Spiking neural networks for embedded event-based vision

Amélie Gruel

UNIVERSITÉ CÔTE D'AZUR

ÉCOLE DOCTORALE
SCIENCES ET TECHNOLOGIES DE L'INFORMATION ET DE LA COMMUNICATION

THÈSE DE DOCTORAT

Réseaux de neurones impulsionnels pour la vision embarquée basée sur les événements

Amélie GRUEL

Laboratoire d'Informatique, de Signaux et Systèmes de Sophia Antipolis (I3S)
UMR7271 Université Côte d'Azur CNRS

Présentée en vue de l'obtention du grade de docteur en INFORMATIQUE **d'** Université Côte d'Azur

Dirigée par :
Pr. Jean MARTINET – I3S, France
Co-encadrée par :
Dr. Laurent PERRINET – INT, France
Soutenue le : 06/10/2023

Devant le jury, composé de:
Pr. Benoit MIRAMOND – LEAT, France
Dr. Chiara BARTOLOZZI – IIT, Italy
Dr. Timothée MASQUELIER – CerCo, France
Pr. Michel PAINDAVOINE – LEAD, France
Pr. Elisabetta CHICCA – CogniGron, Netherlands
Dr. Yulia SANDAMIRSKAYA – ZHAW, Switzerland
Dr. Ryad BENOSMAN – Meta, USA

chist-era

APROVIS3D

RÉSEAUX DE NEURONES IMPULSIONNELS POUR LA VISION EMBARQUÉE BASÉE SUR LES ÉVÉNEMENTS

Spiking neural networks for embedded event-based vision

Amélie GRUEL

Jury :

Président du jury

Benoit MIRAMOND, Full Professor, Université Côte d'Azur, Laboratoire d'Electronique, Antennes et Télécommunications (LEAT), France

Rapporteur·se·s

Chiara BARTOLOZZI, Senior Researcher, Italian Institute of Technology (IIT), Italy
Timothée MASQUELIER, CNRS Research Director, Université Toulouse 3, Centre de Recherche Cerveau et Cognition (CerCo), France
Michel PAINDAVOINE, Emeritus Professor, Université de Bourgogne, Laboratoire d'Etude de l'Apprentissage et du Développement (LEAD), France

Examinateur·trice·s

Elisabetta CHICCA, Full Professor, University of Groningen, Groningen Cognitive Systems and Materials Center (CogniGron), Netherlands
Yulia SANDAMIRSKAYA, Senior Researcher, Zurich University of Applied Sciences (ZHAW), Intel's Neuromorphic Computing Lab, Switzerland

Directeur de thèse

Jean MARTINET, Full Professor, Université Côte d'Azur, Laboratoire d'Informatique, de Signaux et Systèmes de Sophia Antipolis (I3S), France

Co-encadrant de thèse

Laurent PERRINET, CNRS Research Director, Université Aix-Marseille, Institut de Neurosciences de la Timone (INT), France

Membres invité·e·s

Ryad BENOSMAN, Research Director, Meta, USA

Université Côte d'Azur

Amélie GRUEL

Réseaux de neurones impulsionnelspour la vision embarquée basée sur les événements

xiv+253 p.

This document was prepared with LaTeX2e and the class `these-ISSS` version v. 2.10.

Impression : output.tex – 10/1/2024 – 11:01

Révision pour la classe : these-ISSS.cls,v 2.10 2020/06/24 14:16:37 mpelleau

This work was supported by the European Union's ERA-NET CHIST-ERA 2018 research and innovation programme under grant agreement ANR-19-CHR3-0008.
The authors are grateful to the OPAL infrastructure from Université Côte d'Azur for providing resources and support.

Réseaux de neurones impulsionnels pour la vision embarquée basée sur les événements

Résumé

La vision par ordinateur embarquée est récemment devenue omniprésente. Elle englobe des tâches telles que la détection, la reconnaissance et le suivi d'éléments visuels, avec des applications en robotique (conduite autonome), dans l'industrie (évaluation de la qualité des produits, automatisation de tâches répétitives), dans la sécurité, pour l'expérience client, dans les réseaux sociaux, etc. Cet engouement généralisé ne fait que renforcer la nécessité de surmonter les défis posés par ce domaine de recherche, à savoir une consommation d'énergie gargantuesque, une grande mémoire et la prise en charge d'un large éventail d'algorithmes.
Nous pensons qu'une réponse prometteuse à ces défis peut être amenée par l'utilisation combinée de réseaux de neurones à impulsions (SNNs) et de caméras événementielles. Les SNNs sont des réseaux de neurones artificiels bio-inspirés qui visent à imiter la dynamique des neurones biologiques en traitant l'information sous forme de séries d'impulsions. Les caméras événementielles sont un nouveau type de capteur visuel bio-inspiré qui génère des données asynchrones en fonction des changements d'intensité des pixels. Elles sont idéales pour les applications en temps réel, mais la grande quantité d'informations temporisées qu'elles génèrent est difficile à traiter à l'aide de modèles traditionnels de vision par ordinateur. Cependant, les données événementielles se combinent naturellement aux SNNs en termes d'inspiration biologique, d'économie d'énergie, de latence et d'utilisation de la mémoire, pour le traitement dynamique des données visuelles.

Cependant, la nouveauté des SNNs et caméras événementielles laisse place à de nombreuses améliorations en termes de prétraitement optimal des données ainsi que leur traitement subséquent, en tirant le meilleur parti des particularités de ces concepts scientifiques. Dans cette thèse, nous avons identifié plusieurs problématiques liées à ce vaste champ de recherche, que nous avons condensées en deux thèmes principaux.
La première problématique concerne l'optimisation du prétraitement embarqué des données événementielles acquises par une caméra embarquée pour en faciliter l'analyse ultérieure. Nous proposons trois solutions : les événements pourraient soit 1) être réduits dans l'espace ou dans le temps, *online* ou *offline* ; 2) ne conserver que les éléments saillants et rejeter le reste ; 3) être soumis à un mécanisme de fovéation, selon un compromis bio-plausible entre les deux solutions précédentes. Nous avons comparé qualitativement et quantitativement les données obtenues après chaque méthode de prétraitement afin d'évaluer laquelle de ces méthodes amène à un meilleur compromis entre entre la quantité de données (c'est-à-dire le nombre d'événements) conservées et la pertinence de l'information préservée.
Le second défi est l'exploitation de l'adéquation des SNNs pour traiter la temporalité spécifique des événements dans un contexte embarqué.
Dans l'ensemble, nous espérons avoir apporté une contribution utile à l'exploitation des avantages uniques de la combinaison des SNNs avec les caméras événementielles pour la vision par ordinateur embarquée, en particulier en ce qui concerne le prétraitement des données événementielles.

Mots-clés : Réseaux de neurones impulsionnels, Caméras événementielles, Neuromorphique, Vision par ordinateur, Neurosciences computationnelles, Reconnaissance de motifs.

Spiking neural networks for embedded event-based vision

Abstract

In recent years, embedded computer vision has become omnipresent. It encompasses tasks such as detection, recognition and tracking of visual elements, with applications in robotics (autonomous driving), industries (assessment of product quality, automation of repetitive tasks), security, customer experience, social networks, etc. This widespread impetus only reinforces the need to overcome the challenges posed by this area of research, which requires gargantuan energy, high memory, and support for a wide range of algorithms.
We believe a promising solution to these challenges can be found in the combined use of spiking neural networks (SNNs) and event-based cameras. SNNs consist of a bio-inspired artificial neural network aiming to mimic the dynamics of biological neurons by processing the information as spike trains. Event cameras are a novel type of bio-inspired visual sensor which generates asynchronous data according to pixel intensity changes. It is ideal for real-time applications, but the great amount of timed information is challenging to process using standard computer vision models. Moreover, event data make a natural match for SNNs in terms of biological inspiration, energy savings, latency and memory use for dynamic visual data processing.

However, the novelty of SNNs and event cameras leaves room for many improvements in terms of optimal preprocessing of data as well as how this data is processed, making the most of the particularities of these scientific concepts. In this thesis, we identified several questions related to this broad field of research, which we have condensed into two main topics.
The first issue concerns the optimisation of the embedded preprocessing of event data acquired by an onboard camera to facilitate subsequent analysis. We propose three solutions: event data could either 1) be spatially or temporally reduced, either online or offline; 2) keep only salient elements and discard the rest; 3) be subject to foveation, as a bio-plausible compromise between the two previous solutions. We compared qualitatively and quantitatively the event data obtained after applying each preprocessing method to assess which method yields the best trade-off between the amount of data (i.e. events) kept *versus* the relevance of information maintained.
The second challenge is the exploitation of the SNN relevance to processing the specific chronology of event data in an embedded context.
Overall, we hope to have made a valuable contribution to the exploitation of the unique advantages of combining SNNs and event cameras for embedded computer vision, especially concerning event data preprocessing.

Keywords: Spiking neural networks, Event-based cameras, Neuromorphic, Computer vision, Computational neuroscience, Pattern recognition.

Acknowledgements

This was, strangely enough, the hardest pages to write in this manuscript. How can I properly thank all the people who helped me during those 3 eventful years of new discoveries, hard labour and intense scientific research ?

Let me start first with my supervisors: I would like to express my gratitude to my PhD director Jean Martinet, who took a chance on me despite my initial lack of training in AI and helped me develop a broad range of new skills, from computer science to project management and internship supervision. Thank you for your help and all the opportunities you have given me, thanks to which I have been able to visit many countries and meet many interesting people. A warm thank you to my co-supervisor Laurent Perrinet, who introduced me to this thesis subject. Thanks to you I was able to apply for it and thus take part in the APROVIS3D project. I greatly enjoyed all our interesting conversations and your enthusiasm for biology and computational neuroscience.

I would like to thank Chiara Bartolozzi, Timothée Masquelier, Michel Paindavoine, Ryad Benosman, Elisabetta Chicca, Benoit Miramond and Yulia Sandamirskaya for accepting to be part of my jury. A special thank you to Chiara, Timothée and Michel for your relevant feedback on my manuscript, and to Benoit for agreeing to preside this jury. It was a source of pride to defend my thesis in front of these renowned researchers and I am delighted that my defence led to such instructive discussions.

A warm thank you to the APROVIS3D members for our enlightening discussions during our meetings as well as for the multidisciplinary aspect of this project, which taught me a lot. I would like to especially offer my appreciation to Michele Magno, Teresa Serrano-Gotarredona and Bernabe Linares-Barranco for their invitation to visit their laboratories in Zurich and Sevilla respectively, and for the work we did together there. Similarly, a special thanks to Antoine Grimaldi, Farnaz Faramarzi, Sotiris Aspragkathos and Dalia Hareb, my fellow APROVIS3D students, for your companionship in this project.

I would also like to thank the SPARKS – BioInfo team and the I3S administrative team for their support during those 3 turbulent years. A special thank you to Pierre Alfarroba for all our lively discussions, to Jean-Paul Comet for your help and advice both as team leader and director of EDSTIC, and to Magali Richir for your unfailing support and your comforting, ever-present help. Additionally, I would like to thank Hugo Bulzomi, who started as one of our interns but whose determination and motivation led to many fascinating exchanges and whom I am now happy to call a friend.

Speaking of friends, I would like to thank all the people I met here in Côte d'Azur: I arrived 3 years ago without knowing anyone, but thanks to your warm welcome and particularly to Laetitia's joie de vivre (an additional thank you for your help on the day of my defence !), I was able to integrate easily into the laboratory. Being part of ADSTIC, first as simple member, then secretary and finally president, also helped me create new friendships and be an active part of the scientific and social life of our laboratory. An additional thank you to my roommates Timothy and Loris for all the parties, the support between fellow PhD students and the lively evenings spent discussing life, the thesis and everything else.

A special thanks to my family: my mother Agnès, my father Guillaume and my brother Nico who believed in me since always, gave me their unconditional love and support throughout my

academic career and helped me reach this goal. An additional thank you to my grandmothers: both to Denise who saw me start my PhD, and Odile who cheered me on to the end.

Finally, I would like to give all my appreciation to my friends from afar with whom I still have the chance to share amazing moments. Thank you to Rey for being part of my life since our licence and for all our mutual admiration and encouragement. A great thank you to Coralie, Vincent and Elsa for all the laughter, the board and escape games, the parties, the endless nights during our master working on projects and association stuff, and above all for keeping up with me even though I moved almost 1,000 km away. Thank you to all my friends at Telligo for putting up with me while I was "taking a break" in summer camps ; there are too many to name here but I believe some do deserve a special shout-out, including the newly met Théodorusse, Trottynettes, Corto, Mélusine and Thalion. I would like to thank in addition Haz, whose growing friendship fills me with joy and whom I can never thank enough for her help in celebrating my thesis defence. I give all my gratitude to my dear Chocolatine for being the little sister I never had and especially the best nurse there is.

At last, I would like to thank from the bottom of my heart my best friends Karavanes and Xymus, who love me as much as I love them and supported me through all the ups and downs, the dramas and the joys of these last 3 years. This thesis would never have been possible without you two by my sides.

To all of you, thank you for everything we lived together and all that is to come,

Dr. Amélie "Arte" Gruel

Table of contents

1 Introduction **1**
- 1.1 How can visual neuroscience improve computer vision ? . . . 1
  - 1.1.1 Why should computer vision draw inspiration from visual neuroscience ? 1
  - 1.1.2 Which neuro-inspired tools should one use to improve computer vision ? 3
- 1.2 Scientific context . . . 4
- 1.3 Our motivations and objectives . . . 5
  - 1.3.1 Neuromorphic data preprocessing for optimised online analysis . . . 5
  - 1.3.2 Neuromorphic embedded processing of spatiotemporal patterns . . . 9
- 1.4 Our contributions . . . 9
  - 1.4.1 How can we optimise the trade-off between quantity and quality of the visual information as a preprocess for any embedded data processing ? . . . 9
  - 1.4.2 How can we exploit the relevance of the spiking neuron to process the specific chronology of event data ? . . . 12
- 1.5 Manuscript outline . . . 14

Notations **15**

State of the art

2 Spiking neural networks **19**
- 2.1 Neuronal dynamics . . . 21
  - 2.1.1 Behaviour of a biological neuron . . . 21
  - 2.1.2 Behaviour of a spiking neuron . . . 23
  - 2.1.3 Parameters . . . 25
  - 2.1.4 Network architecture . . . 26
- 2.2 Neuromorphic hardware . . . 30
  - 2.2.1 Simulation on CPU/GPU . . . 30
  - 2.2.2 Human Brain Project's SpiNNaker . . . 34
  - 2.2.3 Intel's Loihi . . . 34
  - 2.2.4 Center for Project-Based Learning's Kraken . . . 34
- 2.3 Applications . . . 35

3 Event-based cameras **37**
- 3.1 Operation of an event camera . . . 40
  - 3.1.1 Biological retina . . . 40
  - 3.1.2 Silicon retina . . . 40
- 3.2 Event data representations . . . 42
  - 3.2.1 Representations retaining temporal accuracy . . . 42
  - 3.2.2 Representations losing temporal accuracy . . . 42
  - 3.2.3 Tools for event data handling . . . 42

3.3 Applications and corresponding datasets . . . 43
3.3.1 DVS 128 Gesture . . . 43
3.3.2 N-MNIST . . . 44
3.3.3 DDD17 . . . 44
3.3.4 DSEC . . . 45
3.3.5 Lip reading . . . 46
3.3.6 Overview of existing datasets . . . 47
3.4 Limits . . . 49

4 Spiking neural networks applied to an embedded event camera **51**
4.1 Embedded systems . . . 53
4.2 Why is it such a good idea ? . . . 53
4.3 Applications . . . 54
4.3.1 Classification . . . 54
4.3.2 3D reconstruction . . . 56
4.3.3 Visual attention . . . 59

5 Concepts from neuroscience **61**
5.1 Visual attention . . . 63
5.1.1 Biological visual attention . . . 63
5.1.2 Neuromorphic visual attention . . . 67
5.2 Foveation . . . 70
5.2.1 Biological foveation . . . 70
5.2.2 Foveation in computer vision . . . 70
5.2.3 Foveated event-based camera . . . 72
5.3 Delay learning and detection of spiking motifs . . . 72
5.3.1 Biological delay learning . . . 72
5.3.2 Neuromorphic models for delay learning . . . 74

Contributions

6 Event data downscaling **79**
6.1 Introduction . . . 81
6.2 Spatial downscaling . . . 82
6.2.1 Spatial funnelling . . . 82
6.2.2 Log luminance reconstruction . . . 83
6.2.3 Events to video to events . . . 89
6.2.4 SNN pooling . . . 90
6.3 Temporal downscaling . . . 94
6.3.1 Temporal funnelling . . . 94
6.3.2 Structural downscaling . . . 95
6.4 Experimental comparison . . . 96
6.4.1 Comparison according to activity measures . . . 97
6.4.2 Comparison of real-time downscaling . . . 100
6.4.3 Comparison according to performance on a classification task . . . 103

6.5 Human assessment study . . . 106
6.5.1 Human machine interface and protocol . . . 106
6.5.2 Assessing human performance . . . 109
6.5.3 Comparing human and machine performance . . . 112
6.6 Conclusion . . . 114

7 Saliency detection by event density **117**
7.1 Introduction . . . 119
7.2 Saliency detection . . . 121
7.2.1 Input layer . . . 121
7.2.2 Saliency detector – soft exponential Winner-Takes-All . . . 121
7.2.3 Saliency detector – weight adaptation rule . . . 123
7.3 Spatiotemporal filtering — saliency detection of one object of interest at a time . . . 124
7.3.1 Network architecture . . . 124
7.3.2 Threshold adaptation rule . . . 124
7.3.3 Implementation on neuromorphic hardware . . . 126
7.4 Multi spatiotemporal filtering — simultaneous saliency detection of multiple objects of interest . . . 128
7.5 Experimental validation . . . 131
7.5.1 Visual input data . . . 131
7.5.2 Validation of the attention selection of one object of interest . . . 132
7.5.3 Validation of the simultaneous attention selection of multiple objects of interest . . . 142
7.5.4 Comparison with State-of-the-Art approaches . . . 144
7.6 Conclusion . . . 144

8 Neuromorphic foveation **147**
8.1 Introduction . . . 149
8.2 Methodology — Binary foveation . . . 150
8.3 Experimental validation . . . 152
8.3.1 Performance on a classification task . . . 152
8.3.2 Performance on a segmentation task . . . 155
8.3.3 Quantitative assessment of the saliency detection . . . 156
8.4 Discussion . . . 156
8.5 Conclusion . . . 159

9 Towards embedded applications — First steps and perspectives **161**
9.1 Introduction . . . 163
9.2 Coastline detection . . . 164
9.2.1 Adapting an ANN classifier into an embedded SNN classifier . . . 164
9.2.2 Delay learning . . . 167
9.2.3 Line detection . . . 175
9.3 3D reconstruction . . . 177
9.3.1 Embedding the model on SpiNN-3 . . . 177
9.3.2 Stereo-matching performance . . . 178
9.3.3 Live demonstrator . . . 179

9.4 End-to-end neuromorphic lip reading . . . 180
9.4.1 i3S dataset . . . 181
9.4.2 Neuromorphic lip reading model . . . 183
9.4.3 Experimental validation . . . 185
9.4.4 Limitations and perspectives . . . 186
9.5 Conclusion . . . 188

10 Conclusion **189**
10.1 Summary of contributions . . . 189
10.2 Perspectives . . . 191
10.2.1 Short term perspectives . . . 191
10.2.2 Medium term perspectives . . . 192
10.2.3 Long term perspectives . . . 193
10.3 Next steps . . . 193

Bibliography **195**

List of figures **221**

List of tables **225**

Annexes

A Visualisation of RoIs detected by our spatiotemporal visual attention mechanism 231
B Visualisation of different spatial downscaling methods. . . . 233
C Visualisation of the stakes of neuromorphic foveation . . . 237
D Pipelines for event data handling . . . 240
D.1 From AEDAT2 to CSV files . . . 240
D.2 From CSV to NPZ files . . . 240
E Analysis of DVS and RGB coastline dataset . . . 241
F Line detection . . . 245
G 3D reconstruction . . . 248
G.1 "Two fans" dataset . . . 248
G.2 "Pendulum" dataset . . . 249
G.3 DSEC . . . 250
G.4 Communication between the SpiNNaker board and an external device . . . 251
H i3S dataset . . . 252
H.1 Composition of the i3S dataset . . . 252
H.2 Tree structure of the i3S dataset . . . 252

CHAPTER 1

Introduction

Human beings have always drawn inspiration from the biological world surrounding us to create new technologies. Over 500 years ago, Leonardo da Vinci was one of the first people to theorise the concept of bio-mimicry — *"Go take your lessons from nature, that's where our future lies"* — and his inventions paved the way for bio-inspired technologies such as aeroplanes, Velcro, solar panels, turbines, pressure-resistant helmets, efficient heating systems, hammers, vaccination, ... and more recently, artificial intelligence.

Talos, Golems, Frankenstein and other manufactured intelligent humanoids have been part of the popular consciousness since ancient times. Still, it was only in 1950 that Alan Turing published his landmark paper "Computing Machinery and Intelligence", where he discussed machines' potential consciousness and thinking ability. Six years later, the historic workshop Dartmouth Summer Research Project on Artificial Intelligence was the birthplace of the field of artificial intelligence (AI) research, especially the Logic Theorist program, which aimed to mimic human problem-solving skills. This conference was followed by a long period of successive success and setbacks, which finally ended in the 2000s with the successful application of machine learning (ML) to academia and industry tasks combined with the increase in computer hardware performance and the size of available data.

1.1 How can visual neuroscience improve computer vision ?

1.1.1 Why should computer vision draw inspiration from visual neuroscience ?

In recent years, the performance of computer vision algorithms increased tremendously. In 2010, the first winner of the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) (Deng et al., 2009) reached a 52.9% accuracy (Russakovsky et al., 2014), which AlexNet, i.e. the first deep convolutional neural network (CNN) applied to ImageNet, quickly outperformed in 2012 (63.3% accuracy) (Krizhevsky, Sutskever, & Hinton, 2012). According to (Alom et al., 2018), this "significant breakthrough in the field of ML and computer vision for visual recognition and classification tasks [...] is the point in history where interest in deep learning increased rapidly". From this point on, the successive ILSVRC winners produced increasingly good results, reaching in 2023 an impressive 91.1% accuracy on a global dataset of more than $14,2M$ samples distributed in almost $22K$ classes (X. Chen et al., 2023). It is worth noting that these DL models' performance overcame the human performance in 2015 (see Fig. 1.1), only three years after the breakthrough bought by AlexNet. Indeed, it was experimentally assessed that trained human annotators reach a maximum accuracy of 94.9% (top-5 classification error) while classifying samples from ImageNet (Russakovsky et al., 2014), which was surpassed by ResNet-152's validation error of 95.4% (He, Zhang, Ren, & Sun, 2016). An "optimistic" human

accuracy was approximated at 97.6% — equally surpassed by a DL model in 2018 (Mahajan et al., 2018).

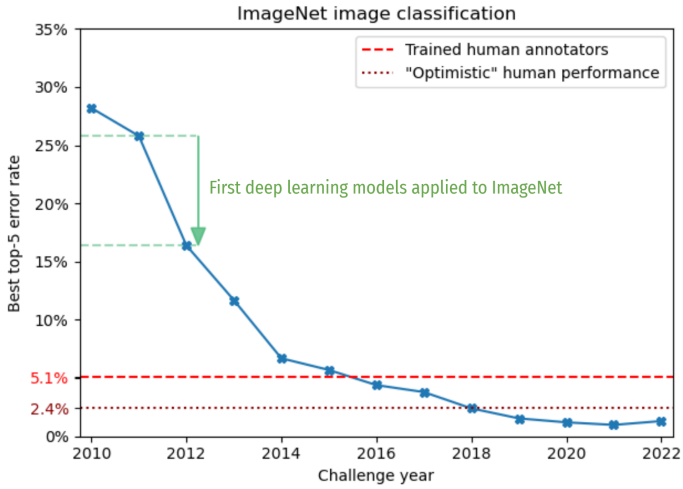


Figure 1.1 – Evolution of ImageNet image classification performance over time.

Overall, these recent achievements of DL models could suggest that despite its early bio-inspiration, ML no longer needs visual neuroscience to improve its performance. As recently declared by Yann LeCun, one of the scientific leaders in AI research, the latter should no longer "be blinded by biology" (Cox & Dean, 2014).

However, this progress must be nuanced: the superhuman performances are to be counterbalanced by the proportionate size of these networks and by the gargantuan quantity of energy consumed to produce these results. Let us take a closer look at some of the models mentioned previously: ResNet-152 used $40M$ parameters ; AlexNet consists of 13 layers, including 5 convolutional ones, implemented with $60M$ parameters ; and the current ILSVRC 2023 laureate reached a substantial $2440M$ parameters. While the computer performance in terms of floating-point operations per second (FLOPS) is known * for each of these architectures, the energy consumption has not yet been studied — however, it has been estimated that a recognition task among 1,000 different objects performed on a standard computer consumes about $250W$ (Weerasinghe, 2022).

If we direct our gaze to more modern systems, Chat-GPT3 uses 175 billion parameters, and its energy consumption during training was approximated at $1,287MWh$ (Patterson et al., 2021), i.e. around 550 tonnes of CO_2eq. This corresponds to the equivalent of 550 round trips from Paris to New York by plane and 250 times what a European citizen is supposed to consume per year to achieve carbon neutrality ! We do not know the actual energy consumption of the most recent version of Chat-GPT (based on GPT4), but we can only assume from its staggering number of parameters (170 trillion) that it must approximate a $1,000$ times the consumption of GPT3.

∗. According to (Desislavov, Martínez-Plumed, & Hernández-Orallo, 2021), AlexNet performs at 1.52 GFLOPs and ResNet-152, 22.6 GFLOPs. The number of the current ILSVRC winner has not been made publicly available yet ; however, we can note that the 2021 laureate ViT-G/14 reached an accuracy of 91.1% using $1843M$ parameters and performing at 5270 GFLOPs (Dai, Liu, Le, & Tan, 2021).

In contrast to the neural networks described above, the human brain from which their designers draw inspiration is significantly more energy efficient: with a power budget of only $20W$, it allows us to "make sense of a complex and ever-changing world, to plan complex actions, to navigate our social environment and intuit the minds of others, and to learn and remember across our entire lifespans" (Cox & Dean, 2014). Its architecture enables these impressive feats: over 100 billion neurons operating simultaneously, interconnected by 100 trillion synapses. The latter propagate action potentials at a relatively low speed ranging from 1 to $100m/s$, with an additional slowdown caused by dendritic and axonal delays ; the visual signal perceived by the retina reaches the primary visual cortex after $50ms$, and the later stages of visual processing take place after almost $200ms$. However, even though these propagation times are way slower than manufactured silicon computers performing up to billions of computational cycles per second, the cerebral structure's high parallelism and dense connectivity allow information to span between distant brain regions rapidly (Liao, Vasilakos, & He, 2017).

All the figures described above underline the brutal difference in energy consumption between computer vision and a human being, which even the superhuman performances mentioned previously do not make up for. To close this gap and reduce the impact of the digital world on the environment (responsible for 3.8% of greenhouse gas emissions and consuming 4.2% of primary energy in 2019 (Bordage, 2019)), it is crucial to draw inspiration from visual neuroscience to improve computer vision.

1.1.2 Which neuro-inspired tools should one use to improve computer vision ?

Introduced as the "third generation of neural network models", SNNs represent an asynchronous type of artificial neural network (ANN) closer to biology than traditional artificial networks, mainly because they seek to mimic the dynamics of neural membrane and action potentials over time (Maass, 1997). Their event-driven nature (the SNN's neurons with no spikes remain idle while the ANN's units are always active), as well as their summation operations (binary inputs compared to the ANN's floating point values) that are inherently less computationally expensive, contribute to their energy efficiency over a given period (Roy, Jaiswal, & Panda, 2019). The SNNs' bio-inspiration allows for them to be implemented on neuromorphic hardware, i.e. silicon architectures whose elemental building blocks work as biological neurons and transmit the information as electric impulses (the "spikes").

Event cameras are a novel bio-inspired sensor, akin to a "*silicon retina*" (Mahowald & Mead, 1991), which encodes the visual information as spikes ; they are bringing an emerging vision paradigm by mimicking the biological retina. In 2008, Lichtsteiner, Posch and Delbruck presented the first complete design of a Dynamic Vision Sensor (DVS) (Lichtsteiner, Posch, & Delbruck, 2008a) responding only to temporal brightness change in a scene with no consideration for colours, similar to the organic retina. This asynchronous sensor records visual data with a microsecond temporal resolution and a high dynamic range ($120dB$ compared to the human eye's $35dB$ and traditional red-green-blue (RGB) camera's $60dB$), with significantly low power ($10mW$ compared to the human eye's $10mW$) (Gallego, Delbruck, Orchard, & al., 2020). Event data require low memory to be stored, 1000 times less than RGB videos (Kilobytes versus Megabytes) (Rebecq, Gallego, Mueggler, & Scaramuzza, 2018).

SNNs receive and process information as spike trains, meaning as a non-monotonous sequence of activations. Therefore, they make for a suitable candidate for the efficient processing of incoming event patterns measured by event-based cameras: each event is assimilated to an activation

spike between two spiking neurons. As stated by (Roy et al., 2019), "a suitable spiking neuron model with proper synaptic plasticity while exploiting event-based, data-driven updates is a major goal among neuromorphic engineers to enable computationally efficient intelligent applications".

Both those scientific concepts fall within the spectrum of "neuromorphic computing", a concept invented in the 1980s by Prof. Carver Mead at Caltech "which combines biology, electrical engineering, computer science and mathematics to create ANNs able to handle information as a human brain and nervous system would" (L, 2021). Nowadays, this concept includes both different processing strategies, such as spintronics, in-memory computing, SNNs, photonics, etc (Christensen et al., 2022), as well as various sensors. We can mention some for each of the five senses: (Mahowald & Mead, 1991) and (Lichtsteiner et al., 2008a) for the sight, (Lyon & Mead, 1988) and (Chan, Liu, & van Schaik, 2007) for the hearing, (Koickal et al., 2007) and (Koickal et al., 2007) for the smell, (Yang et al., 2023) and (LeBow et al., 2021) for the taste and (Birkoben, Winterfeld, Fichtner, Petraru, & Kohlstedt, 2020) and (Macdonald, Lepora, Conradt, & Ward-Cherrier, 2022) for the touch.

In this thesis, we propose tackling this challenging task by capitalising on two compatible scientific concepts: SNNs and event cameras, both allowing for neuromorphic computing. We present below the scientific context and motivations of this work, as well as the contributions achieved.

1.2 Scientific context

This PhD work has been conducted under the supervision of Prof. Jean Martinet, within the SPARKS Team (i3S, 2023b) at the I3S laboratory in Sophia Antipolis, France (i3S, 2023a); and has been co-supervised by Dr Laurent Perrinet, a researcher at the Institute of Neurosciences of Timone in Marseille, France (INT, 2023).

This thesis is undertaken in the context of the European APROVIS3D project (April 2020-September 2023) (APROVIS3D, 2023), supported by the European Union's ERA-NET CHIST-ERA 2018 research and innovation programme under grant agreement ANR-19-CHR3-0008. The objective of the APROVIS3D project is to design and implement ML methods based on SNNs to extract visual features and infer useful information from the visual scene using stereo event cameras. In particular, the project focuses on two use cases: coastline tracking and depth estimation.

The consortium from four EU countries shares a common interest in analog-based computing and computer vision and offers a unique combination of expertise this project requires. It was created with the following objective: SNNs specialists from various fields, such as visual sensors (IMSE, Spain), neural network architecture and computer vision (UCA, France and University of Lille, France) and computational neuroscience (INT, France), will team up with robotics and automatic control specialists (NTUA, Greece) and low power integrated systems designers (ETHZ, Switzerland) to help geoinformatics researchers (UNIWA, Greece) build a demonstrator UAV for coastal surveillance, evaluated at software level for comparison with standard image-based approaches in terms of accuracy and energy consumption, before hardware integration.

The key challenges of this project will be to design an end-to-end analog system (from sensing to AI-based control of the UAV and 3D coastal volumetric reconstruction) which is energy efficient and adapted to real conditions. The project relies on SpiNNaker applied to a stereopsis system dedicated to coastal surveillance using an aerial robot. This project is additionally the oc-

casion to embed and experimentally assess the neuromorphic foveation mechanism as designed by IMSE (Serrano-Gotarredona, Faramarzi, & Linares-Barranco, 2022). APROVIS3D aims to show that such a bioinspired analog design will bring considerable benefits regarding energy efficiency and adaptability needed to make coastal surveillance with UAVs practical and more efficient than digital approaches.

1.3 Our motivations and objectives

As described above, the main objectives of the APROVIS3D project are coastline detection for autonomous navigation of a UAV and 3D reconstruction from a stereo stream of events, using an SNN implemented on the neuromorphic hardware SpiNNaker and interfaced with two DVSs, both embedded on a UAV. Our work falls within this broad spectrum and, more specifically, addresses the issue of optimising [†] the preprocessing of event data acquired by an event camera to facilitate subsequent analysis. We specifically target their combined use for embedded systems, i.e. hardware systems with software designed to perform real-time operations for a dedicated function either independently or as a part of a larger system (HEAVY.AI, 2022). The UAV application task of APROVIS3D dictates this choice, which leads to the development of frugal neural networks.

1.3.1 Neuromorphic data preprocessing for optimised online analysis

Indeed, embedded systems and even early realised neuromorphic embedded processors are quite limited in terms of memory bandwidth. A UAV has available low processing power and low memory: its limited battery lifetime leads to limited flight time and requires planning UAV missions to minimise energy consumption (Abeywickrama, Jayawickrama, He, & Dutkiewicz, 2018).

While the data produced by event cameras are inherently sparse and seem ideally suited to minimise the issue of embedded energy consumption and data storage, the event flow can be very high. Event cameras have a very high temporal resolution (in the order of microseconds), meaning that one pixel can theoretically generate 1 million events per second at most. A DVS128 of resolution 128×128 could produce thus 16.4 billion events per second. While this case never happens in real use due to the events' intrinsic sparsity and non-redundancy, several million events per second have been recorded in highly dynamic and textured scenes (Tayarani-Najaran & Schmuker, 2021). These heavy event streams require significant resources to handle and may be too dense to be correctly processed by the state-of-the-art low-power embedded system. Indeed, as stated in (Sorbaro, Liu, Bortone, & Sheik, 2019): since most neuromorphic architectures require the neuron state to be fetched from memory at each synaptic event then be rewritten and each of those operations consume a certain amount of energy (usually around $10^{-11}J$ (Indiveri & Sandamirskaya, 2019) – $8x10^{-9}J$ for SpiNNaker (Stromatias, Galluppi, Patterson, & Furber, 2013) and $23.6x10^{-12}J$ for Loihi (Davies et al., 2018)), "reducing the number [of synaptic operations] is therefore the most natural way to keep energy usage low". Furthermore, recent study shows that in certain lighting conditions, high-resolution event cameras produce data susceptible to temporal noise and with an increasingly high *per-pixel event rate*, thus leading to the decreased performance of some traditional computer vision tasks (D. Gehrig & Scaramuzza, 2022). While

†. Disclaimer: although we often use the term "optimal" or "optimisation" in this manuscript, we are in no way involved in combinatorial computing. Our use of the term "optimal" in this work corresponds to its literary definition, meaning ideal or most desirable.

desktop event vision solutions can afford considerable computing resources, embedded processing can suffer from hardware limitations such as energy resources, memory, computational power and communication bandwidth. In such situations where a dense event stream can neither be stored nor processed online, it is likely that some (if not most) events will be randomly dropped, which will obviously impede correct processing.

Additionally, this heavy flow of information may not always be relevant to the task at hand. Some work simply chose to reduce this information by pooling the events into frames, then processing this data as standard visual images (Fang, Yu, Chen, Masquelier, et al., 2021 ; Huang, 2021). A concrete example can be found in real-time event-based odometry applied to robotics, which requires only movement information and whose needs can typically be answered by frame analysis (Rebecq, Horstschaefer, & Scaramuzza, 2017). However, its main drawback is that the asynchronicity of the event data, which is a real advantage of event data, is disregarded ; all events are gathered, and their timestamps are discretised at a higher temporal grain.

Other works aim to reduce the induced high storage requirements by implementing event data compression methods, such as entropy coding (Gallego et al., 2020), dictionary-based compression (Ziv & Lempel, 1977 ; Alakuijala et al., 2018), Internet of Things specific compression (Blalock, Madden, & Guttag, 2018 ; Gallego et al., 2020), fast integer compression (Khan, Iqbal, & Martini, 2020, 2021), etc. These compression techniques can be lossy — see Poisson Disk Sampling-based Lossy Event Compression (Banerjee, Wang, Chopp, Cossairt, & Katsaggelos, 2021) — or lossless — see (Khan et al., 2020, 2021). Their major disadvantage is their inability to compress data in real-time during acquisition.

Furthermore, deep SNN models can have difficulties handling a great amount of data simultaneously while minimising their energy consumption, especially at a high temporal resolution. Some chose to compress these models using attention, as proposed in (Kundu, Datta, Pedram, & Beerel, 2021).

All in all, drawing from the multiple examples presented above, we are convinced that the issue of event data optimisation for onboard analysis must be addressed. Therefore the first core problem of this thesis is: **How can we optimise the trade-off between quantity and quality of the visual information as a preprocess for any embedded data processing ?** In other words, what is the best method to reduce the number of events to be processed while retaining a maximum amount of information relevant to this process to optimise the embedded system performance ?

Event-data downscaling Our first approach to solving this wide-ranging issue is to reduce the event stream by simply downscaling the spatial and temporal resolution. Downscaling vision data is an essential feature in a system, especially if embedded, so as to be able to adjust the complexity of data to the available resources, such as processing capability and energy consumption. Reducing the size of visual data is straightforward for grayscale and RGB images with interpolation. When it comes to event data, size reduction is much less trivial.

It is important to note here that our main goal above all is to reduce the number of events in the input stream above. To solve this problem, we opted to reduce either the spatial dimension of the sensor or the grain size of the data recording time. These solutions should reduce the number of events by decreasing the number of possible spatial or temporal coordinates. However one should note that the impact of event reduction on the energy consumption depends on the downstream system: if it is event-based, the goal of lower energy consumption is valid. However, in a clock-based system, the downscaling of the temporal resolution is more impacting than a lower number

of events: a lower temporal resolution leads to a smaller number of timesteps to simulate, thus a lower energy consumption and processing time ; which is not necessarily the case for a lower number of events. Regarding spatial downscaling, the reduced map size is beneficial for both event- and clock-based systems. Additionally, one could argue that the reduced data provided as input would only have an impact on the resources consumed by the first layer of the downstream network, especially in DL models. This argument can easily be dismissed here, as we are looking to facilitate embedded event data processing and such DL models are rarely embedded due to their high energy consumption. Furthermore, it is easily observable in many deep SNN models that the output activity is proportional to the output activity.

Most existing event vision approaches handle event data preprocessing by simply downscaling the pixel coordinates, which is actually a max-pooling operation in the event domain. Such as strategy is adopted by the *Tonic*'s "Downsample" method (Lenz et al., 2021) (see Section 3.2.3), which is, to the best of our knowledge, the only existing tool for event data downscaling post-recording. One obvious issue of this simple event "funnelling", especially in the spatial dimension, is that the resulting event density drastically increases in the reduced spatial dimension, causing a critical information loss. Indeed, the precision of (x, y) coordinates decreases and the fine details of the recorded objects are lost.

Attempts to overcome this issue have been made by using hardware-integrated filters to reduce the event rate. For example, the event camera Prophesee Gen4 implements a row-wise arbitration, introducing systematic errors in the event timestamps, thus randomly skipping events (Finateu et al., 2020). The authors in (Delbrück, Graca, & Paluch, 2021) introduce fixed-step feedback controllers, which automatically adjust the camera parameters according to the input event rate and noise: they adjust the threshold and refractory period to regulate the event rate and adapt the bandwidth to regulate the noise. However, these filters significantly alter the visual data and therefore are not a viable solution.

In neuromorphic ML, SNNs consider each pixel as an input neuron, and each event is produced as an input spike to the neural model. However, they usually cannot handle the flow of information the event data represents as this first layer is often too important for efficient computation ; thus, most authors applying this type of neuromorphic computing on event data resolve to convolve the input. For example, (Cordone, Miramond, & Ferrante, 2021) apply multiple convolution layers to the dataset *DVS128 Gesture* before processing it. We can also mention the residual graph convolutional techniques, which handle each input event as a node in the graph (Bi, Chadha, Abbas, Bourtsoulatze, & Andreopoulos, 2019): this leads to low computational efficiency and a substantial simulation run and only produces satisfying results on a sparse dataset with well-defined empty areas like the one they use.

The aforementioned examples are just a few reasons why event downscaling is a real challenge in neuromorphic computer vision and needs to be formalised as a step further towards sparse data processing.

Visual attention for salient selection of OoIs Our second approach to solving the wide-ranging issue of event data preprocessing is to process only the events belonging to the salient elements of the visual scene, i.e. the elements noticeable and/or relevant for the task at hand. We identify regions of interest (RoIs — or salient regions) as a set of pixels recording a high spatiotemporal density of events. The objects of interest (OoIs) thus correspond to events that cluster together spatiotemporally to form the structure of a visual object.

We believe that detecting RoIs in the original visual scene to select the corresponding OoIs (i.e. the events taking place in the detected RoIs) is the missing element to optimising the trade-off between the quantity and the quality of the reduced event data. Additionally, it is a promising approach to increase accuracy, reducing both latency and memory requirements. Indeed, the study of many biological organisms highlighted the importance of limiting the processing of sensory RoIs, thus minimising the energy spent on this task as well as the reaction delay to what is perceived (Tsotsos, 1990). For prey animals, for example, this means a quicker detection of a potential predator and a more efficient escape from danger. Attention was first described as the coalition of "focalisation, concentration and consciousness" by William James in 1890 (James, 1890). According to (Borji & Itti, 2013), visual attention can be defined as the behavioural and cognitive process of selectively focusing on a discrete aspect of sensory cues while disregarding other perceivable information. In psychology, this mechanism is more recently defined as the allocation of limited cognitive processing resources on one or a few relevant environmental elements while ignoring others (Anderson, 2005). It can be either subjective or objective, as well as voluntary or instinctive.

Taking inspiration from the primates' visual attention mechanisms (Moosmann, Larlus, & Jurie, 2006 ; Itti & Koch, 2001 ; Martinet, Lablack, Lew, & Djeraba, 2009 ; Aydemir, Hoffstetter, Zhang, Salzmann, & Süsstrunk, 2023), many computer vision tasks (classification, object tracking, autonomous navigation, etc.) rely on small salient items in the global scene to limit the amount of data to be processed. As technologies for sensory information processing expanded, an increasing number of them assimilated this idea for their own purposes. In the deep learning community, for instance, this concept was converted into a neural network's component weighting features by the task's level of importance. Attention can thus be applied to regions in images, words in text, phonemes in speech, etc.

Works that implement neuromorphic visual attention applied to event data are rare in the literature. Only a few models genuinely take advantage of the intrinsic dynamics of SNNs and the uniqueness of event data: in particular, (Renner, Evanusa, & Sandamirskaya, 2019) make use of the mathematical model of DNF (Sandamirskaya, 2014) as a soft Winner-Takes-All (WTA) to implement tracking of salient objects, which position in the sensor was known due to an initial man-made pre-activation. Therefore, we believe that the neuromorphic salient selection of OoIs is a promising axis of research to optimise the event data to be processed.

Neuromorphic foveation We propose a third and last approach to optimise the on- and off-line processing of event data as a combination of the two previously targeted solutions: foveation, a visual neuroscience mechanism allowing the complex eye to acquire relevant information selectively. We define here neuromorphic foveation as a feedback loop between an event camera and a neuromorphic saliency detector, merging events at multiple resolutions according to identified RoIs. Teresa Serrano-Gotarredona and Bernabé Linares-Barranco, two APROVIS3D collaborators, recently patented an electronically foveated DVS (e-Fov DVS) (Serrano-Gotarredona et al., 2022), whose development was considerably delayed by the sanitary crisis (see Section 5.2.3). As a result of this impediment, this sensor has not been associated with a feedback loop saliency detection mechanism and validated on a computer vision task within the timeline of this work — our objective is to implement software neuromorphic foveation as a first step towards this direction.

1.3.2 Neuromorphic embedded processing of spatiotemporal patterns

While this PhD thesis' core objective is the optimisation of event data preprocessing for embedded neuromorphic analysis (as described in the previous section), it only answers part of the APROVIS3D project goals. Embedded coastline detection and 3D reconstruction will benefit significantly from our results but cannot be achieved solely thanks to them. We thus turned our attention to solving these tasks.

Our analysis of these problems raised the importance and relevance of applying SNNs to event cameras to solve computer vision tasks dedicated to spatiotemporal pattern processing. The second problem we addressed is thus the following: **How can we exploit the relevance of the SNN to exploit the specific chronology of event data in an embedded context ?** In other words, how do we handle precise movements recorded by an event camera using the crucial suitability of SNNs ? The specific tasks we identified as linking this problem to APROVIS3D are coastline detection using delay learning or simple SNN architectures ; 3D reconstruction ; and lip-reading.

1.4 Our contributions

We established in the previous section our motivations to tackle the following two wide-ranging issues: How can we optimise the trade-off between quantity and quality of the visual information as a preprocess for any embedded data processing ? And how can we exploit the relevance of the SNN to process the specific chronology of event data ? In the paragraphs below, we describe the work carried out and the results obtained during this thesis.

1.4.1 How can we optimise the trade-off between quantity and quality of the visual information as a preprocess for any embedded data processing ?

We sought to address this issue by experimenting with several neuromorphic and non-neuromorphic mechanisms.

Event data downscaling As a first approach to solve the problem posed by the excessive flow of events, we implemented ten event data downscaling methods:

- six "spatial" methods, i.e. influencing the spatial resolution according to a spatial dividing factor with no effect on the temporal resolution:
  - spatial funnelling ;
  - event count ;
  - linear log-luminance reconstruction ;
  - cubic log-luminance reconstruction ;
  - SNN pooling with separate handling of the polarity ;
  - SNN pooling with mutual handling of the polarity ;
- four "temporal" methods, i.e. influencing the event flow temporally according to a temporal dividing factor with no effect on the spatial resolution:
  - synchronous temporal funnelling ;
  - asynchronous temporal funnelling ;
  - stochastic structural downscaling ;
  - deterministic structural downscaling.

To compare these various methods, we study quantitative and qualitative descriptors of the two event-based datasets DVS 128 Gesture (Amir et al., 2017) and N-MNIST (Orchard, Jayawant, Cohen, & Thakor, 2015), downscaled using these methods. The quantitative descriptors correspond to the number of events, temporal density, information entropy (as defined in Shannon's information theory) and the computation time for on- and off-line downscaling (respectively corresponding to the reduction of event data as it is being recorded or of the whole dataset at once). The qualitative descriptors correspond to the optical coherence (as estimated by our collaborators at IMSE, specialists in neuromorphic sensors) and the classification performance of the two neuromorphic classifiers using respectively the Parametric Leaky Integrate-and-Fire (PLIF) neuron model (Fang, Yu, Chen, Masquelier, et al., 2021) and the Spike Layer Error Reassignment (SLAYER) framework (Shrestha & Orchard, 2018). We also describe the results of a gesture classification task performed by a set of human users to the latter.

Considering that our application is embedded processing of the reduced data, we give more weight to the criteria "number of events", "online computation time" and "classification performance" (either machine or human). The SNN pooling methods are both implemented on the neuromorphic hardware SpiNNaker (S. B. Furber et al., 2013) ; the results tend to show that the best choice to perform spatial event reduction on an embedded system to adjust the size of data to the available resources, such as processing capability and energy consumption, can be found among the SNN pooling methods. Suppose that we do not limit our criterion to spatial downscaling. In that case, the stochastic structural method that consists in randomly dropping events shows exciting performances, with a classification accuracy that remains very high even when dropping a high ratio of events.

Our human gesture classification assessment showed that the spatial dividing factor needs to be maintained at a certain threshold (8 for a 128×128 sensor) to ensure that human performance does not fall below the chance level. Overall, the quality of the data obtained from our downscaling methods is not uniform, and we identified some surprising discrepancies between human and ML approaches to gesture classification using event data: human accuracy is higher than machine accuracy in specific dividing factors for specific methods, such as the synchronous temporal funnelling and mutual and separate SNN techniques. This study sheds light on the potential limitations of event data downscaling and provides insights into the human perception of gesture classification using event data.

Visual attention for salient selection of OoIs Following the results presented above, we hypothesised that the trade-off between the quantity of events *versus* the quality of the retained information might be improved by only processing events corresponding to OoIs, i.e. salient elements of the visual scene. In collaboration with the groups of Michele Magno (partner of APROVIS3D) and Luca Benini, both at ETH Zürich, we implemented to this end a low-latency neuromorphic model of visual attention and salient spatiotemporal selection, which simultaneously outputs one or multiple OoIs detected in an event-based scene. Our architectures are designed to be lightweight enough to enable running in real-time. The RoIs are detected thanks to intrinsic SNN dynamics that accumulate spikes with a leaky behaviour The generalisability of the model is ensured by dynamic adaptation rules applied to synaptic weights and neural thresholds.

We implemented various versions of our visual attention model:

— a saliency detector, which detects RoIs ;

— a model for the salient selection of one OoI at a time ;

— a model for the simultaneous salient selection of multiple OoIs at a time.

Those models have been developed in close collaboration with fellow members of APROVIS3D at ETH Zürich and IMSE. They have been experimentally validated on a central processing unit (CPU) and on the neuromorphic hardware Loihi (Davies et al., 2018) and the academic platform Kraken (Di Mauro, Scherer, Rossi, & Benini, 2022).

We demonstrated the quality of the visual attention mechanism implemented thanks to quantitative and qualitative descriptors. After salient selection of one OoI in the DVS 128 Gesture dataset (where we hypothesise the OoI to be the moving arms), we obtained a repartition of events in space reduced by 80% and a classification accuracy maintained at an acceptable 70% of the one achieved on the original dataset. On a custom-made dataset with multiple OoIs, we confirmed that our model performs a saliency detection akin to biological visual attention. When confronted with a visual scene with multiple OoIs and asked to select only one, we expect a human viewer to always select either the one with the highest density of information (i.e. highest number of events and spatiotemporal density) or the one that comes first when the shift between the multiple OoIs is big enough. Our model exhibited similar behaviour in our results, which could be improved by experimental validation of our hypothesis concerning the human-like visual attention our model mimics.

Additionally, we demonstrated the usefulness of dynamic rules for online weight and threshold adaptations, which, to our knowledge, cannot be implemented on Loihi: the results obtained with the adaptation rules (on CPU and Kraken) are significantly better than those obtained without those rules (on Loihi).

Our innovative model for simultaneous salient selection of multiple OoIs at a time equally achieves good results. While it only accurately filters out n OoIs out of n initially present in the input event data, it reaches a selection latency of $15ms$ ($5ms$ in best-case scenarios). It selects OoIs with a quality leading to a classification performance reaching 73% of the performance obtained on original data while reducing the activated surface of the sensor (i.e. the average number of pixels activated over time) by 1000, all with no training phase and a reduced number of neurons and connections compared to state-of-the-art salient detection methods.

Implementing the models described above on SpiNNaker (S. B. Furber et al., 2013) is currently being undertaken to align with APROVIS3D constraints.

Neuromorphic foveation We finally worked on drawing benefits from both approaches described below (event data reduction and neuromorphic visual attention mechanisms) to offer a third and last solution to the problem posed by the excessive flow of events. Our neuromorphic foveation model merges events at multiple resolutions according to the RoIs detected using the first model described in the previous section. We experimentally validated this mechanism on software as a preliminary work to its hardware implementation. We implemented a "binary" version of this mechanism, akin to the combination of sample events in high resolution and low resolution using a mask. We applied it to the two datasets DDD17 (Binas, Neil, Liu, & Delbruck, 2017) and DVS 128 Gesture (Amir et al., 2017) to study the performance evolution for two computer vision tasks, respectively semantic segmentation and classification, according to the remaining number of events. We deepened the study using additional quantitative descriptors, i.e. the spatial and temporal densities of the RoIs.

The semantic segmentation results demonstrated that the foveated data managed to keep on average the same performance as with the original dataset while decreasing by two-thirds the mean number of events to reach the low-resolution value. On the classification side, the average number of events per sample drops to 1% of the original metric after reduction and 50% after

foveation, while the performance decreases respectively by 63% and only by 25%. This difference is explained by the intrinsic distinction between the datasets and tasks: classifying a DVS 128 Gesture sample can be done by identifying the movement of the hands, i.e. by following the trajectory of the RoIs while segmenting a driving scene requires more information and cannot rely solely on the data provided in the RoIs, although these greatly facilitate the task. Combined observations confirm our core thesis that neuromorphic foveation leads to the best trade-off between information quantity and quality, at least on a software level.

Furthermore, we studied the performance evolution of the dataset pre-processed with various methods (reduction, visual attention and neuromorphic foveation) when reduced using an increasing factor of stochastic structural downscaling. Interestingly, while all three data types show the same behaviour, the averaged foveated data outperforms the high-resolution data when the 60% threshold of decrease is crossed. This is explained by the fact that most events kept in the foveated dataset provide relevant information to the semantic segmentation model. In contrast, a significant part of the events in the original dataset is not as valuable.

In summary, we demonstrated the stakes of foveation applied to event data for semantic segmentation and classification. Our work shows that such a strategy does concurrently preserve the accuracy of event data processing and greatly reduces the amount of data needed for the task, at least with the current software implementation. Further work will implement the hardware neuromorphic foveation feedback process by concurrently using the foveated DVS presented in (Serrano-Gotarredona et al., 2022) and the neuromorphic chip SpiNNaker (S. B. Furber et al., 2013) to record foveated event data in an embedded fashion.

1.4.2 How can we exploit the relevance of the spiking neuron to process the specific chronology of event data ?

We identified multiple computer vision tasks answering this problem while falling within the spectrum of APROVIS3D, the results of which are detailed below.

Coastline detection We participated in implementing this application, i.e. the main use case to be addressed within the APROVIS3D project, thanks to three approaches.

The first one corresponds to ANN-to-SNN learning and is carried out by a dedicated APROVIS3D workgroup composed of NTUA, INT and UCA collaborators. NTUA records event data and RGB videos of the sea and the coastline ground, separated and mixed, using cameras embedded on a UAV. Once the data is acquired, UCA verifies the provided videos' quality and translates the data into a universal format. This modified dataset is then forwarded to INT: having implemented the Hierarchy Of event-based Time-Surfaces (HOTS) classifier (Lagorce, Orchard, Galluppi, Shi, & Benosman, 2016) in previous work, they update this architecture to differentiate between sea and ground patches efficiently. Once correctly adapted to this task, UCA will implement the HOTS network with the hyperparameters learned during training on the neuromorphic hardware SpiNN-3. The final topology will finally be embedded in the UAV set up by NTUA. At UCA, we handle the translation of data between different file formats (from `AEDAT2` to `NPY`) as well as the verification of the quality of the data, as the embedded DVS required multiple adjustment steps to provide a usable dataset.

The second approach corresponds to delay learning, i.e. an SNN learning rule which, instead of changing the synaptic weight, changes the delay of the electrical spike journey in the presynaptic neuron's axon (Hüning, Glünder, & Palm, 1998). We aim to implement a delay learning

mechanism to classify between spatiotemporal patterns in an event scene, mainly applied to our application task of coastline detection. Indeed, depending on whether the events recorded correspond to the ground or the sea, their movements are quite different: the events corresponding to the ground are generated by the sensor's ego-motion (Chapel & Bouwmans, 2020) while the movement of the events corresponding to the sea is the sum between the movement generated by the ego-motion (vertical motion) and by the waves (mostly a motion parallel to the coastline). To this end, we extended the bio-plausible unsupervised delay learning model introduced in (Nadafian & Ganjtabesh, 2020) to a greater number of convolution layers and noisy and noiseless synthetic data. Additionally, we applied various metrics (accuracy, recall, F1-score, GINI index) to assess the training phase and ease the future adaptation to real-world data.

3D reconstruction Another use case of the APROVIS3D project targets the reconstruction of the 3D architecture of a visual scene from two stereo event streams recorded by two DVS interfaced to a neuromorphic platform. Building on (Dikov, Firouzi, Röhrbein, Conradt, & Richter, 2017) model for neuromorphic 3D reconstruction, we embed the model on a SpiNN-3 by 1) downscaling the input data, 2) modifying the network's architecture, and 3) cropping the data. Once we experimentally validated our results on the toy datasets provided in (Dikov et al., 2017), we produced some first results on the real-world dataset DSEC (M. Gehrig, Aarents, Gehrig, & Scaramuzza, 2021).

Neuromorphic lip reading Finally, we distance ourselves from the APROVIS3D spectrum and propose an innovative application task that would strongly benefit from the SNN's intrinsic temporal processing and event data's crucial chronology: lip reading. In order to tackle this task, we recorded an event-based dataset of lip movement corresponding to various words. This dataset, only the second in state of the art, subsequently allowed us to develop a neuromorphic lip reading model classifying short sequences of events into word categories. This SNN architecture leverages the advantages of neuromorphic computing, making it suitable for real-time embedded scenarios. We experimentally validated this approach ; we demonstrated that this model, the first to our knowledge to apply an SNN to event-based data of lip movement, produces promising results while keeping a relatively small size. This task is specifically relevant to our scientific problem because *a contrario* to the others, it could not have been solved using a small set of frames (consisting of accumulated events) but only thanks to the fine temporal resolution of events.

1.4.2.1 Overview of publications

All in all, the different articles and extended abstracts we produced during this PhD answer the different problems described above according to the following repartition:

Subject:	**How can we optimise the trade-off between quantity and quality of the visual information as a preprocess for any embedded data processing ?**			**How can we exploit the relevance of the SNN to process the specific chronology of event data ?**
	Event reduction	Visual attention	Foveation	
Conference proceedings	VISAPP 2022 WACV 2023 CVPRw 2023 (human performance assessment)	CBMI 2021 AICAS 2022 AICAS 2023 IJCNN 2023 MVA 2023	/	CVPRw 2023 (lip reading)
Extended abstracts	/	ORASIS 2023	ORASIS 2021 NeuroVision 2022	JOBIM 2022 (delay learning)
Journals	/	/	Biological Cybernetics 2022	Brain Sciences 2023 (delay learning)

TABLE 1.1 – Overview of publications published during this PhD thesis.

1.5 Manuscript outline

This manuscript is organised as follows: Chapter 1 introduces the thesis by discussing its motivations, scientific context and various objectives. The state-of-the-art is distributed between the Chapters 2, 3, 4 and 5. SNNs and event cameras are introduced, and their history, mechanism and applications are described in detail respectively in Chapter 2 and 3. The benefits of the combined use of embedded SNNs and event cameras are amply demonstrated and documented in Chapter 4. Finally, Chapter 5 develops the various concepts from neuroscience, which potential or achieved application to computer science brings a significant performance boost. Chapters 6, 7, 8 and 9 constitute the contributions part. Our first issue concerning the trade-off between quantity and quality is addressed in 6, 7 and 8, which respectively present the design, implementation and achieved results of event data reduction, salient detection of OoIs by event density and neuromorphic foveation. Chapter 9 tackles the issue of drawing benefits from SNNs temporality to handle the chronology of event data by describing the various embedded applications and achieved results. Finally, Chapter 10 concludes this work and summarises the results achieved during this PhD thesis.

Notations

Scientific acronyms

AI	artificial intelligence
ML	machine learning
SNN	spiking neural network
ANN	artificial neural network
DNN	deep neural network
CNN	convolutional neural network
IF	integrate-and-fire
LIF	leaky integrate-and-fire
FEF	frontal eye field
WTA	winner-takes-all
RoI	region of interest
OoI	object of interest
DNF	dynamic neural field
UAV	unmanned aerial vehicle
RGB	red green blue (images)
DVS	dynamic vision sensor
ATIS	asynchronous time-based image sensor
DAVIS	dynamic and active-pixel vision sensor
CPU	central processing unit
GPU	graphics processing unit
ASIC	application-specific integrated circuit
FPGA	field programmable gate array
SNE	sparse neural engine
USB	universal serial bus
PLIF	parametric leaky integrate-and-fire
SLAYER	spike layer error reassignment
HOTS	hierarchy of event-based time-surfaces
RMSE	root-mean-square error
MIoU	mean intersection over union
FLOPS	floating-point operations per second

APROVIS3D collaborators

APROVIS3D	Analog PROcessing of bioinspired Vision Sensors for 3D reconstruction
UCA, **i3S**, **SPARKS**	Université Côte d'Azur, Laboratoire d'Informatique, Signaux et Systèmes de Sophia Antipolis, Scalable and Pervasive softwARe and Knowledge System
INT	Institut de Neurosciences de la Timone
IMSE	Instituto de Microelectrónica de Sevilla
NTUA	National Technical University of Athens
ETHZ	Eidgenössische Technische Hochschule Zürich
UNIWA	University of West Attica

State of the art

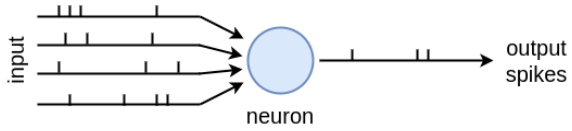
CHAPTER 2

Spiking neural networks

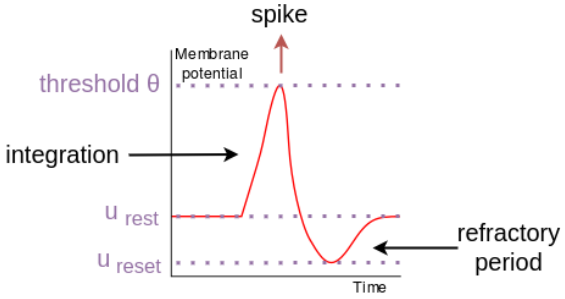
This chapter introduces the concept of spiking neural networks, a type of artificial neural network closer to biology than traditional artificial networks. Indeed, spiking neural networks receive and process information in the form of spike trains, meaning as a non-monotonous sequence of activations. Therefore, they make for a suitable candidate for efficiently processing incoming event patterns measured by silicon retinas.

2.1 Neuronal dynamics 21
2.1.1 Behaviour of a biological neuron 21
2.1.1.1 Anatomy of the human nervous system 21
2.1.1.2 Chemoelectric mechanism of the biological neuron 22
2.1.2 Behaviour of a spiking neuron 23
2.1.2.1 Hodgkin-Huxley model 24
2.1.2.2 Integrate-and-Fire models 24
2.1.2.3 Leaky Integrate-and-Fire model 24
2.1.2.4 Izhikevich model 25
2.1.2.5 Spike Response Model 25
2.1.3 Parameters 25
2.1.3.1 Neuronal parameters 25
2.1.3.2 Synaptic parameters 25
2.1.4 Network architecture 26
2.1.4.1 Information encoding 26
2.1.4.2 Dynamic adaptation rules 27
2.1.4.3 Learning rules 28
2.2 Neuromorphic hardware 30
2.2.1 Simulation on CPU/GPU 30
2.2.2 Human Brain Project's SpiNNaker 34
2.2.3 Intel's Loihi 34
2.2.4 Center for Project-Based Learning's Kraken 34
2.3 Applications 35

SNNs are a new type of ANN closer to biology than traditional artificial networks, also known as the third generation of neural network model (Maass, 1997). In accordance with this high bio-inspiration, they seek to mimic the dynamics of neural membrane and action potentials over time (Paugam-Moisy & Bohte, 2012). Additionally, whereas information was precedently encoded and processed as multi-bit numbers, SNN encodes the signal using sequences of spikes (or activations) with varying weights, delays and frequencies. SNNs receive and process information in the form of spike trains, meaning as a non-monotonous sequence of activations, as represented in Fig. 2.1a. Therefore, they make for a suitable candidate for efficient processing and classifying of incoming event patterns measured by silicon retinas.



(a) A spiking neuron receives spike trains as input and processes this information to produce a new sequence of activations.



(b) Evolution of the neuron's membrane potential over time when activated by input spikes.

Figure 2.1 – Behaviour of a spiking neuron.

2.1 Neuronal dynamics

2.1.1 Behaviour of a biological neuron

2.1.1.1 Anatomy of the human nervous system

In biology, the term "neuron" designates one type of the cells forming the nervous system. An adult human brain comprises roughly 85 billion neurons, each $0.01 - 0.05mm$ in diameter and dedicated to "sensing the environment, communicating the perceived changes and commanding the body's response to these sensations" (Bear, Connors, & Paradiso, 2007). As seen in Fig. 2.2, each neuron is composed of the *soma* (i.e. the body of the neural cell, around $20\mu m$ in diameter and also called *perikarion*) and two types of neurites, the axon and the dendrites.

The axon transfers information over long distances in the nervous system, ranging from $1mm$ to over a meter long and with a diameter varying between 1 and $25\mu m$ in the human brain. We'll develop on the impact of both those factors on the speed of information transmission in Section 5.3.1. At most $2mm$ long and forming a "dendritic tree", dendrites extend from the *soma*

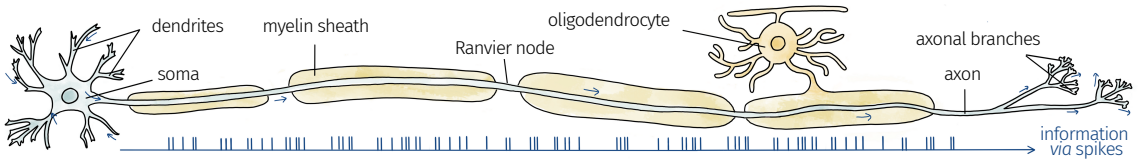


Figure 2.2 – Biological neuron.

and act as the "antennae" by connecting with the axons of other neurons to receive their incoming signals. The connection point between a first neuron's axon and a second neuron's dendrite (or, in certain occasions, *soma*) is called a synapse, derived from the Greek "to fasten together". Each dendrite of each neuron is covered with thousands of synapses.

In order to transit from the presynaptic to the postsynaptic neuron, the information is transmitted from the presynaptic neuron's *soma* to the synapse as a train of electrical impulses, i.e. "action potentials", that triggers the delivery of neurotransmitters from the pre- to the postsynaptic neuron which finally translates the received information back into an electro-chemical format. These are mediated by neurotransmitters, which are chemical substances endogenous to the nervous systems and responsible for the transmission of various information ; the human body contains more than 40 different types of neurotransmitters, including acetylcholine, norepinephrine, dopamine, gamma-aminobutyric acid and serotonin (Vasković, 2023).

Scientists have developed various classification mechanisms to distinguish between different types of neurons according to their number of neurites, the shape of their dendritic tree, the length of their axon, the organs with which they connect (either sensory surfaces of the body, muscles or other neurons) or the differences in gene expression.

Neurons are not the only cells present in the nervous system ; another broad category is the glial cells, whose representatives are just as numerous as neurons. Also known as the "sleeping giants of neuroscience", glia "contribute to brain function mainly by insulating, supporting and nourishing neighbouring neurons" (Bear et al., 2007). Without it, the brain would be unable to perform the vast array of tasks it is responsible for.

Many different types of glial cells exist, of which the most represented are the astrocytes. Its role is not yet entirely clear, but it is believed to regulate the neurites' growth by controlling the chemical content of the extracellular space. Oligodendrocytes and Schwann cells compose the myelinating glia, i.e. responsible for the concentric wrapping of myelin, a thin protein sheet interspersed with lipid layers, around the axon. Scientific experiments have proven its essential role in information transmission and facilitation of both the neural circuit function and the behavioural performance (see Section 5.3.1 for more details). Other, less represented glial cells also exist in the nervous system: we can cite the ependymal cells (cell migration the organism development) and microglia (phagocytosis).

2.1.1.2 Chemoelectric mechanism of the biological neuron

As stated above, the neuron transmits information along its axon as a train of "action potentials", electric impulses encoding data in their frequency, pattern and timing.

This mechanism is made possible by the chemical balance between the intracellular and extracellular environment surrounding the neurons. There is a high concentration of potassium within

the neuron, alimented by a high concentration of sodium and calcium in the extracellular environment enabling the activity of the membrane sodium-potassium pumps. This disequilibrium leads chemically to the creation of an outward potassium movement thanks to membrane potassium channels, leaving the inside of the neuron negatively charged.

Due to this difference in electrical charge, the neuronal membrane has a resting potential of $-65mV$. When the neuron is activated, its extracellular environment becomes positively charged relative to the extracellular one during $1ms$ at most: this corresponds to the depolarisation or integration. The action potential, also called "nerve impulse", "discharge", or "spike", takes place only if the membrane potential reaches a specific value called "threshold" during the depolarisation phase. In this case, the neuron enters a phase of hyperpolarisation where it undershoots below the resting value and cannot be activated again before increasing back to the resting value. Otherwise, the membrane potential will slowly decrease back to its resting value. This behaviour is presented in Fig. 2.1b.

2.1.2 Behaviour of a spiking neuron

As seen in Fig. 2.1b, spikes are emitted by each neuron depending on its membrane potential, which increases according to the input activation (much like the biological membrane potential increases when the biological neuron is electrically activated) and decreases over time in the absence of stimulation (thus simulating the leak observed in the biological neurons) (Paugam-Moisy & Bohte, 2012). Once the membrane potential u crosses the threshold θ with a positive slope, a spike is produced, and the membrane potential is reset at u_{reset}. This is coherent with a biological neuron's behaviour when an action potential: those two steps correspond respectively to the neuron's depolarisation (or overshoot) and hyperpolarisation (or undershoot) as described in Section 2.1.1.2.

By definition, a spiking neuron follows a model based on parameters describing its internal state and its reaction to the input current with more or less faithful biological realism. A complete review of those models is presented in (E. M. Izhikevich, 2004) (see Fig. 2.3), but we describe the most common ones below.

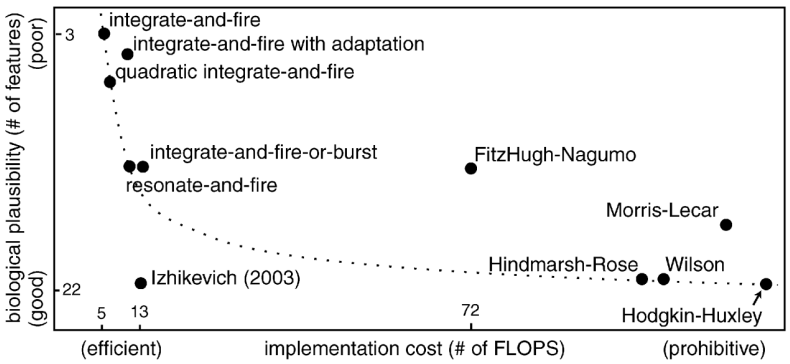


Figure 2.3 – Comparison of the neurocomputational properties of spiking models, from (E. M. Izhikevich, 2004). The x-axis label "# of FLOPS" is an approximate number of floating point operations (addition, multiplication, etc.) needed to simulate the model during a 1 ms period.

2.1.2.1 Hodgkin-Huxley model

Historically the first spiking neuron model, the Hodgkin-Huxley model was developed in 1952 (Hodgkin & Huxley, 1952) and describes the neuronal dynamic by its variations of ionic concentrations in the intra and extracellular environments. It is part of the so-called "conductance-based" models and has been “successfully compared to numerous data from biological experiments on the giant axon of the squid” (Paugam-Moisy & Bohte, 2012). Its high degree of realism makes it far too complex to be of any real use in SNN simulation.

2.1.2.2 Integrate-and-Fire models

Drawing inspiration from Hodgkin-Huxley, the Integrate-and-Fire (IF) model places itself at a less precise level of detail to be more computationally efficient (Lapicque, 1907). It integrates the spikes and increases the membrane potential u only at the time of the input, according to Eq. 2.1:

$$C\frac{d}{dt}U = -\frac{u(t) - u_{\text{rest}}}{R} + I(t) \tag{2.1}$$

The spike firing time t_f is defined in the following threshold crossing equation:

$$\begin{cases} u(t_f) = \theta \\ u'(t_f) > 0 \end{cases} \quad \Rightarrow \quad u(t_f) = u_{\text{reset}} \tag{2.2}$$

The main difference with Hodgkin-Huxley is the simplification of the action potential processing. While the latter considers the shape of the action potentials, the IF (and LIF described below) consider each spike as a uniform event defined only by its timing.

2.1.2.3 Leaky Integrate-and-Fire model

The Leaky Integrate-and-Fire (LIF) model varies from its inspiration IF by adding a leaky behaviour: if incoming spikes activate the neuron, the membrane potential increases. If it does not reach the threshold, it decreases slowly back to its resting value when the input stops (leak). If the membrane potential overcomes a threshold, an output spike is produced, and the membrane potential is reset.

The dynamics of the LIF neuron’s membrane potential u are described by the equations 2.3:

$$\tau_m \frac{du}{dt} = u_{\text{rest}} - u(t) + RI(t) \tag{2.3}$$

where τ_m is the membrane’s time constant and I the input current modulated by a resistance R. Without any input current, the membrane potential is at rest and is of value u_{rest}. When activated, it increases according to the input current. Moreover, at each timestep a slow decrease towards u_{rest} is driven by the time constant τ_m, thus modeling the voltage leakage. The firing time t_f is defined by the threshold crossing Equation 2.2.

While implementing SNN simulations, the IF and LIF models are often preferred because of their simplicity, conferring an advantage on the issue of efficiency for computational resources.

2.1.2.4 Izhikevich model

The Izhikevich model makes a good compromise between biophysical plausibility and computational cost: according to the authors, "using this model, one can simulate tens of thousands of spiking cortical neurons in real time ($1ms$ resolution) using a desktop PC"(E. M. Izhikevich, 2004).

2.1.2.5 Spike Response Model

Defined by Gerstner in 1993, the Spike Response Model is a phenomenological model of the neuron based on the occurrence of spike emissions, which makes it more intuitive and more straightforward to implement (Gerstner, Ritz, & van Hemmen, 1993 ; Paugam-Moisy & Bohte, 2012). It generalises the IF model and its implementation abandons coupled differential equations (as used by IF, LIF and Izhikevich) in favour of the use of kernels.

2.1.3 Parameters

Overall, the computational implementation of a spiking neuron requires setting different parameters influencing the network performance. The list of parameters might vary between different models ; below are described most of the LIF parameters and their impact on neuron behaviour.

2.1.3.1 Neuronal parameters

This section brings together the parameters influencing the neuron directly.

Membrane potential Three parameters, expressed in mV, define the membrane potential:
- the resting membrane potential u_{rest}, corresponding to the value at rest (around $65mV$ in the biological neuron) ;
- the reset potential u_{reset}, corresponding to the value reached by the membrane potential during the hyperpolarisation phase (see Fig. 2.1b) ;
- the threshold θ, setting the potential limit before a spike is emitted.

Membrane time constant The membrane time constant τ_m corresponds to the "amount of time (in ms) it takes for the change in potential to reach 63% (i.e. $1 - \exp^{-1}\%$) of its final value" (Byrne, 2020). In other words, the higher the value, the longer it will take to finish its depolarisation phase, and the later the spike will be produced from the incoming excitation, and *vice versa*.

Refractory period Expressed in ms, the refractory period τ_{refrac} is the time needed by the membrane after the hyperpolarisation phase to increase from the reset potential back to the resting potential, i.e. the amount of time during the neuron cannot be excited after it produced a spike.

2.1.3.2 Synaptic parameters

This section covers the parameters influencing the synapse.

Weight The weight ω defines the importance of the influence of a synapse activity on the post-synaptic neuron. In traditional ANN, the input data is multiplied by it within the hidden layer. In SNN, it increases the activation or inhibition of the post-synaptic neuron following the activation of the presynaptic neuron. It can be assimilated to the post-synaptic transmitters released in the biological synapse: the higher the value, the more important the effect of the information transmitted.

Delay The delay δ corresponds to the amount of time it takes for the information to leave the presynaptic neuron's *soma* and reach the post-synaptic *soma*. Biologically speaking, this parameter corresponds to the combination of two elements: the dendritic delay and the axonal delay. In addition, it can be learned using a variety of biological mechanisms and is at the origin of many neuronal behaviours, all of which are described in more detail in Section 5.3.1. While many may consider that this parameter concerns the neuron, we decided to include it in the "synaptic parameters" as it influences the synaptic connection between two neurons, and most simulation libraries implement it at the synapse level.

Excitatory and inhibitory decay time The decay time simulates synaptic fatigue, a biological phenomenon taking place when the synapse is too highly activated as a way to regulate the nervous system activity physiologically. It results in the temporary inhibition of neuron activity and influences either excitatory or inhibitory synapses, depending on its type.

2.1.4 Network architecture

An SNN is constructed using populations of neurons linked together with connections with respect to certain rules and specific architecture. A population or layer reunites neurons in a unit that often shares the function and the initial parameters. The neurons can be arranged either linearly or in 2D or 3D space (depending on the simulator used), allowing for the implementation of distance-dependant rules.

The connections can be either excitatory, meaning that they activate the post-synaptic neuron; or inhibitory, indicating that the post-synaptic neuron is inhibited. The two most common types of connective patterns are One-To-One, meaning that each neuron of the presynaptic population is connected to only one neuron of the postsynaptic population at the same index in the layer; and All-To-All, meaning that each presynaptic neuron is connected to all the postsynaptic neurons.

2.1.4.1 Information encoding

By definition, SNNs use time as a basis for information coding. Indeed it has been experimentally demonstrated that biological neurons transmit information with impressive efficiency: for example, a human being requires only $100-150ms$ to process complex visual stimuli (S. J. Thorpe & Fabre-Thorpe, 2001). Scientists used to theorise that the information was encoded using a Poisson-like spike rate; however, the visual system is composed of multiple layers firing at an average rate of $10ms$, too slowly to allow for rate coding. Instead, it seems more likely that the information is encoded using the precise timing of the spikes.

Thorpe defined in 2001 various bio-plausible encoding mechanisms for SNN relying on time (S. Thorpe, Delorme, & Van Rullen, 2001). Among them can be found the latency coding, where the information is transmitted in the time it took for the spike to reach the postsynaptic

neuron after a stimulus ; and the rank order coding, where the information is encoded in the order in which the spikes reached the postsynaptic layer. A visual example highlighting the importance of the choice of information encoding is presented in Fig. 2.4.

Furber later introduced an additional temporal encoding mechanism: N of M coding counts the input spikes using a feedback inhibition circuit and blocks their spiking once a certain number has been reached, thus implementing a k-WTA and allowing for the detection of specific combinations of spikes (S. B. Furber, John Bainbridge, Mike Cumpstey, & Temple, 2004). The implications of these different strategies on neuromorphic circuits are discussed by (Panzeri, Janotte, Pequeño-Zurro, Bonato, & Bartolozzi, 2023).

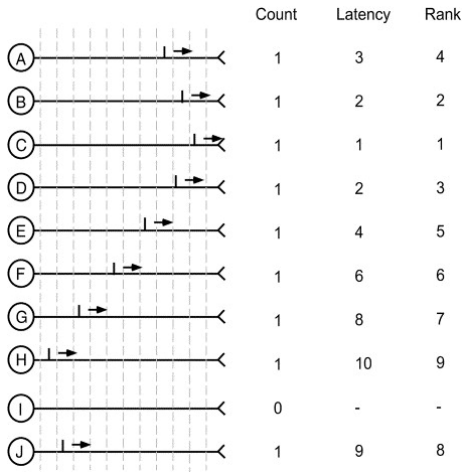


Figure 2.4 – Comparison between three SNN coding schemes that can operate in a short time window, from (S. Thorpe et al., 2001).

2.1.4.2 Dynamic adaptation rules

An SNN can be applied to various tasks (some examples can be found below in Section 2.3) that can be achieved with and without learning. In the second case, it relies on intrinsic dynamic behaviours and specific network architectures.

A first adaption mechanism worth mentioning is the WTA, observed in the cortical architecture, is a first adaption mechanism worth mentioning. This mechanism allows for the selective activation of a neuron, enabled by a pattern of lateral inhibition retroactively regulating the layer. It is implemented using an All-To-All inhibitory connection, without autapses (self-connections) and where the input and output layers are the same. Each neuron activation leads to the inhibition of the others to prevent too many efferent neurons from receiving too many spikes from a particular afferent region, thus ensuring that different neurons code different inputs (Rolls & Milward, 2000). In the brain, lateral inhibition leads to WTA situations. Yet, other forms exist, such as inter-group WTA competition (instead of intra-group), leading to a cohort of winner-neurons rather than a single one (Delorme, Perrinet, Thorpe, & Samuelides, 2001). Despite WTA's core purpose being enhanced selectivity, it also increases feature sparsity (Falez, Tirilly, Bilasco, Devienne, & Boulet, 2019) and maintains synaptic weights within bounds (Masquelier & Thorpe, 2007). An example of such a mechanism is provided in the model described in Section 7.2.2 and Fig. 7.2.

Homeostasis makes a second interesting adaptation mechanism. Globally speaking, we call "homeostasis" any mechanism that allows a system to regulate a key factor to maintain its beneficial value, guaranteeing flexibility. Applied to SNN, this mechanism is "crucial in providing a more efficient solution to the emergence of independent components" (Perrinet, 2010). It corresponds to the dynamic adaptation of neuronal thresholds often applied to reducing the high spike frequency induced by complex patterns and networks, as observed in the nervous system (W. Zhang & Linden, 2003 ; Perrinet, Samuelides, & Thorpe, 2004).

2.1.4.3 Learning rules

In biology, learning is "achieved *via* local plasticity mechanisms that operate at various spatial and temporal scales" (Christensen et al., 2022). SNN can learn patterns and specialise for specific tasks by updating their parameters at each timestep until a specific aim is achieved. The modified parameter is more often than not the weight ; however, other parameters can also be the subject of learning, such as the synaptic delay (see Section 5.3), the membrane time constant (Fang, Yu, Chen, Masquelier, et al., 2021 ; Rathi & Roy, 2021), the neuronal threshold (Rathi & Roy, 2021 ; J. Zhang et al., 2022), or even the membrane potential which distribution is "rectified" into a Gaussian (Y.-Z. Guo et al., 2022 ; S. Wang, Cheng, & Lim, 2023).

The strategies adopted to perform learning with SNN can be differentiated into various groups: conversion-based versus spike-based (Roy et al., 2019), online versus offline (J. Wang, Belatreche, Maguire, & McGinnity, 2010), on-chip versus off-chip (Christensen et al., 2022), supervised versus unsupervised (Paugam-Moisy & Bohte, 2012).

Conversion-based versus spike-based. (Roy et al., 2019) proposes to segregate the SNN learning mechanisms between conversion-based and spike-based approaches. The first one corresponds to the conversion of a set of parameters trained by an ANN into parameters understandable by an SNN with a similar architecture ; this approach produces the best results on large-scale SNN (for example, in the context of ImageNet classification (Sengupta, Ye, Wang, Liu, & Roy, 2018)). However, it is to be noted that this method leads to a significantly large inference time, so an increased latency and decreased energy efficiency.

The second spike-based approach is more biologically plausible because of its exploitation of the intrinsic temporal aspect of SNNs. Two research directions have been adopted: unsupervised (trained without labelled data) and supervised (trained with labelled data) learning. The first one uses local STDP-based learning rules, where the STDP is a bio-inspired unsupervised rule reinforcing the synaptic connections between neurons activating closely in time. This mechanism is heavily inspired by Hebb's rule: "cells that fire together wire together" (Hebb, 1949). It allows for the selective selection of repeating patterns of input spikes (Delorme et al., 2001 ; Perrinet & Samuelides, 2002 ; Masquelier & Thorpe, 2007) and is suitable for energy-efficient on-chip implementation (see below). However, it is difficult to adapt to large-scale SNN due to the "vanishing forward-spike propagation", i.e. the decreased firing rate in deeper layers, and current research are trying to solve this issue using feedback-based learning methods (Roy et al., 2019).

Initiated with the early ReSuMe (Ponulak & Kasiński, 2009) and tempotron (Gütig & Sompolinsky, 2005), the research on the supervised spike-based approach has recently directed its attention to the implementation of a "global backpropagation-like spike-based error gradient descent to enable supervised learning in multi-layer SNNs" (Roy et al., 2019). Recent works in this vein include SpikeProp (Bohté, Kok, & Poutré, 2000) and variants, which implement a differentiable

approximate function for SNN using a specific output spike train as the target; others perform stochastic gradient-descent on membrane potentials (Lee, Delbrück, & Pfeiffer, 2016; Mostafa, 2016). Even though promising, these supervised methods have not been able yet to overcome conversion-based approaches.

Offline versus online learning In (J. Wang et al., 2010), the authors underline the segregation of SNN learning approaches between offline and online learning and the lack of representation of the latter despite its suitability for large datasets and non-stationary tasks. The offline learning method updates the trained parameters only once the whole dataset has been presented; new data cannot be added without retraining the whole network. It is thus quite fast but subject to local minima issues. Among those methods can be found both supervised — SpikeProp (Bohté et al., 2000), Synfire Chains (E. Izhikevich, 2006) — or unsupervised — STDP (Masquelier & Thorpe, 2007) and Hebbian learning mechanisms.

However, online learning updates the network parameters at each new training sample to minimise the error per sample. It is adapted to continuous environmental changes due to its "always-on" approach and is thus significantly more bio-coherent. Few examples of such a mechanism exist: we can cite the four-layer hierarchical model for face recognition introduced in (Wysoski, Benuskova, & Kasabov, 2006) and the gustatory model for taste recognition based on an evolving learning algorithm in (Soltic, Wysoski, & Kasabov, 2008), which both lack adaptive fine-tuning of the learning parameters as well as suffer from an excessive training speed. It has been more recently applied to robotics with the self-adaptation model for optimal navigation in dynamic environments implemented in (Alnajjar, Zin, & Murase, 2008), and to event-based object recognition in (Grimaldi, Boutin, Ieng, Benosman, & Perrinet, 2023). Online learning is specifically relevant in the current research field since it overcomes the issue of generalisability, i.e. the mismatch between the training dataset and the real world, by learning continuously and correcting errors as they unfold (Christensen et al., 2022).

Off-chip versus on-chip learning Finally, the training phase can either occur off or on a synaptic circuit (Christensen et al., 2022). In the first case, the SNN is trained on a separate computer with significant memory and computational power, then deployed on neuromorphic hardware (see Section 2.2. This approach generally achieves the best inference accuracy on practical tasks since it performs well on large available datasets and often makes use of convolutional architectures (Esser et al., 2016).

On-chip learning consists primarily of learning dynamics taking inspiration from computational neuroscience, such as the STDP variants or, more recently, the 3F rules (Ercsey-Ravasz et al., 2013), and allow for online learning. However, this approach requires a large amount of memory and routing resources, thus limiting the size of the architectures; promising hardware alternatives are currently being developed to allow for the storage and computation to take place at the same place (Christensen et al., 2022). Overall, this approach helps overcome the challenge posed by the high power requirement of state-of-the-art AI technologies (Christensen et al., 2022).

A third option can be found in hybrid learning, also known as "chip-in-the-loop", combining both approaches while allowing for online learning.

A detailed state-of-the-art of recent SNN models with their corresponding learning mechanism can be found in Tables 2 and 3 from (Yamazaki, Vo-Ho, Bulsara, & Le, 2022). It is interesting to

note that the increasing size of SNN induces many challenges, amplified compared to those faced with smaller-scale systems: controlling the neuronal activity with homeostasis, compensating local failures with adaptation mechanisms, the application of unsupervised learning mechanisms and online learning, and synaptogenesis and neurogenesis (Christensen et al., 2022).

2.2 Neuromorphic hardware

As of today, SNN simulations on traditional von Neuman computers require a substantial simulation time ; indeed, “their simulation requires the sequential computation of several time steps, and simulation time thus scales with the deepness of the network as well as the time resolution of the simulation” (Bouvier et al., 2020).

However, researchers have recently designed hardware specifically adapted to the fast and low power simulations of SNNs, using electronic circuits that faithfully reproduce the dynamics of neurons in real time (Sorbaro et al., 2019). This novel technology is called "neuromorphic hardware" and can be divided into three different designs (Javanshir, Nguyen, Mahmud, & Kouzani, 2022):

— analog designs model neurons using physical processes leading to a very efficient implementation of operations with low power consumption thanks to the natural dynamics of the system ;
— digital designs model neurons with bits, allowing for a “controllable and guaranteed precision of variables" (Javanshir et al., 2022) but with a significant impact on memory requirements and energy consumption ;
— mixed designs take the best of both analog and digital, for example implementing the spike communication digitally while modeling neurons with analog elements.

Those designs are often implemented on Application-Specific Integrated Circuits (ASICs) or Field Programmable Gate Array (FPGA) devices: while most known neuromorphic boards are developed on ASICs (such as the digital SpiNNaker (S. B. Furber et al., 2013), TrueNorth (Akopyan et al., 2015) and Loihi (Davies et al., 2018), or mixed BrainScaleS (Schemmel et al., 2010) and Neurogrid (Benjamin et al., 2014)), FPGAs are gaining momentum as “a promising candidate for acceleration of SNN, achieving better speed-up than CPUs and less energy consumption than graphics processing units (GPUs)" (Javanshir et al., 2022). We can mention the recent applications of SNNs on FPGAs to edge-detection (Qi, Zhang, Taha, Chen, & Hasan, 2014), bio-mimetic pattern generation (Ambroise, Levi, Joucla, Yvert, & Saïghi, 2013) and event-driven vision processing (Yousefzadeh, Serrano-Gotarredona, & Linares-Barranco, 2015).

We present in this section first a short review of CPU and GPU simulators, then a selection of neuromorphic hardware dedicated to SNN simulation. A more comprehensive review of existing neuromorphic hardware and their particularities can be found in (Bouvier et al., 2020), (Roy et al., 2019) and (J. Wang, 2022).

2.2.1 Simulation on CPU/GPU

SNN simulators can be divided into two approaches, depending on their synchronicity (Javanshir et al., 2022). The “event-driven” simulators are asynchronous, i.e. it only updates the neuron when there is an incoming spike. They allow for a high operation speed and good bio-plausibility at the cost of an increased implementation complexity, thus are more suitable for

sparsely activated networks. The "clock-driven" simulators however update the neuron synchronously "at every tick of a clock" (Javanshir et al., 2022) thus allowing for parallel computing and an easier implementation on CPUs (small networks) and GPUS (greater networks with higher mathematical complexity).

We adapt and complete in Table 2.1 below the list of available software frameworks for SNNs simulation from (Yamazaki et al., 2022), with the additional information provided in (Pehle & Pedersen, 2021).

Table 2.1: Overview of existing SNN simulators on CPU.

Name	Description	Simulation type	GPU	Backpropagation
Auryn (Zenke & Gerstner, 2014)	C++ simulator for recurrent SNN with synaptic plasticity	NA	No	No
BindsNET (Hazan et al., 2018)	Python package geared towards the development of biologically inspired algorithms for ML on CPU or GPU, using PyTorch functionality	Clock-driven	Yes	No
Brian2 (Romain & Dan, 2008)	free, open-source Python simulator, advertised as "widely-used, easy to use and reliable"	Clock-driven	Yes	No
CARLsim (Nageswaran et al., 2009)	C and C++ library for simulating large-scale SNN on multiple CPU and GPU cores	Clock-driven	Yes	
cuSNN (Paredes-Valles et al., 2020)	C++ GPU-accelerated simulator for large-scale networks including STDP learning rules	Clock-driven	Yes	No
GeNN (Yavuz et al., 2016)	GPU-accelerated SNN simulation environment based on NVIDIA CUDA technology	NA	Yes	No
NeMo (Fidjeland et al., 2009)	High-performance SNN simulator of networks of Izhikevich neurons on CUDA-enabled GPUs	Clock-driven	Yes	No
Nengo (Bekolay et al., 2014)	Python package for building, testing and deploying neural networks, with a useful user interface and an extension dedicated to deep learning	Clock-driven	Yes	No
NEST (Gewaltig & Diesmann, 2007)	Python simulator of dynamics, size and structure of neural systems, often used in medical and biological applications but with poor performance on large datasets and deep learning	Event- and clock-driven	Yes	No
NEURON (Hines & Carnevale, 2001)	Python simulation environment for building and using SNN on a local machine, in the cloud or on an HPC and providing a graphical interface	Event-driven	Yes	No
Norse (Pehle & Pedersen, 2021)	PyTorch extension to implement deep-learning compatible SNN components	Clock-driven	Yes	Yes
PyGeNN (Knight et al., 2021)	Python library for GeNN simulator	NA		
PyNN (Davison et al., 2009)	Python interface allowing to define and simulate SNN on different backends, including software simulators (NEST, NEURON) and neuromorphic hardware (SpiNNaker, BrainScaleS)	Event- and clock-driven	No	No
PySNN (BasBuller, 2019)	Python SNN framework building on PyTorch to implement efficient simulation both on CPU and GPU, inspired from BindsNET and cuSNN	Clock-driven	Yes	No
Rockpool (SynSense, 2019a)	Python package developed by SynSense for training, simulating and deploying SNN on several simulation backends (such as Brian2, Torch, JAX, Numba or Numpy)	Clock-driven	Yes	Yes
Sinabs (SynSense, 2019b)	Python library for development and implementation of spiking CNNs, developed by SynSense	Clock-driven	Yes	Yes

Continued on next page

Table 2.1: Overview of existing SNN simulators on CPU. (Continued)

SLAYER (Shrestha & Orchard, 2018)	C++ backpropagation framework for SNN, also known as Spike LAYer Error Reassignment	Clock-driven	Yes	Yes
SLAYER PyTorch (Shrestha & Orchard, 2018)	PyTorch port of the original SLAYER framework, to be used in Python and allowing for the implementation of SNN on Loihi	Clock-driven	Yes	Yes
SNN toolbox (Rueckauer et al., 2017)	Python package for automating the conversion of pre-trained network analog to SNN	Clock-driven	Yes	No
snnTorch (Eshraghian et al., 2021)	Python simulator extending PyTorch for deep learning and SNN	Clock-driven	Yes	Yes
SpikingJelly (Fang, Yu, Chen, Masquelier, et al., 2021)	Python SNN simulator extending PyTorch, using stateful neurons	Clock-driven	Yes	Yes

2.2.2 Human Brain Project's SpiNNaker

SpiNNaker is a neuromorphic hardware platform developed at the University of Manchester under the Flagship Human Brain Project. Each SpiNNaker chip communicates and process spike events: it contains a router that delivers each incoming spike to the corresponding neural processor where the spike should arrive. The SpiNNaker computation is based on the availability of 18 parallel ARM processor cores per chip. Each ARM core is tightly coupled to local memory (TCM), where the neuron parameters and states are stored. Additionally, each chip is packaged with an external memory communicated through a DMA (direct memory access) controller allowing direct data transference between the internal and external memory. This memory stores the synaptic connectivity between the different neural populations allocated in the chip.

Each SpiNNaker core can simulate up to 256 spiking neurons – the initial aim was to simulate 1,000 neurons, but it proved to be a goal too optimistic. A SpiNNaker chip can thus allocate about 4,500 neurons (S. Furber & Bogdan, 2020). Additionally, each neuron can receive 1,000 incoming synapses at most, hence a SpiNNaker chip can simulate up to 4,500,000 synapses.

Two versions of SpiNNaker boards are available. The first SpiNN-3 board contains 4 SpiNNaker chips, i.e. 72 cores. However, only 63 cores are available out of this total of 72, as the remaining are used for OS, monitoring, data I/O, etc. A SpiNN-3 board can thus implement SNN with a maximum of $18K$ neurons. Currently available SpiNN-5 boards contain 48 chips, having the capacity for $195K$ neurons.

SpiNNaker allows the description of the SNN system to be allocated in the hardware using sPyNNaker, a library developed for SpiNNaker in the PyNN language (University of Manchester, 2021). The network and corresponding connectivity described using the PyNN SpiNNaker library are mapped to the corresponding hardware. A software tool partitions and maps the defined network and connectivity into the physical cores and generates the corresponding routing and synaptic tables.

2.2.3 Intel's Loihi

Intel Loihi is a recent neuromorphic research chip (Davies et al., 2021) that can be used as a processing platform. In particular, the Kapoho Bay platform comes in a universal serial bus (USB) form factor with 2 Loihi chips with a total of 256 neuro-cores able to simulate 262.144 neurons and up to 260Mn synapses. It can be easily interfaced for live communication with a host system by programming the three embedded x86 processors used for monitoring and I/O spike management embedded on the Loihi chip.

2.2.4 Center for Project-Based Learning's Kraken

The Kraken chip (Di Mauro et al., 2022) comprises three main subsystems composing a heterogeneous architecture. The first subsystem, called the Fabric Controller, is built around a 32-bit RISC-V core that acts as the central control unit for the entire System-on-Chip. The Fabric Controller contains the main interconnection busses to the main L2 memory and the Advanced Peripheral Bus, which controls all the System-on-Chip peripherals. Additionally, the Fabric Controller hosts a complete set of on-chip peripherals and a power management unit.

The second subsystem is a RISC-V-based general-purpose accelerator known as the "Cluster", which hosts eight RISC-V cores enhanced with specialised Instruction Set Architecture exten-

sions. The cluster also features a 128 KiB L1 Tightly Coupled Data Memory and a dedicated hardware block for fast event management, parallel thread dispatching, and synchronisation.

The third subsystem, the External Hardware Processing Engine, hosts two accelerators, including the Sparse Neural Engine neuromorphic accelerator and CUTIE, a ternary weight neural network accelerator. The accelerators operate on two independent clock domains and are programmed via memory-mapped register interfaces.

The chip features a vast set of IO peripherals that can generate interrupts depending on data transmission events. An autonomous IO subsystem called the *µDMA* hosts all the IO peripherals. It can be programmed to orchestrate data transfers from the peripherals to the L2 memory and vice versa. Kraken also has a power management scheme that allows the on/off switching of unused system parts to control the overall energy consumption.

The Kraken chip is mounted on an evaluation board incorporating external components essential to execute complete applications. A USB-C connector is used to power the board and for JTAG and UART data transfer. The three power domains of Kraken are supplied by buck converters that are individually runtime-configurable, permitting application-controlled DVFS. External memory includes a combined HyperFlash/HyperRAM chip and a quad-SPI flash memory chip that can store application code for standalone booting. Additional connectivity is provided by Arduino headers, and a Camera Parallel Interface ribbon connector and level shifters are present between each off-chip connector and Kraken's I/O pins.

2.3 Applications

SNN can be applied to multiple application tasks, ranging from computer vision to gustative and tactile classification models. In 2012, (Paugam-Moisy & Bohte, 2012) outlined a first review of existing applications of SNN to pattern recognition ; we can mention the tasks of speech processing, computer vision, robotics, motor control, and rehabilitation in medicine. In 2022, (Yamazaki et al., 2022) published an SNN review where they summarise recent applications of SNN in computer vision (Table 2) and robotics (Table 3). Additionally, the same year, (Christensen et al., 2022) presented a "roadmap on neuromorphic computing and engineering" with a complete section dedicated to SNN applications, including robotics, self-driving cars, audition, olfaction and chemosensation.

It is interesting to mention here the recent advances in deep SNNs: these networks aim to "bridge the gap between the big success of deep learning in AI applications and the promise of SNNs" (Pfeiffer & Pfeil, 2018). Indeed, since its invention, the application of deep learning methods allowed for significant improvements in "the state-of-the-art in speech recognition, visual object recognition, object detection and many other domains such as drug discovery and genomics" (LeCun, Bengio, & Hinton, 2015). Deep neural networks (DNNs) are composed of multiple layers (divided into one input, one output and at least two hidden ones) allowing for multiple levels of data representation and abstraction. Thanks to the feature discovery hierarchy, the different features are not designed by humans but acquired from data using non-linear activation functions with increasing complexity, abstraction and invariance (Tavanaei, Ghodrati, Kheradpisheh, Masquelier, & Maida, 2018). Even though DNNs strongly drew inspiration from neuroscientific models of cortical hierarchies, such as pattern recognition in the primate's brain, they lack fundamental bio-

plausible elements compared to the brain, most notably the communication of information as spikes.

The introduction of deep SNNs therefore aims to overcome this issue. They seem particularly promising for processing event-based visual and auditory inputs while deployed on neuromorphic hardware (Tavanaei et al., 2018). The SNN intrinsic behaviours should lead to fast propagation of salient information through multiple layers with low-power consumption (Pfeiffer & Pfeil, 2018) — however, these promises are not yet fulfilled. Some notable limiting factors are the lack of spike-oriented training algorithms, benchmarks not suited to the bio-inspired aspect of SNNs (more suited to tasks based on continuous data perceived in the real world than frame-based datasets) and evaluation metrics measuring efficient real-world performance. To bypass this pitfall, recent works took a heightened interest in spike-based training methods and local learning rules in order to develop a neural network “as efficient and biologically plausible as SNNs but as powerful as DNNs in performing different tasks" (Tavanaei et al., 2018).

Of note, the recent application of surrogate gradient to SNNs, although distancing itself from biology, seems to offer a “particularly flexible and efficient method to overcome the aforementioned challenges" (Neftci, Mostafa, & Zenke, 2019). This strategy consists in applying to SNNs the supervised learning algorithm “back-propagation”, popular in DL and which updates synaptic weights at each timestep in order to minimise the error, i.e. the distance between the current output and the desired one (Masquelier, 2021). Applying back-propagation to SNNs is not trivial as it requires differentiable activation functions, which are inherently absent in SNNs. (Neftci et al., 2019) propose the Surrogate Gradient-Learning, a global approach commonly acclaimed which approximates the LIF basis equations into a recurrent relation for the potential, thus allowing the application of back-propagation with the help of a “surrogate gradient”. Thanks to this method’s approximations, SNN models can maintain low firing rates while obtaining nearly state-of-the-art accuracy. We can also mention the S4NN, a latency-based backpropagation for static stimuli which estimates the gradients of the loss with respect to all the weights but is constrained to bellow state-of-the-art performance due to its “at-most-one-spike-per-neuron” limitation (Kheradpisheh & Masquelier, 2019).

CHAPTER 3

Event-based cameras

In this chapter, we introduce the notion of event-based cameras. Also called silicon retinas, they represent a new kind of visual sensor that measures pixel-wise changes in brightness and accordingly outputs asynchronous events. This novel technology allows for a sparse, non-redundant and energy-efficient recording and storage of visual information, which has recently led to a substantial increase in its use for computer vision tasks, notably in robotics.
With the recent advent of event cameras, we seek to use this unusual type of data as input to neural networks applied to selected vision tasks (scene segmentation, feature tracking, depth estimation, etc.). Additionally, event data streams are highly compatible with spiking neural networks due to the similarity in the information encoding ; therefore, these features make their combined use unavoidable for the future of real-time embedded vision systems.

3.1 Operation of an event camera 40
3.1.1 Biological retina 40
3.1.2 Silicon retina 40
3.2 Event data representations 42
3.2.1 Representations retaining temporal accuracy 42
3.2.2 Representations losing temporal accuracy 42
3.2.3 Tools for event data handling 42
3.3 Applications and corresponding datasets 43
3.3.1 DVS 128 Gesture 43
3.3.2 N-MNIST 44
3.3.3 DDD17 44
3.3.4 DSEC 45
3.3.5 Lip reading 46
3.3.6 Overview of existing datasets 47
3.4 Limits 49

Conventional video cameras are based on the periodic acquisition of frames. Each frame is a static representation of the visual scene acquired by measuring the average light intensity during a certain sub-period, commonly known as exposure time. The intensity of each pixel is periodically measured (with a typical frame period of 20-30ms) and communicated regardless of whether it contains relevant information. Furthermore, objects moving at high speed relative to the frame time appear blurred in the image frame.

However, biological vision systems are not frame-based, but the transmitted information is coded as spikes. The idea of a novel bio-inspired event-based sensor, akin to a "*silicon retina*" (Mahowald & Mead, 1991), has been developed since the 1990s. Pixels of such sensors emit spikes whenever relevant information is captured in the visual field. In the last two decades, different technological developments of spiking silicon retinas implementing spatial and temporal filtering have been published: retinas converting luminance to spike frequency, spatial contrast retinas, etc. They did not reach the maturity of commercial applications due to several limitations such as high fixed pattern noise, low fill factor, and complex circuitry resulting in low sensor resolution.

In 2008, Lichtsteiner, Posch and Delbruck presented the first complete design of a DVS (Lichtsteiner et al., 2008a): it implements a simplified retinal model where each pixel computes autonomously the relative time difference of the illumination received on its photosensor and responds only to temporal brightness changes in a scene with no consideration for colours, similar to the biological retina. When pixel illumination increases and its relative change exceeds a certain threshold, the pixel generates a positive ON output spike. Similarly, the pixel generates a negative OFF output spike if the illumination decreases and its relative change exceeds a certain negative threshold. DVS circuitry can be implemented with compact circuitry, which results in low fixed pattern noise and higher resolution sensors. Furthermore, DVS exhibit high temporal resolution (below 1 microsecond) and intrascene dynamic range as high as 120dB (Lichtsteiner et al., 2008a). Due to these features, these sensors have emerged as the first commercially available neuromorphic sensors and have overcome the integration density and mismatch limitation exhibited by previous neuromorphic retinas. Advanced megapixel DVS sensors have been developed (C. Li, Longinotti, Corradi, & Delbruck, 2019; Suh et al., 2020; Kubendran, Paul, & Cauwenberghs, 2021; M. Guo, Huang, & Chen, 2017) and new high-speed vision applications and systems based on DVS sensors are emerging.

The main advantages of such an event-based camera are (Gallego et al., 2020):

— the high temporal resolution (microsecond time grain), thanks to which an event can be emitted on the timescale of microseconds and avoid motion blur;
— the high dynamic range (up to $120dB$), which makes it possible to use them for extremely dim as well as under bright sun illumination;
— the high contrast range, which allows for highly contrasted images avoiding dazzling effects caused by sudden illumination changes, thus scenes with illumination expanding over $120dB$ can be sensed without suffering saturation;
— the low latency and asynchronicity enabled by the independence between each pixel;
— the absence of redundancy in the information transmitted, as compared to frame-based sensors (1,000 times fewer data produced by an event camera than in RGB videos (Prophesee, 2019));
— the low power consumption ($10mW$ at the die level and less than $100mW$ for the whole embedded system (Gallego et al., 2020)), following the model of biological retinas ($3mW$

for the retina and less than $10mW$ for the whole eye (Skorka & Joseph, 2011)) and substituting the biological photoreceptors with photodiodes in the electrical circuits.

By definition, a DVS generates sparse data: since each DVS pixel generates a string of spikes which respond to the temporal variation of the illumination, the pixel remains silent under a static background, saving communication and computation energy of the subsequent recognition network. This spatiotemporal filtering eliminates data redundancy over time and space ; this technology therefore allows for an energy-efficient recording and storage of data evolving over time and space. However, the increasing resolution of DVS prototypes (S. Chen & Guo, 2019) has motivated the research on active reduction methods for DVS data to save communication, computation bandwidth, and energy.

3.1 Operation of an event camera

3.1.1 Biological retina

In biological retinas, photoreceptors sense the visual scene and lead to continuous and asynchronous spike generation whenever relevant information is detected, i.e. when high spatial or temporal contrast is caught in the scene (Meister & Berry, 1999).

The human retina performs vision thanks to a wide array of photosensitive cells, of which around 100 million (the rods) are responsible for black-and-white vision and 4 million (the cones) for vision in colour (Steffen et al., 2019). These photoreceptors make one of the three retinal layers presented in Fig. 3.1 ; the other two are the outer and inner plexiform layers, respectively containing bipolar cells and ganglion cells.

The bipolar cells can be divided into ON and OFF-types, depending on their response to photoreceptor activation linked to their glutamate receptors: an OFF cell is hyperpolarised, thus "turned off", when subjected to light whereas an ON cell is depolarised, i.e. "turned on' (Bear et al., 2007)'. The receptive field of a bipolar cell is divided into its center, from which comes direct photoreceptor input, and its surround, receiving input from the horizontal cells (see Fig. 3.1) ; the activation of the center and the surround lead to different responses from the membrane potential. The bipolar cells are thus said to have "antagonistic center-surround receptive fields". A similar ON-OFF and center-surround organisation can be found in the ganglion cells (Bear et al., 2007).

The bipolar and ganglion sensory cells lower the activity rate produced by the photoreceptors and received *via* the horizontal cells before transmitting the information through the optic nerve to be processed by the neural layers of the visual cortex. This spike-based coding of the visual scene results in high temporal resolution and low communication.

3.1.2 Silicon retina

Fig. 3.2 shows the conceptual schematic of an event camera pixel. A photosensor produces a current I_{ph} proportional to the illumination. A trans-impedance logarithmic circuitry converts the detected current into a voltage $V_{\log} = A_T \log(I_{ph})$, which depends logarithmically on the photocurrent. This voltage is the input to a differencing capacitive amplifier so that the voltage difference

$$\Delta V_{\text{diff}} = -C_1/C_2 \Delta V_{\log} = -C_1/C_2 A_T \frac{\Delta I_{\text{ph}}}{I_{\text{ph}}} \tag{3.1}$$

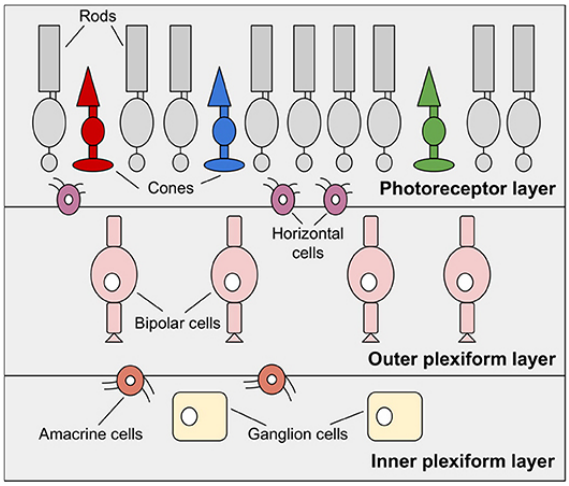


Figure 3.1 – Human retina, as presented in (Steffen et al., 2019), reduced to essential layers for neuromorphic visual sensors: the photoreceptor layer, the outer plexiform layer including bipolar cells and the inner plexiform layer made up of ganglion cells. Additionally, horizontal cells and amacrine cells connect these layers.

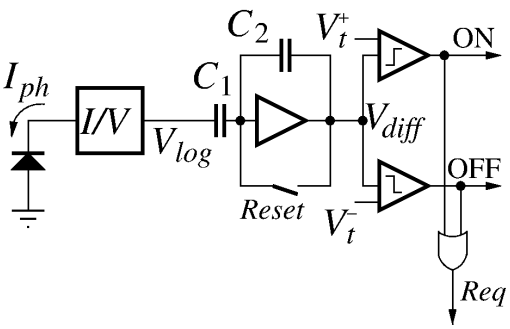


Figure 3.2 – Block diagram of an event camera pixel.

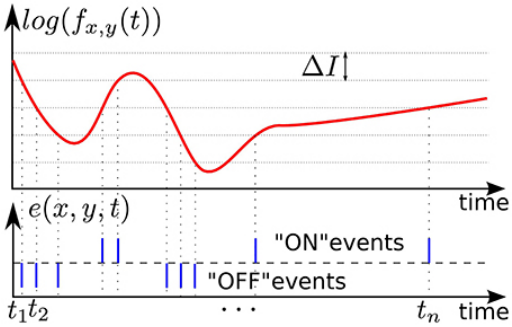


Figure 3.3 – Illustration of event-based encoding of visual signals with respect to the predefined threshold ΔI, from (Lagorce et al., 2015).

That way, voltage V_{diff} is proportional to the relative temporal variation of the photocurrent. Voltage V_{diff} is compared with upper and lower voltage thresholds. Each time it goes over the upper (or below the lower) threshold, the pixel will generate an output ON (or OFF) event and voltage V_{diff} is reset to a resting value V_R.

Fig. 3.3 illustrates the resulting behaviour of the event camera pixel. The upper subfigure plots the variation of a pixel photocurrent over time, while the lower subfigure illustrates the generated ON and OFF output events. As can be observed, each time the photocurrent increases (or decreases) by a certain relative variation given by,

$$\frac{\Delta I_{\text{ph}}}{I_{\text{ph}}} = \frac{C_2(V_t - V_R)}{C_1 A_T} \tag{3.2}$$

the pixel generates an ON (OFF) output event.

Multiple event camera designs have been developed since the first bio-inspired sensor was built by Misha Mahowaldin in 1991. We can differentiate between three main architectures, whose characteristics are outlined in Table 2 of (Steffen et al., 2019): the Dynamic Vision Sensor (DVS), the Asynchronous Time-based Image Sensor (ATIS), and the Dynamic and Active Pixel Vision Sensor (DAVIS). (Gallego et al., 2020) presents a comprehensive comparison of existing commercial and prototype event cameras in their Table 1.

3.2 Event data representations

Formally, an event is a tuple (x, y, t, p) that indicates the pixel x, y where the event occurred, a timestamp t (with $10^{-6}s$ time resolution), and a polarity p that indicates the direction of change: positive for brighter and negative for darker. However, event data is often preprocessed and transformed into alternative representations more adapted to given tasks (Gallego et al., 2020). These representations can be divided into two categories described below according to whether they retain temporal accuracy or not.

3.2.1 Representations retaining temporal accuracy

The four representations maintaining temporal accuracy are individual events (before any preprocessing), which is a format particularly suited for SNN handling (Paredes-Valles et al., 2020; Weikersdorfer & Conradt, 2012); the voxel grid, a 3D histogram of events where each voxel represents a particular pixel and time interval (L. Wang, Mostafavi, Ho, & Yoon, 2019; A. Zhu, Yuan, Chaney, & Daniilidis, 2019); and the 3D point set, akin to a graph where each event is treated as a point in 3D space (Benosman, Clercq, Lagorce, Ieng, & Bartolozzi, 2014; Sekikawa, Hara, & Saito, 2019).

3.2.2 Representations losing temporal accuracy

Many methods represent events by accumulating them over time, thus losing their temporal accuracy. Among them can be found the frame or 2D histogram, where the events are summed and/or accumulated into 2D grids (Kogler, Sulzbachner, & Kubinger, 2009; Liu & Delbrück, 2018; Maqueda, Loquercio, Gallego, Garcıa, & Scaramuzza, 2018); the time surface, a 2D map where each pixel stores the timestamp value of the last event at that pixel (Delbruck, 2008a; Lagorce et al., 2016; Grimaldi, Boutin, Ieng, Benosman, & Perrinet, 2022); the event packet, which gathers events in a spatiotemporal neighbourhood to process them together (Reinbacher, Munda, & Pock, 2017; Mueggler, Gallego, Rebecq, & Scaramuzza, 2018); reconstructed brightness images that are more motion-invariant than time surfaces (Rebecq, Ranftl, Koltun, & Scaramuzza, 2019b); and motion-compensated event image where the events are wrapped to a reference time, and their alignment is maximised (Gallego & Scaramuzza, 2017; Gallego, Rebecq, & Scaramuzza, 2018).

3.2.3 Tools for event data handling

As seen above, event cameras produce a novel kind of data whose novel format complicates the processing and application of methods traditionally used in computer vision. A particular com-

plication can be found in the wide array of file formats (`AEDAT`, `CSV`, `NPY` or `NPZ`, `MAT`, `DAT`, `HDF5`... and others presented in Tab. 3.1) and the possible incompatibility of specific formats and computer languages (for example, between `AEDAT2` and Python).

Multiple tools were thus made available over time to handle event data, with various specifications described in (Robotics & Perception Group, 2020).

Among these, we were particularly interested in *Tonic*. This Python library for neuromorphic data handling has been created on an original idea dating back from the 2019 Telluride neuromorphic workshop (Lenz et al., 2021). It facilitates dataset downloads and their conversion to different event representations (frames, time surfaces, voxel grids, etc.). It also proposes event transformation, such as denoising, event dropping, polarities merging, spatial and temporal jittering and downsampling.

3.3 Applications and corresponding datasets

The event cameras' particularities grant them advantages in many application cases compared to more traditional visual sensors (Gallego et al., 2020). Because of their operating mode, event-based cameras are helpful in tasks featuring movements, such as object tracking (Glover & Bartolozzi, 2016), gesture recognition (Amir et al., 2017), optical flow estimation (Orchard, Benosman, Etienne-Cummings, & Thakor, 2013 ; Paredes-Valles et al., 2020), Simultaneous Localisation and Mapping (Kim, Leutenegger, & Davison, 2016 ; Vidal, Rebecq, Horstschaefer, & Scaramuzza, 2018), etc. Their high resolution and contrast range is also valuable for astronomical studies (Cohen et al., 2017 ; Chin, Bagchi, Eriksson, & Schaik, 2019). Research on their combined use with traditional RGB cameras becomes more common, with applications on high-resolution image reconstruction (Z. Zhang, Yezzi, & Gallego, 2021 ; Rebecq et al., 2019b) and video deblurring (L. Zhang et al., 2021). A significant field of applications is robotics: autonomous vehicles, drones and other embedded systems facing limitations in terms of energy resources, memory, computational power, and communication bandwidth.

In 2020, (Gallego et al., 2020) presented a complete survey of event-based vision according to the tasks addressed. The authors keep an up-to-date list of existing event-based datasets and corresponding applications publicly available at (Robotics & Perception Group, 2020). We introduce in the paragraphs below the datasets most commonly used in current neuromorphic research and their corresponding applications. The last paragraph brings together 38 datasets (including the ones detailed previously) summarised in table form.

3.3.1 DVS 128 Gesture

Hand gesture recognition is a skill used daily in human society and is tightly integrated with verbal communication, hence the need for computational learning of such data. Since it relies heavily on temporal information (i.e. a static representation is not enough to solve this task), this task is well-suited for event-based computation. Building on this notion, Amir et al. presented at CVPR 2017 a complete hand gesture neuromorphic dataset called *DVS128 Gesture* (Amir et al., 2017), which has now become a standard benchmark in event data classification. To this end, 29 subjects were recorded performing 11 different hand gestures under three kinds of illumination conditions by a DVS128 camera. A total of about 133 samples are available for each gesture, each composed roughly of 400,000 events and of dimension 128×128 pixels, for a duration of approximately 6 seconds. The dataset is split into two sub-datasets to facilitate ML training: the

train set consists of 80% of the recorded samples and the test set 20%, with a balanced distribution of the 11 gestures between them.

3.3.2 N-MNIST

The *Neuromorphic-MNIST* dataset is an event translation of the original *MNIST* dataset (for Mixed National Institute of Standards and Technology), which was presented in 1998 by Y. LeCun and which gathers size-normalised handwritten digits for pattern recognition (Lecun, Bottou, Bengio, & Haffner, 1998). To record this neuromorphic adaptation, Orchard et al. presented an average of 7,000 MNIST samples on a screen for each digit from 0 to 9 during 500 ms to an ATIS sensor of resolution 28×28 pixels mounted on a motorised unit, moving in saccades (Orchard et al., 2015). This produces 80,000 neuromorphic samples, each roughly composed of 4,200 events. The set was split into train (88%) and test (12%) sub-sets for ML purposes. The authors showcase the advantages its small resolution brings, which is a "reduce[d] processing time, allowing for rapid testing and iteration of algorithms when prototyping new ideas".

N-MNIST is particularly heavy in events: according to the parameters presented in the above paragraphs, N-MNIST displays, on average, 10.71 events per pixel per second, whereas DVS128 Gesture only displays about 4.06 events per pixel per second.

3.3.3 DDD17

The DAVIS Driving Dataset 2017 (DDD17) (Binas et al., 2017) contains 40 different driving sequences of event data captured by an event camera (around $447GB$). However, since the original dataset provides only both grayscale images and event data without semantic segmentation labels, (Alonso & Murillo, 2019) provides segmentation labels they produced from 20 different sequence intervals taken from 6 of the original DDD17 sequences as well as a multi-channel representation of the corresponding events (normalised sum, mean and standard deviation for each polarity). One can extract the corresponding original events from DDD17 with the traditional $< x, y, p, t >$ structure using DDD20 tools [*]. The resulting dataset is split into a training dataset consisting of 15,950 frames and a testing one consisting of 3,890 frames.

Ev-SegNet is a semantic segmentation model applied to DDD17 and built by (Alonso & Murillo, 2019), which outperforms all existing studies using event cameras in this kind of task. Semantic segmentation is a crucial task in visual scene understanding and corresponds to a visual classification problem which assigns a label, i.e. a given class, to each pixel in the image. This model is inspired by current state-of-the-art semantic segmentation CNNs, slightly adapted to use the event data encoding. As shown in Fig. 3.4, it consists of an encoder-decoder architecture: an encoder represented by the Xception model in which all the training is concentrated, and a light decoder connected to the encoder via skip connections to help deep neural architecture to avoid the vanishing gradient problem and also to make the fine-grained details learned in the encoder part used in the decoder to construct the initial image. Moreover, (Alonso & Murillo, 2019) uses an auxiliary loss which increases convergence speed.

The model takes as input six channels representing the count, mean and standard deviation of the normalised timestamps of events happening at each pixel within an interval of $50ms$ for

∗. https://github.com/SensorsINI/ddd20-utils

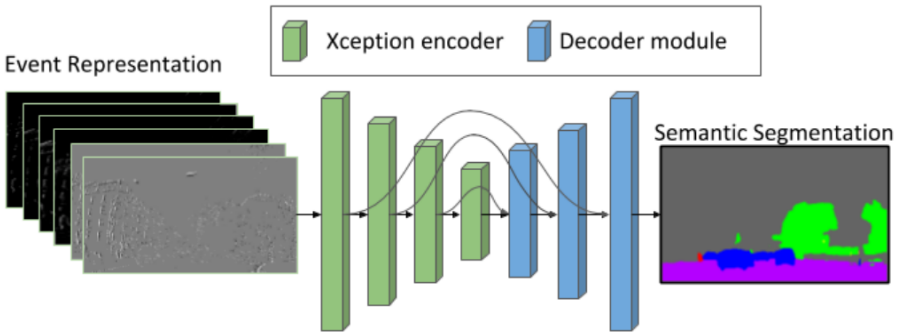


Figure 3.4 – CNN architecture of Ev-SegNet, from (Alonso & Murillo, 2019).

the positive and negative polarities. It is applied to the DDD17 dataset described previously. Finally, the training is performed via backpropagation to minimise the soft-max cross-entropy loss measured by summing the error between the estimated pixels' classes and the true ones.

The semantic segmentation performance is measured thanks to standard metrics of semantic segmentation: the Accuracy (Eq. 3.3) and the Mean Intersection Over Union (MIoU — Eq. 3.4).

$$\begin{aligned} \text{Accuracy}(y, \widehat{y}) &= \frac{1}{N} \sum_{i=1}^{N} \delta(y_i, \widehat{y}_i) \\ &= \frac{TP + TN}{TP + TN + FP + FN} \end{aligned} \tag{3.3}$$

$$\begin{aligned} \text{MIoU}(y, \widehat{y}) &= \frac{1}{C} \sum_{j=1}^{C} \frac{\sum_{i=1}^{N} \delta(y_{i,c}, 1)\delta(y_{i,c}, \widehat{y_{i,c}})}{\sum_{i=1}^{N} max(1, \delta(y_{i,c}, 1)\delta(\widehat{y_{i,c}}, 1))} \\ &= \frac{TP}{TP + TN + FP + FN} \end{aligned} \tag{3.4}$$

where y and $\widehat{y}$ are the desired output and the system output respectively. C is the number of classes. N is the number of pixels and δ denotes the Kronecker delta function. TP, TN, FP, and FN respectively stand for true positive, true negative, false positive, and false negative.

3.3.4 DSEC

The Driving Stereo Even t Camera dataset *DSEC* is a hybrid stereo event camera and video camera dataset in driving scenarios, in both favourable and very challenging illumination conditions (M. Gehrig et al., 2021). The scenes are recorded using two event cameras of resolution 640×480, combined with an RGB camera and a lidar. The dataset is composed of 53 sequences in urban, suburban and rural areas in Switzerland, amounting to a total of 3193 seconds. The authors wished to overcome issues present in existing similar datasets (such as the Multi Vehicle Stereo Event Camera Dataset MVSEC, DDD17 or DDD20) and evaluate the datasets' quality using several established metrics in stereo matching.

3.3.5 Lip reading

Tan et al. (Tan et al., 2022) proposed a novel dataset of event-based recordings of lip motions and an associated architecture to perform event-based lip reading: the MSTP network. As discussed in (Tan et al., 2022), the existing event-based action recognition methods, such as point-cloud-based, graph-based, and fixed-frame-based approaches, are not suitable for the lip reading task since it requires the perception of fine-grained spatiotemporal features from the event data.

The performance of the MSTP on this task was experimentally proven to be superior to the state-of-the-art event-based action recognition models and video-based lip reading models. Within the MSTP network, the input events are consequently converted to low-rate and high-rate event frames with different temporal bins to better preserve the spatiotemporal information of the event stream. These two types of event frames are then fed into a multi-branch network with message flow modules between different branches designed to perceive both complete spatial features and fine temporal features from the event data. Following that, a sequence model decodes the visual features into words. In their study, SNNs are mentioned but discarded because they lack an efficient backpropagation algorithm to train them. Hence, SNNs are yet to be applied to this problem.

3.3.6 Overview of existing datasets

Table 3.1: Overview of existing event-based datasets.

Name	Task	Classes	Size and format
ASL DVS (Yuhuang, 2016)	Object recognition	Sign language for 24 letters (A to Y, excluding J)	19.9GB(`MAT`)
ATIS Planes (Afshar et al., 2019)	High speed detection and unsupervised object extraction	Different plane models	322.8MB (`MAT`)
Color Event Camera Dataset (Scheerlinck et al., 2019)	Multiple applications	Object and human motion, interior scenes, driving, calibrations with and without infrared	32.9GB (binary `rosbag`)
CIFAR10-DVS (H. Li et al., s. d.)	Object recognition	Transportation, animals	7.81GB (`AEDAT`, `DAT`)
Combined Dynamic Vision / RGB-D Dataset (Weikersdorfer et al., 2014)	SLAM	5 scenarios	7.8GB (`TSV`, `BVH`)
DDD17 (Binas et al., 2017)	Autonomous navigation	Scenes of highway and urban driving	447GB (`AEDAT`, `HDF5`)
DDD20 (Hu et al., 2020)	Autonomous navigation	Scenes of highway and urban driving	931GB (`AEDAT`, `HDF5`)
DET (Cheng et al., 2019)	Lane extraction, autonomous navigation	Labelled lines	44.31GB (`BIN`, `BMP`)
DHP19 (Calabrese et al., 2019)	3D human pose recognition	33 movements	160GB (`AEDAT`)
DND21 (S. Guo & Delbruck, 2022)	Background activity denoising	Driving and interior scenes, moving dots	44.8GB (`AEDAT2`)
Driving Event Camera Dataset (Rebecq et al., 2019b)	Autonomous driving	Highway, street and parking with or without sun	124GB (`rosbag`)
DSEC (M. Gehrig et al., 2021)	Autonomous driving and stereovision	Driving in a variety of illumination conditions	152GB (`H5`)
DVS09 (Delbruck, 2008b; Lichtsteiner et al., 2008b)	Recognition, surveillance and tracking	Walking, hand motion, driving, eye tracking, mouse activity	450MB (`DAT`)
DVS 128 Gesture (Amir et al., 2017)	Gesture classification	11 hand gestures	2.7GB (`AEDAT`, `NPZ`)
DVSACT16 (Hu et al., 2016; Hu, 2016)	Object tracking, action recognition and object recognition	Conversion of 4 existing dynamic RGB datasets, i.e. VOT Challenge 2015 dataset, TrackingDataset, UCF-50 Action Recognition Dataset and Caltech-256 Object Category Dataset	45.1GB (`AEDAT`, `HDF5`)
DVSFLOW16 (Rückauer & Delbruck, 2016)	Optical flow	Synthetic and real-life motions	3GB (`AEDAT`)
DVSMOTION20 (Almatrafi et al., 2020)	Optical flow	Camera motions (checkerboard, classroom, conference room) and object motion (hands, cars)	849MB (`AEDAT`, `MAT`)
DVSNOISE20 (Baldwin et al., 2020)	Denoising	Interior scenes	7.9GB (`AEDAT`, `MAT`)

Continued on next page

Table 3.1: Overview of existing event-based datasets. (Continued)

EBSSA (Cohen et al., 2017; Afshar et al., 2020)	Detection and tracking for space situational awareness	Daytime and nighttime recordings of spatial objects	12.6GB (MAT)
EDFLOW (Liu & Delbruck, 2022)	Optical flow	Outdoor scenes	15.8GB (AEDAT)
Event-based, Direct Camera Tracking (Bryner et al., 2019; Gallego, Lund, et al., 2018)	Camera tracking	Indoor and outdoor camera trajectories	244MB (ROSBAG)
Event-based Lip-reading (Tan et al., 2022)	Lip reading	Close-ups of lips pronouncing words (100 classes)	87.4MB (NPY)
EVIMO (Mitrokhin et al., 2019, 2020; Parameshwara et al., 2021)	Motion segmentation	Various backgrounds	87.4MB (BAG, NPZ, TXT)
GEN1 (de Tournemire et al., 2020)	Automotive, pedestrian detection	Cars, pedestrians	unknown (DAT)
HQF (Stoffregen et al., 2020)	Acquisition of quality ground truth	Camera motions, indoor and outdoor scenes	2.7GB (BAG)
IROS2018 (Mitrokhin et al., 2018)	Moving object detection and tracking	Objects with varying degree of occlusions and brightness	200MB (TXT, PNG, BAG)
MNIST-DVS and **MNIST-FLASH-DVS** (Serrano-Gotarredona & Linares-Barranco, 2013)	Handwriting recognition	Numbers from 0 to 9	6GB (AEDAT, MAT)
MVSEC (A. Z. Zhu, Thakur, et al., 2018)	Optic flow and 3D reconstruction	Stereo sequences of indoor and outdoor scenes, recorded from a driving car, a hexacopter, a motorcycle or a human hand	74.7GB (rosbag, HDF5)
N-Caltech101 (Orchard et al., 2015)	Object recognition	256 categories	4GB (binary data)
N-CARS (Sironi et al., 2018)	Car classification	Car and background	4.13GB (AEDAT)
N-MNIST (Orchard et al., 2015)	Number classification	Handwritten numbers (10 classes)	9GB (BIN, NPZ)
PAF (Miao et al., 2019)	Pedestrian detection, action recognition and fall detection	Pedestrian, actions (10 classes) and postures (4 classes)	4.13GB (AEDAT)
POKER-DVS (Pérez-Carrasco et al., 2013)(Serrano-Gotarredona & Linares-Barranco, 2015) and **SLOW-POKER-DVS** (Rodríguez et al., 2017)	Symbol recognition	4 classes	12MB (AEDAT)
PRED18 (Moeys et al., 2016, 2018)	Predator/prey robot chasing	Left, right, center and non-visible	298GB (AEDAT)
ROSHAMBO17 (Lungu et al., 2017)	Symbol recognition	Rock, paper, scissor and no hand	79GB (AEDAT)
SL-ANIMALS-DVS (Vasudevan et al., 2020, 2022)	Action recognition	Sign language words for 20 animals	749MB (AEDAT)

3.4 Limits

While the event camera provides the significant advantages described above, one could ponder whether its microsecond resolution is relevant in any situation. In recent works, some of its application cases could have been solved using a less precise temporal grain. for example, human pose estimation and gesture recognition are studied in the widely used benchmark dataset DVS 128 Gesture described above — but does one really need a microsecond resolution when one gesture is completed in roughly 1 second ? It could be of greater interest to apply this technology to domains where the phenomenons of interest are significantly brief, such as in high-energy physics or detonics.

Additionally, due to their differential imaging mechanism event cameras are more subject to noise than standard RGB cameras: both are subject to ‘inherent shot noise in photons and transistor circuit noise, but the additional quantisation of temporal contrast in event cameras is more complex to control (Gallego et al., 2020). The noise in event data is mainly caused by to movements of background objects (Padala, Basu, & Orchard, 2018). When the overall brightness of the visual scene decreases, the noise in the recorded event data increases due to the camera continual sampling and the logarithmic conversion (Ding et al., 2023). To overcome this issue, different strategies to implement denoising either at sensor level or in post-processing were developed in recent works (see Section 1.3.1). (Dilmaghani, Shariff, Ryan, Lemley, & Corcoran, 2022) introduces in 2022 a complete evaluation of the different biases (i.e. sensor settings) proposed by Prophesee in their Gen4H camera and compares the quality of output events thanks to the “average image gradient magnitude" metric assessing their sharpness. It has also been demonstrated that the contrast sensitivity of an event camera influences the amount of noise and pixel-to-pixel mismatch: the contrast should be set above a lower bound to avoid “a storm of events" (Gallego et al., 2020).

CHAPTER 4

Spiking neural networks applied to an embedded event camera

The joint use of spiking neural networks and event cameras, respectively introduced in previous Chapters 2 and 3, in an embedded setting is a promising combination for dynamic visual data processing. Both technologies have recently emerged separately about a decade ago from electronics and neuroscience communities, sharing many features: biological inspiration, temporal dimension, model sparsity, aim for higher energy efficiency, etc. This chapter further develops this initial statement and presents an overview of state-of-the-art embedded models interfacing event cameras and SNNs.

4.1 Embedded systems 53
4.2 Why is it such a good idea ? 53
4.3 Applications 54
4.3.1 Classification 54
4.3.1.1 SLAYER benchmark classifier 54
4.3.1.2 PLIF-based classifier 55
4.3.1.3 HOTS 55
4.3.2 3D reconstruction 56
4.3.2.1 Osswald *et al.*, 2017 56
4.3.2.2 Dikov *et al.*, 2017 57
4.3.2.3 Risi *et al.*, 2020 59
4.3.3 Visual attention 59

Because of their asynchronous operation principle, event cameras are a natural match for SNNs: each event can be assimilated to an activated spike between two neurons. Furthermore, the behaviour of the log-luminance recorded by the sensor displays a substantial similarity with one of a LIF neuron, as shown in Fig. 6.10. Their combined use is of such high interest from the point of view of biological inspiration, energy savings, decision latency, and memory use that it is gaining momentum in the field of embedded computer vision (Gallego et al., 2020).

4.1 Embedded systems

According to De Micco *et al.*, an embedded system is an "electronic equipment with a computing core which, unlike a personal computer, is designed to meet a specific function and is usually optimised to satisfy strict requirements of processing time, reliability, power consumption, size and cost" (De Micco, Vargas, & Fierens, 2020). As underlined in (Biglari & Tang, 2023), those requirements still hinder the implementation of most ML algorithms. One must consider the specific application to identify the appropriate embedded system and its equipment, such as the sensors used. A review of the state-of-the-art research in the embedded ML area is presented in (Biglari & Tang, 2023), according to the application tasks (autonomous driving, security, healthcare, agriculture, drones, etc.). Among the sensors available, they mention LIDAR, electrocardiograms, microphones, radars, motion sensors and RGB, depth, thermal and 360-degree cameras. They make the remarkable omission of event-driven cameras, which are nevertheless particularly well suited to onboard computer vision due to their high dynamic range and temporal grain as well as their low memory and low energy consumption (see Chap. 3).

4.2 Why is it such a good idea ?

The emerging event-based vision technology promises to significantly improve energy efficiency and latency in the next generation of computer vision applications (Gallego et al., 2020). However, classic computer vision algorithms which rely on frame-based data for applications such as object classification (Song et al., 2019) and scene segmentation (Peng et al., 2019) are unable to handle sparse, asynchronous event data coming from DVS. Approaches combining classic methods with event-based data as input require a first step of data pre-processing at the cost of losing temporal information (A. Z. Zhu, Yuan, Chaney, & Daniilidis, 2018).

However, recent work has shown that SNN can operate on raw event data and achieve similar performance as conventional methods (Paredes-Vallés, Hagenaars, & de Croon, 2021). Thanks to neuromorphic chips, and ultra-fast low-power processors, SNNs can be implemented on hardware and interfaced directly to an embedded event camera. They thus make for a promising alternative for the embedded handling of event-based data.

Indeed, even though the maturity of SNN-based vision algorithms is still far from state-of-the-art conventional methods, which provide excellent results for vision tasks with traditional cameras, asynchronous event sequences require special handling for which SNNs are better suited. Indeed, many previous works have demonstrated the benefits of SNN and the perfect match with event-based cameras to reach both exceptionally low latency and energy efficiency (Gallego et al., 2020) — all of this, however, without achieving equivalent performance to DNNs.

Pushed by the objective of closing the gap with DNN accuracy, many previous works do not consider the limits of memory and complexity that state-of-the-art embedded neuromorphic pro-

cessors pose. Another drawback that limits the success of today's event-based cameras and neuromorphic computing, especially for embedded low-power systems, is the fact that the promised low power, low latency and energy efficiency are overpowered by the interface's high power consumption, often due to non-standard communication protocol. Although nowadays data is mainly acquired using a high-power FPGA and a USB interface (Di Mauro et al., 2021), novel embedded processors are being developed, providing the research community with reference platforms and resources to address this issue. Large-scale, neuromorphic processors like SpiNNaker (S. B. Furber et al., 2013) or Intel Loihi (Davies et al., 2021) can simulate hundreds of thousands or even millions of spiking neurons in real time. To achieve genuinely power-efficient real-time operation, such platforms need to be even smaller, reach even lower power consumption and integrate the camera interface directly on board. Such an example is given by Kraken (Di Mauro et al., 2022), which can interface directly with event-based cameras without power-hungry FPGAs or external interface adapters.

4.3 Applications

Traditional methods for standard vision tasks (e.g. recognition, tracking, segmentation, motion analysis, etc.) cannot be applied straightforwardly to event-based cameras due to their unconventional output type. Silicon retinas bring considerable potential in indexing and retrieval, yet only a limited number of methods have been explored in this direction. In 2020, Gallego et al. (Gallego et al., 2020) establish *inter alia* a state of the art of existing SNN algorithms for feature detection, object tracking, 3D reconstruction and motion segmentation applied to event data. This state of the art is maintained up to date on the team's Github (Robotics & Perception Group, 2020).

We present below some state-of-the-art models applied to different application tasks and extend those relevant to the experiments performed during this PhD.

4.3.1 Classification

A classification task aims to assign a label to each dataset sample.

4.3.1.1 SLAYER benchmark classifier

Shrestha and Orchard developed in 2018 a Python framework called SLAYER (for Spike Layer Error Reassignment in Time) (Shrestha & Orchard, 2018), based on the ML library PyTorch (Paszke et al., 2019) and designed to simulate "backpropagation based SNN learning" on GPU. A specific "SLAYER Loihi" module has been implemented to run SNN models initially developed on SLAYER, on Intel's Loihi neuromorphic chips (Davies et al., 2021) without further adaptation to the code. This module and, more globally this framework, were benchmarked on a classification task applied to different spiking and non-spiking visual datasets, such as MNIST (Lecun et al., 1998), N-MNIST (Orchard et al., 2015) and DVS128 Gesture (Amir et al., 2017), and the audio dataset TIDIGITS converted into spikes (Leonard & Doddington, 1993). According to the authors, when the input is a spiking dataset, "the spike data from the DVS is directly fed into the classifier" which removes any step of pre-process and allows the SNN to be trained using "significantly less number of neurons and layers" than any state-of-the-art CNN classifier.

4.3.1.2 PLIF-based classifier

Fang et al. introduced during ICCV 2021 an SNN classifier using a new spiking neuron model called PLIF (Fang, Yu, Chen, Masquelier, et al., 2021). This model aims to better represent the heterogeneity of biological neurons by learning the membrane time constant, whereas membrane parameters usually correspond to hyperparameters set before training. To reproduce the biological dynamic where all neighbouring neurons share similar properties, the time constant is the same for all neurons belonging to the same layer but differs between layers, thus implementing diverse phase-frequency responsiveness. The authors also opted for a max-pooling method instead of the average-pooling method traditionally used in SNN, thus preserving the asynchronous characteristic of neuron firing.

This model allows the authors to apply a backpropagation learning algorithm to a classification task. They present the results obtained when classifying traditional RGB datasets, as well as neuromorphic datasets such as DVS128 Gesture (Amir et al., 2017), N-MNIST (Orchard et al., 2015) and CIFAR10-DVS (H. Li et al., s. d.).

It is important to note that the authors chose to process the neuromorphic datasets as frames precisely because of the high number of events. The events E are thus translated from their traditional representation $E(x_i, y_i, p_i, t_i)$ (with x_i and y_i the pixel's coordinate of the i^{th} event, p_i the polarity and t_i the timestamp) into T slices of same time length. The frames F are represented as follows:

$$F(j, p, x, y) = \sum_{t=t_{\text{min}}}^{t_{\text{max}}} \delta(x_t, x).\delta(y_t, y).\delta(p_t, p) \tag{4.1}$$

with t_{min} and t_{max} the minimal and maximal timestamp in the j^{th} slice and δ the Kronecker delta function. When run on neuromorphic datasets, this classifier considers each frame as the input for one timestep ; T is the number of timesteps on which the classifier is run for one epoch.

4.3.1.3 HOTS

The online classification algorithm introduced in (Grimaldi et al., 2023) is an extension of a previous study entitled *HOTS: A Hierarchy of Event-Based Time-Surfaces for Pattern Recognition* (Lagorce et al., 2016). In this work, they make object recognition on a stream of events through a feedforward hierarchical architecture using *time surfaces*, an event-driven analog representation of the local dynamics of a scene. Using a form of Hebbian learning, the network can learn in an unsupervised way progressively more complex spatiotemporal features which appear in the event stream. Once trained, one layer of this network transforms any incoming event from the stream of events into a novel event as it is selected in a layer of spiking neurons. Using it as a building block, such layers can be stacked together, each layer's output address space defining a novel input address space for the next layer.

Inspired by this dynamical processing of the visual information, (Grimaldi et al., 2023) added a Multinomial Logistic Regression as an online classifier and transformed the previous *post hoc* classification process, which counted the activity of the neurons of the last layer, into an always-on decision process. The Multinomial Logistic Regression layer also takes *time surfaces* as input, and a formal demonstration was made to assimilate this algorithm to an SNN with Hebbian learning and homeostasis.

The classifier introduced in (Grimaldi et al., 2023) will be referred to as "HOTS" in the remainder to facilitate its designation.

4.3.2 3D reconstruction

Stereo vision is the computation of depth based on the binocular disparity between the images of an object in the left and right eyes. This requires matching up features in the two eyes, that is, identifying features in the left and right retinas that are both images of the same point in the visual scene. This allows to obtain the 3D information from a pair of images by estimating the relative depth of points in the scene. These estimates are represented in a stereo disparity map, which is constructed by matching corresponding points in the stereo pair. The disparity is computed as:

$$\text{disparity} = x - x_i = \frac{B \times f}{Z} \tag{4.2}$$

where x and x_i are the distance between points in the image plane corresponding to the 3D scene point and their camera centre, B is the baseline that represents the distance between two cameras, Z is the depth, and f is the focal length of the camera.

The models we detail below will tackle this computer vision task using a stereo-matching approach. It is interesting to note that it is far from the only existing neuromorphic approach one could apply: for example, the StereoSpike model introduced in (Rançon, Cuadrado-Anibarro, Cottereau, & Masquelier, 2021) reconstructs depth from a stereo-event stream using a modified U-Net-like encoder-decoder architecture, which performs supervised learning using a surrogate gradient and achieves good results with a remarkably low number of parameters compared to the state-of-art.

As stated above, the models we studied in this thesis tackle 3D reconstruction using stereo matching. This “correspondence problem” is the problem of finding a pair of corresponding pixels or patches in a set of stereo images. Having the matching objects or pixels, we can calculate the depth, which is inversely proportional to the disparity—the displacement along the epipolar lines—between the matches.

The geometrical characteristics of the stereo cameras can be formulated to map a single pixel of one image into a set of possible corresponding pixels in the other image. In classic computer vision, there are two general approaches for matching pixel pairs: area-based and feature-based matching. The use of DVSs introduces a separate clue that can be used for matching: time. Just as with its biological counterpart, the asynchronous operation of the silicon retina opens the possibility of matching events based on their time of appearance. However, due to inherent sensor variations and spatial aliasing, event coincidence alone is not reliable enough for event matching and has to be combined with additional constraints to achieve a reasonable matching quality.

4.3.2.1 Osswald *et al.*, 2017

(Osswald, Ieng, Benosman, & Indiveri, 2017) introduced in 2017 a “spiking neural network model of 3d perception for event-based neuromorphic stereo vision systems”, composed of three neuronal populations: the retinal population, the coincidence detectors and the disparity detectors. Retina cells serve as input to the network and project excitatory connections towards the coincidence detectors. The disparity detectors pool the responses from the coincidence detectors *via*

excitatory and inhibitory connections. The network outputs a disparity event only when the spikes produced by a disparity neuron and a corresponding coincidence neuron (i.e. located in a nearby representation in disparity space) take place within a few milliseconds.

This architecture is implemented on an FPGA interfaced with the neuromorphic processor ROLS (Qiao et al., 2015), comprising an array of 256 analog IF neuron circuits and 512×256 dynamic synapses. All in all, this model successfully solves the stereo correspondence problem, as evidenced by the small local average disparity error ϵd, inferior to 1 pixel.

4.3.2.2 Dikov *et al.*, 2017

The "spiking cooperative stereo-matching" model introduced in (Dikov et al., 2017) is composed of a three-dimensional array of cells, where each cell encodes a belief in matching a heterolateral pair of pixels from the left and the right visual scenes from the stereo event stream. Due to intrinsic structural constraints, this model aims to respect the biologically plausible principles devised by (Marr & Poggio, 1976):

— within-disparity continuity, implemented using a weak potentiation between neighbouring cells representing similar disparity and leading to a smooth disparity map ;
— cross-disparity uniqueness, implemented *via* a strong inhibition between cells to allow for the pairing of two corresponding pixels ;
— prevention of the homolateral matching, i.e. the activation of disparity neurons solely due to the activity of one retina.

The corresponding connectivity patterns are implemented *via* excitatory and inhibitory synaptic pictured in Fig. 4.1.

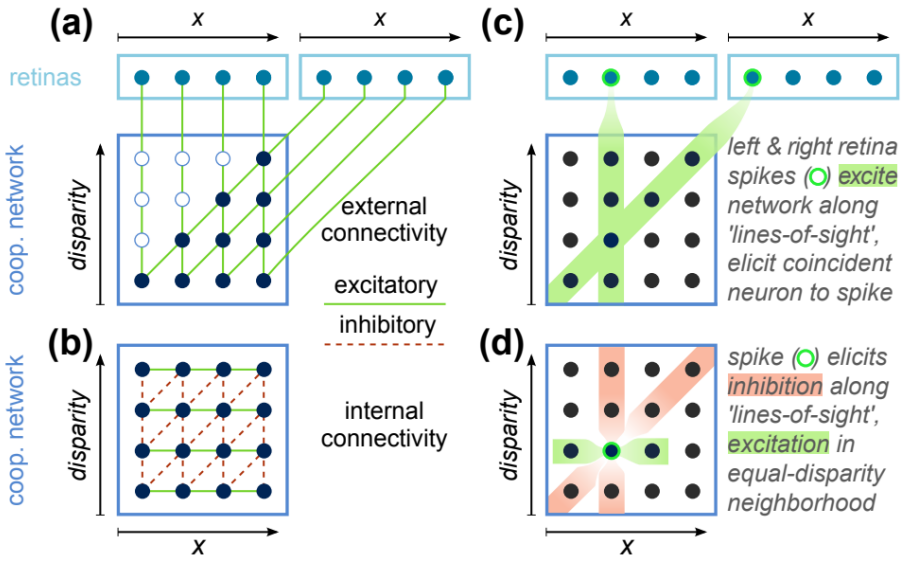


Figure 4.1 – Network topology of the cooperative stereo-matching network introduced in (Dikov et al., 2017). Figure from (Dikov et al., 2017). (a),(b) Connectivity in terms of excitatory and inhibitory connections: neurons/cells (dots) are framed into populations and interconnected by synaptic projections (lines). (c),(d) Inhibition and excitation pattern for an exemplary pair of retinal input spikes (c) leading to a match, i.e. spike, in the cooperative network (d) which in turn triggers internal inhibition and excitation according to Marr's physical constraints.

The third constraint, i.e. the prevention of homolateral matching, leads to the implementation of each cell (black dots in Fig. 4.1) as a micro-ensemble instead of a simple neuron. As depicted in Fig. 4.2, a micro-ensemble is composed of two retina pixels R_r and R_l, one disparity-sensitive collector neuron C and two intermediary blocker neurons B_l and B_r, serving as inhibitory interneurons. When one retina is activated, the corresponding spike excites the collector neuron C as well as the ipsilateral blocker neuron B. Once activated, the latter inhibits the collector neuron C. At the same time, this retinal spike inhibits the contralateral blocker neuron B. The synaptic delays δ between the micro-ensemble neurons are adjusted so that the spike propagation times along the $R \rightarrow C$ and $R \rightarrow B_{\text{contra}} \rightarrow C$ are equal.

This mechanism prevents homolateral matching by providing for two situations:

— first scenario: only one retina is activated, and C must not activate. In this case, the retina spike leads to the activation of C and the end of C inhibition by the contralateral blocker; but this overall activation is hindered by the activation of B, itself inhibiting C which thus cannot spike.

— second scenario: both retinas are activated, and C must produce a disparity spike. In this case, both retinas inhibit their contralateral blockers, thus preventing them from inhibiting C. The latter is now only activated on both sides and can activate.

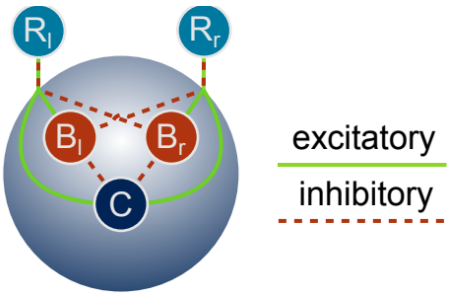


Figure 4.2 – Topology and working principle of neural micro-ensembles to prevent homolateral self-matching, from (Dikov et al., 2017).

This network measures the disparity from the event stream recorded by two parallel stereo event cameras. It is implemented using PyNN (Davison et al., 2009) on the neuromorphic board SpiNNaker (S. B. Furber et al., 2013).

The disparity is contained in a range $[d_{min}, d_{max}]$. In this work, the disparity is strictly positive; since the cameras are parallel, $d_{min} = 0$ corresponds to very distant objects. As the neural micro-ensembles are arranged in each layer according to a trapezoid, and each layer corresponds to one coordinate y, we can calculate the total number of neural micro-ensemble N_{nme} as:

$$N_{nme} = n \times y_{max} \text{ with } n = \frac{w_{baseline} + w_{topline}}{2} \times h \tag{4.3}$$

where n is the number of neural micro-ensembles in each layer and $w_{baseline}$ and $w_{topline}$ are respectively the wide and small bases of the trapezoid, computed according to Eq. 4.4.

$$\begin{cases} h & = d_{max} - d_{min} + 1 \\ w_{baseline} & = x_{max} - d_{min} \\ w_{topline} & = x_{max} - d_{max} \end{cases} \tag{4.4}$$

All in all, knowing that each neural micro-ensemble contains three neurons, the number of neurons N_n in the network is:

$$N_n = 3 \times (x_{max} - \frac{d_{min} + d_{max}}{2}) \times (d_{max} - d_{min} + 1) \times y_{max} \tag{4.5}$$

With a sensor size of $x_{max} = y_{max} = 128$ with a $[0, 127]$ disparity range, the network implementation requires 3.15 million neurons, corresponding to at least 12,288 SpiNNaker cores, 723 SpiNNaker chips or 15 SpiNN-5 boards.

This spiking cooperative stereo-matching network is experimentally validated on three datasets: a synthetic dataset presenting letters in space at various depths, with no jitter or noise; and two experimental datasets, "two fans" and "pendulum", recorded in a controlled environment. In the "two fans" dataset, two identical 3-blade fans with a diameter of $35cm$ are recorded during $10s$ by two event cameras set up with a horizontal offset of $10cm$. The fans are respectively $1.8m$ and $2.8m$ away from the cameras and generate approximately $140,000$ events per second per camera. In the "pendulum" dataset, a pendulum (i.e. a tennis ball suspended on a string) swinging back and forth at around $0.6Hz$ is recorded during $10s$ by two event cameras set up with a horizontal offset of $10cm$. The pendulum swings along the line-of-sight of the left camera and generates around $20K$ events per second per camera. The results are thus obtained on three datasets, one synthetic and two real-world tests, of which one is quasi-stationary and one dynamic.

The network achieves an accuracy of 86% on the synthetic dataset. For the "two fans" dataset, 97% of the output events around the distant fan indicate a disparity $d = 7 \pm 1$ and 06.7% of the events around the proximate fans indicate a disparity $d = 10 \pm 1$. Finally, 62.2% of all events produced when presented to the "pendulum" dataset are within ± 2 pixels of that metric. A systematic error causes this relatively bad matching performance: the network tends to fuse the ball images mirror-symmetrically in this dynamic scene. According to the authors, this behaviour can be counteracted by object-level matching or a simple ordering constraint.

4.3.2.3 Risi *et al.*, 2020

(Risi, Aimar, Donati, Solinas, & Indiveri, 2020) proposes a spiking stereo neural network adapted from the architecture presented above (Osswald et al., 2017). Firstly, each coincidence detector is connected to the retina cells *via* a slow NMDA-like synapse on one side and a fast AMPA-like synapse circuit block on the other. This mechanism demands that both synapses be stimulated rapidly so that the coincidence detector neuron fires. Secondly, the authors implement each cell as a micro-ensemble to prevent homolateral matching, similarly to (Dikov et al., 2017).

This architecture is implemented on an FPGA platform interfacing two DAVIS cameras with three neuromorphic processors DYNAPs (Moradi, Qiao, Stefanini, & Indiveri, 2018). It is experimentally validated on multiple synthetic datasets and in real-time scenarios.

4.3.3 Visual attention

An extensive review of SNN models of visual attention applied to event data can be found in Section 5.1, along with the biological basis of visual attention.

CHAPTER 5

Concepts from neuroscience

This chapter extends Chapter 4 by reviewing applications combining spiking neural networks and event cameras inspired by biology and the associated biological mechanism.

5.1 Visual attention 63
5.1.1 Biological visual attention 63
5.1.1.1 Eye and visual cortex in the context of the visual attention 64
5.1.1.2 Neural mechanisms 66
5.1.1.3 Neuromodulators 66
5.1.2 Neuromorphic visual attention 67
5.1.2.1 Attention mechanism with SNNs 67
5.1.2.2 Attention mechanisms applied to silicon retinas . 68
5.1.2.3 Attention mechanisms with SNNs applied to silicon retinas 69
5.2 Foveation 70
5.2.1 Biological foveation 70
5.2.2 Foveation in computer vision 70
5.2.3 Foveated event-based camera 72
5.3 Delay learning and detection of spiking motifs 72
5.3.1 Biological delay learning 72
5.3.2 Neuromorphic models for delay learning 74

This chapter introduces SNN models applied to event data to solve application tasks inspired by biological behaviours. The corresponding biological behaviour is described in detail; each application is the object of a comprehensive review. The three biological behaviours adapted to neuromorphic computing studied in this PhD are: visual attention, foveation and delay learning.

5.1 Visual attention

Visual attention can be defined as the behavioural and cognitive process of selectively focusing on a discrete aspect of sensory cues while disregarding other perceivable information. This biological mechanism, more specifically saliency detection, has long been used in computer vision to drive the analysis only on relevant parts of images or videos for further processing.

It is important here to differentiate the attention we are interested in with the concept of attention as most often used in AI. Indeed the AI community comprehends attention as a piece of additional information calculated and updated during runtime for each input vector in order to focus learning on the most relevant input elements. This attention mechanism was originally applied to recurrent neural networks such as encoder-decoder models, which tended to give more importance to the last input elements. It was then used as building blocks for the transformer model, where they are now computed in parallel instead of sequentially (Vaswani et al., 2017). Transformers were recently adapted to SNNs with the recent example of Spikformer (Zhou et al., 2023).

What interests us in this work is the attention behaviour observed in biological organisms, as described in further detail below.

To limit the amount of data to be processed, several approaches inspired by primates' visual attention mechanisms (Itti & Koch, 2001; Martinet et al., 2009) have proposed to focus the processing only on the most salient parts of the scene. Indeed, the study of many biological organisms highlighted the importance of limiting the processing of sensory RoIs, thus minimising the energy spent on this task as well as the reaction delay to what is perceived (Tsotsos, 1990). For prey animals, this means a quicker detection of a potential predator and a more efficient escape from danger. Attention was first described as the coalition of "focalisation, concentration and consciousness" by William James in 1890 (James, 1890). In psychology, this mechanism is now defined as the allocation of limited cognitive processing resources on one or a few relevant environmental elements while ignoring others (Anderson, 2005). It can be either subjective or objective, as well as voluntary or spontaneous.

As technologies for sensory information processing expanded, an increasing number of them assimilated this idea for their own purposes. In the DL community, for instance, this concept was converted into a neural network's component weighting features by the task's level of importance. Attention can thus be applied to regions in images, words in text, phonemes in speech, etc. For example, some studies aim to exploit this mechanism in order to optimise CNNs (Jetley, Lord, Lee, & Torr, 2018). We believe that studying the biological aspect of visual attention will allow for the development of new models applicable to event vision.

5.1.1 Biological visual attention

Early work by Jonides (Jonides, 1983) distinguish two stages of visual attention, defined by the type of processing: it can be processed either sequentially when it is concentrated on a specific

area of the visual scene ; or it can be processed in parallel in case of a more uniform distribution over the scene. Multiple models of visual attention have been defined over time, all coexisting with one another. The earlier one is the spotlight model, stating that "attention has a focus, a margin and a fringe" (James, 1890), meaning it does not entirely prevent the perception of elements outside the attended visual area. The zoom-lens model (Eriksen & St. James, 1986), the attentional engagement theory (Raftopoulos, 2009) and the gradient model (LaBerge & Brown, 1989) were all three elaborated between 1986 and 1989 and corresponded to non-exclusive psychological and biological observations in humans. Treisman went one step further and produced the Feature Integration Theory (Treisman & Gelade, 1980), according to which visual searches are done through combined parallel and serial stages ; the blended use of those two processes enables the integration of the global features of a complex visual scene.

Selective visual attention can relate to other well-known bio-inspired concepts. Curiosity can be considered as a driver for visual attention since a novel element catches more of the eye than a known element (Shi, Zhang, & Zeng, 2020). In addition, selective visual attention suffers from the exploration versus exploitation trade-off: should one spend more time studying a specific element with the risk of missing crucial information or losing energy processing the entire visual scene at all times ?

5.1.1.1 Eye and visual cortex in the context of the visual attention

This section outlines what is known about the organs and pathways enabling human vision to review human visual attention.

Human eye and retina Humans have complex eyes, allowing for colour detection and binocular vision, thus depth perception. It is composed of three parts: the outermost layer comprises the *cornea* and the *sclera*, whereas the middle layer contains the iris and muscles. The innermost layer, situated at the back of the eye, is mainly composed of the retina, which permits the detection of light and colours. The retina contains two types of light-sensing cells: the photoreceptor cells (rods and cones) and the photosensitive ganglion cells, emulated by the event cameras (see Chapter 3).

The inside of the human eye is filled with humour which the light rays cross until they reach the retina. They are primarily captured by the *macula*, a retina spot responsible for central colour vision. A subdivision of this macula is the *fovea*, where the cones are more closely packed than anywhere else in the retina. Since cones are responsible for colour discrimination and delicate detail perception, the *fovea* corresponds to the central spot where the vision is optimal in bright light. Hence any attention mechanism will aim to direct the eyes to point the "foveal gaze" towards the interesting visual feature. It should be noted that the small size of the *fovea* limits the overall perception of fine details. The organism overcomes this issue by redirecting the eye in saccades, up to 3 times per second. A similar mechanism could be implemented, allowing for a higher resolution on the spot targeted by the "foveal gaze", thus letting a larger amount of specifically relevant information be processed.

Visual cortex Once the visual information has travelled from the eyes and through the thalamus, it is processed by the visual cortex in the brain's occipital lobe. Each hemisphere has a visual cortex cross-handling the information output by the opposed eye. The visual data is received firstly by

the primary visual cortex (V1); it then journeys through the extrastriate areas V2, V3, V4 and V5 (see Fig. 5.1).

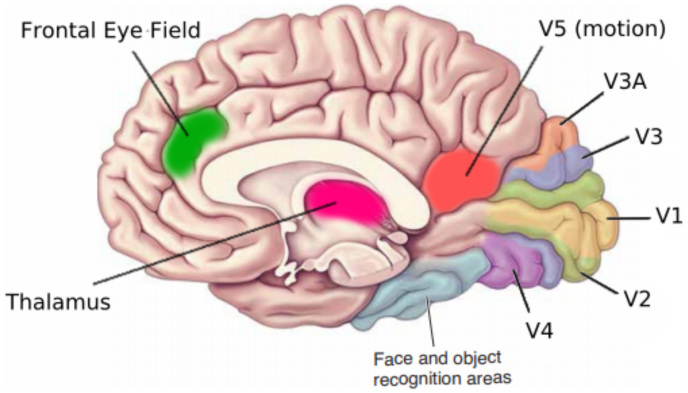


Figure 5.1 – Schematic depiction of the visual cortex, adapted from (Bear et al., 2007).

Located in the frontal cortex, the Frontal Eye Field (FEF) has a role in visually guided saccades' target selection and selective spatial attention (Moore & Zirnsak, 2017). It encodes spatial information into retinocentric coordinates, that is, coordinates established using the retina as a reference. It receives inputs from and projects onto most of the visual cortex.

According to Knudsen *et al.* (Knudsen, 2018), high activity in the FEF brings an increased neuronal response in V4 in a "space-specific, attention-dependant manner". Indeed, the prefrontal cortex integrates the task according to a non-retinocentric frame of reference and translates it into retinocentric signals by sending it to the lateral intra-parietal area. The FEF convert those signals into topographical maps and finally outputs those maps to the retinocentric visual regions of the *posterior cortex* and *superior colliculus*. This may hint at a pathway enabling the influence of saccade-related signals on visual representation. Once understood, this pathway could be implemented as a neural network to specifically direct attention towards the retinocentric space privileged by the FEF.

Forebrain and midbrain networks. It is interesting to note that the forebrain and midbrain networks are strongly involved in attention (Knudsen, 2018). The forebrain selects information based on its relevance to the task at hand or its saliency. It encodes visual information via enhanced distributed representations regulated by heterogeneous dynamics. A similar type of information encoding, with each element interpreted according to its saliency, would speed up its processing significantly.

The midbrain enables spatial attention by monitoring the environment for behaviourally relevant stimuli and then directing the gaze to the region of highest interest thanks to a powerful inhibitory competition. It represents the visual input from the retina and the visual forebrain as a topographic map of space following a retinocentric frame of reference. A midbrain-like neural network would allow for the identification of the data relevant to the task at hand in order to optimise the bandwidth of information to process.

5.1.1.2 Neural mechanisms

Visual attention can be categorized using three dichotomies, differentiated by distinct and divergent neural mechanisms (Moore & Zirnsak, 2017).

The first dichotomy can be defined between *top-down* and *bottom-up* attention. The top-down one corresponds to a selective type of attention depending on a previously set motivation or rule. This "endogenously generated signal" influences visually-driven signals in primates. The bottom-up depends on physical saliency, including brightness, movements and colours. Those two interdependant types of attention interact strongly during visual search, which helps to focus on salient features relevant to the task at hand and disregard the less fitting ones.

A second dichotomy exists between *spatial*, *temporal* and *feature-based* attention. Spatial attention is directed towards a specific location in space thanks to the occulomotor system, which leads to the prioritisation of an area in the visual field. Furthermore, temporal attention focuses on a specific instant in time ; this type is often tapped into for video processing and emphasises critical video frames. It could be particularly interesting to exploit this kind of attention in human action recognition. Finally, feature-based attention selects elements based on their resemblance to a behaviourally relevant object by representing the corresponding features in the prefrontal *cortex anterior* and the ventral *pre-arcuate* (Bichot, Heard, DeGennaro, & Desimone, 2015).

The last dichotomy opposes *overt* and *covert* attention. It respectively corresponds to the presence (overt) or absence (covert) of motor commands leading to saccadic eye movements. Those two types of attention are simultaneous and complementary: overt attention comes from orientating eye movements, actively guided by salient features determined thanks to concurrent covert attention. When attention comes to orientating movements (overt), the *posterior parietal cortex* activity increases. In addition, the ventral stream visual area is actively involved in guiding this motion according to salient features determined thanks to concurrent covert attention. This is confirmed by the observation of a significant temporal correlation between the visual processing of targets and eye movement (Moore & Zirnsak, 2017). Interestingly, covert attention is the one most often studied by visual neuroscientists. It improves the detection and discrimination of features (Chevallier & Tarroux, 2008) both at the fovea and in the visual periphery, thanks to a visual enhancement in V4 and the inferior temporal cortex.

5.1.1.3 Neuromodulators

Neuromodulators are a class of extraneuronal molecules released within cortical and subcortical structures which allow and influence the signal transmission between neurons. According to Moore and Zirnsak (Moore & Zirnsak, 2017), three neuromodulators have a particularly relevant role in attention: acetylcholine, dopamine and norepinephrine.

Acetylcholine is synthesised by the *nucleus basalis* of Meynert, the substantial *innominata* and the basal forebrain. When collected by a certain type of receptors present on the neuron surface (metabotropic muscarinic or ionotropic nicotinic receptors), it can enhance selective visual attention. Moreover, the release of acetylcholine seems to play a role in the enhanced processing of sensory information: its increase triggers glutamate release from retinal ganglion cells. This release of extracellular molecules by those photosensitive cells boosts their effect and amplifies the visual data captured from the corresponding retinal space.

Another neuromodulator of interest is dopamine: synthesised by midbrain *nuclei*, it appears to alter the strength and reliability of converging glutaminergic synapses in the prefrontal cortex,

thus affecting the FEF and selective attention. A link has been established between this molecule and Attention Deficit Hyperactivity Disorder.

Also implicated in this disorder, norepinephrine is involved in the selective response to salient sensory stimuli, depending on the relevance of the stimuli for the task. It is synthesised by the *locus cæruleus*.

Much like dopamine pathways were studied and imitated when developing various models of reinforcement learning, it would be highly appealing and relevant to tailor those neuromodulators' mechanisms in order to enhance computational models applied to computer vision tasks.

5.1.2 Neuromorphic visual attention

Applications of visual attention to computer vision have been studied for the past two decades. However, the possible impact of SNNs and event-based cameras on this domain has been little studied so far. In the following section, we establish a review of state-of-the-art attention mechanism models applied to event camera data and/or using SNNs.

Itti and Koch presented a complete overview of bottom-up visual attention models in 2001 (Itti & Koch, 2001). They define saliency as a bottom-up type of attention, operating very quickly and independently to the nature of the task. Most of the computer vision domain work on saliency relies upon this definition.

In 2014, Le Callet and Niebur reviewed the various existing applications of visual attention to multimedia technologies (Le Callet & Niebur, 2013). Their convincing synthesis of the domain emphasises the benefit of attention particularly regarding "multimedia delivery, retargeting and quality assessment of image and video, medical imaging, and the field of stereoscopic 3D images applications". It strengthens our conviction that applying attention mechanisms to computer vision, using SNNs and event data, is a highly enticing line of scientific inquiry.

5.1.2.1 Attention mechanism with SNNs

The use of SNNs instead of traditional artificial neurons has been prioritised in the modelling of attention for some years, maybe due to their common bio-inspiration. Chevallier *et al.* (Chevallier, Cuperlier, & Gaussier, 2010) compared the relevance of their use for this task and concluded in favour of spiking neurons over traditional ones, following their previous work on attention in the exploration of a prey-predator environment (Chevallier, Paugam-Moisy, & Lemaître, 2005).

SNNs have thus been used to model different types of attention mechanisms applied to various tasks. For instance, Katayama *et al.* (Katayama, Yano, & Horiguchi, 2004) used SNNs to implement overt attention through selective visual attention with gaze shift. This model used two layers: one representing the virtual cortex and the other the hippocampal formation. The correlation of firing times of spikes output by these two layers determines three states: the attention state, the non-attention state and the shift of attention.

Another application of SNNs is the modelling of covert attention (visual attention without eye movement): Chevallier and Tarroux (Chevallier & Tarroux, 2008) use SNNs to extract saliency and focus on the attention of a moving stimulus. This is implemented using a neural filter, a saliency map and a focus map. The filter thresholds enforce a convolution on low and high spatial frequencies. The first map gathers information from visual features on different spatial frequencies thanks to synchrony detectors. The second map finally allows covert attention using a self-

connection mask. In this model, saliencies are temporally coded and arise in hierarchical order, close to the forebrain representation of the visual scene. (Thakur et al., 2017) implement a neuromorphic model of visual saliency in RGB data on digital hardware, using stochastic computation and achieving very low power consumption. They believe their model to be useful for "information triaging, compression and analysis in computer vision tasks". The neuromorphic dynamic visual saliency model introduced in (Molin, Thakur, Niebur, & Etienne-Cummings, 2021) is bottom-up, feed-forward and built on the concept of proto-objects, i.e. low-level elementary features formed rapidly and in parallel in the visual cortex. The authors compare their results to the ground truth obtained from human eye prediction as well as implement their model on a hybrid FPGA to enable real-time applications.

Chick *et al.* (Chik, Borisyuk, & Kazanovich, 2009) defined a model for selective visual attention applied to the sequential selection of objects. This top-down, temporal, feature-based attention uses a 2-layer architecture of inhibitory, excitatory and peripheral neurons. This SNN model used two types of inhibition to implement respectively attention focus and shift thanks to short-term plasticity.

SNNs can also allow for object detection with saliency indexing, mimicking top-down attention. Such a model has been implemented by Wu *et al.* (Wu, McGinnity, Maguire, Cai, & Chen, 2013), in the form of three successive stages: a network inspired by the biological retina extracts the low-level image features, which are then decomposed into multiple visual pathways. Finally, "*attention area maps*" similar to the retinocentric output of the FEF are created, approximated by spike rate maps and enabling the detection of RoIs.

A model of interest has been defined by Bernert and Yvert (Bernert & Yvert, 2019) with an application to spike sorting, a common pattern recognition problem in neurosciences. Spike sorting consists of detecting action potentials emitted by biological neurons, thanks to the classification of the patterns present in the membrane potential measured in said neurons. This unsupervised network reproduces overt temporal attention after a short learning period with little data: an attention neuron modulates intermediate and output layers according to a combined short-term plasticity, hysteresis and threshold adaptation mechanism.

Finally, since curiosity is one of the main drivers of attention, we can mention the work recently introduced in (Shi et al., 2020). To perform object recognition on MNIST data, the authors regard attention as the novelty of a stimulus. Thus their SNN model learns by estimating the novelty of the visual samples and updating those samples when the novelty exceeds a certain threshold. According to the authors, the attention mechanism takes place in one of the cerebral pathways responding to curiosity: more specifically, it follows the pathway starting from the ventral tegmental area and reaching the hippocampus *via* the prefrontal cortex and the *precuneus*.

5.1.2.2 Attention mechanisms applied to silicon retinas

The processing of event-based data often takes place *via* the adaptation of computer vision models initially dedicated to classic RGB visual information. One example is given by Cannici *et al.* (Cannici, Ciccone, Romanoni, & Matteucci, 2018), who designed two visually attentive models for event-based data. One aims to locate RoIs using event activity within the field of view, while the other is based on the *Deep Recurrent Attentive Writer* neural model. This model was designed in 2015 by Gregor *et al.* (Gregor, Danihelka, Graves, & Wierstra, 2015) to generate complex images and emulate the foveation of the human eye, thus implementing a spatial attention mechanism. Another example of adaptation from RGB to event data is given by the evProto model introduced

in (Iacono et al., 2019): the authors extend the algorithm for proto-object detection and spatial-scale estimation in RGB data introduced in (Russell, Mihalas, von der Heydt, Niebur, & Etienne-Cummings, 2013), based on border ownership — which reacts to edges potentially belonging to a border — and grouping — strongly activating a region enclosed by several object borders. (Ghosh et al., 2022) further develops the previous work into a depth-based attention model by combining it with a biologically inspired stereo disparity estimation algorithm.

Feng *et al.* (Feng et al., 2020) implemented an algorithm applied to denoising of event data: the background of a scene recorded by a DVS camera is denoised thanks to the detection of event density in a spatiotemporal neighbourhood, which can be assimilated to covert visual attention.

5.1.2.3 Attention mechanisms with SNNs applied to silicon retinas

Works addressing visual attention by combining SNNs and event data are rare in the literature.

Recently, some authors have studied the combination of SNNs used on event-based data in order to convey an attention mechanism. An example of such a combination is given by Bogdan *et al.* (Bogdan et al., 2019), where they develop an unsupervised decomposition of elementary motion detectors. According to them, "fast localised motion detection is crucial" for an efficient visual attention mechanism.

Additionally, neuromorphic saliency detection models applied to event data are most often than not derived from existing models implemented with traditional neural networks and applied to RGB or grayscale images: (D'Angelo, Perrett, Iacono, Furber, & Bartolozzi, 2022) is for example adapted from (Iacono et al., 2019), itself originally adapted from (Russell et al., 2013). Another noteworthy example is given by the model recently introduced in (Lele, Fang, Anwar, & Raychowdhury, 2022), where the authors combine CNN and SNN applied respectively on frames and event data to achieve embedded target localisation in a prey/predator scenario. Both networks are implemented on a drone and their fusion outperforms CNN-only processing in realistic 3D virtual environments. The CNN is trained for prey detection on the intensity image reconstructed from the embedded DVS, while the SNN allows for the cancellation of events generated from the drone ego-motion. This combination is inspired by insects and primates' biological behaviours: their vestibular sensors interact with the vision "to cancel the activity corresponding to the predator's self-motion".

Only a few models genuinely take advantage of the intrinsic dynamics of SNNs and the uniqueness of event data. One popular bioinspired mechanism applicable to the task of saliency detection is the Dynamic Neural Field (DNF), a spatiotemporal mathematical description of activity patterns in neuronal populations in the visual and motorcortex (Sandamirskaya, 2014). Renner *et al.* (Renner et al., 2019) make use of this mechanism as a soft WTA to implement saliency tracking of pre-activated objects, by manually increasing the activity in a pre-defined region. While neuromorphic, this approach is not applicable in real scenarios where the RoI is not known beforehand. DNF was also used by Rasamuel *et al.* (Rasamuel, Khacef, Rodriguez, & Miramond, 2019) to extract a saliency map from combined DVS and standard camera, with an application for real-world exploration and exploitation. Renner *et al.* also proposed a combined use of SNNs and silicon retina, applied to object tracking (Renner et al., 2019): they implemented a recurrent SNN emulating the activity of large homogeneous populations of biological neurons. *Via* a soft WTA, this model thus adopts the attractor dynamics of DNFs.

5.2 Foveation

5.2.1 Biological foveation

The majority of biological visual sensors have very irregular sensor grids. For example, some insects have a peripheral vision grid used for navigation, while another is specialised to a very specific location in the visual space and dedicated to mating behaviour. Most predatory mammals have a central visual area on the retina with a high density of photoreceptors and the ability to move their eyes, and thus the localisation of this area of high acuity around the centre of the gaze. This action is driven by visual attention mechanisms (see previous Sec. 5.1.1) and can be learned by means of a saccadic mechanism (Daucé, Albiges, & Perrinet, 2020) that has been shown to optimise the efficiency of information gain at each saccade (Daucé & Perrinet, 2020). The wide variety of anatomical configurations illustrates that this arrangement is closely related to the behavioural repertoire of each animal (Land, 2018), and the introduction of such an irregularity, which we refer to here as *foveation* for simplicity, can provide significant improvements in information processing in event cameras.

As described in Paragraph **Human eye and retina**, foveation is a visual process in which the eye directs its focus to a particular point of interest, typically by moving or adjusting the position of the eye. In the human eye, the fovea is the central area of the retina responsible for sharp and detailed vision. By directing the fovea towards the point of interest, the visual system can acquire more information and process it with higher accuracy and resolution.

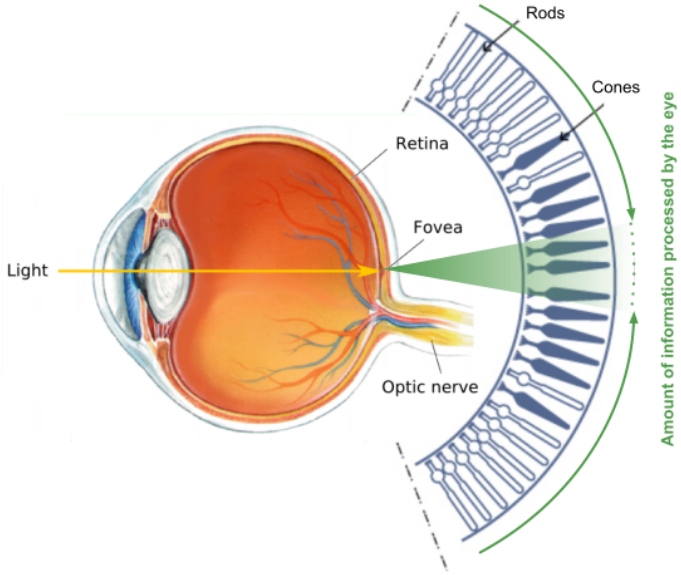


Figure 5.2 – Biological foveation mechanism, adapted from (Bear et al., 2007).

5.2.2 Foveation in computer vision

Foveation is important in various applications, such as image and video compression, where selective compression is applied to regions of low visual acuity, and in robotics, where foveated sensors can reduce the amount of data and processing required for visual tasks.

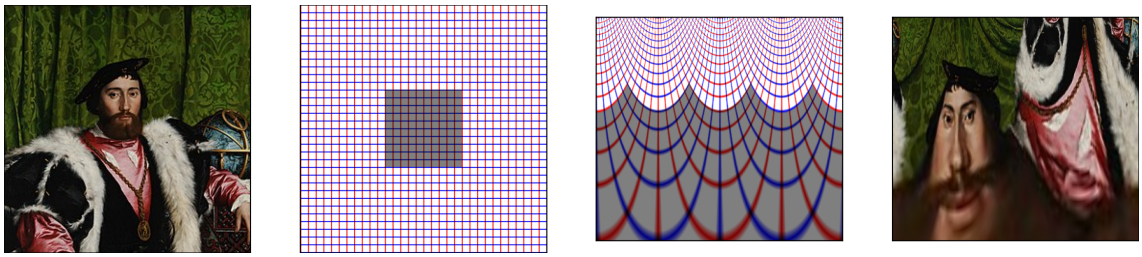


Figure 5.3 – Visual representation of retinotopic foveation, from (Perrinet, 2023). “A picture (extract from the painting "The Ambassadors" by Hans Holbein the Younger) can be represented on a regular grid represented by vertical (red) and horizontal (blue) lines. Retinotopy transforms this grid, and in particular, the area representing the fovea (shaded gray) is over-represented. Applied to the original image of the portrait, the image is strongly distorted and represents more finally the parts under the axis of sight (here the mouth).”

The concept of foveation has been explored in (Geisler & Perry, 1998) in the late 1990s with a foveated multiresolution pyramid with the same objective to reduce the amount of data transmitted in remote vision tasks such as vehicle teleoperation, surveillance, or telemedicine. While a pointing device was used in their experiments to select the foveated area, authors recommend a gaze tracker like in (Martinet et al., 2009) as a preferable solution to select areas of interest. It has been demonstrated that the retinotopic representation of images (as described in Fig. 5.3) is beneficial in Spatial Transformer Networks among others “to learn CNN to be invariant to affine transforms” (Dabane, Perrinet, & Daucé, 2022).

Most artificial sensors, such as the CMOS chip that powers the cameras in the average smartphone, have evenly spaced sensors. This is also the case for all event cameras. This is an optimal choice when considering the trade-off between the increasing miniaturisation of each pixel and the growing demand for higher image resolutions. However, the introduction of an irregularity such as the one described above, which we refer to here as *foveation* for simplicity, can provide significant improvements in information processing in event cameras.

A foveated sensor that mimics the retinotopic layout of a biological retina will generally use fewer pixels than a conventional sensor and will therefore be more energy efficient. Indeed, this would allow us to maintain a high amount of meaningful information while significantly reducing the amount of raw data to be processed. In the particular case of neuromorphic architectures such as SpiNNaker or Intel Loihi, the power consumption is directly proportional to the number of spikes/events processed, and reducing this number is an efficient way to reduce power consumption, a heavy constraint for embedded applications. However, we believe that the development of a mechanism mimicking foveation would greatly improve the processing of event-driven data beyond the energy efficiency aspect. Indeed, recent studies have shown that the particular geometrical layout of the log-polar mapping observed in the human retina has several advantages (Hao et al., 2021). In particular, a rotation or a zoom is transformed into translations into a log-polar mapping (Traver & Pla, 2003). Further rotation and zoom-invariant processing can for example be easily implemented in a CNN, thus improving the classification performance (Jérémie, Daucé, & Perrinet, 2023). Moreover, foveation can lead to several improvements in image compression (Araujo & Dias, 1997), in the efficiency of image registration (Sarvaiya, Patnaik, & Bombaywala, 2009), or for object recognition (Pramod, Katti, & Arun, 2022). Yet, it is still not known whether such foveation could be beneficial for event-driven imaging.

A mechanism akin to neuromorphic foveation was introduced by D'Angelo and colleagues in 2020 (D'Angelo et al., 2020) and drew inspiration from the Spiking Elementary Motion Detector, which detects optical flow in event data, as well as from varying sizes of receptive fields in the mammalian retina. D'Angelo and colleagues extend this work in order to "decrease the computational resources required, by filtering the number of incoming events into the visual field while maintaining a fine resolution in the centre of the visual field", which comes close to the foveation definition. However, they extrapolate the visual attention from the centre of the optical flow — which may not contain all the salient elements of the visual scene.

5.2.3 Foveated event-based camera

The introduction of foveation at the sensor level for event-driven sensors should reduce the energy and bandwidth consumption from the sensor level up to the computing system by dynamically allocating more bandwidth and pixel resolution to the RoIs and reducing the information of peripheral (non-interesting) regions. Recently, an electronically foveated DVS has been proposed (Serrano-Gotarredona et al., 2022) where DVS pixels can be dynamically configured in high-resolution foveal regions or grouped into low-resolution regions with arbitrary sizes. The sensor can attend in parallel to an arbitrary number of foveal high-resolution regions. Furthermore, the electronic reconfiguration of foveal regions makes it faster and has lower energy compared with the configuration of the centre of the RoI using mechanical control of the sensor position.

5.3 Delay learning and detection of spiking motifs

As described in Chapter 2, an SNN is constructed using populations of neurons linked together with connections, with respect to a certain architecture and certain rules allowing it to learn a behaviour. These rules often concern the evolution of the synaptic weight between two pre- and post-synaptic neurons, i.e. the amount of neurotransmitters released in the synapse following the pre-synaptic neuron's excitation. Hebb's rule can be mentioned: "cells that fire together wire together" (Hebb, 1949).

Delay learning is another of those learning rules which, instead of changing the synaptic weight, changes the delay of the electrical spike propagation in the pre-synaptic neuron's axon (Hüning et al., 1998). Depending on the definition, the delay refers to either the time a spike takes to travel through the presynaptic axon, the postsynaptic dendrites or both.

We present below a short review of biological delay learning and neuromorphic models implementing this bio-inspired learning. A more complete review of this topic can be found in our review article (Grimaldi et al., 2022).

5.3.1 Biological delay learning

The first evidence of any neuronal delay in the information propagation within the animal brain came from the interaural time difference, allowing for the azimuthal localization of sound by barn owls. According to Gerstner in 1996, there is a true paradox in auditory neural systems since "neural networks encode behaviourally relevant signals in the range of a few μs with neurons that are at least one order of magnitude slower" (Gerstner, Kempter, van Hemmen, & Wagner, 1996). Various biological experiments have thus revealed the existence of a biological axonal delay precisely adjusted according to variations of parameters in the brainstem (with our without

application of plasticity); one of them, the "Jeffress model" is presented in Fig. 5.4. We can also mention the Reichardt detector, a model inspired by the drosophila's elementary motion detector for basic motion detection (Reichardt & Rosenblith, 1961; Franceschini, Ruffier, & Serres, 2009) whose mechanism is presented in Fig. 5.5.

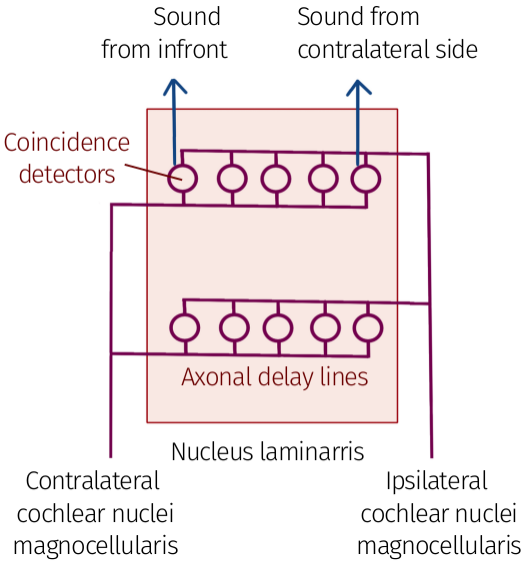
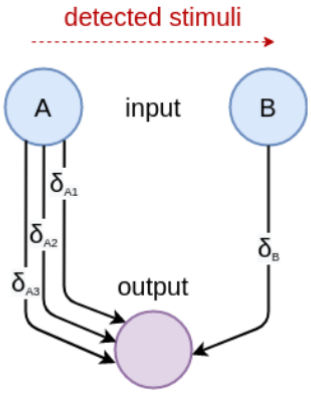


Figure 5.4 – Schematic showing the Jeffress model for sound localization as applied to chicken and barn owl, adapted from (Kubke & Carr, 2006). Neurons located in the dorsal edge of the *nucleus laminaris* respond maximally to sounds originating from far contralateral space, and neurons located in more ventral positions respond maximally to sounds originating from the front.



with δA1 >> δA2 >> δA3 > δAB

The output neuron connected to the output with...	δA1	δA2	δA3
...fires when the speed of the detected stimuli is	slow	moderate	fast

Figure 5.5 – Reichardt detector model as introduced in (Reichardt & Rosenblith, 1961). Two input neurons A and B connect to a common output. The input from B and A are delayed according respectively to a value δ_B and comprised in $[\delta_{A1}, \delta_{A2}, \delta_{A3}]$. If the stimulus to detect, moving from A to B with a time $t_{stimulus}$, is timed with those two connections so that $\delta_{Ax} = \delta_B + t_{stimulus}$, then the spikes will reach the output at the same time and the model will fire.

Over and over, myelin has been identified as one of the parameters mentioned above. Indeed, this multilaminar coating formed by the glial cells in the Vertebrates' nervous system facilitates both the neural circuit function and the behavioural performance (Fields, 2015). Experiments on mammals show that myelination is directly related to learning and memory consolidation, especially motor training and in enriched environments, both at an early age and in older animals, due to its involvement in coupling the activity of distant neuron populations. It is worth highlighting that myelination becomes increasingly important in larger brains where conductance delays are substantial, and brainwave rhythms are critical; synchrony errors can lead to neuropsychiatric and neurological dysfunctions. Some further studies show that oligodendrocytes may detect neuronal activity thanks to growth factors or neurotransmitters released through ion channels or via exocytosis. Fig. 5.6 outlines the biological mechanism behind myelination and its impact on learning.

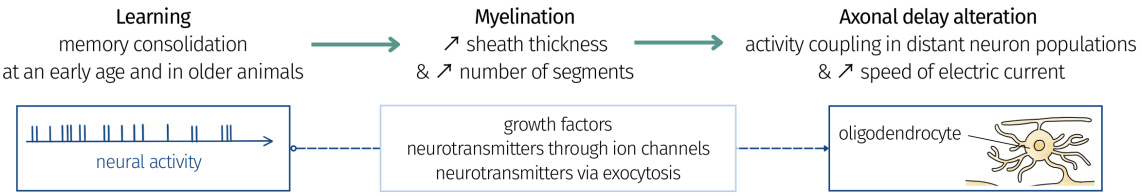


Figure 5.6 – Biological mechanism at the origin of delay learning *via* myelination.

5.3.2 Neuromorphic models for delay learning

Delay learning is a striking example of a computer science concept effectively reinforced by ongoing neuroscience work. Indeed, more and more SNN models are being developed with an impetus to learn by updating delays, not just synaptic weights. One convincing instance among many is given by (Paugam-Moisy, Martinez, & Bengio, 2008), which combines delay adaptation and polychronisation for reservoir computing using an STDP and a supervised delay learning algorithm. (Wright & Wiles, 2012) presents a Spike-Delay-Variance Learning which adapts the shape of the postsynaptic profiles according to temporal error. (Kerr, Burkitt, Thomas, Gilson, & Grayden, 2013) selectively potentiates recurrent connections with different axonal and dendritic delays during oscillation activity. (Matsubara, 2017) implements an unsupervised delay learning algorithm adjusting conduction delays and synaptic weights.(Nadafian & Ganjtabesh, 2020) proposes an STDP extended to the delay learning repeating spatiotemporal patterns described in further detail in the paragraph below. More recently, (Grimaldi, Besnainou, Ladret, & Perrinet, 2022) introduces a supervised SNN model akin to a Reichardt detector learning polychronous groups, i.e. repeated activations of prototypical spiking motifs ; this work was then extended in (Grimaldi & Perrinet, 2023).

Bio-plausible unsupervised delay learning (Nadafian & Ganjtabesh, 2020) introduce a non-supervised learning mechanism of synaptic delays in an SNN model composed of four convolution layers with a kernel size of 5×5 and with shared weights and delays (i.e. all the weights and delays of one convolution layer is updated homogeneously). The network is given as input a set of stimuli formed by a dot moving diagonally during 5 timesteps randomly in either one of four directions (northeast, north-west, south-east and south-west - see Fig. 5.7), with additional noise. With such an architecture and input, each convolution layer specialises in the recognition of one specific pattern among the four given as input (see Fig. 5.8, thanks to four unsupervised learning rules described below.

The authors propose an STDP updating both weights and delay as follows:

$$\begin{aligned} \Delta t_{i,j} &= t_j - t_i - d_{i,j} \\ \Delta\delta_{i,j} = G(t_{i,j}) &= \begin{cases} -B_- \times \exp \frac{-\Delta t_{i,j}}{\sigma_-} & \text{if } \Delta t_{i,j} \geq 0 \\ B_+ \times \exp \frac{\Delta t_{i,j}}{\sigma_+} & \text{if } \Delta t_{i,j} < 0 \end{cases} \end{aligned} \tag{5.1}$$

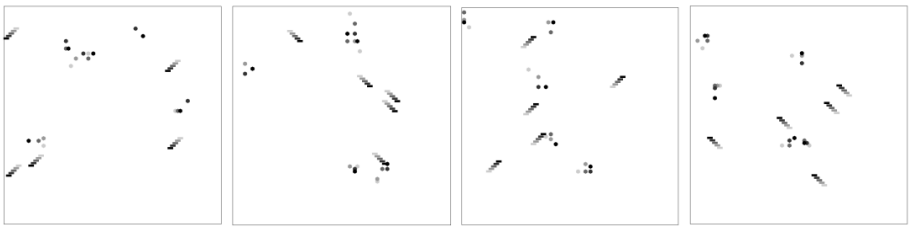
where i and j are respectively the pre and post-synaptic neurons, t_x is the spiking time of the neuron x, $d_{i,j}$ is the synaptic delay between i and j, $\Delta t_{i,j}$ is the delay variation after STDP, B_+ and B_- are the parameters controlling the value of synaptic delay reduction and increment and σ_+ and σ_- are the STDP time constant.

Secondly, they implement a homeostasis mechanism based on the neural activity: the activity of each neuron is compared to a targeted frequency; in case of low activity, their weight will be decreased and delay increased to approach this objective (and *vice versa* on the contrary) according to the following equation:

$$\begin{aligned} K(t+1) =& \frac{R_{\text{target}} + R_{\text{observed}}}{R_{\text{target}}} \\ \delta(t+1) =& \delta(t) - \lambda_\delta \times K(t+1) \\ \omega(t+1) =& \omega(t) - \lambda_\omega \times K(t+1) \end{aligned} \tag{5.2}$$

where R_{target} and R_{observed} are respectively the targeted and observed activation frequencies, $\omega(t)$ is the current weight, $\delta(t)$ is the current delay and λ_ω and λ_δ are the learning rate of homeostasis mechanism, respectively applied to weights and delays. Additionally, the thresholds are linearly adapted according to the neural activity: the threshold of a neuron is increased when it spikes and decreased otherwise.

Finally, they include in their model a stop condition, developed to avoid the convergence of all delays to zero (corresponding to the biological hyper-myelination) and the loss of any temporal feature in the learnt pattern. If one of the synaptic connections of a neuron has a delay δ inferior to the constant c (with $c > B_-$), the learning is stopped and the STDP is no longer applied to any synaptic connections of this neuron.



(a) $-135°$ *angle* **(b)** $-45°$ *angle* **(c)** $45°$ *angle* **(d)** $135°$ *angle*

Figure 5.7 – Four samples of the dataset employed in (Nadafian & Ganjtabesh, 2020). The intensity of each dot shows the location of it over time (black indicates the most recent location).

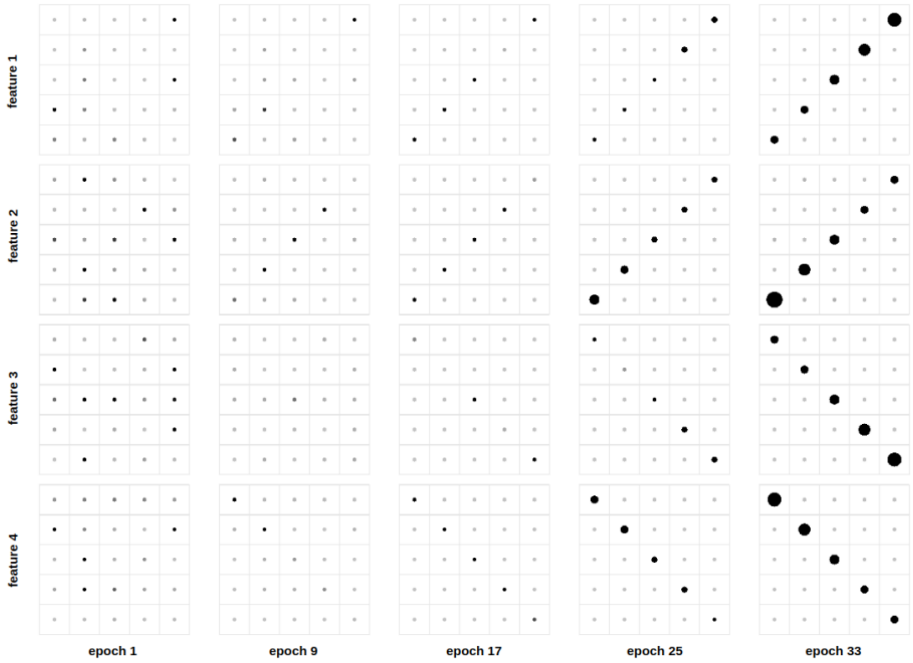


Figure 5.8 – Weights and delays of each convolution at different training epochs, from (Nadafian & Ganjtabesh, 2020). The size (intensity) of each circle represents the delay (weight) of its corresponding synapse, where the bigger circle illustrates lower synaptic delay (black indicates the higher synaptic weight).

Contributions

CHAPTER 6

Event data downscaling

This chapter presents our work on embedded event data downscaling. Indeed, embedded systems face limitations in terms of energy resources, memory, computational power, and communication bandwidth. Hence, the increasing use of event cameras in embedded computer vision tasks calls for a way to reduce the amount of events to be processed while keeping the relevant information for the task at hand. We thus believe that a formal definition of event data reduction methods will provide a step further towards sparse data processing. Ten methods will be introduced in this chapter, achieving either spatial or temporal downscaling in order to reduce the number of events in the dataset. For each, we will study the trade-off between their capacity to handle events in real-time, the amount of information kept after reduction and the performance of a computer vision task on the processed data.

6.1 Introduction 81
6.2 Spatial downscaling 82
6.2.1 Spatial funnelling 82
6.2.2 Log luminance reconstruction 83
6.2.2.1 Event count 83
6.2.2.2 Linear interpolation 84
6.2.2.3 Cubic interpolation 85
6.2.3 Events to video to events 89
6.2.4 SNN pooling 90
6.2.4.1 Separate handling of polarity 90
6.2.4.2 Mutual influence of positive and negative events . 91
6.2.4.3 Influence of hyperparameter tuning on SNN pooling 91
6.2.4.4 Real-time data downscaling on SpiNN-3 93
6.3 Temporal downscaling 94
6.3.1 Temporal funnelling 94
6.3.2 Structural downscaling 95
6.4 Experimental comparison 96
6.4.1 Comparison according to activity measures 97
6.4.2 Comparison of real-time downscaling 100
6.4.3 Comparison according to performance on a classification task 103
6.4.3.1 SLAYER benchmark classifier 103
6.4.3.2 PLIF-based classifier 103
6.5 Human assessment study 106
6.5.1 Human machine interface and protocol 106
6.5.1.1 Event data 106
6.5.1.2 Interface 108
6.5.1.3 Protocol 108
6.5.2 Assessing human performance 109
6.5.3 Comparing human and machine performance 112
6.6 Conclusion 114

6.1 Introduction

Neuromorphic computer vision and event sensor-based applications are receiving increasing interest from the computer vision community (classification, detection, tracking, segmentation, etc.), especially for robotics or autonomous driving scenarios.

Downscaling event data is an important feature in a system, especially if embedded, so as to be able to adjust the complexity of data to the available resources such as processing capability and energy consumption. Reducing the size of visual data is straightforward for grayscale and RGB images with interpolation. When it comes to event data, size reduction is much less trivial: event downscaling is a real challenge in neuromorphic computer vision and needs to be formalised as a step further towards sparse data processing.

To this end, we designed different methods to reduce the number of events in a dataset or in an event stream. We decided to tackle this problem by reducing the spatial dimension of the sensor, or by reducing the grain size of the data recording time. In total, we implemented ten event data downscaling methods: six reducing the spatial resolution (designated as "spatial" methods) and four temporally reducing the event flow ("temporal" methods). We aim to design methods which can handle events in real time so that they can be deployed on an embedded system and reduce the input flow of events as they come to ease their processing. Additionally, the spatial methods are developed so that the reduced events correspond to the events that would be produced by an event camera of smaller dimensions than the original, recording the same scene.

In order to study the impact of spatial and temporal downscaling, we compare several features of the resulting data, such as the total number of events, event density, information entropy, *optical consistency*, computation time and performance of a computer vision task as assessment criteria.

We also carried out what is to the best of our knowledge the first-ever human perception study with event data, where we assess and compare human performance on a gesture classification task. The implementation of this study was the subject of a work placement by three Master students, Ms Lucía Trillo Carreras, Ms Marina Bueno Garcia and Ms Ewa Kupczyk, carried out under our supervision. The contributions of this study are 3-fold: (1) we establish a size threshold under which the human performance falls behind the chance level, (2) we compare several spatial and temporal event downscaling methods and show that all methods give unequal data quality, and (3) we highlight some unexpected discrepancies in a comparison between human vs machine performance.

To the best of our knowledge, the proposed methods make the first attempt to formalise event data downscaling. The methods were implemented in Python [*].

The results presented in this chapter were published on three occasions, in the proceedings of the VISAPP 2022 conference (spatial funnelling and log-luminance reconstruction methods — (Gruel, Martinet, Serrano-Gotarredona, & Linares-Barranco, 2022)), the WACV 2023 conference (neuromorphic spatial methods, i.e. SNN pooling — (Gruel, Martinet, Linares-Barranco, & Serrano-Gotarredona, 2023)) and the CVPR 2023 Workshop on Event-based Vision (temporal methods and human assessment — (Gruel, Trillo Carreras, Bueno Garcia, Kupczyk, & Martinet, 2023)).

∗. The Python code is available online at https://github.com/amygruel/EvVisu

6.2 Spatial downscaling

The following subsections present the six methods that we designed and implemented to downscale spatially the data produced by an event camera. It should be noted that a seventh method is introduced but quickly discarded as it does not meet our criterion for embedded data downscaling: indeed it cannot be performed online and requires previous training of an ANN model (see Section 6.2.3). Each method takes as input a sample of events at maximum resolution as well as the reduction factor for this sample and returns the downscaled events as illustrated in Fig. 6.1. The temporal dimension is not affected by the reduction. Depending on the downscaling method used, the number and frequency of the downscaled events will not necessarily equal those of the events occurring in the corresponding fullscale regions. These specific methods are particularly relevant for ML models limited in memory and bandwidth (i.e. limited in terms of the spatial resolution of the data they handle) or tasks requiring multi-scale resolution — see Chapter 8 and the work conducted in (D. Gehrig & Scaramuzza, 2022)).

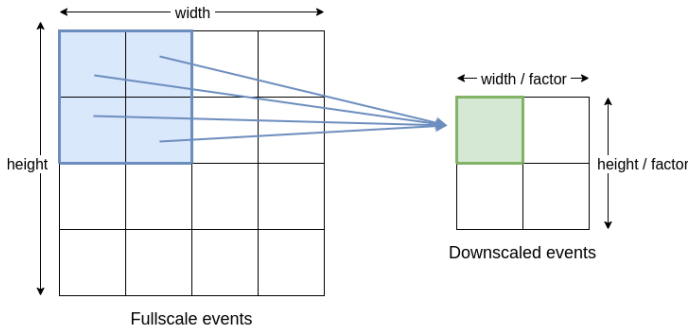


Figure 6.1 – A set of fullscale events captured by a sensor of size height $\times$ width is r according to a downscaling factor div, set to 2 in this example. Here, all the events of the original sample occurring in the blue area's pixels will correspond to the events occurring in the green pixel after reduction.

6.2.1 Spatial funnelling

A first intuitive solution to reduce event data is to simply divide each event's coordinates in x and y by the downscaling factor, then remove all the reduced events duplicates (i.e., keep only one occurrence of each event for a specific set of coordinates (x, y, t, p) – see Fig. 6.2). This method follows the same logic as *Tonic*'s downscaling approach, with the slight difference that *Tonic* keeps duplicates while our implementation discards them.

From a computational point of view, this downscaling method consists simply of updating the memory address of the event's x, y coordinates. Since this relies on one elementary operation repeated n times, with n the number of events processed, it has a complexity $O(n)$.

Thanks to its quickness and simplicity, this solution could answer the need for downscaling events in real time on embedded systems. However, the main drawback is that this process does not drop any event (except the duplicates). The reduced area undergoes a high density of events, yielding an information loss due to the accumulation of pixel activation: the precision of (x, y) coordinates decreases and the fine details of the recorded objects are lost (see Fig. 6.2). Spatial funnelling drastically increases pixel activity (see Fig. 6.2), especially when the downscale factor

is significant. In the extreme yet unlikely case where the data is downscaled to a 1×1 target size, the resulting pixel is likely to be constantly active. Finally, the mechanism does not correlate with the physical behaviour of an actual event camera: the optical consistency is very low for this method.

6.2.2 Log luminance reconstruction

From an optical point of view, downscaling the spatial size of an event recording boils down to averaging the luminance captured by a subset of pixels in the original sensor. In a physical implementation of an event camera formed by a 2D array of pixels as the one shown in Fig. 3.2, this method could be implemented by averaging the photocurrents I_{ph} generated by neighbour pixels and performing the trans-impedance logarithmic conversion and posterior voltage amplification and differentiation (as explained in Chapter 3). In order to implement this behaviour, the log-luminance is reconstructed for each pixel in the full-scale dataset: each event occurring on the same pixel increases (if positive) or decreases (if negative) the log-luminance level by one unit. The mean log-luminance is then computed for each region of size $factor^2$ in the sensor. Lastly, the downscaled events are generated: an event occurs at every timestep where the average log-luminance crosses a threshold line ; the event is positive if the gradient is positive, and vice-versa.

To achieve a pixel's log-luminance reconstruction, all events from this pixel are translated into points of coordinates (x shows the timestamps, y is the level reached) where the initial level is arbitrarily set to 0 and the contrast threshold set to 1 (see Fig. 3.3).

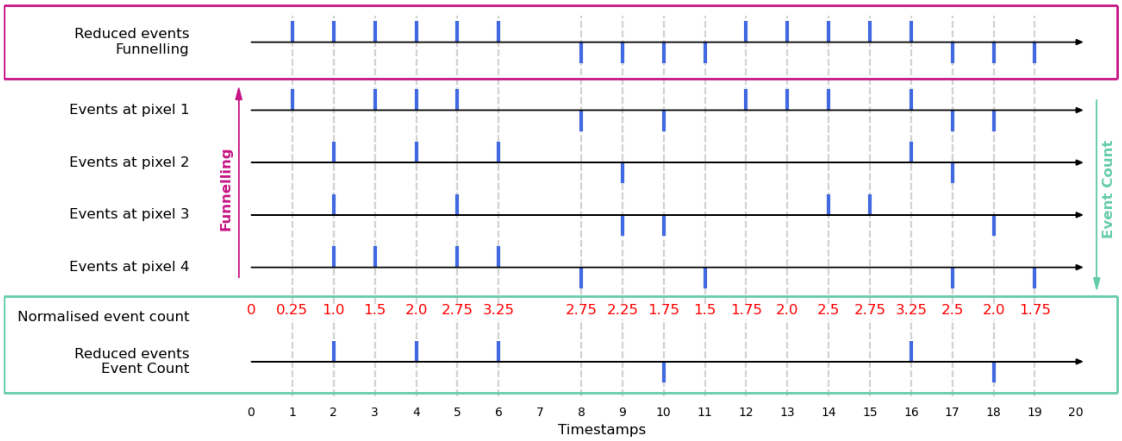


Figure 6.2 – Schematic illustration for spatial funnelling and event count methods. A sensor of size 2×2 outputs negative and positive at each pixel across time, which are downscaled using spatial funnelling (upper red frame) and log-luminance event count (lower green frame). In this example, the downscaling factor is set to 2.

6.2.2.1 Event count

A first approach to log-luminance reconstruction consists in estimating the mean log-luminance value reached by a downscaled pixel every time an event occurs in the corresponding fullscale region. In other words, we realise a normalised event count in the fullscale region as

presented in Fig. 6.2. If its value crosses a contrast threshold, an event is produced; an additional value keeps track of the slope in order to set this event's polarity depending on the slope. It is also necessary to handle the constraint posed by the fact that an event is only produced if the entire threshold has been crossed since the last event. The slope value allows us to ensure this by identifying the extrema, thus preventing events from occurring when the log-luminance crosses a threshold directly following an extremum. Therefore, no event is produced at timestamps 8,13 et 17 in Fig. 6.2, consistently with the native behaviour of an event camera.

Since the complexity of this method relies on the number n of events, it is therefore also $O(n)$.

This approach's main disadvantage is the loss of neuromorphic asynchrony. A reduced event can only occur at the timesteps where a full-scale event took place, which would not be the case with actual log-luminance reconstruction, as depicted in the following paragraphs. This drawback is also an asset: this reduction method can easily downscale events online as they are being recorded, and its algorithm simplicity is a plus for an embedded system.

6.2.2.2 Linear interpolation

The second approach consists in actually reconstructing the log-luminance curve as a polyline, where each event in the log-luminance graph is linked to its neighbours in time by a line as presented in Fig. 6.3. This allows for a quick computation and interpolation of the intersections between the thresholds and the curve.

However, it is not as faithful to the physical reality as we would like it to be: the log-luminance curves are reconstructed as polylines and not cubic curves similar to the actual behaviour of log-luminance recorded by an event camera. Furthermore, it has one disadvantage that it shares with the cubic estimation approach described below: both those approaches cannot be computed strictly in real time since the interpolation requires knowing the future behaviour of the curve, i.e. the next events that will occur on the monitored pixel, which are obviously unknown at the current time. In real-time scenarios, these methods require an adaptation: we propose to work around this problem by inserting a slight delay between the data recording and processing.

The proposed adaptation strategies describe the behaviour of the reconstructed log-luminance curves when no input event is given at a time t:

— *constant* strategy: the reconstructed curve remains constant until new events are triggered, follows the equation $y = c$ with c a constant equals to the log-luminance value reached last (see Fig. 6.4);

— *interpolation* strategy: the reconstructed curve is interpolated (linear/cubic) given the slope at the location of the last triggered event when it crossed the previous log-luminance threshold (see Fig. 6.5).

Note that the *interpolation* strategy presented in Fig. 6.5 leads to negative values, which is physically incoherent. Indeed, the log-luminance curve recorded by an event camera would not go into the negative (as it is not physically sound) by would have a minimum bound of 0. In our implementation, we made the choice to allow for negative values for the two following reasons:

1. the curves are interpolated for an "unseen future", i.e. for an unknown time up until the arrival of an event at the corresponding pixel. We thus expect the next event to arrive quickly enough for the curve not to go too low into the negatives.
2. the curves are interpolated using a null log-luminance value at the origin. Since we do not know when the event camera began to record, this value is arbitrary and we could decide to set it at a much higher value, thus preventing the issue of interpolated negative

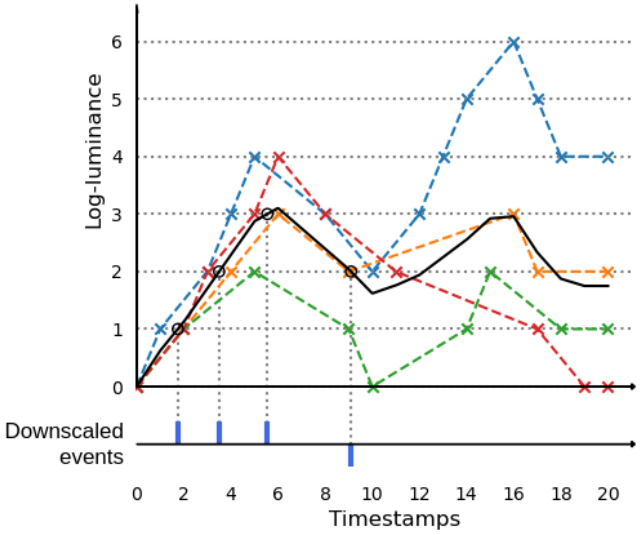


Figure 6.3 – Schematic illustration for linear log-luminance reconstruction. A sensor of size 2×2 outputs negative and positive events at each pixel across time (crosses in blue, yellow, red and green, each colour corresponding to one pixel), which are downscaled using log-luminance reconstruction with linear interpolation. The black line corresponds to the downscaled log-luminance values, which are computed as the average of the log-luminance values recorded by each of the 4 original pixels. In this example, the downscaling factor is set to 2.

log-luminance value. The only information we need is the time at which the interpolated curves cross the thresholds; here we set the cross with the origin at 0 to facilitate the reader's comprehension.

This note also concerns the *interpolation* strategy applied to the cubic lo-luminance reconstruction described below (Fig. 6.8).

6.2.2.3 Cubic interpolation

In this last approach, the log-luminance of each neuron is interpolated as a cubic curve strictly running through the events on the log-luminance graph as seen in Fig. 6.6. The cubic interpolation was implemented using *Scipy*'s method *PChipInterpolator*, an implementation of the Piecewise Cubic Hermite Interpolating Polynomial. This algorithm is a third-degree piecewise polynomial function which seeks to match the derivatives of each point with its neighbours. Thus its extrema coincide with the data's extrema, which is not the case for all cubic interpolation algorithms that tend to greatly overshoot (Rabbath & Corriveau, 2019). Similarly to other cubic interpolation algorithms, this cubic interpolator I takes as input a set of values (x, y, x') and returns the y' data interpolated for all x' values with respect to the relation between y and x:

$$I(x, y, x') = y' \text{with } x' = \sum_{x=\lfloor \frac{x_{min}}{threshold} \rfloor + threshold}^{\lfloor \frac{x_{max}}{threshold} \rfloor} (x) \tag{6.1}$$

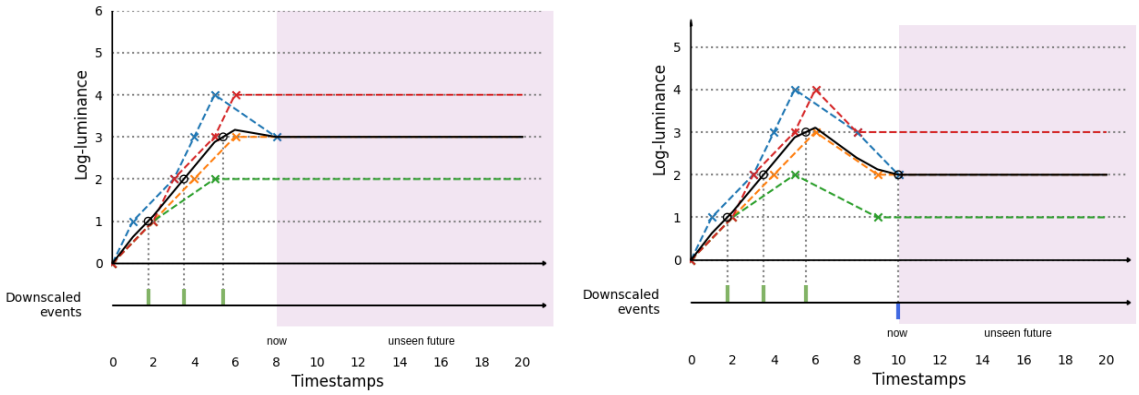


Figure 6.4 – Illustration of the *constant* strategy applied to linear log-luminance reconstruction. The input events and downscaling factors are similar to those used in Fig. 6.3.

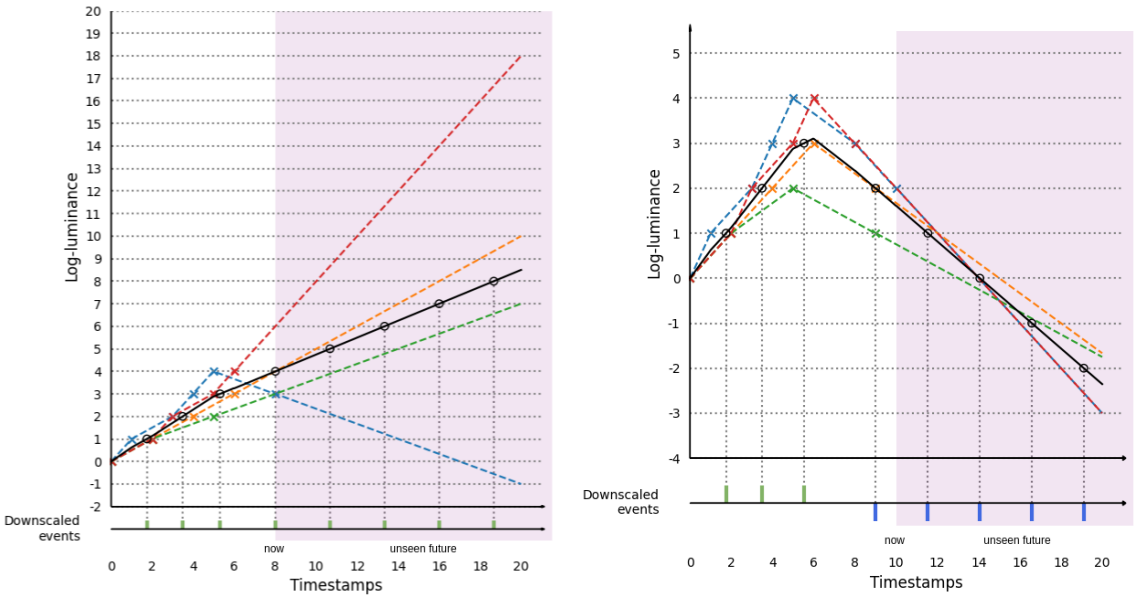


Figure 6.5 – Illustration of the *interpolation* strategy applied to linear log-luminance reconstruction. The input events and downscaling factors are similar to those used in Fig. 6.3.

In our case, the coordinates are those of the events in the corresponding log-luminance graph: the x-axis corresponds to the timestamps and the $y - abscisse$ to the events' log-luminance. From a set of known log-luminance levels x', the cubic interpolation outputs the timestamps y' where the events' log-luminance crosses those levels.

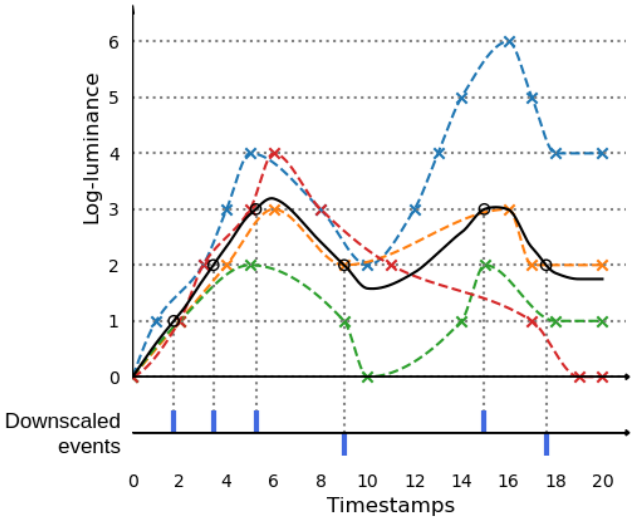


Figure 6.6 – Schematic illustration for linear log-luminance reconstruction. A sensor of size 2×2 outputs negative and positive at each pixel across time, which are downscaled using log-luminance reconstruction with cubic interpolation. In this example, the downscaling factor is set to 2.

This approach reconstructs a curve close to an actual log-luminance curve recorded by an event camera, granting a high optical consistency to this approach. Indeed, this method emulates the real-time signal averaging among neighbouring pixels that is implemented in hardware in the electronically foveated DVS proposed by our collaborators at IMSE (Serrano-Gotarredona et al., 2022). That way, the log-luminance reconstruction generates the output event flow that most closely resembles the output event flow obtained by the electronically foveated DVS real hardware observing the same scene driven by the same attention mechanism.

However, the existing tools allowing the cubic interpolation of y values according to a set of x parameters require the x set to be strictly monotonous, without any duplicates. The first hinder can be removed by splitting the (x, y) dataset presented in Eq. 6.1 at the different log-luminance's extrema:

$$S_i(x, y) = \sum_{i=0}^{n_{ext}} \sum_{t=t_{y_{ext_i}}}^{t_{y_{ext_{i+1}}}} (x_t, y_t) \tag{6.2}$$

with S_i the i^{th} subset of the events comprised between two extrema, n_{ext} the total number of extrema in the dataset i.e. the number of segments S_i, and $t_{y_{ext_i}}$ and $t_{y_{ext_{i+1}}}$ respectively the timestamps of the i^{th} and $(i+1)^{th}$ extrema.

The second hindrance mentioned above is more complex to avoid: although it is a rare occurrence, multiple events still happen at the same coordinates and in the same timestamps. This

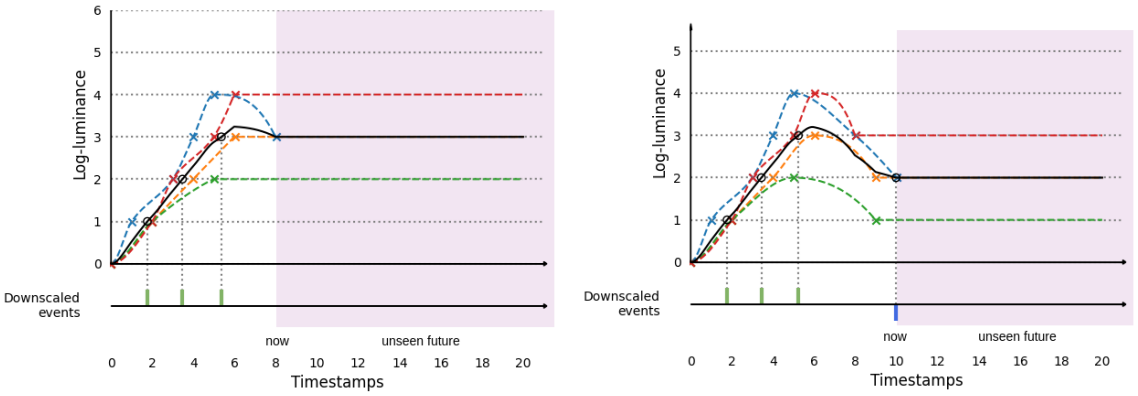


Figure 6.7 – Illustration of the *constant* strategy applied to cubic log-luminance reconstruction. The input events and downscaling factors are similar to those used in Fig. 6.6.

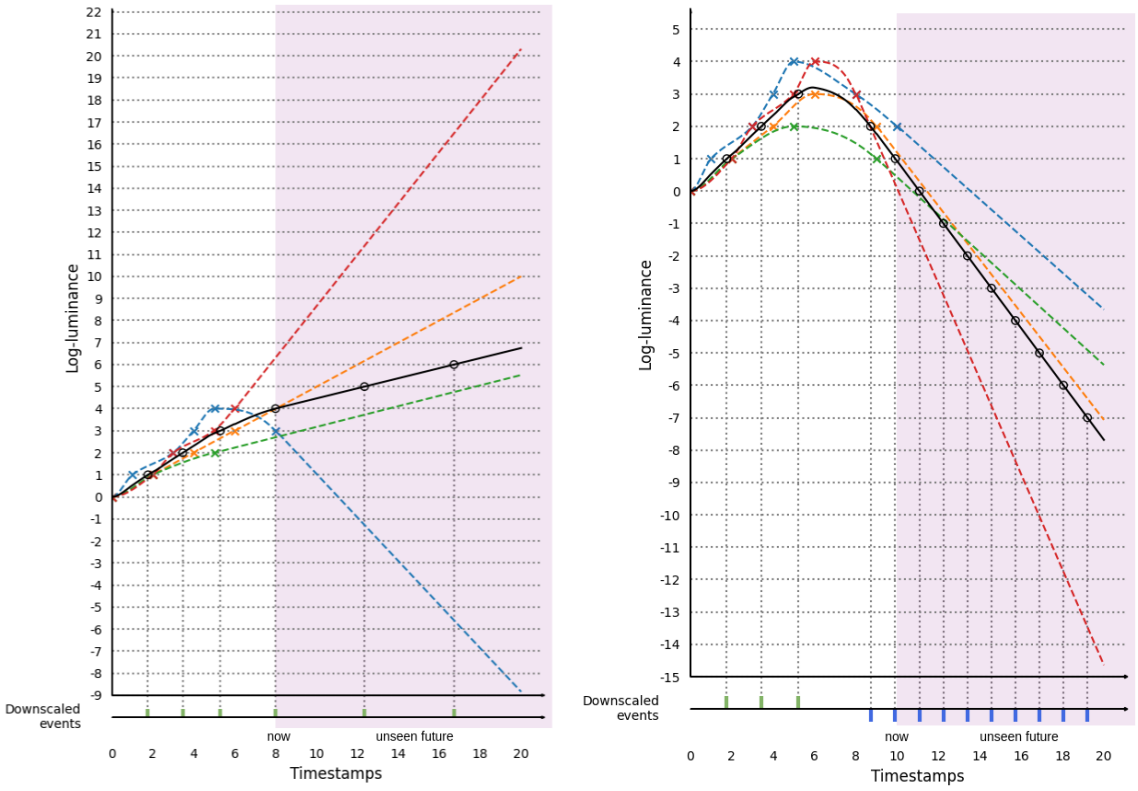


Figure 6.8 – Illustration of the *interpolation* strategy applied to cubic log-luminance reconstruction. The input events and downscaling factors are similar to those used in Fig. 6.6.

requires us to only keep one specimen of duplicates, thus lightly skewing the data processing.

The handling of event data in real-time by the cubic log-luminance interpolation method follows the strategies described for linear log-luminance reconstruction: the behaviour of the *constant* strategy is described in Fig. 6.7 and the behaviour of the *interpolation* strategy in Fig. 6.8.

6.2.3 Events to video to events

Rebceq and colleagues introduced in (Rebecq et al., 2019b) a method to reconstruct grayscale videos from an event stream using a recurrent neural network trained on a large amount of simulated event data. Using this model, a simple technique would be to use standard image downscaling methods on grayscale frames reconstructed from the event data and then translate this reduced image back into events.

This is equivalent to reconstructing a downscaled luminance and computing downscaled events from it, but it has some important drawbacks. The main one lies in the complete loss of event data's asynchronicity since the grayscale video produced from the event data is built from the accumulation of information into frames. This method needs two sets of grayscale data to be created during the process (fullscale from events, then downscale) leading to high memory consumption, while embedded systems are limited in data storage. Moreover, its implementation in practice is computationally heavy, which is not suited for embedded applications. Lastly, the chance of errors in event data reduction is quite high due to the two approximations and one interpolation required by this method.

A pipeline implementing such a downscaling process is the following:

1. translate the event data into grayscale frames thanks to *Events-to-Video* library introduced in (Rebecq, Ranftl, Koltun, & Scaramuzza, 2019a). This module actually consists of a ML model trained on a set of datasets comprising neuromorphic and grayscale recordings of the same scenes.
2. rescale the grayscale frames thus obtained with a standard interpolation using one of the many existing image processing tools.
3. finally translate the downscaled grayscale frames into events by using the library *Video-to-Events* from (D. Gehrig, Gehrig, Hidalgo-Carrió, & Scaramuzza, 2020). This library reconstructs the log-luminance variations between each grayscale frame, and then outputs the corresponding events according to user-defined negative and positive contrast thresholds.

Some works implement multi-scale resolution of event data using part or all of the pipeline described above. For example, the authors in (D. Gehrig & Scaramuzza, 2022) implement steps 2 and 3 to assess the evolution of the performance of various computer vision tasks.

This pipeline requires the event dataset one wishes to downscale either to be one of the datasets the *Events-to-Video* model has been trained on or to contain a set of grayscale frames recorded simultaneously as the event data. Since this method is very arguable considering the motivations for event downscaling, we did not pursue it further during this work. Note that the dataset used in our experiments does not meet either of these two criteria (see Section 3.3).

6.2.4 SNN pooling

This subsection presents two novel SNN architectures for spatial event data downscaling by SNN pooling, introduced in (Gruel, Martinet, Linares-Barranco, & Serrano-Gotarredona, 2023). Here, our core objective lies in designing a fully integrated energy-efficient SNN model able to run dynamically on a neuromorphic platform. We claim the importance of providing downscaling methods for event data in order to filter out events and reduce the data flow with care, as this preprocessing step should be achieved while preserving the information conveyed. The design of such a model is not straightforward, and it requires some hyper-parameter tuning to achieve the results presented below. As far as we know, it is the first time an SNN has been studied as a method to downscale event-based data. Knowing that the combination of SNNs with event-based cameras seems to prove its worth in terms of compatibility of data types (Cordone et al., 2021), information retention (Hagenaars, Paredes-Valles, & de Croon, 2021), energy efficiency (Cordone et al., 2021 ; Debat et al., 2021) and processing latency (Debat et al., 2021) reinforces the relevance of studying event data reduction using SNNs.

In some ways, this event processing has often been used by SNN models handling neuromorphic data: indeed the use of a stridded convolution or max pooling layer e.g. in (Cordone et al., 2021) is close to event downscaling by SNN pooling, as we present here. The events are directly translated as input spikes in a 2D layer of size height $\times$ width connected to a smaller 2D layer of size $\frac{\text{height}}{\text{ratio}} \times \frac{\text{width}}{\text{ratio}}$, which implements a convolutional layer with a kernel size $ratio \times ratio$, a stride $ratio$, without padding. Each spike output from this smaller layer is then interpreted as an event, with each neuron construed as a pixel of a smaller sensor. The connection between each output neuron and the corresponding input region is fully connected (i.e., all the spikes produced by the region are sent to the same neuron). As we will describe in the following paragraphs, we implemented two SNN architectures depending on whether the positive and negative events are considered separately or not.

Both methods were implemented in this work using the Python SNN simulator PyNN (Davison et al., 2009) to produce downscaled datasets for further comparisons. However, since this library runs simulations on CPU, it leads to a significant run time. We thus adapted the networks to be executed on the SpiNNaker neuromorphic chip (S. Furber & Bogdan, 2020): we chose SpiNNaker as a simulator instead of NEST (which runs on CPU) and update the implementation of the fully connected connections to be comprehended by the sPyNNaker library.

6.2.4.1 Separate handling of polarity

A first method of SNN pooling is proposed, where the positive and negative events are handled differently, although during the same simulation. The architecture used is depicted in Fig. 6.9: the 2D input layer is of size height $\times$ width $\times$ 2 with two 2D grids of LIF neurons stacked together, each one receiving either positive or negative events. Likewise, the output layer is of size $\frac{\text{height}}{\text{ratio}} \times \frac{\text{width}}{\text{ratio}} \times 2$; each of its LIF neurons is wired to the corresponding input region by an excitatory connection. No learning is necessary here: the weight ω of the connection between the input and the output is set after fine-tuning phase (as is the sensitivity of an event camera recording a scene — see below) and should not be adapted along the simulation. Spatially downscaling using SNN pooling relies entirely on the LIF neuron's intrinsic dynamics, which come close to the behaviour of a DVS pixel (see Chap. 4).

Most SNNs taking events directly as input, with no intermediate preprocessing of the data, translate each of them into a spike at the corresponding coordinates in the input 2D layer of the network with no distinction between positive and negative events ; any time a pixel is triggered, a spike is emitted by the corresponding input neuron (see (Cordone et al., 2021)). This may not have an impact on some computer vision tasks, but in our case we aim to reduce event data while keeping the information as close to the original one as possible, thus preserving polarity. From this stems our choice for the particular architecture described above.

However, separately handling the event polarities disregards the physics behind event cameras and embodies the main flaw of this first architecture. In the case of a burst of negative and positive events grouped together and recorded by the same pixel, the luminance is actually quite stable and varies around one value with a variation range slightly higher than the sensor's sensitivity threshold. As can be seen in Fig. 6.10, downscaling this data should lead to a smaller luminance variation range, thus to fewer events produced since they compensate each other.

6.2.4.2 Mutual influence of positive and negative events

To resolve the aforementioned problem posed by separate processing of polarities, we implemented a second architecture illustrated in Fig. 6.9 (including the grey dotted box). It builds on the structure of the first one and adds to it a mechanism of mutual inhibition between the polarities. Each receptive field of the input layer activates the corresponding output neuron of the same polarity, but it also inhibits the corresponding output neuron of the opposite polarity with the same weight. This way each activation of the output neuron's membrane potential due to an event of a certain polarity is offset by any potential event of the opposed polarity taking place in a close time span to obtain the expected behaviour presented in Fig. 6.10.

This represents more faithfully the behaviour of an event camera of a smaller resolution, recording the same luminance as the sample being downscaled. It should also be noted that the sensitivity of this simulated recording of a downscaled luminance can be adapted simply by adjusting the weights of the connections, as easily as adjusting the sensitivity of a DVS: the stronger the weights, the higher the activation of the output neurons and the more events produced, and reciprocally. Both methods described above will respectively be referenced to as "separate SNN" and "mutual SNN" methods in the following pages.

6.2.4.3 Influence of hyperparameter tuning on SNN pooling

As with most SNNs, hyperparameter tuning greatly influences the activation of the different layers and the overall spiking dynamic. Hyperparameters include the synaptic weights, the neurons' thresholds, the membrane time constant, etc (see Section 2.1.3). This hyperparameter tuning is not inconsistent with the physical operation set up of the DVS: indeed in a DVS camera, different parameters can be set to record a scene, such as the contrast threshold or the refractory period.

It was thus necessary to study the activity measures of both separate and mutual architectures described in the previous section with varying hyperparameter tuning in order to select the most optimal one. As seen earlier, many hyperparameters can be tuned in order to obtain the ideal results. Here, we limit our study to the synaptic weight ω (see Fig. 6.9) since we wish to simulate the behaviour of a smaller event sensor and the weight can be assimilated to a DVS contrast threshold. The rest of the hyperparameters are set to values commonly used to initialise a LIF

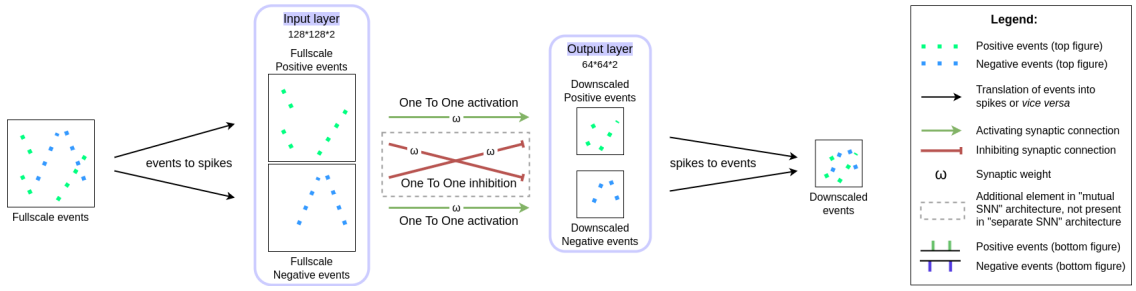


Figure 6.9 – Architectures of separate and mutual SNN downscaling methods. The two architectures proposed by the authors for SNN Pooling either handle the positive and negative events separately (excluding the grey dotted box in the figure) or take into account their mutual influence (including the grey dotted box in the figure). The green arrows represent One-To-One excitatory connections between an input receptive field and the corresponding output neuron of the same polarity, and the red arrows All-To-All inhibitory connections between different polarities. The synaptic weight ω is homogeneous. The green dots correspond to positive events and the blue dots to negative events. The first step of this model is to translate these input events into input spikes while separating the polarity ; the output spikes are translated back into events in the final step. In this example, both architectures are applied to data recorded with a sensor size of 128×128 pixels, which is downscaled by a factor of 2. The output layer of each polarity is thus of size $\frac{128}{2}^2 \times \frac{128}{2}^2 = 64^2 \times 64^2$.

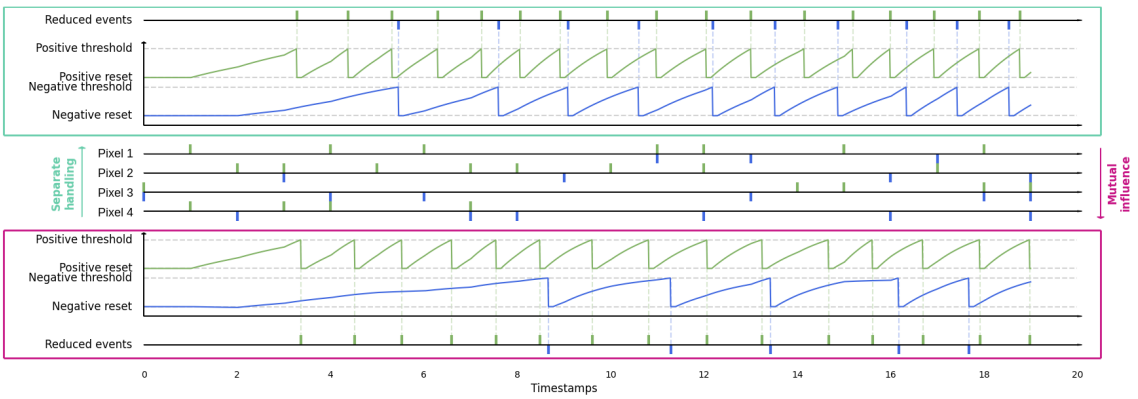


Figure 6.10 – Schematic illustration for SNN pooling. A sensor of size 2×2 outputs events both negative (blue ticks) and positive (green ticks) at each pixel across time, which are downscaled with SNN pooling handling separately the events (green frame on top) or allowing a mutual influence (pink frame at the bottom). In this example, the downscaling factor is set to 2. Best seen in colours.

neuron in an SNN (presented in Tab. 6.1). As pictured in Fig. 6.11, a strong ω will allow for a stronger activation of the output layer, thus letting more events through the connections and leading to a higher output event density. If we consider the standard deviation of the density per pixel as a measure of information provided by the event data, we could set an upper and lower bound for the hyperparameter in order to ensure the production of suitable downscaled data (i.e. which output number of events is smaller than the input, but still high enough to provide information, and which output spatial density is proportional to the input one). Conversely, a lower bound should also be provided in order to keep enough events in the output to have a minimum of information. This is coherent with the contrast threshold used to set up a DVS camera: the higher the threshold, the more events are filtered and the less information the event data provides. Hyperparameter tuning is therefore an important aspect to keep in mind when using SNNs to downscale events.

Fig. 6.11 also interestingly shows that the mutual SNN tends to output fewer events in a lower temporal density than separate SNN, which will be confirmed in Fig. 6.14. Indeed the mutual inhibition tends to normalise the membrane potential of each neuron (see Fig. 6.10), which can be assimilated to the log luminance recorded by each sensor's pixel.

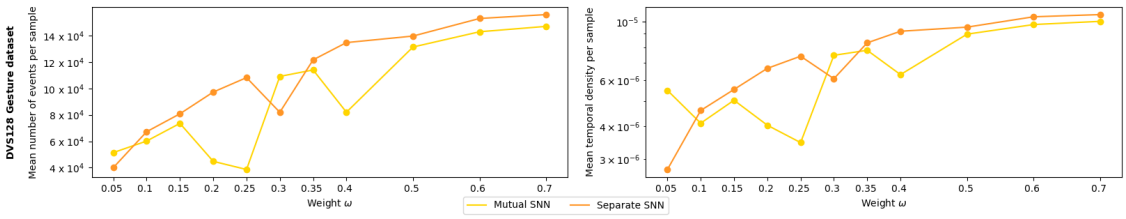


Figure 6.11 – Comparison of activity measures according to a varying synaptic weight ω for separate and mutual SNN downscaling methods applied to DVS128 Gesture.

Parameter	value
Resting membrane potential	-65 mV
Reset membrane potential	-65 mV
Neuronal threshold	-50 mV
Membrane time constant	20 ms
Refractory period	0.1 ms
Excitatory decay time	5 ms
Inhibitory decay time	5 ms

TABLE 6.1 – Hyperparameters used to initialise the separate and mutual SNN downscaling methods (see description of each variable in Section 2.1.3).

6.2.4.4 Real-time data downscaling on SpiNN-3

In order to investigate the relevance of the SNN downscaling method for real-time event handling, we implemented both SNN downscaling methods on the SpiNN-3 board (see Section 2.2.2). The run time of this method thus suffers from the limitations of the SpiNN-3 board. The number of input events per simulation time-step does not influence the simulation run time, contrary to the number of neurons and connections. As described in Eq. 6.3, the proposed architecture is convolutional, meaning that the number of connections c depends on the number of neurons n_{input} and

n_{output} in both layers as well as on the convolutional kernel size $kernel$. The latter is actually the same as $ratio$ in the proposed topology: $kernel = ratio$. Interestingly, as described in Eq. 6.3 below, for a given $width$ and $height$, the number of connections c does not depend on $ratio$, whereas $ratio$ strongly influences the number of output neurons to be simulated on the board.

$$\begin{aligned} n_{input} &= 2 \times \text{height} \times \text{width} \\ n_{output} &= \frac{n_{input}}{ratio^2} \\ c &= kernel^2 \times n_{output} = n_{input} \end{aligned} \tag{6.3}$$

Technically, PyNN uses a *SpikeSourceArray* (resp. a *SpikeInjector*) data structure for off-line (resp. online) neurons that do not need to be simulated by the SpiNN-3 board. This important detail means that although the architecture requires $n_{input} + n_{output}$ neurons, the board will only need to simulate n_{output} neurons and their c connections. Furthermore, the SpiNN-3 board does not handle the definition of connections over a smaller group of neurons than the whole population, which hinders the implementation of the connections in our models. This issue was bypassed using the *FromListConnector* which allows to set the weights between each neuron of the input and output layer in an All-To-All connection, thus enabling the convolution by setting some synapses weight to zero (University of Manchester, 2021). However, this solution requires the implementation of synapses that will not be used during the simulation ; this reduces the overall amount of available synapses that can be implemented on the SpiNN-3 (which upper bound is set to 1,000 synapses according to Section 2.2.2) and might prevent scaling up to larger sensors. Depending on whether the separate or the mutual SNN downscaling version is used, the number of connections varies ; indeed the mutual method adds an inhibitory connection between neurons corresponding to positive events in input to those corresponding to negative events in output and reciprocally. Thus the number of connections c doubles between the separate and mutual SNN methods.

Considering all this information, the simulation run time should decrease when the reduction factor $ratio$ increases for both SNN downscaling methods, while the mutual SNN computation time is likely to remain higher than the separate SNN. This is confirmed below in Section 6.4.2 and Fig. 6.17.

6.3 Temporal downscaling

In this section, we introduce four temporal downscaling methods, reducing the data flow by sub-sampling events and with no impact on the spatial resolution. These methods are adapted to scenarios that highly differ from the ones where a spatial downscaling method would be used (i.e. in cases where a model would not have enough memory to handle an input layer of the sensor size): we believe their use is especially relevant for embedded models, whose memory and computing power may not be sufficient to record and process all events at the time of their arrival, but which do not necessarily have a limit on spatial resolution.

6.3.1 Temporal funnelling

The temporal funnelling method comes close to *Tonic*'s temporal "Downsample" method (Lenz et al., 2021). Similarly to the spatial funnelling method we implemented (see Section 6.2.1), it involves the pooling of events according to their timestamp rather than spatial coor-

dinates and removes any duplicates. Event data is expressed with a μs temporal precision. Pooling them according to their timestamp amounts to keeping only one event of the current polarity at the current pixel in the time window defined by a temporal factor t (see Fig. 6.12), which leads to changing the temporal precision.

We establish two types of temporal funnelling (see Fig. 6.12):

— asynchronous temporal funnelling: each event kept per pixel, per polarity and per time-window maintains its original timestamp, with a thinner temporal grain than the one defined by t.

— synchronous temporal funnelling: each event is kept but with a timestamp rounded to the end of the current time window set by t. This aims to illustrate the behaviour of an embedded system which would accumulate the kept events into frames once all the events of the corresponding time window are processed.

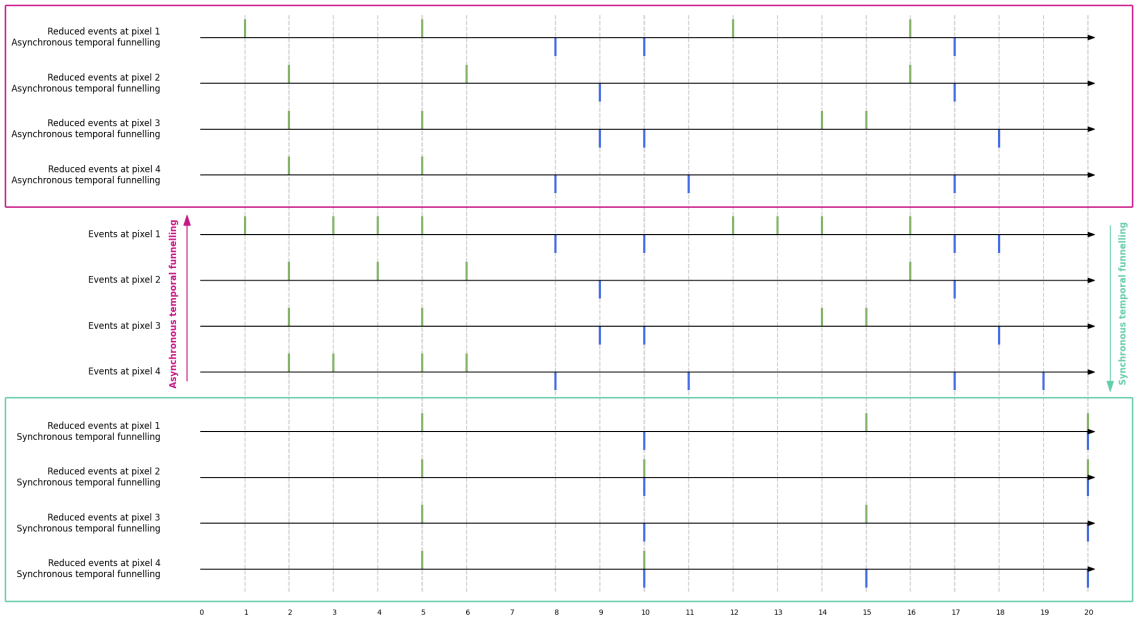


Figure 6.12 – Schematic illustration of the temporal funnelling downscaling method. A sensor of size 2×2 outputs negative and positive events at each pixel across time, which are downscaled using the asynchronous (upper red frame) or synchronous temporal funnelling (bottom green frame) methods with a factor $t = 5$.

The main disadvantages of this method are the induced loss of temporal precision and the increased event density. As for the spatial funnelling, this amounts to simply updating the content of the memory address of the event temporal coordinates, thus having a complexity $O(n)$ (with n the number of events).

6.3.2 Structural downscaling

We subdivide the structural downscaling methods into a deterministic one and a stochastic one. The deterministic method (see Fig. 6.13 top) keeps every k^{th} event out of N. The stochastic method (see Fig. 6.13 bottom) filters events with a probability p. This strategy has the benefit

of maintaining the original time and space scales. It handles events efficiently, simply deciding whether to keep or discard each new event.

It should be highlighted that this strategy simulates the real-world situation where an embedded system is overflowed with a dense event stream and is not able to process them all: some events will just be dropped (Gallego et al., 2020).

One can easily compare both structural downscaling methods by converting k in p as described in Eq. 6.4.

$$p = 100 \times \frac{1}{k} \tag{6.4}$$

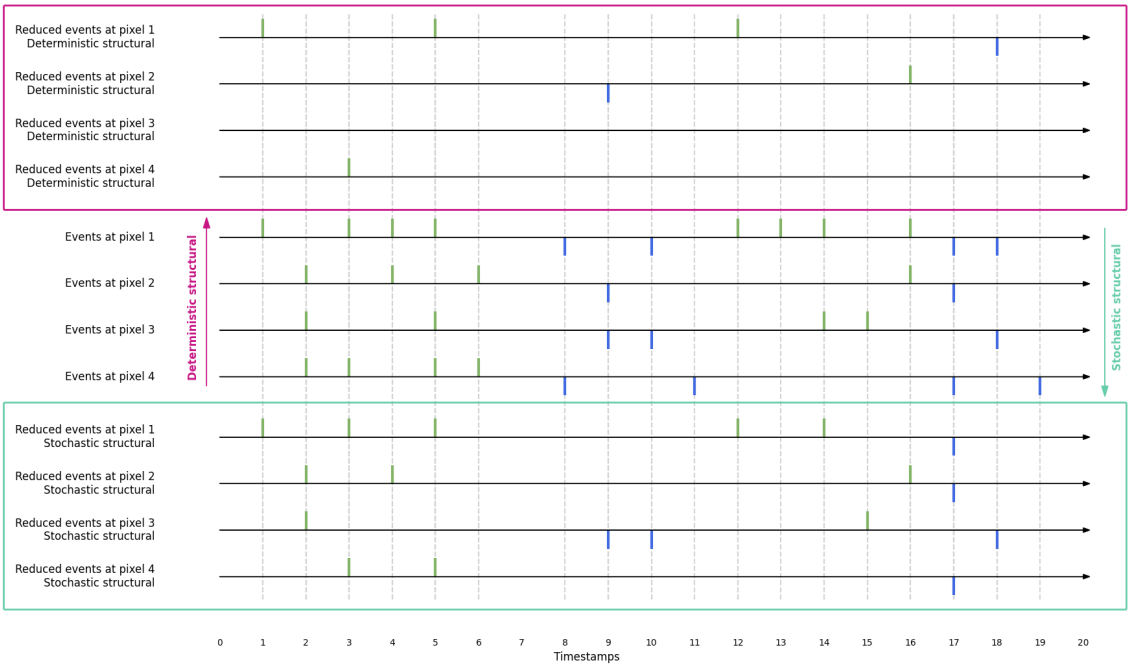


Figure 6.13 – Schematic illustration of structural downscaling methods. A sensor of size 2×2 outputs negative and positive events at each pixel across time, which are downscaled using the deterministic structural (upper red frame) with a factor $k = 6$; and the deterministic structural (lower green frame) with a factor $p = 60\%$.

6.4 Experimental comparison

To assess the performance of the different methods of event data downscaling described above, we applied them to the neuromorphic datasets DVS128 Gesture (Amir et al., 2017) and N-MNIST (Orchard et al., 2013), widely used as a benchmark in event data processing, and we assessed the results according to the criteria described in the following sections. These datasets were chosen according to their relevance to our motivations: DVS128 Gesture represents dynamic information to be processed temporally, which would be the case in most embedded systems. As for N-MNIST, it is particularly heavy in events: according to the parameters presented in its description, N-MNIST displays on average 10.71 events per pixel per second, whereas DVS128

Gesture only displays about 4.06 events per pixel per second (see Sections 3.3.1 and 3.3.2). All in all, they are different but complementary in many aspects (sensor size, amount of events, visual information, etc).

6.4.1 Comparison according to activity measures

Methods	**Number of events**	**Temporal density**	**Information entropy**	**Generation time**	**Optical consistency**
Original fullscale data	194,398	$1.95e^{-7}$	$2.94e^{-6}$	-	-
Spatial Funnelling	186,414	$2.95e^{-6}$	$3.65e^{-5}$	103.20 ms	*
Tonic Downscale	194,398	$5.40e^{-5}$	$4.31e^{-4}$	3.20 ms	*
SNN Pooling					
Separate handling	145, 680	142.27	0.15	648.86 s	**
Mutual influence	113, 431	10.77	0.15	680.43 s	***
Log-luminance					
Event Count	7,778	$1.25e^{-7}$	$1.88e^{-6}$	0.83 s	**
Linear Estimation	624	$8.58e^{-9}$	$1.56e^{-7}$	2.63 s	***
Cubic Estimation	435	$8.92e^{-9}$	$1.66e^{-7}$	2.94 s	****

TABLE 6.2 – Features of the original DVS128 Gesture dataset and of the six spatial downscaling methods, for a downscaling factor of 4.

The spatial downscaling of event data aims to significantly reduce the number of events per sample while keeping a sufficient amount of relevant information.

Table 6.2 compares the six spatial downscaling methods presented above and the *Tonic*'s "Downsample" tool according to 5 criteria described in the following section. The mean number of events per sample was calculated over the whole *DVS28 Gesture* dataset at full scale and downscaled by a reduction factor 4.

Events' temporal density D_t corresponds to the activation probability of pixels averaged over the whole sensor:

$$D_t = \frac{\sum_{x\in\{1,w\}} \sum_{y\in\{1,h\}} P_{x,y}}{w \cdot h} \tag{6.5}$$

where w and h are, respectively, the width and height of the sensor. The activation probability $P_{x,y}$ of one pixel of coordinates (x, y) is calculated as the number of events (positive or negative) occurring at a given pixel divided by the time length of the sample:

$$P_{x,y} = \frac{\sum_{t\in\{t_{min},t_{max}\}} \delta(x_t, x) \cdot \delta(y_t, y)}{t_{max} - t_{min}} \tag{6.6}$$

with t_{min} and t_{max} respectively the minimum and maximum timestamp of the sample, and δ the Kronecker delta function, which returns 1 if the variables are equal, and 0 otherwise.

If we consider the information brought by positive (or negative) events according to Shannon's information theory, we can estimate the entropy H (in bits) of the dataset as the average over all pixels and polarities p of the pixel entropy $H_{x,y,p}$:

$$\begin{aligned} H &= \frac{\sum_{p\in\{-,+\}} \sum_{x=1}^{w} \sum_{y=1}^{h} H_{x,y,p}}{2 \cdot w \cdot h} \\ \text{with } H_{x,y,p} &= \sum_{P_e \in \{P_{x,y,p}, \overline{P_{x,y,p}}\}} -P_e \cdot log_2(P_e) \end{aligned} \tag{6.7}$$

where $P_{x,y,p}$ is the probability of *event* (probability for the pixel of being active), and $\overline{P_{x,y,p}} = 1 - P_{x,y,p}$ is the dual probability of *no event*, for the polarity p at location (x, y). Overall, we believe that an event camera provides interesting information when the sensor is neither too little nor too much activated, i.e. when the pixels have an activation chance close to 50%. $H_{x,y,p}$ becomes lower when the pixel is either never active or always active. Entropy values in Table 6.2 are averaged from both polarities.

Finally, the computation time of each method presented in this paper was calculated on a panel of over $500M$ events. The values presented in Table 6.2 correspond to the average calculation period for the downscaling of $100K$ events, expressed in seconds. Optical consistency is a qualitative estimation of the resemblance between the method implemented and the actual behaviour of an event camera of smaller resolution recording the same scene ; this estimation was performed by our collaborators at IMSE, specialising in micro-electronics and event cameras.

It should be noted that the values produced for the SNN Pooling method are not obtained from the same panel of event data used by the other methods. As a matter of fact, since the computation time for one sample using either technique exceeds 10 min, we chose to assess our criteria on a smaller set of downsampled events using SNN Pooling.

The values of these criteria confirm the hypotheses we outlined above. *Tonic*'s "Downscale" method yields the highest number of events, thus the highest event density ; this spatial reduction does not lead to a significant drop of events and thus does not answer our need for reduced data to handle. The same goes for spatial funnelling, which ignores all event duplicates. Even though the computation time of those two methods is low, the output data is almost as large as the input. We should also point out that the information entropy of the spatial funnelling is quite low due to the constant activation of the sensor's pixels. The log-luminance techniques are the most consistent with the actual behaviour of event cameras. It drops a large amount of data, which might significantly improve any embedded computation, while maintaining a relatively high level of information entropy as intended. SNN Pooling is also promising by its behaviour, closely related to one of an event camera, but it is extremely sensitive to the weight of its connection: here the weight is probably too strong hence its poor results.

Fig.6.14 shows that the different event spatial downscaling methods implemented vary strongly by their activity measures. The plot on the left shows quite clearly that the number of events per sample decreases to a varying extent depending on the downscaling method used. Since we aim for a drop in the number of events post-reduction, the least convincing method by this criterion

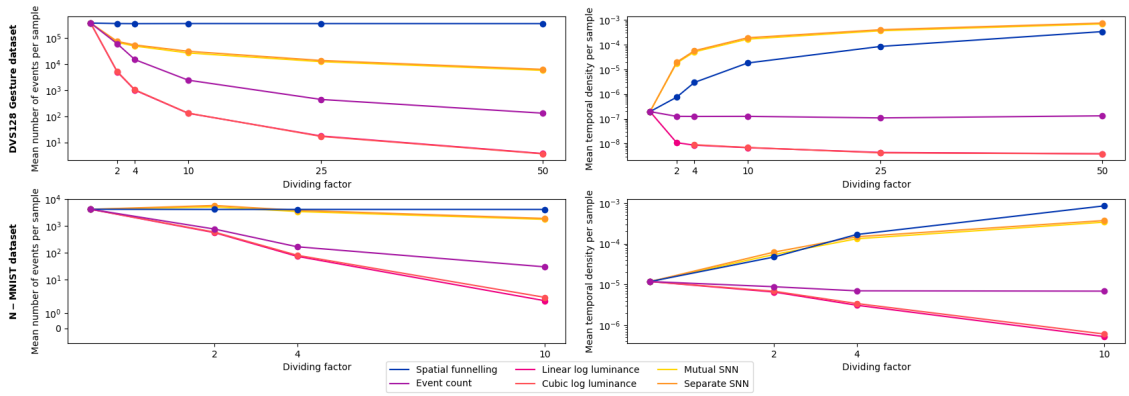


Figure 6.14 – Comparison of activity measures between different methods and reduction factors for DVS128 Gesture (above) and N-MNIST (below) datasets. The y-axes are in log scale. Since N-MNIST has a lower resolution than DVS128 Gesture, we assess its activity measures up until a reduction factor of 10 (instead of 50). We see here that the funnelling method keeps a high number of events and has an increasing temporal density, whereas all the other methods (except for SNN pooling) strongly decrease the number of events kept. The log luminance reconstructions lead to the decrease of temporal density, while the event count maintains one similar to the original data.

is obviously the funnelling one: the number of events stays stable no matter the reduction factor, whereas the most pronounced drop is obtained with the log luminance reconstruction methods.

The plot on the right represents the temporal event density, which we additionally study as the number of events is not a sufficient criterion for deciding which method provides the best preservation of information with an increasing reduction factor. This criterion shows the distribution of the reduced events over the sensor: a high temporal density corresponds to event data produced by a highly triggered sensor. Fig. 6.14 shows that the funnelling and event count methods have a high temporal density, i.e. all the sensor's pixels tend to trigger a lot (significantly more than 50% of the time) across time thus reducing the information transmitted. *A contrario*, the log luminance reconstruction seems to provide informative data, and both SNN methods seem a good compromise between these two *extrema*.

Since the spatial downscaling methods are more likely to be applied on an embedded system, it is interesting to consider the computational cost of each method, which give an estimate of the energy consumed. The funnelling method reduces data with nearly zero energy, as the funnelling can be applied in the event transmission from the sensor to the processor by just ignoring some bits in the address of the events (and removing any duplicates). On an embedded computer, the log luminance reconstruction methods are implemented using an algorithm of higher complexity thus requiring a greater energy expenditure. However, if we consider this reduction at the sensor level, it can be produced at zero energy cost. Finally, the SNN methods are to be directly performed by an embedded neuromorphic chip, whose energy consumption is directly proportional to the number of events given as input and the number of synapses implemented between the input and output layer, i.e. proportional to the reduction factor.

6.4.2 Comparison of real-time downscaling

In this section, we propose to compare the different spatial downscaling methods we implemented regarding offline and online data handling. Indeed, while event sensors show a number of advantages over frame-based cameras such as high temporal resolution, low energy consumption, high dynamic range sensing, etc, that make them unavoidable for the future of real-time embedded vision systems, the limited processing resources make it often hard to deal with a high event density in real-time embedded vision applications when handling events as they come †. This dilemma calls for data reduction techniques able to handle input events in real time. We investigate the time each of the methods introduced precedently takes to process $1s$ of input data from the DVS 128 Gesture dataset while preserving the quality of the information provided by the output data. The run-time is calculated both for the "real-time" processing of events, i.e. the event data is reduced as it is being recorded, or for the "deferred" processing, where the event data is first recorded as a whole and then downscaled. Indeed, some methods need a different implementation depending on this choice of temporality (see the description of log-luminance reconstruction methods above). Note that the online downscaling time of temporal methods, which were implemented more recently, could not be measured at the time of redaction of this manuscript. The online and offline event downscaling studied in this section was run on an Intel 8-core i9-10885H CPU at 2.4GHz.

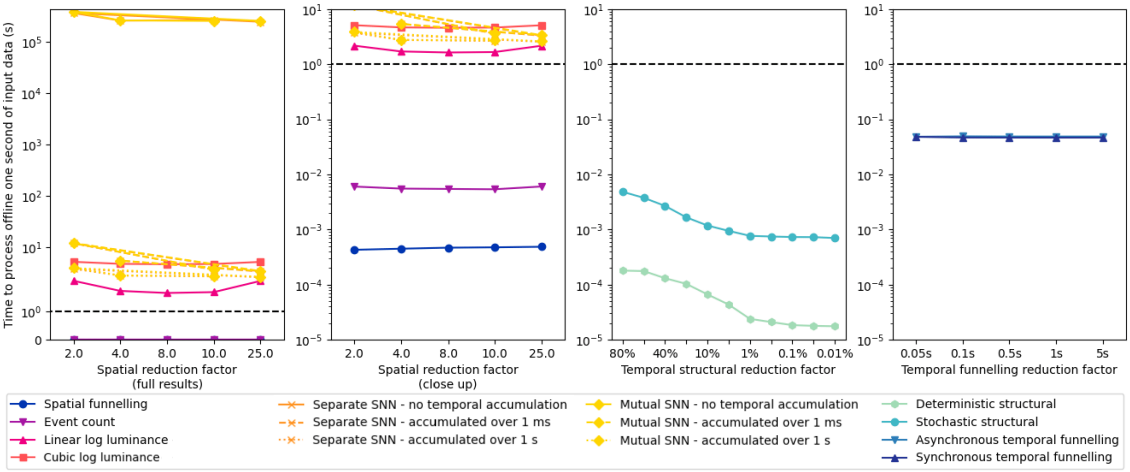


Figure 6.15 – Run time evolution for offline spatial and temporal downscaling of $1s$ of real-time event data. The dashed line shows the real-time bound. The SNN methods are run on SpiNN-3. Run offline, the spatial funnelling method is the quickest one as expected. Considering the temporal methods, all allow for quicker-than-real-time downscaling.

Fig. 6.15 confirms the results presented in Tab. 6.2: spatial funnelling is definitely the quickest method among the spatial downscaling ones, followed closely by the event count. Among the temporal methods, the funnelling methods run with the same time and are the slowest out of the four — however, it still outperforms the log-luminance reconstruction methods and the SNN pooling. The deterministic structural method outruns all methods (spatial included). Furthermore,

†. This issue of processing event data on the fly might not be as great when accumulating events to process them as frames. However, we believe in the usefulness of retaining the high temporal accuracy of events for further process, which calls for pre-processing the data event *per* event.

Fig. 6.15 shows that the temporal and structural methods do process events online relatively fast and stably. When the temporal reduction factor increases, the deterministic structural factor k increases or the structural stochastic filtering probability decreases.

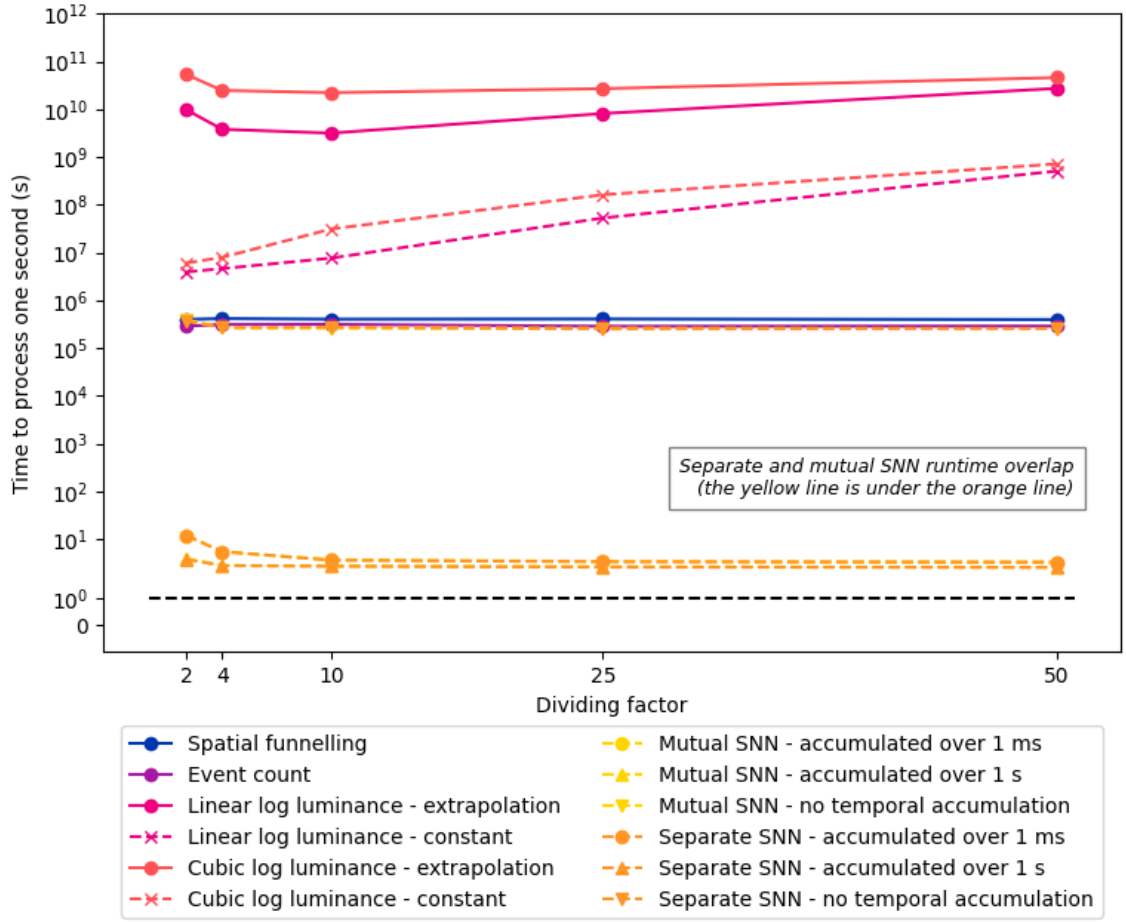


Figure 6.16 – Run time evolution for online spatial downscaling of $1s$ of real-time event data. The dashed line shows the real-time bound. The SNN methods are run on SpiNN-3, while the log-luminance reconstruction methods are evaluated for the two strategies *constant* and *interpolation*. The SNN methods are undoubtedly the quickest, but still two orders of magnitude slower than real-time.

Fig. 6.16 illustrates the computational run-time evolution. It should be noted that this CPU can only perform a maximum of 2.4 billion cycles per second, whereas an event camera can output up to $1M$ events per second per pixel. For the DVS 128 Gesture dataset, the event camera is of size 128×128 pixels, thus has the capacity to output up to $16B$ events per second[‡]. The DVS 128 Gesture comprises an average of $30k$ events per second, as described in Section 3.3.1. The speed of our CPU is thus too low to simply cycle through the events in real time, so the real-time reduction is unattainable using this hardware.

‡. The number of 16 billion events per second is hypothetical and corresponds to the worst case scenario where the DVS pixels are constantly triggered during the whole second. It has been calculated as follows: $1M$ events per second per pixel $\times\ 128^2$ pixels $= 16,384M$ events per second $\approx 16B$ events per second.

It is interesting to note that while the event count is slower than the funnelling when run offline on the whole dataset at once (see Fig. 6.15), it is quicker in an online scenario. This is due to the fact that the event count needs by definition to be run timestamp per timestamp — which is how we believe any online downscaling method should be run — thus giving it an advantage over the spatial funnelling. As for linear and cubic log-luminance reconstructions, they suffer from an important drop in performance accuracy and an increase in run time when the reduction factor increases ; but it is to be noted that the constant strategy is significantly quicker than the interpolation one.

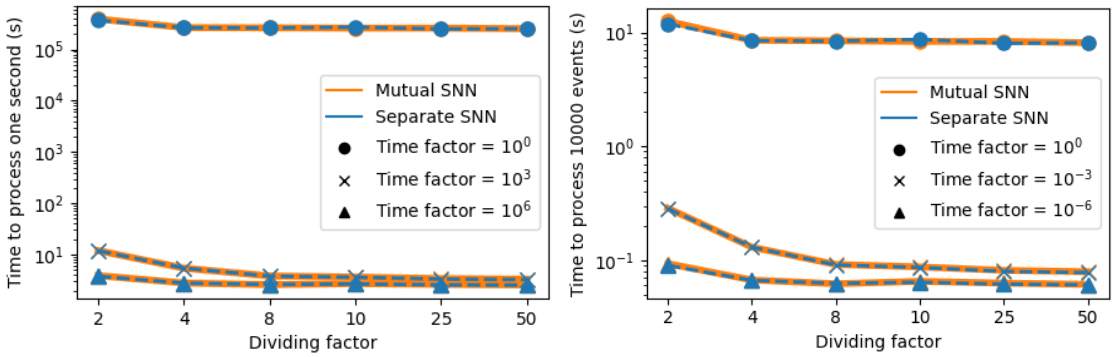


Figure 6.17 – Time required by both SNN downscaling methods to process one second of input events (left) or 1000 incoming events (right) while run on the SpiNN-3 board, with a previous asynchronous temporal funnelling of the input data.

The simulation run time for the SNN downscaling methods on SpiNN-3 is displayed with more details in Fig. 6.17. We combine the SNN pooling methods with the asynchronous temporal funnelling to reduce event data both spatially and temporally in order to assess the limitations of the SpiNN-3 board in terms of simulation time for the simple networks that are the SNN pooling methods. Note that this is one of many method combinations for spatiotemporal reduction that one could implement using the methods we introduced. We downscale the event data temporally by a factor $t \in [1, 1e^3, 1e^6]$ (in μs). The temporal factor $t = 1$ corresponds to the absence of preliminary temporal reduction.

The time required to spatially downscale $1s$ of input data in real-time without preliminary temporal reduction is significantly higher than $1s$, even when the reduction is run on the SpiNN-3 board. This seems to point towards a limitation of the SpiNN-3 board: to simulate a simple network composed of two layers (one of which is the input, thus not simulated as SNN by SpiNNaker) and one to two sets of connections, the board requires minimum time significantly greater than 10^{-6} seconds (grain of an event timestamp) to simulate a time step. SpiNN-3 therefore seems to require a prior temporal reduction in order to be able to simulate event data in a reasonable time. Indeed, Fig. 6.17 shows that a temporal funnelling of at least 10^3 is needed.

As discussed previously, the simulation run time decreases when the reduction factor increases. Surprisingly, we do not observe a significant difference in run time between mutual SNN and separate SNN methods. Similar behaviours are observed when studying the evolution of the run-time needed by the SNN methods to process 1000 input events, i.e. around $30ms$ of input data.

6.4.3 Comparison according to performance on a classification task

In the best-case scenario, an optimal spatial downscaling method for event data drastically reduces the number of events while conserving relevant data. This section aims to discuss the relevance of our proposed approaches compared to existing ones regarding this matter. Activity measures are studied above as a first attempt to answer this interrogation. We further investigate the relevance of the retained information by looking at the performance of each method against the specific computer vision task of classification.

We have pre-trained the classifiers implemented using SLAYER and PLIF (that will be designated as "SLAYER benchmark classifier" an "PLIF-based classifier" in the remainder of this chapter — see Section 4.3.1) on the full-scale DVS 128 Gesture and N-MNIST datasets. In order to optimise the simulation run time, we loaded and fine-tuned this classifier's weights in a transfer learning strategy on the downscaled datasets.

6.4.3.1 SLAYER benchmark classifier

We apply the SLAYER classifier (see Section 4.3.1.1) on the DVS128 Gesture and N-MNIST datasets reduced using the six spatial methods introduced previously [§]. The resulting performances are presented in Fig. 6.18. The optimal downscaling method should have the best accuracy performance since it retains the most relevant information necessary for effective classification.

Unlike other classification models which accumulate input events into frames (such as the PLIF-based classifier (Fang, Yu, Chen, Masquelier, et al., 2021) — see subsection below), the SLAYER benchmark classifier takes as input sparse events and shows more interesting and coherent results than the PLIF-based one. Fig. 6.18 shows that the downscaling methods introduced in this paper perform well, especially for the DVS 128 Gesture dataset. Regarding N-MNIST, as seen earlier, the large number of events retained for a small reduction factor depends entirely on the SNN pooling hyperparameters, which should be tuned for each new dataset. Moreover, this shortcoming is compensated by the significant advantage of fast computation time for SNN pooling compared to other methods (see Fig. 6.16). In contrast, the "Spatial funnelling" remains on the lower right even when the reduction factor increases ; moreover, past a small reduction factor (2), its performance is lower than most methods and the number of retained events is higher. It should be noted that all methods lead to a decrease in performance when the reduction factor increases, which validates the activity measures study performed in the above section.

6.4.3.2 PLIF-based classifier

Fig. 6.19 presents the performances obtained using the PLIF-based classifier (see Section 4.3.1.2), which leads to the conclusion that the funnelling method is by far the most appropriate downscaling method, at least in the context of a classification task. This is quite surprising compared to the results obtained using SLAYER (see Fig. 6.18. An explanation of this difference can be found in the fact that the PLIF-based classifier, which takes in input frames, will have a better recognition performance when the number of events is higher, as the corresponding reconstructed frames will provide more information than in a sparser case. Indeed classifiers based on convolution such as the PLIF-based one tend to underperform on sparse data: they can handle

§. Due to unforeseen computational problems, we were unable to test SLAYER on the more recently introduced temporal methods. However, these are evaluated on the PLIF-based classifier in the next section.

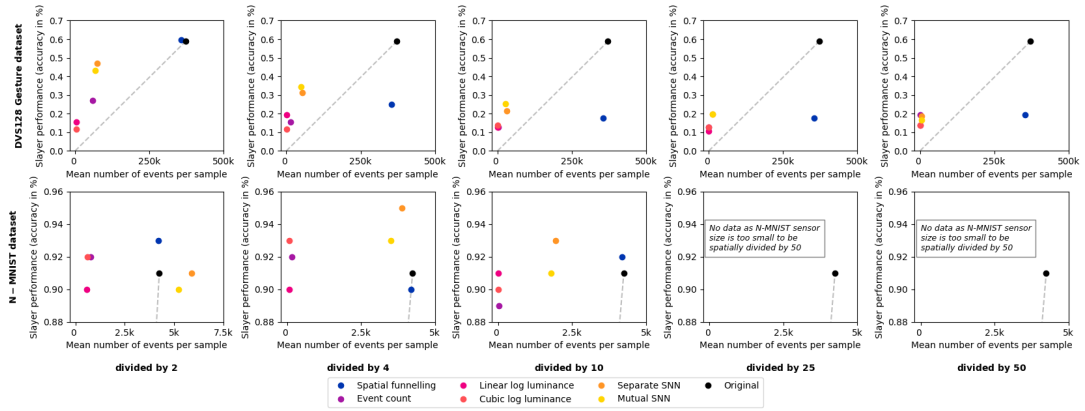


Figure 6.18 – Evolution of the accuracy performance of the SLAYER benchmark classifier applied to DVS128 Gesture (first row) and N-MNIST (second row) when the spatial diving factor increases (columns). The dashed line connects the original data to the origin $(0, 0)$ of the reference system. Note that the points located in the upper left side of the dashed line correspond to results showing interesting trade-offs for performance over data size, i.e. a high accuracy with a small number of events. Globally, most spatial downscaling methods lead to a good trade-off, with the notable exception of the "Spatial funnelling" which remains in the lower right side even when the reduction factor increases.

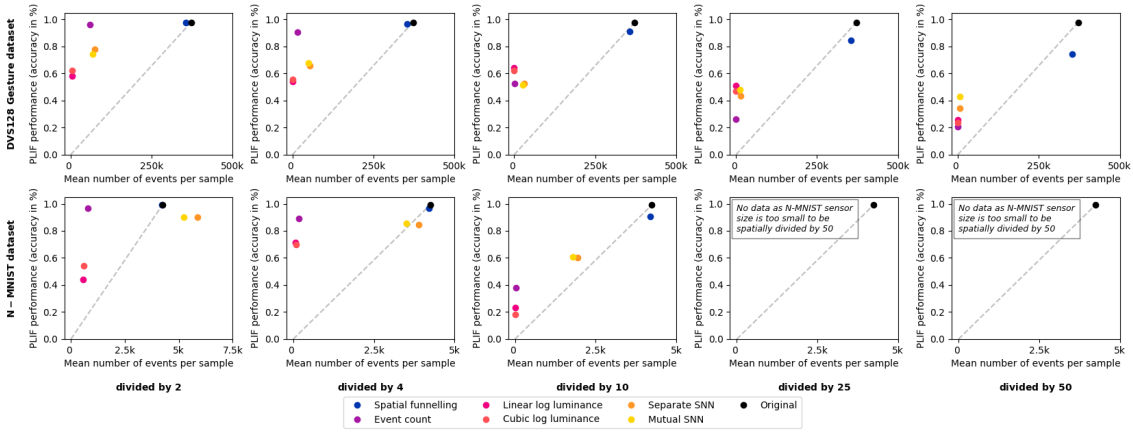


Figure 6.19 – Evolution of the accuracy performance of the PLIF-based classifier applied to DVS128 Gesture (first line) and N-MMNIST (second line). The dashed line separates each plot between what we aim for – a decreased number of events for a significantly good accuracy (upper left) – and what we do not aim for – a decreased accuracy for a great number of events (lower right). Globally most results are in the part that we aim for, with the notable exception of the "SNN pooling" results.

fullscale event data but quickly lose performance when the number of events drops, i.e. when the data is reduced using any methods except funnelling.

Reduction factor	Deterministic structural	Stochastic structural
80	0.96875	0.97569
60	0.96875	0.97569
40	0.96875	0.98263
20	0.96875	0.95486
10	0.95833	0.95833
5	0.95833	0.96181
1	0.93055	0.95486
0,5	0.88541	0.94097
0,1	0.73611	0.73611
0,05	0.65277	0.625
0,01	0.45833	0.35763

TABLE 6.3 – Accuracy of the PLIF-based classifier applied to DVS128 Gesture reduced using the temporal structural downscaling methods.

Reduction factor	Asynchronous temporal funnelling	Synchronous temporal funnelling
50000	0.97916	0.96875
100000	0.97569	0.97916
500000	0.9548	

TABLE 6.4 – Accuracy of the PLIF-based classifier applied to DVS128 Gesture reduced using the temporal funnelling downscaling methods.

The accuracy performance of the PLIF-based classifier obtained on DVS 128 Gesture downscaled temporally using the structural and funnelling methods are presented respectively in Tab. 6.3 and 6.4. According to these results, these methods maintain a quite high amount of information even with a high reduction factor: keeping only 0.05% of the original events, the two structural methods still reach an accuracy exceeding 60%; when accumulating events over $5s$ (i.e. more than 80% of the average sample length), the temporal funnelling methods both remains at 95% accuracy.

An interesting aside worth highlighting is that a large part of the motion information is encoded in the event data polarity, as pointed out in (Oudjail & Martinet, 2019). In some examples, the motion direction can easily be estimated using simply the spatial information (i.e. the repartition of events of different polarities in space) without the help of temporal information. As explained in Chapter 9 and more specifically in Fig. 9.16, the polarities displayed on a single frame reconstructed from $50ms$ of a DVS 128 Gesture sample allow us to identify the direction of the hand motion. We confirm this intuition by comparing the results obtained on a classification task on event data with only available polarity using the PLIF-based classifier, a model known to translate events into 2-channel frames (one channel per polarity) and then train on those frames.

We observe in Fig. 6.20 that the PLIF-based classifier did not obtain the same results whether the classification for DVS128 Gesture is performed on both polarities or solely one (here, negative). Instead of learning the dynamics of the events, the PLIF-based classifier easily recognises classes by identifying the direction of the arm's movement according to the distribution of the positive and negative events on the frame. This questions the hardness of the dataset as well as underlines the need to process event data using a temporal model, able to rely on the temporal evolution of data and/or features.

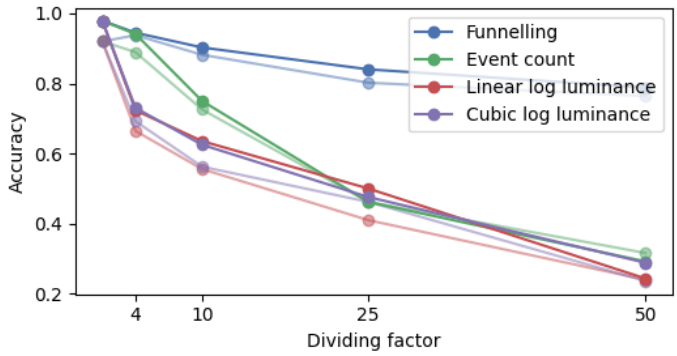


Figure 6.20 – Evolution of the accuracy performance of the PLIF-based classifier applied to DVS128 Gesture when the spatial reduction factor increases, with both polarities and only on negative events (clear lines).

6.5 Human assessment study

In this chapter, we introduced a wide range of downscaling methods for event data and assessed their usefulness by comparing quantitatively and qualitatively between them. We wished to identify the best method, either overall or depending on the use case, relying on the evaluators described above — however, a last question remained: what would one really "see" in the downscaled data ? The results brought by the qualitative assessment using classifiers were a first step towards answering this broad question from a machine point of view. We completed these results to assess the human point of view by carrying out a human perception study, which protocol and results are described in this section.

With the help of our interns Lucía, Marina and Ewa, we performed a study to assess and compare human performance in a gesture classification task using event data. Original event data from IBM's DVS128 Gesture dataset (see Section 3.3.1) is downscaled with the several spatial and temporal methods described above, and the classification performance is measured with human participants.

6.5.1 Human machine interface and protocol

6.5.1.1 Event data

We aim to evaluate the evolution of human performance according to different event downscaling methods at different resolutions. To achieve this objective, a participant should not correctly classify each of the 11 gestures comprised in DVS128 Gesture but should demonstrate that they were able to correctly separate similar gestures in order to prove that the reduction method studied retained sufficient relevant information. For this purpose, we decided to display to the participants only four gestures instead of eleven:

— gesture "right hand clockwise" (A),
— "right hand counter-clockwise" (B),
— "drums" (C),
— "hand rolls" (D).

The selected gestures were two by two similar together but different between the two pairs ; A and B are both quite similar, only differentiable by the direction of the hand movement ; C and D are

similar through the spatial distribution of events. For each figure, short samples from 5 out of the original 29 recorded subjects were selected and randomly displayed to the participants so that they classify the gesture itself and not the performing subject. Fig. 6.21 presents a screenshot of those four gestures ; the participants could use this support to help them classify the videos shown to them during the experiment.

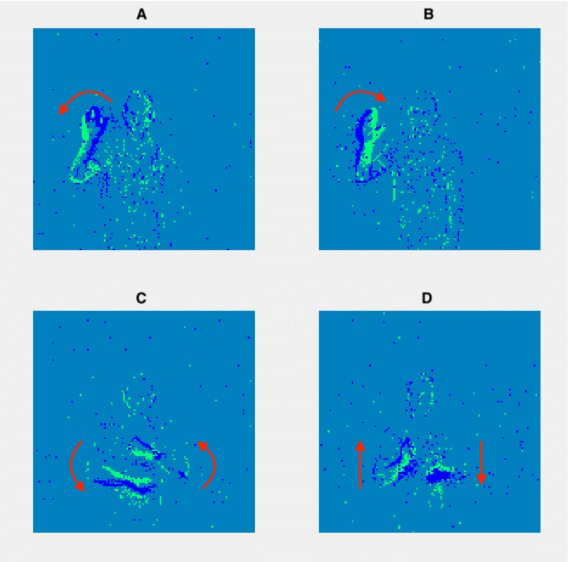


Figure 6.21 – Poster presented to the participants, illustrating the four annotated gestures selected from the DVS128 Gesture dataset to conduct the experiment.

The six spatial event downscaling methods described above were experimentally evaluated in this work at four different reduction factors:

— 2 (resolution of 64×64),
— 4 (resolution of 32×32),
— 8 (resolution of 16×16),
— 10 (resolution of 13×13).

We also experimentally evaluated the four temporal downscaling methods. The two temporal structural downscaling methods were experimentally evaluated in this work at three different structural reduction factors p: 10%, 1% and 0.1%. In other words, the deterministic factor k was respectively set dividing 10, 100 and 1000 and k was converted into p according to Eq. 6.4. The two temporal funnelling downscaling methods were evaluated at five funnelling reduction factors: with time-window lengths of $0.05s$, $0.1s$ and $0.5s$ for the synchronous funnelling, and $1s$ and $5s$ for the asynchronous funnelling. These were agreed upon as it was immediately obvious to us after studying the downscaled datasets that humans would understand the data obtained with the asynchronous method much better and for a longer time window, while they would have a harder time understanding events accumulated into frames displayed at the end of similar time windows. We decided to distinguish between the factors used between the synchronous and asynchronous methods in order to limit the experimental time for participants and not to collect data that we expected or could easily extrapolate.

6.5.1.2 Interface

Our interns Lucía, Marina and Ewa implemented an interface allowing participants to classify different samples of the 4 gestures, with different downscaling techniques and resolutions recorded from 5 different users.

To capture the user input, the interface was implemented using OpenSesame (Mathôt, Schreij, & Theeuwes, 2012), an open-source tool for experiment creation in the fields of psychology, neuroscience, and experimental economics. This interface allows for easy storage of the gesture shown in each sample, its downscaling technique and resolution, enabling also to capture the subject's response and its velocity.

The experiment encompassed a training phase to familiarise participants with the four gestures and a testing phase to record the data for analysis. During both phases, every sample was displayed during $10s$ in `mp4` format. Afterwards, subjects could select one of the 4 options (A, B, C, D) corresponding to the gestures or otherwise select "I don't know" (see Fig. 6.22).

In the training phase, subjects were presented with the 4 different gestures without any downscaling and received short feedback on the correctness of their answer. This training was repeated twice per participant to ensure that they could recognise the original gesture recorded with an event camera, even if they were not previously familiar with event data. Afterwards, in the testing phase, the samples were shown without any feedback.

The data collection for participants starts with a brief experiment explanation, as well as demographic questions, such as gender and age range. To reduce selection bias, participants of different genders across all ages took part in the experiment (see Tab. 6.5). To reduce expectation bias, participants were not informed of the techniques they were classifying at any moment nor were they obliged to complete the experiment within a given timeframe. To minimise order bias, the order of the shown samples was randomised.

Moreover, during the whole experiment, subjects had access to a poster (see Fig. 6.21) with illustrations showing each gesture and corresponding label involved in the study.

Age	18-25 y.o.	More than 25 years old	Total
Female	11	11	22
Male	20	17	37
Other	0	1	1
Total	31	29	**60**

TABLE 6.5 – Demographic repartition of the human assessment experiment participants.

6.5.1.3 Protocol

The study involved two experiments, one to evaluate the spatial techniques and another to evaluate the temporal techniques. Both were performed by 30 subjects each. The participants gave informed consent to participate and were informed of the purpose and nature of the study. The study was conducted in designated quiet rooms intended only for participants, on laptops with screens of similar size and quality. The participants were invited into the room in batches of 3 at different time intervals where they could choose one of the laptops whose positioning prevented access to the answers of other participants (see Fig. 6.23).

In the experiment to assess spatial methods, participants had to identify 122 samples, with an average of $8m30s$ in the total task. In the experiment to assess temporal methods, participants had to identify 44 samples, and we recorded an average of $3m06s$ in the total task. When setting up both experiments, we expected the participants to take around $15s$ to select the label corresponding to each sample they were shown. However, in most cases, the subjects were able to respond faster, in $4.45s$ on average.

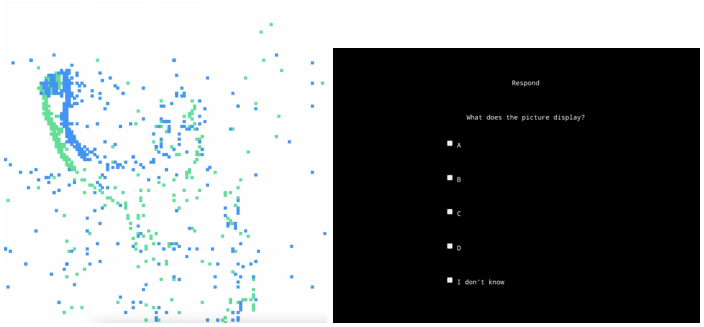


Figure 6.22 – Interface presented to the participants during the human assessment experiment.

Figure 6.23 – Participants performing the human assessment experiment.

6.5.2 Assessing human performance

In this subsection, we assess the quality of human classification by analysing the accuracy, the number of unknown answers, the time to reply and the average number of events per sample for the different downscaling methods. The last criterion "number of events" is a quantitative evaluator relevant to our motivations: indeed, we want to implement event downscaling methods which reduce the quantity (i.e. the number of events) while preserving the quality (i.e. the amount of information carried) in the data. The "unknown" answers correspond to the selection of the "I don't know" choice in the interface, i.e. when participants did not recognise the gesture displayed. Note that the "unknown" answer does not correspond to an additional class *per se*, as the interface does not display at any point a gesture different from A, B, C or D.

As expected, the human accuracy decreases and the percentage of unknown answers increases as the size of the data decreases across all spatial downscaling methods (see Fig. 6.24). We can observe that both SNN-based and spatial funnelling are the techniques with the best human accuracy while log-luminance reconstruction techniques give the worst results.

Human accuracy drops below the chance level (25%) for a spatial diving factor of 10 in the log-luminance and mutual SNN techniques. The biggest drop in human accuracy occurs from factor 4 to factor 8. Surprisingly, the log-luminance techniques lead to accuracy varying little, even increasing slightly but always staying close to the chance level threshold. The notable difference in human accuracy and percentage of unknown answers between techniques does not translate to the time per answer results, which are relatively homogeneous among techniques (see Fig. 6.24 right).

The temporal techniques achieving similar resolutions are assessed in Fig. 6.25. Amongst the structural techniques, the stochastic outperforms the deterministic. As for the temporal funnelling techniques, the asynchronous technique obtains better results, with a small percentage of unknown

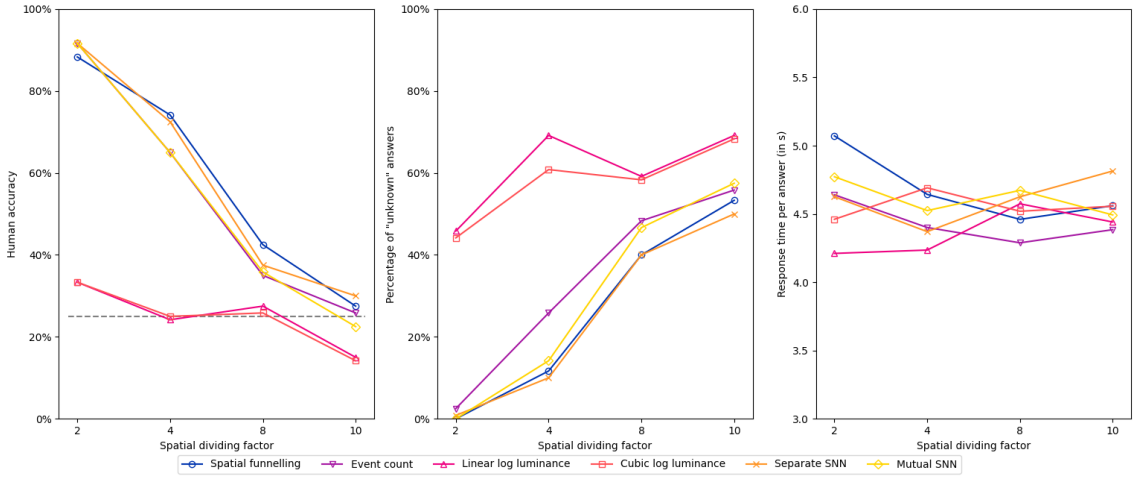


Figure 6.24 – Human accuracy, rate of "unknown" responses and human response time to classify event data downsized spatially. In the figure on the left, the dotted line at 25% of the accuracy corresponds to the chance level.

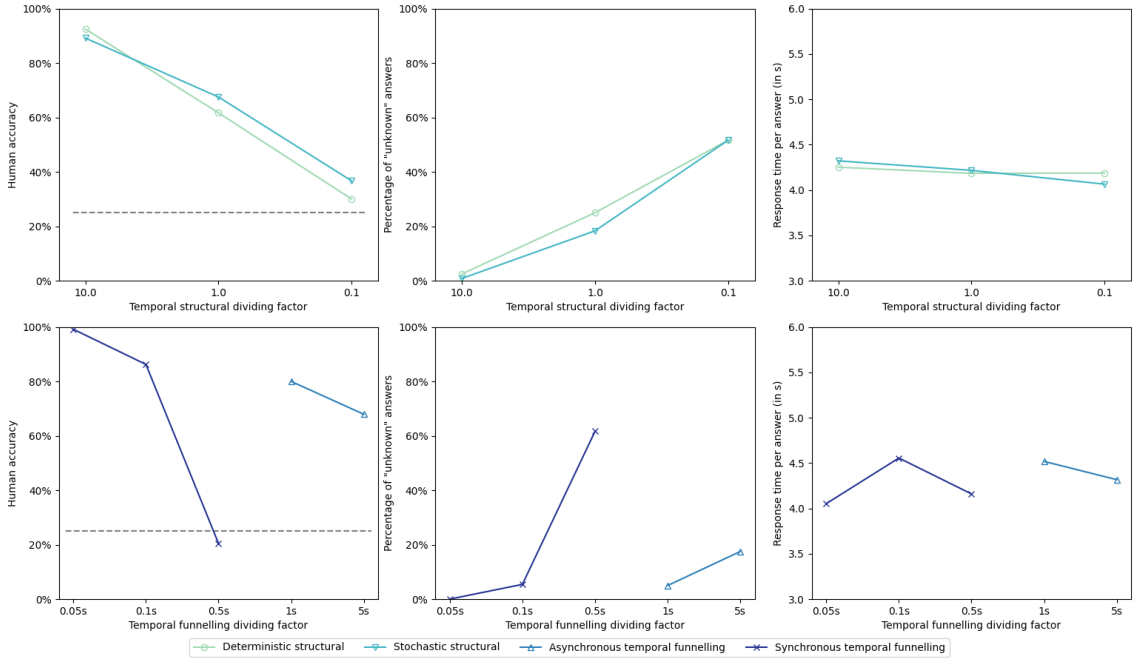


Figure 6.25 – Human accuracy, rate of "unknown" responses and human response time to classify event data downsized temporally. In the figure on the left, the dotted line at 25% of the accuracy corresponds to the chance level.

answers (5%) and a very notable accuracy of 70% despite an accumulation of events over $5s$ — meaning that events appear only once per pixel for 80% of the original samples (that are in average $6s$ long). The temporal downscaling techniques result overall in better human accuracy and a lesser amount of unknown answers than the spatial techniques, which might be explained by the higher number of events. The response time per answer is also shorter than for spatial techniques. Only

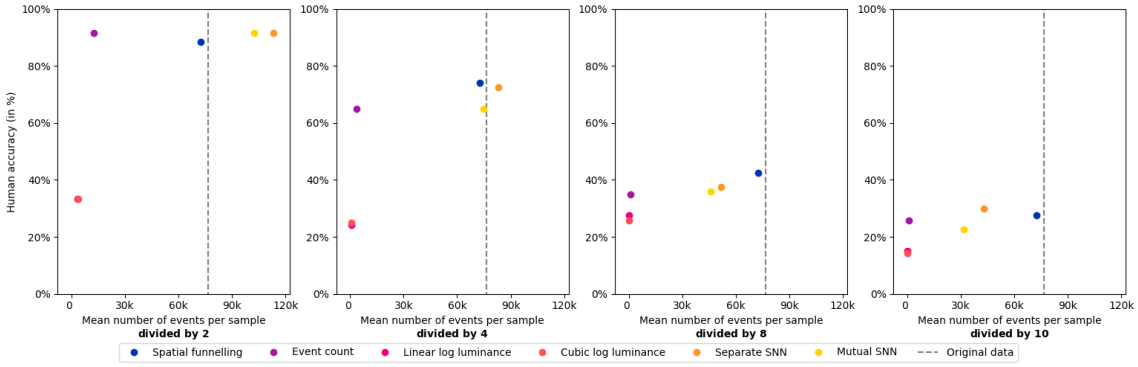


Figure 6.26 – Human accuracy according to the resulting number of events per sample after spatial downscaling methods. The vertical dashed line corresponds to the number of events per sample in the original dataset.

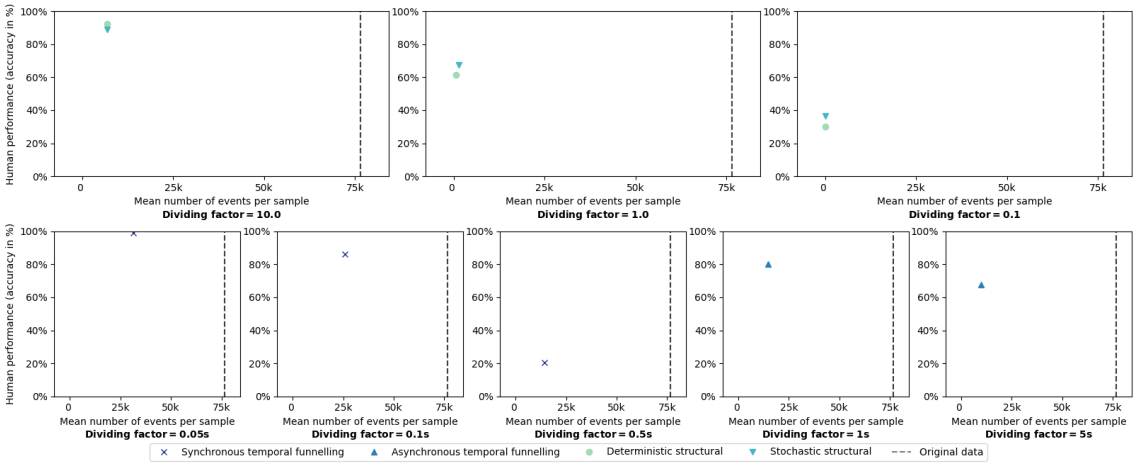


Figure 6.27 – Human accuracy according to the resulting number of events after temporal downscaling methods. The vertical dashed line corresponds to the number of events per sample in the original dataset.

the synchronous temporal funnelling technique falls below the chance level when accumulated over a time window of $0.5s$.

Fig. 6.26 plots the human accuracy by spatial techniques against the resulting number of events. It clearly shows that the three techniques with the best overall human performance, Spatial Funnelling and both SNN pooling, also keep the highest number of events from reduction factor 2 onwards. Moreover, the techniques with the worst human accuracy, the log-luminance reconstruction techniques, give the smallest number of events kept.

Regarding the temporal downscaling techniques (see Fig. 6.27), we can observe that the number of events kept is the same across all reduction factors. As mentioned, the stochastic technique's accuracy exceeds the deterministic's performance.

Thanks to this study on the downscaling of event data for gesture classification using human participants, we can finally conclude that a certain size threshold needs to be maintained to ensure that human performance does not fall below the chance level. For input data of resolution $128 \times$

128, this threshold is close to the factor 8 for spatial methods, 0.1 for temporal structural methods and $0.5s$ for temporal funnelling methods.

6.5.3 Comparing human and machine performance

In this subsection, we assess the quality of human classification compared to the performance achieved by an SNN. We compare the results collected experimentally with the results output by the PLIF-based classifier (see Sec. 4.3.1.2).

As detailed in Chapter 1, many NNs outperform human beings in specialised tasks such as classification. Here we expect the PLIF-based classification to outperform the recorded human performance, especially considering many of the study participants were not familiar with event data. Fig. 6.28 confirms that for the structural and temporal funnelling techniques, the classifier outperforms human results with a minor exception in the synchronous temporal funnelling method with a time window of $0.05s$. The difference between human and machine performance is especially stark for the synchronous temporal funnelling since the ratio displayed in Fig. 6.28 is quite close to 0 for this method. Machines significantly outperform humans, which cannot comprehend a motion from successive frames that are too far apart. This confirms that the PLIF-based classifier used here relies on the spatial repartition of events and does not take advantage of its temporal dimension. This underlines the disadvantages of using a frame-based classifier with event data. For the spatial methods, we encounter more obvious exceptions in the mutual and separate SNN at the reduction factors between 2 and 4.

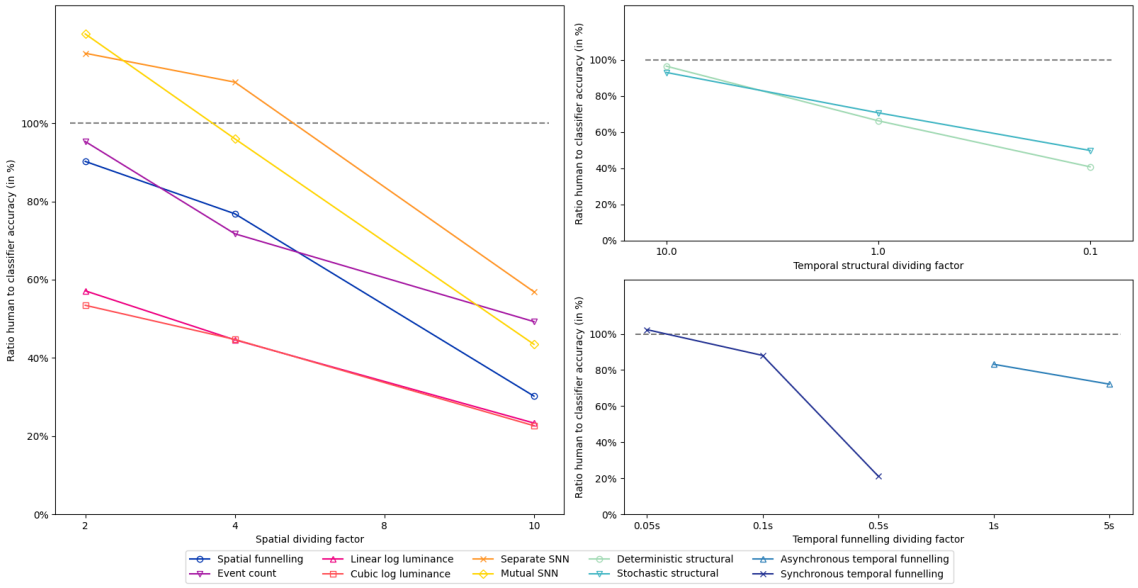


Figure 6.28 – Ratio of the human accuracy to the classifier accuracy, for each downscale factor, for spatial and temporal downscaling methods. When the values exceed 100% (dotted grey line), the human performance is higher than the classifier's; when they fall below 100%, the classifier outperforms the human performance.

Fig. 6.29 sums up the four criteria evaluated until now and adds a fourth evaluation metrics factor to the three evaluated so far (human accuracy, machine performance and number of events

compared to the original data): the downscaling time. The best method would optimise human and machine accuracy while minimising the number of events and downscaling time. Amongst the spatial downscaling methods, we observe once again that the techniques with the best compromise seem to be Spatial Funnelling and SNN. Mutual SNN is slightly better because it achieves an equivalent human accuracy with a significantly lower number of events, even though the downscaling time increments significantly. Once again, log-luminance methods do not provide satisfactory results. Even if they give a very small number of events and are optically coherent with the behaviour of an event camera (see (Gruel et al., 2022)), the accuracy is low and the downscaling time is high.

Amongst the temporal structural downscaling methods, the deterministic technique yields a very similar accuracy to the stochastic method, with a much shorter time taken to downscale. Less obvious, for temporal funnelling methods, the synchronous method offers better global results than the asynchronous technique, with a lower downscaling time and number of events ; however these advantages were traded off for a decrease in human accuracy.

The large standard deviations observed in Fig. 6.29 on the human accuracy measured on five methods (spatial funnelling, event count, SNN separate and mutual and especially synchronous temporal funnelling) can be explained by the wide variation in human performance depending on the intensity of reduction: according to Fig. 6.26 and 6.27, the human accuracy drops when the reduction factor increases for all methods. This reinforces the overall value of the asynchronous temporal funnelling method, which produces event data whose standard deviation is smaller although the performances are measured on significantly higher reduction factors, meaning it is quite consistent in its downscaling.

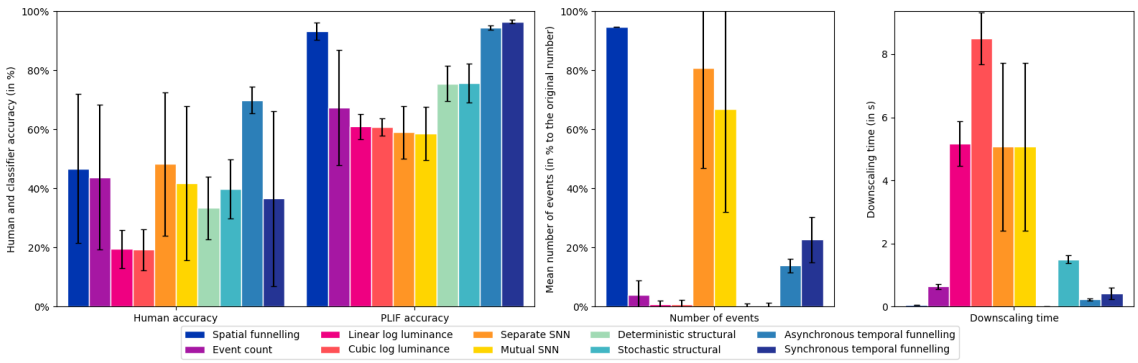


Figure 6.29 – Overall comparison between spatial and temporal event downscaling methods, according to the overall human and machine performance, number of events and downscaling time of $1s$ of input event data (in s). All those criteria are weighted by the downscaling factor: the bigger the reduction, the higher the weight of the corresponding value. The number of events corresponds to the ratio of reduced events to the original number (in %). Each barplot presents the weighted mean of the corresponding value, and the black error bars the weighted standard deviation.

Our results highlight some discrepancies between human and machine-learning approaches to gesture classification using event data. Human accuracy is higher than machine accuracy in specific reduction factors for the synchronous temporal funnelling and mutual and separate SNN techniques. Indeed, this can be explained by the fact that humans are less susceptible to spurious correlation (i.e. associating two samples that are in no way causally connected) in the data than

most neural networks (Srivastava, Hashimoto, & Liang, 2020) and do not suffer from shortcut learning (Geirhos et al., 2020), especially on the relatively small dataset that is DVS128 gesture (only 133 samples in its entirety).

6.6 Conclusion

While standard frame-based data (grayscale and RGB) are easily downscaled – which is a very common pre-processing step –, there existed no straightforward equivalent for event data up until recently.

In this chapter, we introduced and compared ten methods for event data downscaling for computer vision applications: six "spatial" and four "temporal". Our results show that the choice of the reduction methods greatly influences the event features of the reduced dataset, namely the number of events, the density, and information entropy.

While the proposed methods offer a panel of solutions to choose from, this choice is not trivial but depends strongly on the target vision task (classification, detection, tracking, etc), the application context, other event data preprocess it requires and the embedded computer throughput.

Our goal is to identify the best choice to perform spatial event reduction on an embedded system in order to adjust the complexity of data to the available resources such as processing capability and energy consumption (which is often directly linked to the amount of input data — for example, the energy consumed by certain neuromorphic hardware such as Kraken depend on the number of input events (Di Mauro et al., 2022)). It is important to note that our contributions are designed to be used in a global system including downscale and processing ; the implementation of our methods thus yields interest only if the energy consumed by the downscaling mechanism is compensated by a major reduction of the cost of subsequent processes. The results tend to show that this best choice can be found among the SNN pooling methods. Indeed they achieve an efficient trade-off between the optimisation of desired activity measures, performance and energy consumption as well as being significantly closer to real-time data handling than existing methods. Furthermore, when neuromorphic hardware is involved, it can be deemed easier and less costly to use either SNN method described in this chapter as a first preprocessing layer to ML models. As no significant differences can be observed between the run time and the classification performance of mutual and separate SNN pooling, we look at the activity measures to compare both methods. The mutual SNN pooling is definitely the optimal one based on the results of this work regarding these criteria, as it leads to a smaller number of events and a lower temporal density and is more optically coherent with the behaviour of an event camera.

If we do not limit our criterion to spatial downscaling, the structural methods that consist in randomly dropping events show very interesting performances in our experimental settings with the PLIF-based model, with a classification accuracy that remains very high even when dropping a high ratio of events. Additionally, the deterministic method has the lowest downscaling run time in an offline scenario out of all the spatial and temporal methods.

It is interesting to note here that, even though they are standard benchmark event datasets, N-MNIST and DVS 128 Gesture may not be the ideal datasets to assess the impact of temporal downscaling since none of them requires a fine temporal resolution: N-MNIST is essentially a static image translated in space to trigger events, and (Fang, Yu, Chen, Masquelier, et al., 2021) demonstrated that DVS 128 Gesture can be solve with only 20 timesteps of $5ms$ each, i.e. only around $1s$. Thus DVS 128 Gesture and N-MNIST have been useful as a first support to validate

these temporal methods, but a validation on temporally finer data would be welcome, for example on a lip-reading or even faster motion classification test.

Finally, we completed the previous results with a study quantifying the effect of downscaling of event data for gesture classification using human participants. Our study showed that a certain size threshold needs to be maintained to ensure that human performance does not fall below the chance level and that the quality of the data obtained from these methods is not uniform. It also highlights some discrepancies between human and machine-learning approaches to gesture classification using event data. Human accuracy is higher than that of machine in specific reduction factors for the Synchronous Temporal Funnelling and Mutual and Separate SNN techniques. This study sheds light on the potential limitations of event data downscaling and provides insights into the human perception of gesture classification using event data. The findings of this study have implications for the design and implementation of embedded computer vision systems that rely on event data and may also inform the development of more accurate ML algorithms for gesture recognition.

The spatial downscaling methods subsequently allowed us to implement a software version of the neuromorphic foveation architecture described in Chapter 8.

CHAPTER 7

Saliency detection by event density

The combined use of SNN and event cameras is gaining momentum in the field of embedded computer vision as they promise to reduce latency and computational resource requests. However, state-of-the-art embedded neuromorphic models show little interest in modifying input data to optimise model performance, memory usage, latency and power consumption. We propose here to leverage visual attention to optimise this trade-off. Visual attention can be defined as the behavioural and cognitive process of selectively focusing on a discrete aspect of sensory cues while disregarding other perceivable information. This biological mechanism, more specifically saliency detection, has long been used in computer vision to drive the analysis only on relevant parts of images or videos for further processing. The recent advent of event cameras raises the question of adapting attention and saliency to the unconventional type of such sensors' output.
We introduce in this chapter a neuromorphic model of spatiotemporal salient selection, which simultaneously outputs one or multiple segregated objects of interest detected in an event-based scene. We propose to identify regions of interest as those corresponding to a high spatiotemporal density of events, which allows us to rely on intrinsic spiking neural network dynamics and online adaptation rules. We experimentally validate our model on the open DVS128 Gesture Dataset with implementations on CPU, Loihi and Kraken.

7.1 Introduction 119
7.2 Saliency detection 121
7.2.1 Input layer 121
7.2.2 Saliency detector – soft exponential Winner-Takes-All . . . 121
7.2.3 Saliency detector – weight adaptation rule 123
7.3 Spatiotemporal filtering — saliency detection of one object of interest at a time 124
7.3.1 Network architecture 124
7.3.2 Threshold adaptation rule 124
7.3.3 Implementation on neuromorphic hardware 126
7.3.3.1 PyNN 126
7.3.3.2 Loihi 126
7.3.3.3 SpiNNaker 126
7.3.3.4 Kraken 127
7.4 Multi spatiotemporal filtering — simultaneous saliency detection of multiple objects of interest 128
7.5 Experimental validation 131
7.5.1 Visual input data 131
7.5.1.1 Three-gesture dataset 131
7.5.1.2 Control and random symmetric combinations . . 131
7.5.1.3 Prophesee Pedestrian 132
7.5.2 Validation of the attention selection of one object of interest 132
7.5.2.1 DVS 128 Gesture validation 134
7.5.2.2 Three-gesture dataset validation 136
7.5.2.3 Control dataset validation 139
7.5.2.4 Prophesee Pedestrian 140
7.5.3 Validation of the simultaneous attention selection of multiple objects of interest 142
7.5.4 Comparison with State-of-the-Art approaches 144
7.6 Conclusion 144

7.1 Introduction

Embedded systems and even early realised neuromorphic embedded processors are limited in terms of memory bandwidth. During this PhD, we first attempted to address the issue by proposing neuromorphic and non-neuromorphic spatial reduction techniques (see Chapter 6). However, the trade-off between the quantity and the quality of the reduced event data is not optimal, which could be explained by a lack of emphasis on the salient elements of the visual scene during data reduction. Indeed, we believe that detecting RoIs in the original visual scene to select the corresponding OoIs (i.e. the events taking place in the detected RoIs) is a more promising approach to increase accuracy, reducing both latency and memory requirements.

Building on top of the limited number of existing neuromorphic visual attention models (see Section 5.1.2), we implemented a low-latency neuromorphic model of salient spatiotemporal selection, which simultaneously outputs one or multiple OoIs detected in an event-based scene.

Here we based our work on the hypothesis that in an event-driven scene, the human eye is attracted by a cluster of events that are close in time and space that one could easily assimilate to an object and try to comprehend as such. In order to simulate a mechanism of bottom-up visual attention similar to the behaviour described precedently, we have created a model that focuses on the regions corresponding to a high spatiotemporal density of events. In this work, we differentiate between RoIs and OoIs: we consider that the RoI (or "salient region") corresponds to the sensor's spatiotemporal region where interesting events are emitted, while the OoI (or "salient object") is the object one can reconstruct from those events. Let's visualise this difference with the example of the DVS 128 Dataset: in Fig. 7.1, the red square indicates the RoI, i.e. the region where many events take place closely in time and space; and the purple outline indicates the OoI, i.e. the events identified as salient and that can be assimilated to an "object" (here, the hand). One RoI can englobe different OoIs and does not influence their shape.

Figure 7.1 – Visual example of the difference between an RoI (outlined in red) and an OoI (outlined in purple), on a sample extracted from the dataset DVS128 Gesture.

The various mechanisms described in this chapter rely solely on intrinsic SNN dynamics and dynamic adaptation rules applied to synaptic weights and neural thresholds. Additionally, they directly take the event data as input by considering each event (with no regard for the polarity) as a spike emitted by a visual organ (for example, the retina). This is a crucial feature as it leads to minimising the latency since it does not require the conversion of spiking events into a frame. The saliency detector is not specialised for any specific context or any specific shape (in other words, there is no training phase), which allows for a good generalisation ability of the network. Our proposed architectures, implemented with LIF neurons due to their simplicity, are designed to be lightweight enough to run in real time.

In order to validate our models, we experimentally validate their implementation on CPU, Loihi, SpiNN-3 and Kraken (see Section 2.2) in collaboration with Michele Magno's group at ETH Zürich and researchers at IMRA Europe (Sophia Antipolis, France): the research assistant Antonio Vitale deployed the model on Loihi ; Dr Alfio di Mauro and his intern Robin Hunziker took care of the implementation on Kraken ; and the intern Hugo Bulzomi * deployed the model on SpiNN-3 and applied to a pedestrian detection task with the help of his co-supervisor Mr Yuta Nakano. We qualitatively and quantitatively assess the RoI detection and OoI selection performed on the DVS 128 Gesture dataset (Amir et al., 2017). In summary, our contribution:

— is able to simultaneously select multiple OoIs present in the visual scene in the visual scene up to 50% of the time, with no training phase ;
— selects at least one OoI with a delay of less than $15ms$, and reaching $5ms$ in the best cases ;
— selects OoIs with quality leading to a classification performance reaching 73% of the original data with a reduction of the sensor's activated surface by a factor of 1000 ;
— should be implemented on neuromorphic hardware such as SpiNNaker, Loihi or Kraken to optimise the simulation time and energy consumption, as we demonstrate that it is amenable to resource-constrained embedded neuromorphic platforms ;
— achieves all the above with a number of neurons reduced by 10% to 46% compared to the state-of-the-art.

To the best of our knowledge, it is the first neuromorphic model able to simultaneously select multiple OoIs. The implementation in PyNN (Python library) of the mode †. The results presented in this chapter were published in four articles: the model selecting one OoI and its implementation on CPU and Loihi appear in the proceedings of AICAS 2022 (Gruel, Vitale, Martinet, & Magno, 2022), its implementation on Kraken is discussed in the proceedings of AICAS 2023 (Gruel, di Mauro, et al., 2023) and first steps towards a complete implementation on SpiNNaker in the context of pedestrian recognition are described in the proceedings of MVA 2023 (Bulzomi, Gruel, Nakano, Fujita, & Martinet, 2023). Finally, the model of simultaneous multi-OoIs selection is introduced in the proceedings of IJCNN 2023 (Gruel, Martinet, & Magno, 2023a). The latter was also presented during ORASIS 2023 (Gruel, Martinet, & Magno, 2023b). It is to be noted that a first paper presenting a global bibliographic review of visual attention in biology and associated neuromorphic models was accepted at CBMI 2021 (Gruel & Martinet, 2021) ; however, the first implementation of visual attention with SNN on event-data introduced in this paper did not lead to conclusive results and we shifted our research direction to the model described below.

∗. Note that while Hugo Bulzomi was an intern in both i3S and IMRA at the time of submission, this work is technically more of a collaboration as I was not directly his supervisor.

†. The different implementations of the model can be found online at https://github.com/amygruel/FoveationStakes_DVS

7.2 Saliency detection

We introduced a saliency detection by event density which integrates the events produced by each pixel at a low resolution and outputs a set of coordinates for one or multiple RoIs. We define the RoIs as the spatial regions of the sensor where the amount of events received over a certain amount of time is more important than elsewhere over the whole scene, i.e. a region where the number of events is significantly big in a small spatiotemporal window. This visual attention mechanism is thus *bottom up* (i.e. independent from any previously set motivation or rule) and *covert* (i.e. without simulated saccadic eye movements).

Fig. 7.2A presents the architecture of our saliency detector; it is formed by the "Input events" and "Saliency detector" layers.

7.2.1 Input layer

The input layer translates the events generated by the sensor into spikes, with no regard to the polarity. One neuron is assimilated to one pixel of the sensor. The spikes produced by the input layer are sent to the saliency detector via an excitatory downscaling connection. This corresponds to a convolutional layer with a kernel size $S \times S$ and a stride S, without padding. The input neurons are separated into non-overlapping square regions of size $S \times S$. Each neuron in the input layer's subregions is connected to one corresponding neuron in the saliency detector layer, as illustrated in Fig. 7.2A.

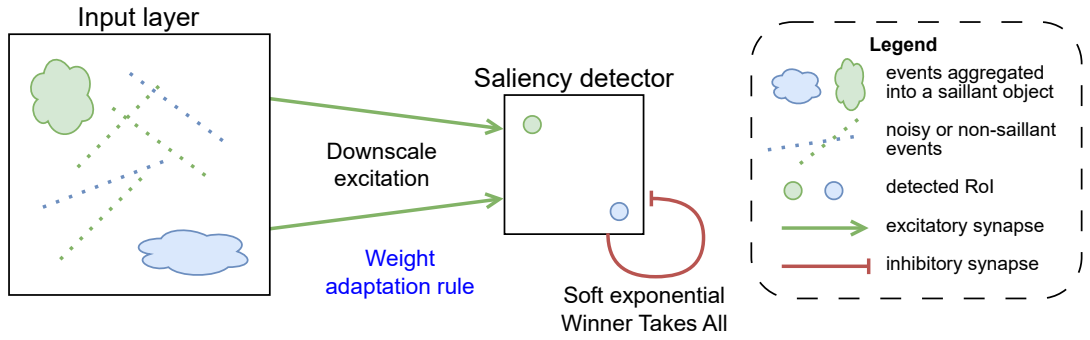
7.2.2 Saliency detector – soft exponential Winner-Takes-All

The model aggregates the neurons activated in the "Saliency detector" layer, corresponding to the salient regions, into distinct segments using a soft exponential WTA. The choice of using WTA instead of another clustering algorithm was made to respect bio-plausibility and implement clustering using a native, SNN-based strategy. Since a strong WTA leads to the activation of only one neuron in the layer and multiple RoI are to be detected by the network, the soft WTA weight has been set experimentally to 0.02. Indeed, the higher the weight, the higher the lateral inhibition and the stronger the selection between the neuronal activations; a strong weight leads to the detection of only one RoI, while a softer one (such as the one we settled on) allows for the simultaneous detection of multiple RoIs.

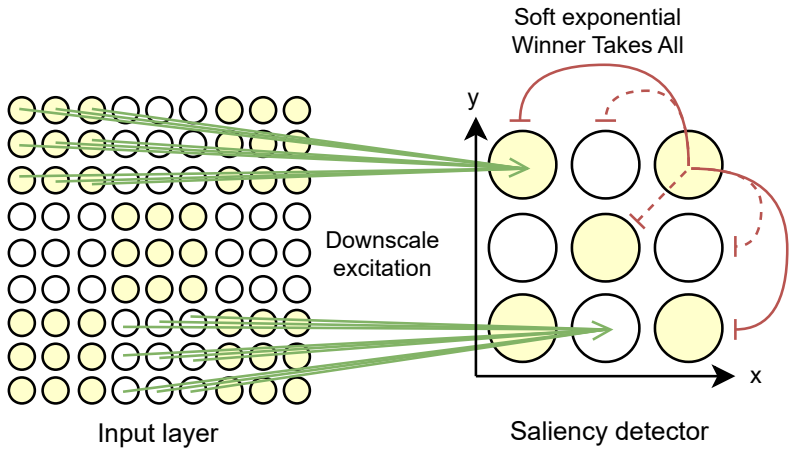
In the case of the saliency detector, a specific exponential WTA is implemented according to the radial basis function Eq. 7.1 in order to allow RoI of arbitrary sizes:

$$\omega_{WTA} = max(\frac{e^d}{w \times h}, \omega_{max}) \tag{7.1}$$

where d corresponds to the Euclidean distance in number of neurons between the active and target neuron subject to inhibition, and w and h are the width and height of the layer. The weight ω_{WTA} has an upper bound of $w\omega_{max} = 50$. The Euclidean distance d can be calculated thanks to the arrangement of the neurons of the "Saliency detector" layer according to an orthogonal reference frame (2D grid): each neuron has a set of (x, y) coordinates. As illustrated in Fig. 7.2B, the higher the distance between the inhibiting neuron and its neighbours, the higher is the inhibition. Another visualisation of this mechanism is provided by Fig. 7.3: the close neighbours of the activated neuron are almost not inhibited, while the neurons further away are increasingly inhibited, up



A) Global architecture



B) Topology

Figure 7.2 – SNN model used to detect saliency by event density.
A) The blue and green dots and shapes in the input layer corresponds to input events: the dots represent noisy or non-saillant events and the "cloud" shapes illustrate events aggregated into saillant objects (OoIs).
B) Precise depiction of the activating convolution between the Input and the Saliency detector layers as well as the soft exponential WTA on the Saliency detector. In this example, the sensor is of size 9×9 pixels and the convolution has a kernel size of $S \times S = 3 \times 3$. Each neuron is depicted as a circle, and the increased size in the Saliency detector layer simply allows better visualisation of the topology. The neurons are spatially positioned as a 2D rectangle. The red connections depict the lateral inhibition implemented by the soft exponential WTA, and the continuity of the lines corresponds to the intensity of the inhibition: neurons far from the activated one are highly inhibited (continuous lines), while those close are softly inhibited (dashed lines).

to $\omega_{WTA} = \omega_{max}$. The maximum inhibition weight corresponds to the weight of the WTA as described above, i.e. the weight by which a traditional WTA inhibits all the neurons in the layer.

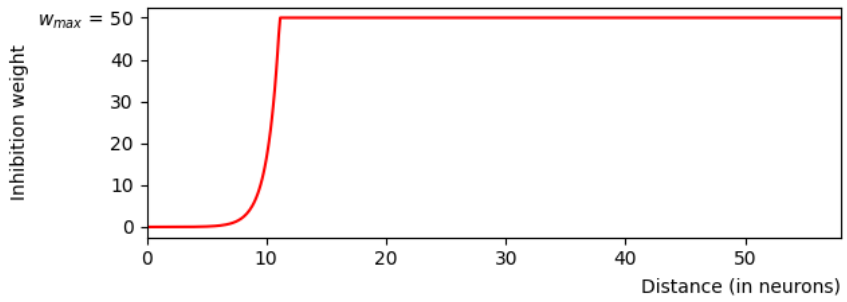


Figure 7.3 – Bounded exponential WTA used as a radial basis function in the attention model.

7.2.3 Saliency detector – weight adaptation rule

Finally, the adaptive detection of saliency in this layer is enabled by a dynamic weight adaptation rule between the input layer and the saliency detector, inspired by Hebb's rule: "cells that fire together wire together" (Hebb, 1949). This rule is implemented by increasing or decreasing the weights of synapses which pre-synaptic neurons have recently fired, as we consider them to be more likely to fire again soon (see Eq. 7.2).

$$\omega(t+1) = \begin{cases} \min(\omega(t) + \Delta\omega, \omega_{max}) & \text{if } ft_{input} \geq t-1 \\ \omega_{reset} & \text{if } ft_{input} < t - \Delta t \\ \omega(t) & \text{otherwise} \end{cases} \tag{7.2}$$

where $\omega(t)$ is the weight at the simulation step t of the synapse to which is applied the dynamic weight adaptation rule, $\Delta\omega$ the positive weight variation at each simulation step, ω_{reset} and ω_{max} the lower and upper bounds of the synaptic weight, ft_{input} the firing time of the last spike transmitted by the synapse and Δt the delay before the synaptic weight is set back to ω_{reset}. Fig. 7.4 illustrates the behaviour of this rule.

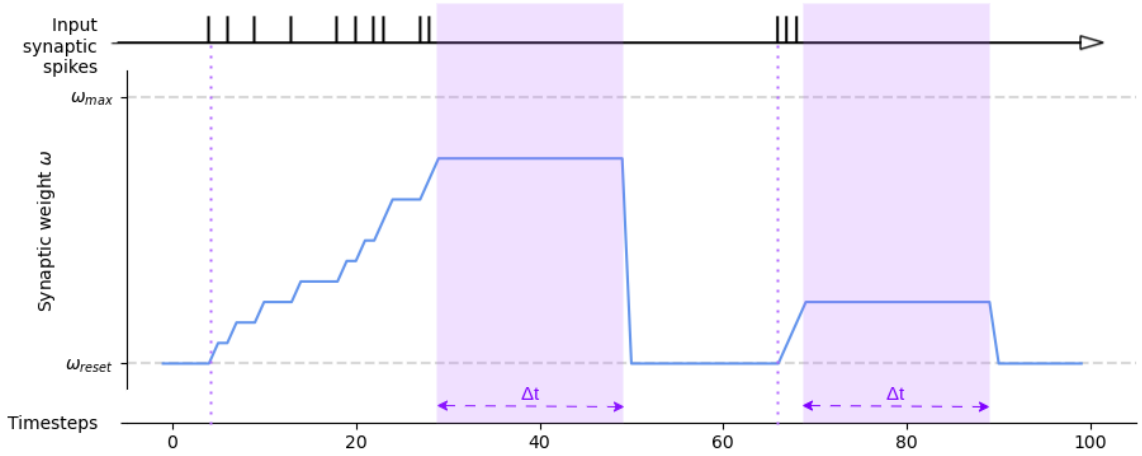


Figure 7.4 – Visual illustration of the behaviour of the weight adaptation rule.

The architecture described is simulated with PyNN, a simulator-independent Python interface for SNN simulators (Davison et al., 2009), combined with NEST (NEural Simulation Tool) (Gewaltig & Diesmann, 2007), a Python simulator for SNN on CPUs.

7.3 Spatiotemporal filtering — saliency detection of one object of interest at a time

We extended the saliency detection architecture described in the section above into a neuromorphic model of spatiotemporal attention on event data, implemented on both the CPU simulator PyNN (Davison et al., 2009), Intel's neuromorphic hardware Loihi (Davies et al., 2021) and Kraken, an academic platform that includes a Spiking Neural Accelerators and RISC-V cores (Di Mauro et al., 2022). This model detects the salient regions of the input sensor ; it outputs accordingly either the first corresponding OoI (i.e., the spikes corresponding to the detected region) or the OoI with the greater event density (see Fig. 7.17 for a complete analysis of this behaviour and its bio-plausibility).

7.3.1 Network architecture

The architecture of this model is described in Fig. 7.5It achieves this task using a light architecture: three SNN layers linked by two sets of excitatory synapses (downscale connection between the input and the RoI detector, and One-to-One connection between the input and the output layers) and two sets of inhibitory synapses implementing WTA mechanisms. Additionally, the model can adapt to any input data thanks to two adaptive mechanisms: the weight adaptation rule described above — which favours the activation of the RoI detector depending on its previous activations ; and a threshold adaptation rule — which promotes the passage of spikes in the output regions corresponding to the salient regions identified by the RoI detector and hinders this passage in other places.

The last layer is the output layer, of the same size as the input layer. It receives the activity of the input layer through one-to-one connectors (i.e. each input neuron is solely connected to the neuron located at the same coordinates in the output layer). Only the spikes corresponding to the RoIs identified by the saliency detector, thus forming the OoIs, are emitted by this layer thanks to an exponential WTA mechanism similar to the one described in Eq. 7.1, implemented with a stronger weight value to prevent other neurons in the output layer to spike when one OoI has already been selected and let through.

7.3.2 Threshold adaptation rule

In order to maintain the same information as the OoIs identified in the original data, we set up a dynamic threshold adaptation rule aiming to facilitate the neuronal spiking at the salient coordinates and to hinder it in other neurons of the output layer. The facilitation is implemented *via* the decrease of the neuronal threshold closer to the resting value of each neuron whenever the corresponding neurons spike in the saliency detector layer. Similarly, the hindering is implemented *via* the increase of the neuronal threshold, further from its resting value, when the saliency detector identifies no activity over a long period of time. This mechanism is described in Eq. 7.3.

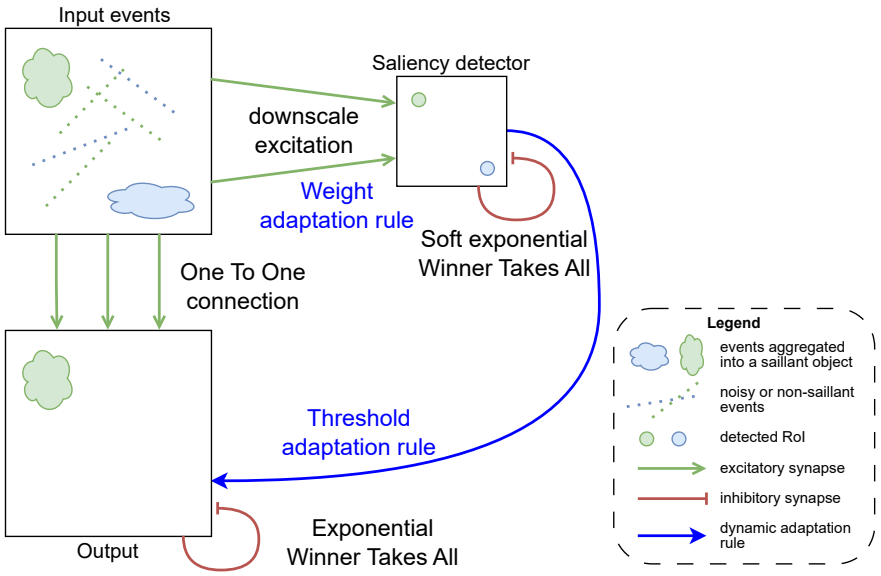


Figure 7.5 – Architecture of the spatiotemporal attention model filtering one OoI at a time.

$$\theta(t+1) = \begin{cases} \max(\theta(t) - \Delta\theta, \theta_{reset}) & \text{if } ft_{saliency} \geq t-1 \\ \min(\theta(t) + \Delta\theta, \theta_{max}) & \text{if } ft_{saliency} < t - t_\delta \\ \theta(t) & \text{otherwise} \end{cases} \tag{7.3}$$

where $\theta(t)$ is the threshold at simulation step t of the neuron to which is applied the dynamic weight adaptation rule, $\Delta\theta$ is the positive threshold variation at each simulation step, $ft_{saliency}$ corresponds to the firing time of the last spike transmitted by the saliency detector in the corresponding neuronal regions and t_δ is the delay before the neuronal spiking is hindered i.e. before the neuronal threshold is increased. Finally, θ_{max} and θ_{reset} are respectively the upper and lower bound (i.e. the reset value) of the neuronal threshold. Note that the $ft_{saliency}$ used here corresponds to the ft_{input} used in Eq. 7.2. However, while the weight adaptation rule in Eq. 7.2 was applied to the synapse linking the input and the saliency detector relies on the activity of its post-synaptic neurons, here the thresholds modifications carried out by the adaptation rule do not rely on the post-synaptic activity (i.e. the output layer's activity) but on the saliency detector acting as an external supervisor. Fig. 7.6 illustrates the behaviour of this rule.

This filtering by dynamic threshold adaptation requires first identifying the RoIs in the sensor before lowering the corresponding threshold, in order to prevent excessive unwanted output at the beginning of the simulation. Thus the initial thresholds are set to a high value in order to let the saliency detector activate first. We demonstrate in the following experimental validation that such a plasticity rule is highly preferable to a simple system of synaptic activations and inhibitions. Indeed, applying an excitating activation to the neurons corresponding to the salient regions identified by the saliency detector would compete with the input activation and saturate these neurons. This would "blur" the spikes from the input data, i.e. create artefact due to the additional activation of the saliency detector, and cause the filter to lose spatiotemporal accuracy.

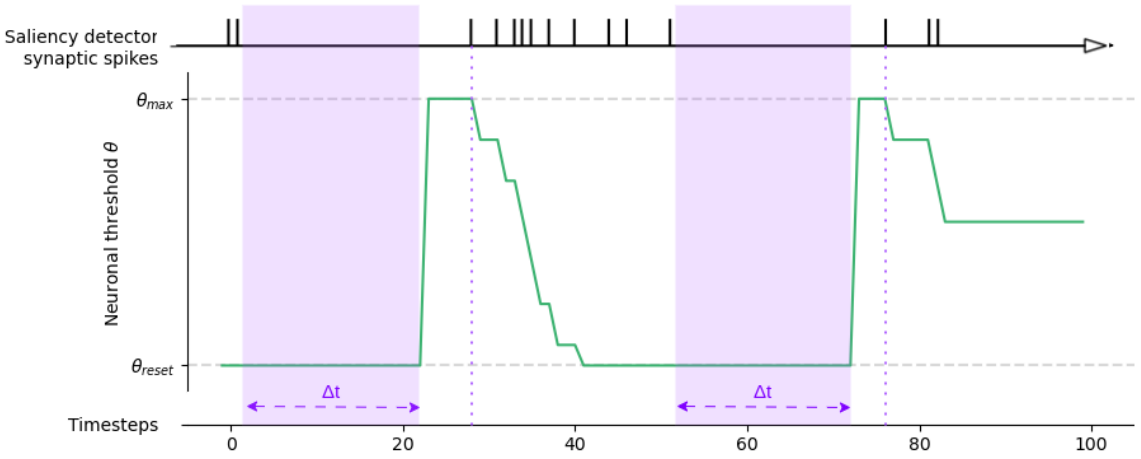


Figure 7.6 – Visual illustration of the behaviour of the threshold adaptation rule implemented in the saliency detection of one output.

7.3.3 Implementation on neuromorphic hardware

7.3.3.1 PyNN

The first experimental validation of the model described in this section was originally performed using PyNN combined with NEST (NEural Simulation Tool) (see Section 2.2), which allows for dynamic adaptation rules. However, this software implementation is significantly limited by the high simulation time.

7.3.3.2 Loihi

Our collaborator Antonio Vitale, from Michele Magno's group at ETH Zürich, then adapted this model with our help to Intel's neuromorphic hardware Loihi. As depicted in Fig. 7.5, the key elements in the presented SNN are both the weight and threshold adaptation mechanisms. Currently, the Loihi platform does not support these features (i.e. the modification of neuronal and synaptic parameters during the simulation) thus the proposed method had to be modified to implement it on the neuromorphic chip and perform an experimental validation. The input and the RoI layer were connected to the output in a way such that at any given timestep the neurons in the output layer would spike only if the corresponding neurons were active in both previous layers. The connections between the input layer and RoI detector layer, as well as the WTA mechanisms, were kept the same. The resulting 3-layer network occupies 68 neurocores on the chip and consists of 4.224 neurons and 81.716 synapses.

7.3.3.3 SpiNNaker

In collaboration with his co-supervisor at IMRA Europe, our intern Hugo Bulzomi deployed this network on SpiNNaker and applied it to a pedestrian detection task. Taking inspiration from the region proposal SNN introduced in (Acharya, Padala, & Basu, 2019), each neuron of the saliency detector is connected to its neighbourhood via excitatory synapses, in addition to being connected to all other neurons via exponential inhibitory connections. The weight adaptation rule

allows the active input regions to out-compete parts of the input with less consistent activity, such as noise. Additional noise-filtering layers such as the one described in (Acharya et al., 2019) are then not needed.

Hugo added a clustering step to this model in order to detect objects in the visual scene and create their bounding boxes. The spikes of the saliency detector are aggregated over a fixed period of time. Since each neuron in the saliency detector corresponds to a patch of size $S \times S$ on the input, we can map their activation to the detection of objects over different parts of the input window. Patches whose corresponding neurons spiked enough times are considered to contain an object. Adjacent activated patches are finally grouped together to form bounding boxes. The overall network is described in Fig. 7.7.

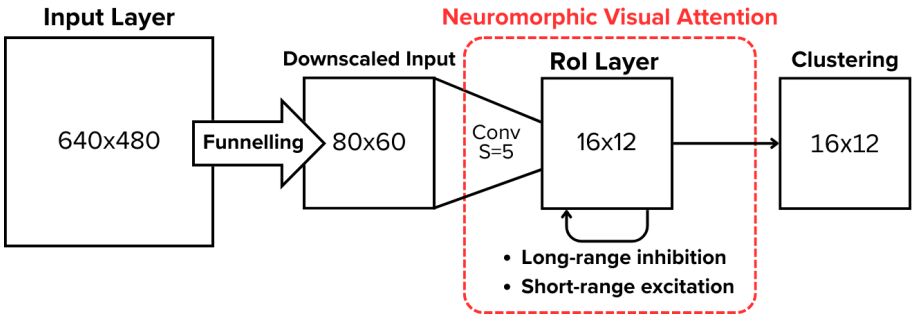


Figure 7.7 – Architecture of the saliency detection model adapted to SpiNNaker.

7.3.3.4 Kraken

A third experiment was performed on ETH Zürich academic platform Kraken, which allows for the online adaptation of neuronal and synaptic parameters and achieved quicker than real-time saliency detection, thus overcoming both PyNN and Loihi's limits. However, the Sparse Neural Engine (SNE), Kraken's dedicated neuromorphic accelerator, is primarily designed to accelerate feed-forward neural networks using the framework PyTorch (Paszke et al., 2019). This framework differs greatly from PyNN which requires a significant adaptation of the architecture's implementation to allow the deployment of the reference model on Kraken. In PyNN, each neuronal population is implemented first and then they are interconnected; PyTorch however implements a neural network by initiating layers and connections together. PyTorch-defined networks are deployed on Kraken by converting them into an intermediate neural network representation format (ONNX). Executable C code is then auto-generated by using a dedicated Kraken deployment toolchain. The low-level generated C code is completely available to the programmer and, therefore, can be modified to include dynamic threshold adaptation rules as well as custom interconnection schemes among neurons belonging to different populations.

We worked conjointly with two other persons, Alfio di Mauro and Robin Hunzinker, from Magno's group to deploy the saliency detection algorithm to Kraken. The original synaptic interconnection schemes connecting the various neuron populations and performing the attention mechanism have been implemented as equivalent neural network layers. The latter accounts for the neuron synaptic connectivity supported by SNE, i.e., 1x1 and 3x3 convolutional and fully connected synaptic connectivity, and the exact LIF neuron model implemented in SNE. Specifically, the downscale connection between the input and the RoI detector, originally implemented

as two populations of LIF neurons connected *via* an excitatory downscaling connection (i.e., a convolutional layer with no padding — see Fig. 7.5), is translated into a fully-connected synaptic connection with no bias and by setting the weights to O in location outside of the convolution kernel, i.e. where connections are not required. Indeed, the PyTorch linear layer is the most appropriate to implement All-to-All connections. The excitatory One-to-One connection between the input and the output layers (see Fig. 7.5) is translated into a convolutional layer with a kernel of size 3×3, padding of size 1 and no pooling nor bias. All the weights are null, except the kernel's central weight, which is set to 1, thus allowing only a connection between corresponding pre- and post-synaptic neurons. Additionally, exponential WTA mechanisms applied to the RoI detector and the output layer are implemented using linear layers with no bias. Since the inhibition is implemented in PyTorch by setting the weights to a negative value (the lower the value, the higher the inhibition), the exponential rule is adapted to enable a correct WTA behaviour as described in Eq. 7.4:

$$\omega_{WTA} = \max(\frac{-e^d}{w \times h} + \omega_{WTA}, 0) \tag{7.4}$$

where the notation is similar to Eq. 7.1. d corresponds to the Euclidean distance between the active and target neuron subject to inhibition, w and h are the width and height of the layer, and w_{min} is the lower bound of the weight ω_{WTA}.

7.4 Multi spatiotemporal filtering — simultaneous saliency detection of multiple objects of interest

In this section, we introduce our saliency detection and multiple OoIs selection model, which is an extension of the visual attention model presented in the previous Section 7.3 to simultaneously detect and select multiple OoIs in an event-based visual scene using intrinsic SNN dynamics. The architecture of the model depends on the number n of OoIs to select in a scene: several lateral output layers are initiated, each connected to the input layer with an All-to-All connection and each selecting a different OoIs in the visual scene. This model is designed to detect n OoIs — however, in an effort to simplify the reader's comprehension, it is depicted in Fig. 7.8 under an architecture which would allow the detection of two OoIs at most.

This new attentional selection of n OoIs features the same saliency detector, exponential WTA and weight adaptation rule as introduced in Section 7.2 ; however, we propose here a new dynamic threshold adaptation rule grounded on the activity of both the saliency detector and the lateral output layers. At each simulation timestep, for each neuron of each output layer, this dynamic rule will first identify the lateral neuronal activity at the same coordinates (see Eq. 7.5) and maximise the threshold if the corresponding lateral neurons are activated (see Eq. 7.7 and Eq. 7.8). The rule will then modify the threshold depending on the activation of the saliency detector in the corresponding neuronal regions (as described in Eq. 7.3.

The binary mask $\alpha_n(t)$, applied to the neuron n, is calculated according to the activity of the lateral layer λ at the simulation step t as follows:

$\forall \lambda \in \lambda_{\text{lateral layers}}, \forall n \in n_{neighbourhood}$,

$$\alpha_n(t+1) = \begin{cases} 0 & \text{if } t \geq \delta_\lambda \times i_\lambda \text{ and } ft_\lambda > t_\delta \\ 1 & \text{otherwise} \end{cases} \tag{7.5}$$

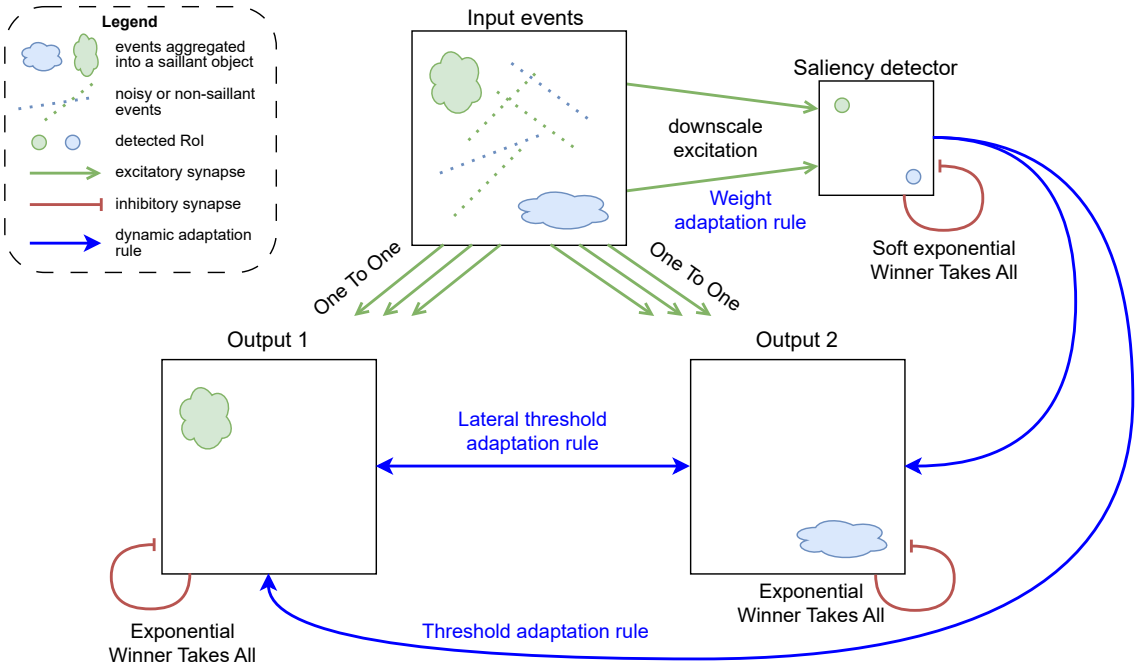


Figure 7.8 – Architecture of the multi spatiotemporal attention. Here, the maximum number of OoIs (i.e. the number of output layers) has been set to 2 for clarity.

where ft_λ corresponds to the firing time of the lateral layer λ's neuron at the same coordinates. δ_λ is the delay applied to each output's influence on its lateral populations: as the output layers are implemented sequentially, i_λ is the arbitrary identifier given to each layer and by which the δ_λ is multiplied to calculate λ's delay. $\alpha_n(t)$ is null if the corresponding neuron in at least one of the lateral layers is activated — thus hindering the selection of spikes emitted in other layers.

The neuronal threshold $\theta_{saliency}(t+1)$ is obtained according to the saliency detector activity as described in Eq. 7.3. It is then updated into $\theta_n(t+1)$ according to the lateral neuronal activity: first, the binary mask is applied to inhibit the neurons whose corresponding neurons are activated in lateral layers ; secondly, this inhibition is decayed over time based on the time elapsed since this lateral activation. In other words, $\theta_n(t+1)$ is calculated according to the result of:

$$\theta_n(t+1) = f_{decay}(f_{inhibition}(\theta_{saliency}(t+1))) \tag{7.6}$$

The function $f_{inhibition}$ introduced in Eq. 7.6 inhibits the neuronal activity according to the newly calculated $\alpha_n(t+1)$, i.e. maximises the neuronal threshold:

$$f_{inhibition}(\theta) = \begin{cases} \theta_{max} & \text{if } \alpha_n(t+1) = 0 \\ \theta & \text{otherwise} \end{cases} \tag{7.7}$$

where θ_{max} is the upper bound of the neuronal threshold. If $\alpha_n(t+1)$ is null, it means that the corresponding neurons in at least one of the lateral layers λ is active and therefore the neuron of the current layer should not fire.

The function f_{decay} introduced in Eq. 7.6 reduces the inhibition applied by $f_{inhibition}$ based on the time elapsed since this lateral activation: $\forall \lambda \in \lambda_{\text{lateral layers}}$,

$$f_{decay}(\theta) = \begin{cases} \theta - \Delta\theta & \text{if } ft_\lambda < t - t_\delta \\ \theta & \text{otherwise} \end{cases} \tag{7.8}$$

where $\Delta\theta$ is the positive threshold variation at each simulation step and $ft_\lambda(t)$ the firing time of the corresponding neuron in the lateral layer λ.

Fig. 7.9 illustrates the overall behaviour of these rules.

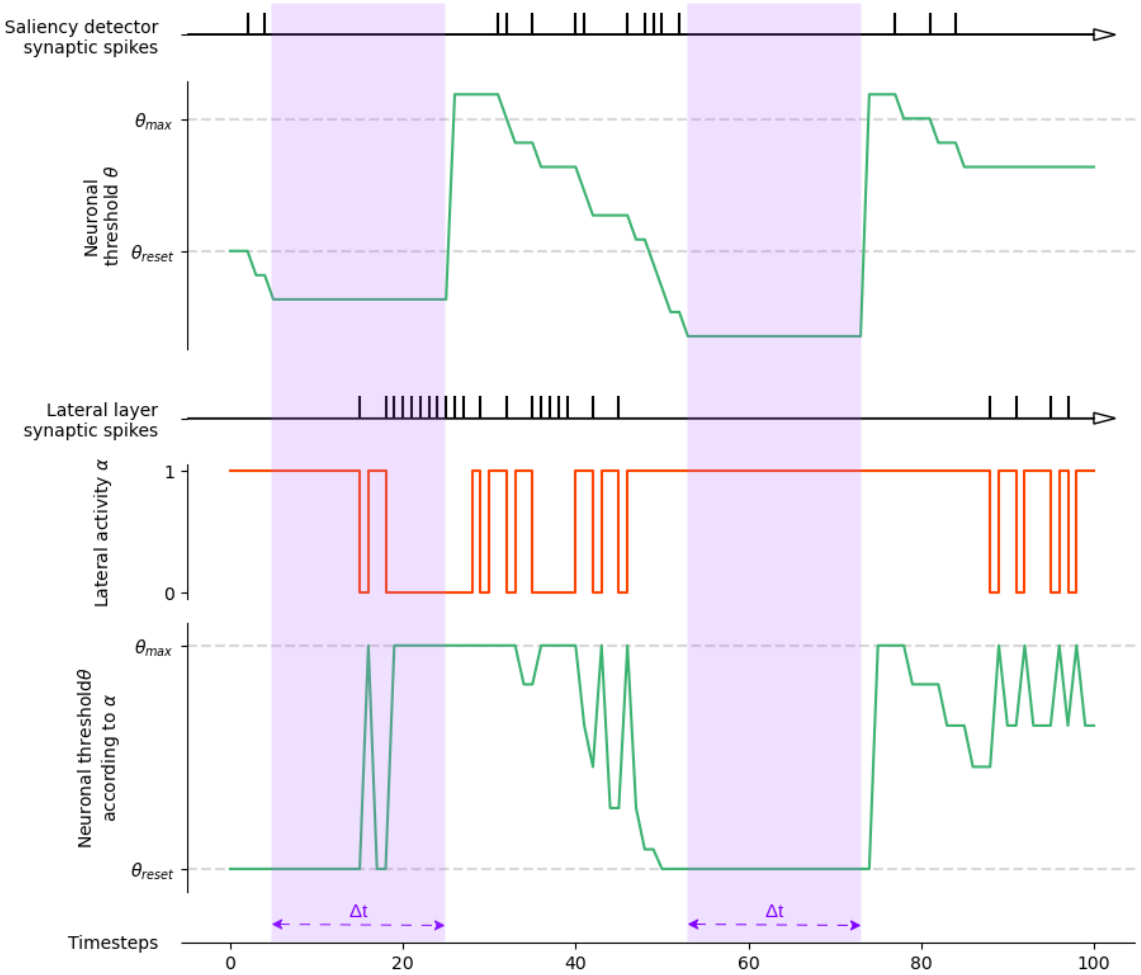


Figure 7.9 – Visual illustration of the behaviour of the threshold adaptation rule implemented in the saliency detection of multiple outputs, according to the activity and the firing time of the saliency detector and lateral layers.

7.5 Experimental validation

In the following section, we aim to demonstrate the neuromorphic models' efficient saliency detection and attentional selection of one or multiple OoIs in an event-based visual scene by verifying their quantitative and qualitative accuracies.

7.5.1 Visual input data

To this end, we need an event dataset with a controlled number of OoIs, with known spatiotemporal coordinates. Additionally, as we wish to assess a qualitative aspect by evaluating the performance of a classification task on the output OoIs, the dataset must include labelled objects corresponding to the output OoIs.

Since (to the best of our knowledge) such a dataset does not exist, we artificially created one meeting the criteria described above by stitching together various samples from DVS 128 Gesture (see Section 3.3.1) according to three protocols described below.

Additionally, we use a fourth dataset for the experiments led using SpiNNaker that will be designated as "Prophesee Pedestrians" in the remainder.

7.5.1.1 Three-gesture dataset

A first combination of DVS 128 Gesture samples was performed to experimentally validate the model implementation on PyNN et Loihi. The samples from 3 classes were used in various combinations for the demonstration of this spatiotemporal attention model: "arm roll" (class 1), "hand clap" (class 2) and "right hand clockwise" (class 7). This small dataset will be referenced as "Three-gesture dataset" in the remainder.

The samples are separated into two sub-types, either "large" or "small", according to the number of events in the sample and the ratio of active pixels in the sensor. We arbitrarily decided that "small" gestures have an active region inferior to 2.5% of the sensor and a number of events inferior to 5000: according to Fig. 7.13, classes 1 and 2 are "small" and class 7 is "large". We differentiate between those two characteristics in order to assess the influence of event flow on the saliency detection. New samples with size 128×256 were generated by concatenating horizontally samples from both types. A varying time shift to the start of the second sample is added, using shifts of $0.5ms$, $1ms$, $5ms$, $10ms$, $20ms$ and $50ms$, to demonstrate the accuracy of the filtering increasing with the time shift. We balance our dataset by allowing the first gesture to start either on the right or on the left.

The ground truth $GT(x, y, t)$ for each experiment was defined as the event stream of the first gesture only and all other events are $GT'(x, y, t)$.

7.5.1.2 Control and random symmetric combinations

Control dataset A second custom-made dataset called "control" was created by randomly selecting 50 DVS 128 Gesture samples, copying them n times and concatenating them horizontally by offsetting their x coordinates accordingly. This leads to a pool of 50 samples on which to detect one to n OoIs (i.e. the hand doing the gesture), whose spatiotemporal localisation is approximately known.

Random symmetric dataset A third custom-made dataset was similarly created by randomly selecting $50 \times n$ DVS 128 Gesture samples and combining them together to produce a final pool of 50 samples. As the intrinsic activity measures of each sample differ according to their class and may affect the saliency detection (see Fig. 7.17), for each combination $n - 1$ others are created by permuting the order of the samples in order to create a random but symmetric custom-made dataset of $50 \times n$ elements.

The two datasets described above will be respectively referred to as "control" and "random symmetric" in the rest of this paper. An example of both custom-made datasets is shown in Annexes, in Fig. A.1, where n is arbitrarily set to 2 to facilitate the reader's comprehension. It is to be noted that the actual data used as input in the following sections have been spatially reduced by 4 using the event count method we introduce in Chapter 6, as seen in Annexes Fig. A.2 and Fig. A.3, due to the limitations of PyNN in terms of the maximum number of simultaneously simulatable neurons and connections. Furthermore, only the first $100ms$ (or $1s$ for any classification task) of each sample was used to reduce the computation time.

Shift Additionally, we introduce some variations within the two kinds of custom-made datasets presented above: the n samples are combined together according to a certain temporal shift, where the $m+1$ sample's temporal coordinate is shifted by a factor $shift$ to the m^{th} sample. In the following experimental results, $shift$ varies between 0 (all the combined samples start simultaneously) and $5000\mu s$.

7.5.1.3 Prophesee Pedestrian

The fourth dataset was provided by collaborators at IMRA Europe and used by Mr Hugo Bulzomi. This dataset contains multiple recordings of both interior and exterior scenes. Objects captured are pedestrians, using a Prophesee Gen 3 of resolution 640×480. From this event representation, an implementation of Firenet (Scheerlinck et al., 2020) was used to generate grayscale video frames that were then annotated by hand with the bounding boxes of each pedestrian. Fig. 7.10 shows some examples from our dataset. Whereas some interior and simple exterior recordings involve relatively low amounts of noise, other exterior recordings in front of busy supermarkets feature large quantities of noise over the whole sensor.

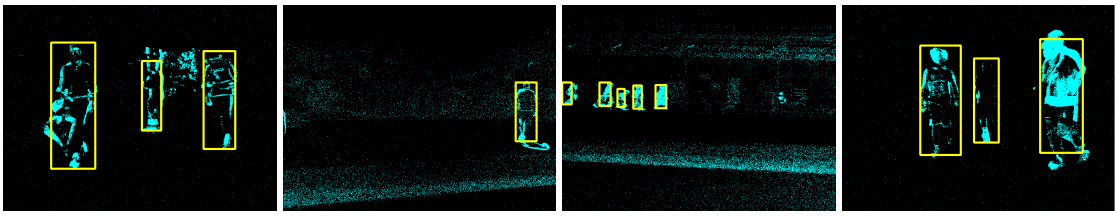


Figure 7.10 – Samples extracted from the Prophesee Pedestrian dataset, where the events are accumulated over time. Yellow squares indicate pedestrians' bounding boxes.

7.5.2 Validation of the attention selection of one object of interest

We performed different experimental validations of the attention selection of one OoI and multi-OoIs during this PhD. This section describes the results concerning the case of one output,

i.e. one OoI selected. The model was implemented on CPU using PyNN, and on two neuromorphic hardware Loihi and Kraken ; the following results compare those three implementations.

The different hyperparameters values used to implement this architecture on PyNN and Kraken are described in Tab. 7.1. The ones used to implement on Loihi were not communicated by our collaborators. The change is specifically stark in Kraken since the neural network weights are quantized to be simulated on Kraken.

Parameters	PyNN	Kraken / PyTorch
RoI detector		
Resting membrane potential	$-65mV$	0
Reset membrane potential	$-100mV$	NA
Neuronal threshold	$-25mV$	0.9
Membrane time constant	$2.5ms$	NA
Refractory period	$0.1ms$	NA
Excitatory decay time	$5ms$	NA
Inhibitory decay time	$5ms$	NA
ω_{WTA}	1	0.1
ω_{reset}	0.7	10
$\Delta\omega$	0.01	5
ω_{max}	2	100
Output layer		
Resting membrane potential	$-65mV$	0
Reset membrane potential	$-65mV$	NA
Neuronal threshold	$-20mV$	0.7
Membrane time constant	$25ms$	NA
Refractory period	$0.1ms$	NA
Excitatory decay time	$5ms$	NA
Inhibitory decay time	$5ms$	NA
ω_{WTA}	50	0.2
θ_{reset}	$-65mV$	50
θ_{max}	$0mV$	100
$\Delta\theta$	$10mV$	1

TABLE 7.1 – Neuronal and synaptic parameters used to implement the spatiotemporal attention model with one output on PyNN and Loihi.

7.5.2.1 DVS 128 Gesture validation

A first validation was performed using the original samples from the DVS 128 gesture dataset (see Sec. 3.3.1).

PyNN Using the PyNN implementation of salient selection of one OoI, we assess the evolution of the quantitative and qualitative properties of event samples from this dataset before and after attentional filtering.

The quantitative properties we evaluated and present in Fig. 7.11 are:

— the activated surface, i.e. the average percentage of pixels activated during a $50ms$ time-window in the sample, over the total number of pixels in the sensor;

— the ratio of OoI's size to the original activated surface, which is the ratio between the averaged width and height of the OoI to the width and height of the activated surface of the sensor in the original data. The width and height are respectively calculated as the difference between the highest and lowest x and y coordinates of events in the identified OoI, accumulated over a $50ms$ time window.

Each of those properties is calculated on the model's output compared to those of the original dataset, averaged for each class. The last plot in Fig. 7.11 presents the classification performance performed by the classifier implemented using PLIF (which will be referred to as "PLIF-based classifier" in the remainder of this chapter – see Sec. 4.3.1.2) on the original and output datasets.

We aim to implement an attentional selection of one or multiple OoIs that on one side significantly reduce the input data to handle, while on the other side maintaining a relatively good quality. Fig. 7.11 assures us that using our saliency detection model allows for an attentional selection of events answering to our needs, with a distribution of events in space reduced by 80% and a classification accuracy maintained at 70% of the original one's.

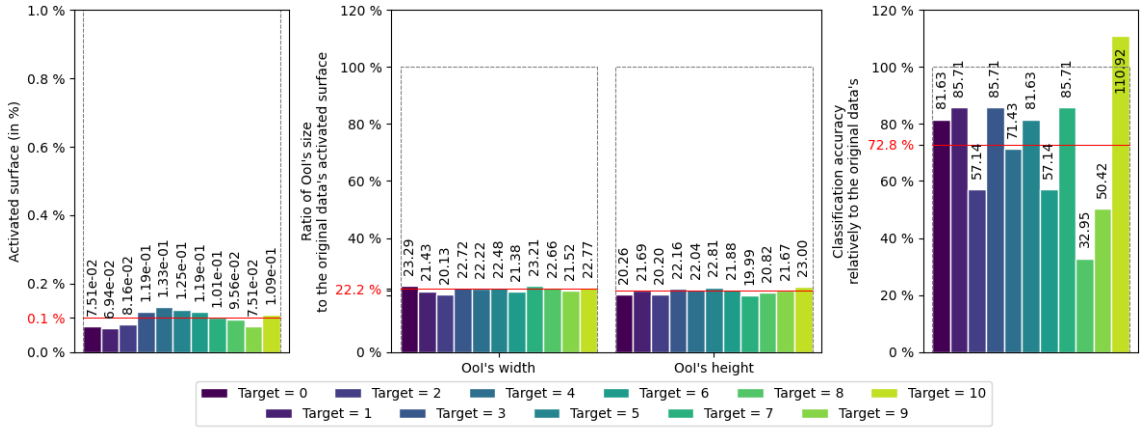


Figure 7.11 – Evolution of quantitative and qualitative properties of the DVS 128 Gesture dataset after attentional filtering, according to the dataset's various targets i.e. various labelled gestures. The values in red indicate the average value over all the targets for each property.

Loihi On Loihi, the input event data is collected for $1ms$ and downsampled by a factor of 4 using an efficient bit-shift operation on Loihi's embedded x86 cores. Each input sequence is $1.5s$

long and thus requires 1500 simulation timesteps to be fully processed. On average, measurements showed that it takes $6ms$ to process $1ms$ of input data. This ratio of 1:6 is much closer to real-time than the one obtained on CPU. These numbers show that processing event-based data on neuromorphic hardware is more efficient in terms of latency. This is an important advantage not only for running offline simulations but also paves the way for potential real-time, closed-loop applications.

Kraken We describe here the experiments conducted on the Kraken platform (see Section 2.2), and we discuss the results of such explorations compared to the performance achieved by our model on PyNN and Loihi (see previous sections).

Table 7.2 provides the energy and latency breakdown for all the relevant algorithm functional modules benchmarked on the SNE accelerator. In our experiments, we used a 6% spiking activity, the highest spiking activity measured across DVS 128 Gesture. Table 7.3 reports the overall energy and execution latency for the entire Kraken platform, including the energy consumption of the RISC-V core (Fabric Controller) that supervises the algorithm execution on SNE. As multiple tiles are executed on the SNE accelerator to run a complete saliency detection, Table 7.3 also reports the SNE idle consumption, i.e. when the accelerator is waiting for weights to be loaded, or the Fabric Controller pre-processes inputs. Kraken achieves real-time simulation run-time as it requires $0.14s$ to execute the saliency detection algorithm on $1s$ of input spiking activity from an event camera, significantly lower than both Loihi and CPU run times. Moreover, compared to the algorithm implementation executed on Loihi and on CPU, it is possible with Kraken to fully implement on-site the dynamic weight and threshold adaptation rules presented in (Gruel, Vitale, et al., 2022), which are a crucial feature for online adaptation.

Layer	Energy (in J)	Latency (in s)
Input to RoI detector	3.46×10^{-4}	1.26×10^{-2}
WTA on RoI detector	2.16×10^{-5}	7.86×10^{-4}
Input to Output	3.63×10^{-4}	1.26×10^{-2}
WTA on Output	3.12×10^{-3}	1.13×10^{-1}
Total	3.85×10^{-3}	1.39×10^{-1}

TABLE 7.2 – SNE latency and energy consumption of the saliency detection functional modules.

Kraken platform	Current (in A)	Simulation time	Energy (in J)
Fabric controller	9.5×10^{-3}	1.4×10^{-1}	8.62×10^{-4}
SNE - active (avg)	44.4×10^{-3}	1.39×10^{-1}	3.85×10^{-3}
SNE - idle	3.635×10^{-2}	3.6×10^{-4}	8.51×10^{-6}
Total	NA	1.4×10^{-1}	4.72×10^{-3}

TABLE 7.3 – Kraken platform energy consumption and latency.

Fig. 7.12 compares the simulation time of the attentional algorithm on a CPU, Loihi and Kraken. We show that with Kraken, it is possible to achieve real-time saliency detection. Furthermore,

we compare the quality of the detected OoI by comparing the classification accuracy obtained on those detected in DVS128 Gesture, performed by the PLIF-based classifier. The classification performance is 20% better on the OoIs detected by Kraken compared to PyNN: this significant increase both confirms the quality of the implementation of the original model on Kraken and can be explained by the greater number of events contained in Kraken's OoIs. Indeed, the PLIF-based classifier accumulates events in frames and performs better on a dataset rich in events.

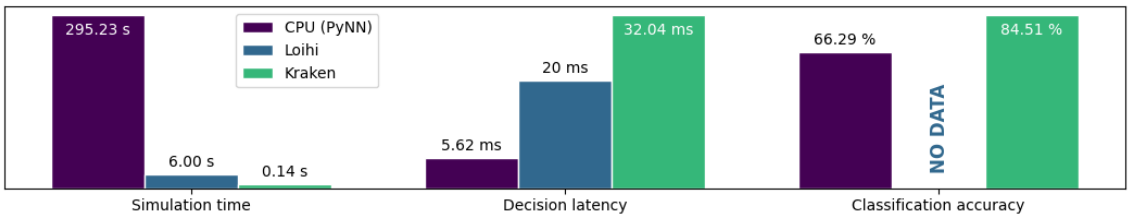


Figure 7.12 – Simulation time for an event input of $1s$, detection latency (i.e. how long should be simulated before identifying the OoIs) and PLIF-based classification accuracy of the OoIs detected in DVS128 Gesture by our saliency selection model, compared between PyNN, Loihi and Kraken implementations. As Loihi's data comes from collaborators no longer working on this project, no classification accuracy could be measured on this hardware.

7.5.2.2 Three-gesture dataset validation

A second experiment was conducted on our custom-made three-gesture dataset. We first assess the quality of event filtering by comparing in Fig. 7.13 the number of events and the active region obtained on output data and on the original data. The values are evaluated according to different temporal shifts applied to the original samples. We observe a decrease in the number of events, as well as an influence of the time shift on the quality of the selection.

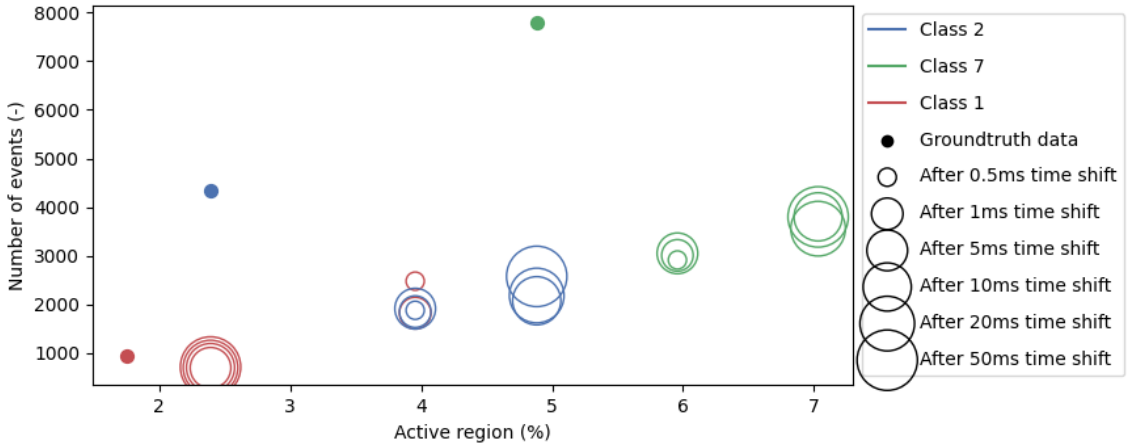


Figure 7.13 – Mean number of events in and percentage of sensor activated by the event data from the three-gesture dataset. The values are computed for the original data ("ground truth data") and for the data output by the spatiotemporal mechanism after different time shifts, using the corresponding original sample as input.

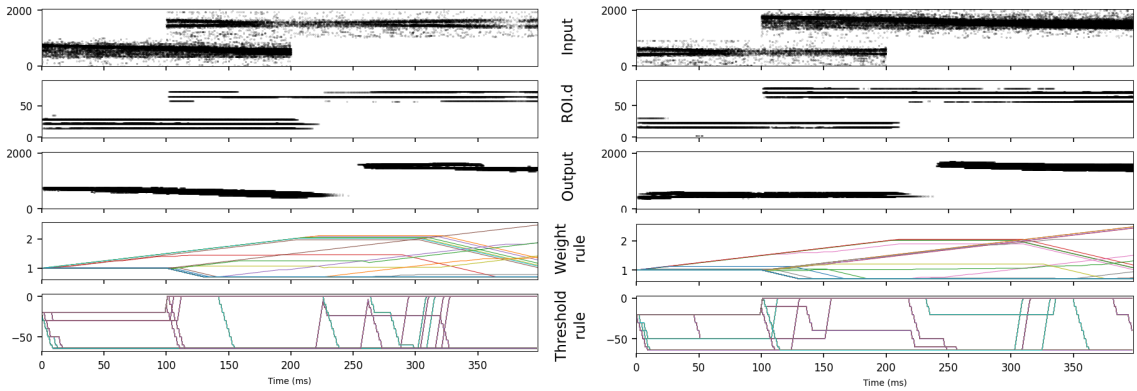


Figure 7.14 – Simulation on PyNN of the attention model applied to two samples with a time shift of 100 ms. The Input, RoI.d and Output plots in the 3 first rows represent the neurons emitting a spike at time t (x axis) respectively in the input, saliency detector and output layers respectively. The Weight and Threshold Rules correspond to the evolution of the weights and thresholds in the output layer.

Fig. 7.14 presents the results of PyNN simulations on two samples from the three-gesture dataset. The left figure presents the selection of one OoI in a sample displaying the gestures "right hand clockwise" (on the left) and "arm roll" (on the right), each gesture lasting $200ms$; while the right figure presents the simulation results on the same two gestures spatially inverted ("arm roll" on the left and "right hand clockwise" on the right). The first gesture to appear on the sensor is the one on the left in both cases. Such a permutation was tested in order to confirm that the saliency is not dictated by the spatial location of the OoI, but indeed by its timing and size (as hypothesised in the introduction of this chapter). The gesture on the left starts first, and overlaps over $100ms$ with the second one. One can clearly see that:

— the saliency detection (second plot from the top, identified by the label "ROI.d") clearly indicates the spatiotemporal RoIs;
— the output layer only s through the events corresponding to the OoI detected first by the saliency detection i.e. the first gesture performed, until the end of the corresponding $200ms$;
— the evolution of the weights of the input to saliency detector connections follows Eq. 7.2, increasing at $\omega_{max} = 2$ when the saliency detector activates or decreasing to $omega_{reset} = 0.7$ otherwise.
— the evolution of the output thresholds follows Eq. 7.3, decreasing at $\theta_{reset} = -65mV$ when the saliency detector activates or decreasing to $\theta_{max} = 0mV$ in absence of activity.

Temporal filtering Both results produced using PyNN and Loihi show that this attention model filters the region of primary interest as a function of time. The output is not significantly influenced depending on the size of the different input RoIs: the first gesture is filtered out (see Fig. 7.14) and as seen in Fig. 7.15 and 7.16, the error stays low and decreases with the shift between the two samples, independently of the characteristics of the first and second occurring input gestures, whether the size or the spatial location vary.

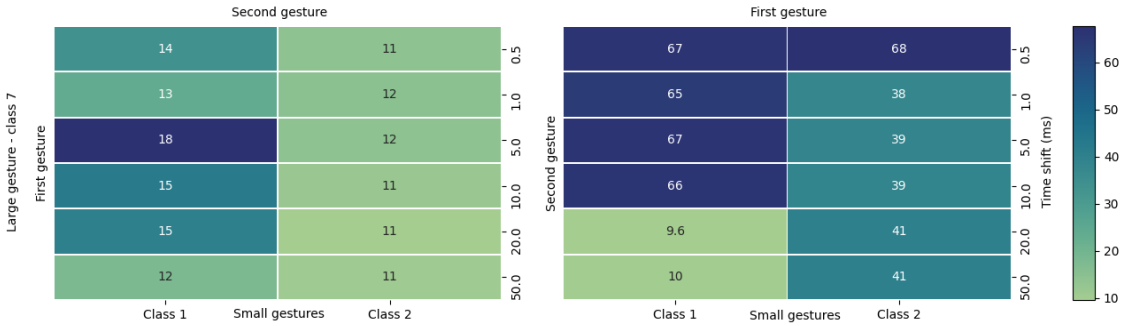


Figure 7.15 – Size invariance of the attention model with selection of one OoI, run on Loihi. Error variation of different combinations between two "small" gestures (classes 1 and 2, x-axis) and one "large" gesture (class 7, y-axis left side), according to different time shifts (y-axis right size). Irrespective of the gesture size, the error decreases as the time shift increases.

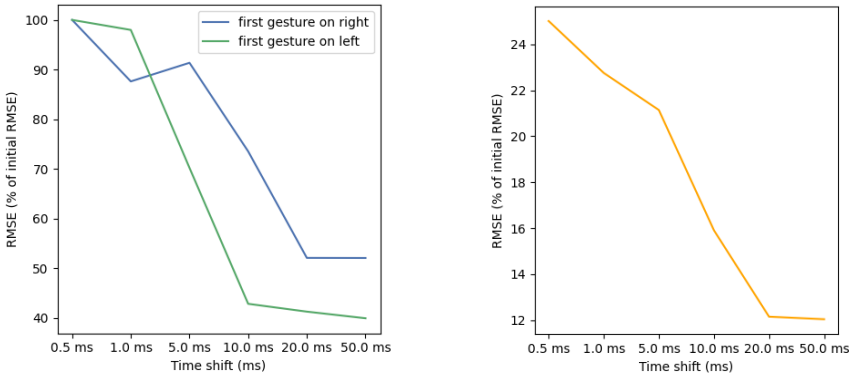


Figure 7.16 – Spatial invariance of the attention model with selection of one OoI run on Loihi, and evolution of the error as the time shift increases, the larger time shit, the better filtering from the attention model.
Right: the RMSE behaves similarly when the first gesture (i.e. the detected OoI) is on the right or on the left. This confirms that the spatial location of the OoI does not influence its detection.
Left: Average RMSE overall: the larger the time shit, the better the selection from the attention model.

Adaptation rules Currently, Loihi does not support online threshold adaptation. Fig. 7.15 and 7.16 present the evolution of an error metric established by our collaborators at ETH Zürich to assess the quality of the OoI selection: assuming that a filtered OoI should have the same activation regions as the ground truth, they implemented a Root-Mean-Square Error (RMSE) evaluating the overlap between the active regions in the ground truth and the output of our model, weighted over $100ms$ time-windows. The lower the RMSE, the better the overlap and the OoI selection, and *vice-versa*. The results displayed in Fig. 7.15 and 7.16 were obtained by running the modified architecture described in Sec. 7.3.3, and not on the PyNN simulator due to the run-time limitations mentioned above. However, to demonstrate the effectiveness of the adaptation rules, we produced some limited results using shorter input data ($400ms$ instead of $1.5s$) on the PyNN simulator: the RMSE equals 0.03 on average in the context described in Fig. 7.14 Left, and 0.16 for Fig. 7.14

Right. This difference can be explained by the number of events in the OoI: in both cases, the OoI is the one detected first, i.e. the "arm roll" (class 7) for the "Left" context and "right hand clockwise" (class 2) for the "Right" context. As can be seen in Fig. 7.13, the "arm roll" has on average a higher number of events and active region, which facilitates its detection as OoI and explains the better RMSE value obtained compared to the “right-hand clockwise”. These results point towards significant improvement using adaptation rules.

7.5.2.3 Control dataset validation

After verifying the performance of the saliency detection model with one OoI in the input data, we now wish to assess its behaviour and its bio-plausibility. Indeed, when confronted with a visual scene with multiple OoIs and asked to select only one, our hypothesis is that we expect a human being to always select either the one with the highest density of information (i.e. highest number of events and spatiotemporal density) or the one that comes first when the shift between the multiple OoIs is big enough. A bio-plausible saliency detection model would follow a similar behaviour; we thus run our model on the pool of custom-made "random symmetric" samples (see Section III.A). Fig. 7.17 displays the results of this experiment: we can see that as expected, the

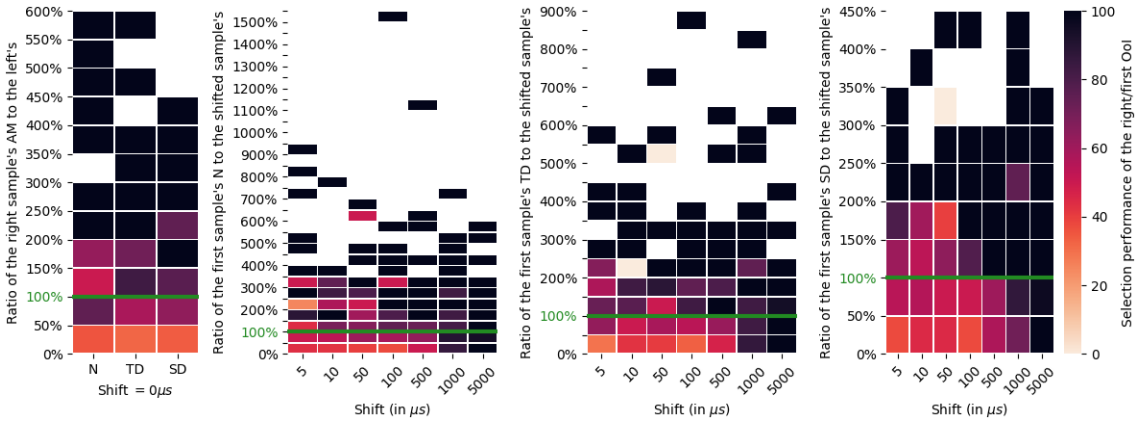


Figure 7.17 – Performance of the attentional filtering according to the shift and diverse activity measures AM. The AM used here are the number of events N, the temporal density TD and the spatial density SD.
The first plot presents the ratio of the values N, TD and SD averaged over the whole "random symmetric" dataset with no temporal shit, between the left and right concatenated samples. The colour of the blocks indicates the chance of selecting the right OoI using our model. According to this first plot, the right OoI is almost always selected first when its N, TD and SD are greater than the left OoI.
The second, third and fourth plots detailed respectively the evolution of the detection performance of the first OoI (i.e. the one appearing first in the sensor) for a set of temporal shifts, according to the ratio of the N, TD and SD respectively between the first and second OoIs. The colour of the blocks indicates the chance of detecting the first OoI: it increases with the temporal shift as well as with the ratio. This means that the longer the timeshift between two OoIs as week as the greater the AM of the first OoI compared to the second, the higher the chance for the model to detect the first one as the most interesting.

performance of detection of the first sample increases strongly with the $shift$, reaching nearly 100% for a $shift$ of $1000\mu s$ and higher. On the other side, this performance is strongly degraded when the value of the activity measure of the first sample is smaller than the second one (i.e. where the ratio of the first sample's value to the second is smaller than 100%, highlighted by a green line on the figure). This is confirmed by comparing these results to the one obtained in case of no shift (plot on the left).

We can thus conclude that detecting the saliency in event data according to the event density is bio-plausible: the detected OoI corresponds either to the first object appearing in the scene (with at least a $1000\mu s$ delay compared to the others) or the one with the highest spatial density of events. We can thus indeed call such a model a spatiotemporal attention mechanism. Additionally, Fig. A.2 in Annexes allows for a visual assessment of the quality of the OoI selection on reduced input event data, whether the dataset used is "control" (Fig. A.2a in Annexes) or "random symmetric" (Fig. A.2b in Annexes).

7.5.2.4 Prophesee Pedestrian

We deployed our saliency model on a SpiNN-3 board and applied it to the dataset Prophesee Pedestrian in order to perform event denoising and pedestrian recognition. We compared our results with two other different approaches, Hynna's detection system and RPN-SNN. Hynna's detection system (Singla et al., 2020) simply applies a connected component labelling algorithm over the input events and is meant for object detection in a similar use case. Inspired by (Acharya et al., 2019), the RPN-SNN is closer to our approach but does not use the presented visual attention mechanisms. Instead, it features an additional refractory layer of the same size as the input window to filter out the noise.

Tab. 7.4 presents the measured precision and recall based on IoU between ground-truth bounding boxes and those predicted by our approach as well as the two models introduced above. One predicted bounding box can match one or several ground-truth bounding boxes as long as the IoU is over an arbitrary threshold.

Method	Precision	Recall	F1
Hynna's detection (Singla et al., 2020)	0.252	0.896	0.393
RPN-SNN (Acharya et al., 2019)	0.513	0.752	0.610
Ours	0.550	0.739	0.631

TABLE 7.4 – Recall, accuracy, and F1 score of compared methods over the whole dataset.

While this table alone suggests that our approach performs similarly to the RPN-SNN, these results do not differentiate performances in low and high noise conditions. To better evaluate whether the attention mechanisms we presented are able to efficiently filter out noise, we evaluated each video of our dataset separately and measured the noise ratio of each video. When the noise ratio comes close to 1, the corresponding data contains a large amount of noise and *vice versa* with 0. Fig. 7.18 presents performances relative to this ratio of noise. According to this figure, RPN-SNN performs best in relatively low-noise conditions: a ratio in $[0, 0.4]$ yields better results than our approach. However, for higher noise values, our networks greatly surpass RPN-SNN both in terms of precision and recall.

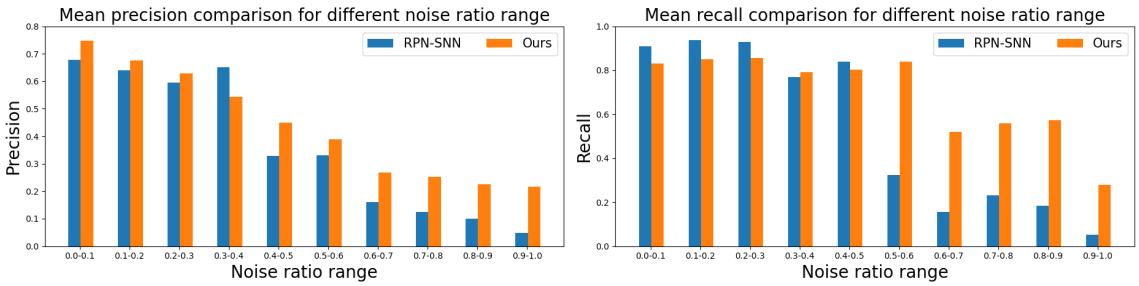


Figure 7.18 – Mean precision-recall comparison in different noise ratio intervals.

Model	Neurons	Synapses	Mean spikes
RPN-SNN (Acharya et al., 2019)	4992	**10972**	7092.6
Ours	**192**	47644	**750.5**

TABLE 7.5 – Size and mean spike counts comparison for an 640×480 input downscaled to 80×60.

Tab. 7.5 presents a size comparison between our network and the RPN-SNN. It shows the mean spike count for both networks over time windows of $150ms$ with an 80×60 input window (640×480 input downscaled by a factor 8). Our network uses 26 times fewer neurons and approximately 4.3 times more synapses while producing almost 10 times fewer spikes. This trade-off is explained by the fact that we do not use an additional refractory layer to filter out noise and instead rely on visual attention mechanisms to ignore it, which requires fewer neurons and more synapses. A lower number of neurons and spike count, as shown in Tab. 7.5, should theoretically lead to lower energy requirements: neurons need to have their potential constantly updated whereas synapses are only used sparsely during the propagation of a spike. To validate this assumption, we will now present measurements directly taken on a neuromorphic hardware implementation of the presented networks.

We chose to implement our model on SpiNNaker to better assess the cost of our approach. The very small size of our network largely allowed it to fit on the small 4-chip SpiNN-3 board which can simulate up to $18K$ neurons. Tab. 7.2 presents energy consumption measurements for the different processes of simulation for a 150ms video segment that have been slowed down by a factor of 10 as the SpiNN-3 could not keep up with the flow of events.

Process	Our network	RPN-SNN(Acharya et al., 2019)
Chips during runtime	**22.3J**	22.7J
Packet transmissions	**1.71mJ**	**1.71mJ**
Mapping	286.9mJ	**271.9mJ**
Loading	47.9J(34.5s)	**35.8J(25.2s)**
Data extraction	905.0mJ	**693.1mJ**
Total	71.5J(35.3s)	**59.6J(25.8s)**

TABLE 7.6 – Energy used by the SpiNNaker implementation. The video has been slowed 10 folds to allow the SpiNN-3 to process every event.

It should be stressed that the values from Tab. 7.6 are only meant to provide approximate energy measurements for the sake of comparison. Still, from this last table we can see that though

our higher number of synapses increases the cost of building and loading the model, our lower neuron count leads to a lower energy consumption by the chips during runtime.

7.5.3 Validation of the simultaneous attention selection of multiple objects of interest

Fig. 7.19 presents the evolution of the performance of detection of two different OoIs depending on the tuning of synaptic and neuronal parameters: ω_{WTA} which influences the number and size of detected RoIs (see Eq. 7.1), $\Delta\theta$ which moderates the impact of salient activity on the output's thresholds (see Eq. 7.3) and t_δ which delays the impact of the lateral output layers (see Eq.7.8). It highlights the importance of parameter tuning in SNN, and the difficulty to identify the correct parameters for each dataset — although we can conclude that it seems that Δt has little impact on the selection performance. According to Fig. 7.19, in order to optimise the multi-OoI detection performance, this work was experimentally validated on PyNN using the hyperparameters defined in Table 7.7.

Parameters	Values of the saliency detector	Values of the output layers
Resting membrane potential	$-65mV$	$-65mV$
Reset membrane potential	$-100mV$	$-65mV$
Neuronal threshold	$-25mV$	$-20mV$
Membrane time constant	$2.5ms$	$25ms$
Refractory period	$0.1ms$	$0.1ms$
Excitatory decay time	$5ms$	$5ms$
Inhibitory decay time	$5ms$	$5ms$
ω_{WTA}	0.5	/
$\Delta\theta$	/	$12mV$
t_δ	/	$50ms$

TABLE 7.7 – Hyperparameters used to implement the multi-OoI selection model.

We aim to assess the multi-OoIs selection performance as well as the quality of the output OoIs. Fig. 7.20 shows the evolution of the performance of multi-OoIs detection according to the temporal shift in the custom-made "control" datasets, as well as the detection latency and PLIF-based classification accuracy compared to the original (in orange). The drop in accuracy compared to the original can be explained by the smaller number of events contained in the multiple output OoIs (see Fig. A.3 in Annexes). Indeed, the PLIF-based classifier (Fang, Yu, Chen, Masquelier, et al., 2021) accumulates events in frames and therefore performs better on a dataset rich in events. Finally, Fig. A.3 in Annexes allows for a visual assessment of the quality of the OoI selection on reduced input event data, whether the dataset used is "control" (Fig. A.3a in Annexes) or "random symmetric" (Fig. A.3b in Annexes).

In certain embedded systems performing autonomous navigation for example, the latency of the model's decision may be crucial to limit any risk of an accident. On Loihi, our attentional model selecting one OoI at the time rejects more than 50% of incoming unwanted events occurring only $20ms$ after activity onset. One might fear that extending this model to multiple OoIs might lead to a significant increase in decision latency — however, Fig. 7.21 demonstrates that our model

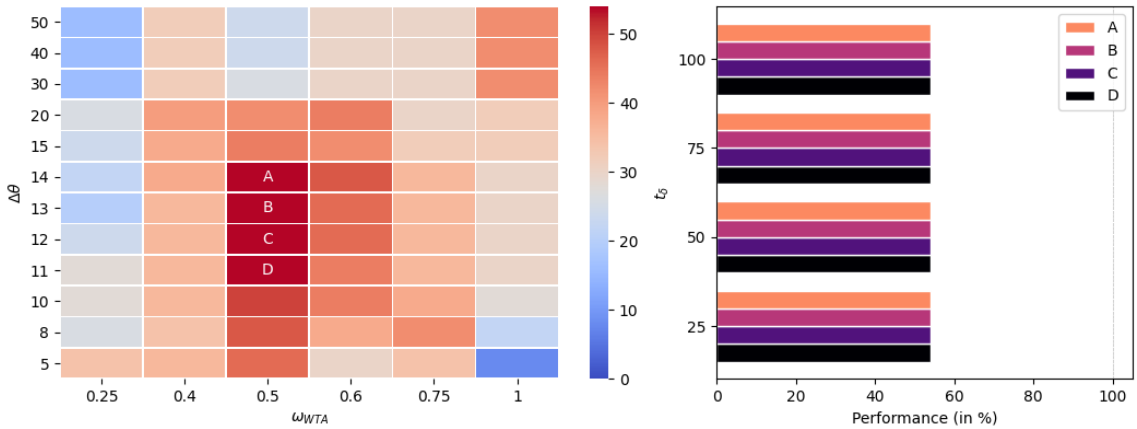


Figure 7.19 – Performance of simultaneous multi-OoIs detection according to the parameters ω_{WTA}, $\Delta\theta$ and Δt. The impact of Δt variation was observed for the parametrisations A ($\omega_{WTA} = 0.5$, $\Delta\theta = 14$), B ($\omega_{WTA} = 0.5$, $\Delta\theta = 13$), C ($\omega_{WTA} = 0.5$, $\Delta\theta = 12$) and D ($\omega_{WTA} = 0.5$, $\Delta\theta = 11$). The plot on the right highlights the lack of influence of Δt on the simultaneous multi-OoI detection.

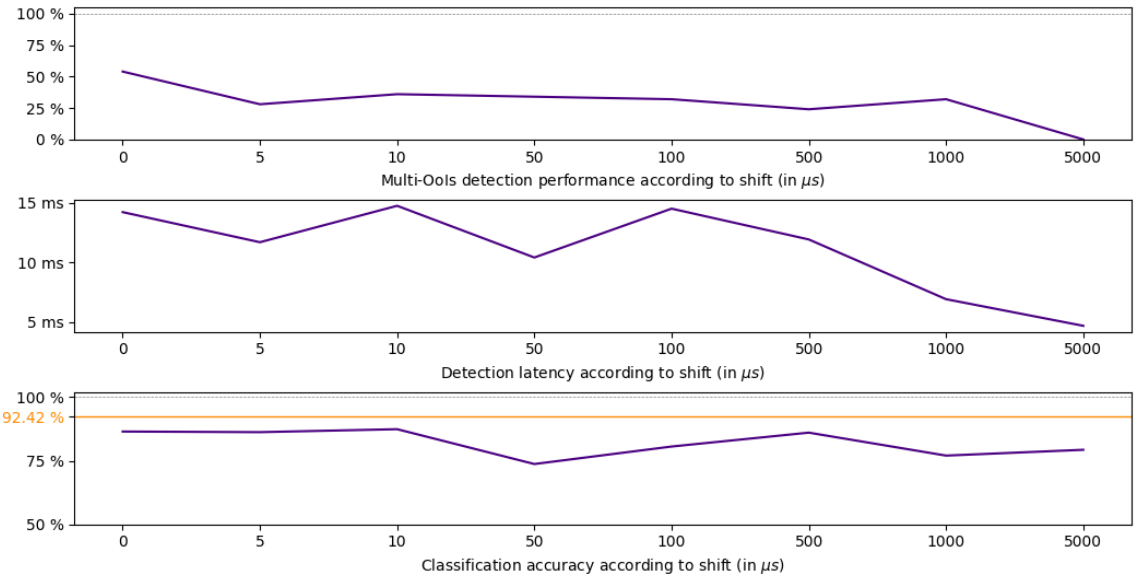


Figure 7.20 – Multi-OoIs detection performance (top), the latency of the OoIs detection (middle) and classification accuracy of the selected OoIs (bottom) of the multi-OoIs selection model on a "control" dataset. The performance presented in the top figure corresponds to the chance of our model detecting two different OoIs (and not twice the same) over the complete "control dataset" for each temporal shift. The orange line on the bottom figure indicates the classification accuracy of the whole dataset. Since we classify between 11 classes, the chance level is around 9%.

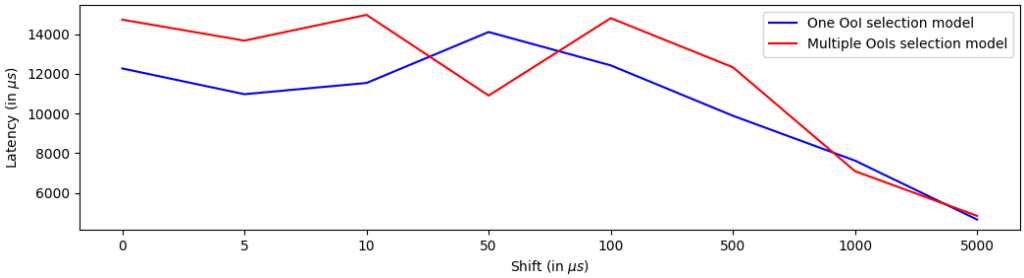


Figure 7.21 – Latency of the OoI selection our saliency detection model for one OoI (in blue) and multiple OoIs (in red) according to the shift. With the exception of the dip observed for a $50\mu s$ shift, the overall latency of OoI selection increases with the number of OoIs to be detected (from one in blue to two in red).

maintains this latency performance, which is revised to the value of $14ms$ maximum without shift and decreases down to $5ms$ for an increasing shift.

7.5.4 Comparison with State-of-the-Art approaches

Table 7.8 compares our contributions with the neuromorphic saliency models implemented in (Renner et al., 2019; D'Angelo et al., 2022). The data presented here were either retrieved directly from the information given by the authors or calculated from the description of each model. Those different metrics enhance our proposed model, whose implementation is resource-efficient while implementing additional features with lower latency.

7.6 Conclusion

We demonstrate in this chapter the significant benefits that might be derived from the application of a visual attention mechanism on event data, selectively focusing on relevant elements of sensory information thus improving the processing efficiency. This chapter described a neuromorphic algorithm for saliency detection in event data simultaneously selecting one or multiple OoIs. We highlighted the significant improvements to our results brought by adaptation rules. Incorporating such mechanisms in the design of future neuromorphic architectures will give rise to novel event-based attention applications.

Our proposed models are able to accurately detect and select n OoIs out of n initially present almost 50% of the time. The multi-OoI detection model selects at least one OoI with a delay of less than $15ms$ at most and reaching $5ms$ in the best cases. The quality of the OoIs selection is such that their classification reaches a performance of 73% the original data's while reducing the activated surface of the sensor by 1000. Our models can and should be implemented on neuromorphic hardware such as Loihi or Kraken to optimise the simulation time and energy consumption, as we demonstrate that it is amenable to resource-constrained embedded neuromorphic platforms and runs on the Kraken SoC platforms in $140ms$. Applied to a pedestrian detection task, our largely surpasses the performances of the detection system of Hynna (Singla et al., 2020) and shows to be more robust to noise than the state-of-the-art (Acharya et al., 2019). Finally, our models achieve all the above with no training phase and a reduced number of neurons and connections compared to state-of-the-art salient detection methods.

Perspectives - Applications and deployments In future works, we wish to validate this novel architecture to an increasing number of n OoIs as well as implement it on the neuromorphic hardware SpiNNaker (S. B. Furber et al., 2013), which allows for the implementation of dynamic adaptation rules, directly interfaced with an event camera. We believe in its usefulness in a real-time embedded task such as multi-object tracking or scene segmentation.

Additionally, we wish to validate the performance on our attention model on CV tasks with faster motion. Indeed, the results presented in this work were obtained on DVS 128 Gesture which, despite being a well-known dataset, requires a significant minimum observation time before decision (at least $1s$ according to (Fang, Yu, Chen, Huang, et al., 2021)), before which the latency of our attention model (at most $5ms$) is negligible.

Perspectives - Towards bio-inspiration The review of attention's biological background presented in Section 5.1.1 allows us to develop some biologically-inspired proposals to adapt the concept of attention to computational models for purposes of responding to computer vision and multimedia tasks. Indeed, if those tasks are easily achieved by the human brain thanks to attention, the neural mechanisms involved deserve to be studied more thoroughly so as to profit from its efficiency and its robust performance.

Firstly, the forebrain presents a promising implementation of top-down attention. Feature-oriented tasks with specific objectives are common in the domain of computer vision and multimedia (such as gesture recognition, object detection, etc.) and correspond fully to the forebrain's attention use.

An alternate perspective would be to exploit neuromodulator mechanisms. For instance, one could reproduce acetylcholine's *modus operandi* to enhance the capture of information from certain RoIs in visual scenes. Combined with a network mimicking the midbrain and its ability to detect the visual data relevant to the task at hand, this would lead to a higher information processing performance.

Another important avenue of investigation involves the exploitation of the link between saccadic eye movements and attention (Daucé et al., 2020), also understandable as the link between overt and covert attention. Such an attention could be achieved by implementing a retinocentric map of saliency, and then processing with a higher resolution on the place where the "*focal gaze*" is directed, i.e. where the saliency is higher.

TABLE 7.8 – Comparison between the state-of-the-art and our contributions, i.e. models of saliency selection of one OoI (fourth column) and multi-OoIs (last column). In the notation, the input data is of size $S \times S$, OL corresponds to the overlapping percentage described in (D'Angelo et al., 2022) and div to the dividing factor between the input layer and the saliency detector in our models. A numerical value was calculated for each theoretical estimation, for $S = 128$, $OL = 5\%$ and $div = 16$.

	(Renner et al., 2019)	(D'Angelo et al., 2022)	Salient selection of one OoI	Salient selection of multi-OoI
Saliency detection				
n_{layers}	2	10	2	2
$n_{neurons}$	$2S^2$	$S^2 \times (1 + \frac{4}{OL^2}) + 5$	$S^2 \times (1 + \frac{1}{div^2})$	$S^2 \times (1 + \frac{1}{div^2})$
For $S = 128$:	32,768	19,010	17,408	17,408
$n_{synapses}$	$S^2 \times (2S^2 - 1)$	$S^2 \times (4 + \frac{20}{OL^2}) + 4$	$S^2 \times (1 + \frac{S^2}{div^4} - \frac{1}{div^2})$	$S^2 \times (1 + \frac{S^2}{div^4} - \frac{1}{div^2})$
For $S = 128$:	536,854,528	65,540	1,063,936	1,063,936
Selection of OoIs				
Selection of OoI	No	No	Yes	Yes
Simultaneous multi-OoI selection	No	No	No	Yes
Detection latency	NA	$16\mu s$	$13\mu s$	$14\mu s$

CHAPTER 8

Neuromorphic foveation

Foveation can be defined as the organic action of directing the gaze towards a visual region of interest, to acquire relevant information selectively. With the recent advent of event cameras, we believe that taking advantage of this visual neuroscience mechanism would greatly improve the efficiency of event-data processing. Indeed, applying foveation to event data would allow to comprehend the visual scene while significantly reducing the amount of raw data to handle.
In this respect, we demonstrate in this chapter the stakes of neuromorphic foveation theoretically and empirically across several computer vision tasks, namely semantic segmentation and classification. We show that foveated event data has a significantly better trade-off between quantity and quality of the information conveyed than high or low resolution event data. Furthermore, this compromise extends even over fragmented datasets. The significant advantage of using foveation on embedded event data highlighted in this chapter reinforces the interest in developing a foveated camera such as the recent e-Fov DVS, which use will be similarly evaluated in future work.

8.1 Introduction **149**
8.2 Methodology — Binary foveation **150**
8.3 Experimental validation **152**
8.3.1 Performance on a classification task 152
8.3.2 Performance on a segmentation task 155
8.3.3 Quantitative assessment of the saliency detection 156
8.4 Discussion **156**
8.5 Conclusion **159**

8.1 Introduction

Traditional and neuromorphic computer vision models can have difficulties handling a great amount of data simultaneously while minimising their energy consumption, especially at a high temporal resolution. We explored in this work two potential remedies for such an issue: event data downscaling (see Chapter 6) and salient selection of OoIs in the event flow (see Chapter 7). However, we believe that the trade-off between information retention and data reduction with existing methods is not yet ideal and can be improved using neuromorphic foveation, a concept designed by Teresa Serrano-Gotarredona and Bernabe Linares-Barranco at IMSE (Sevilla, Spain). This work is a first step towards experimental validation of a neuromorphic foveated sensor.

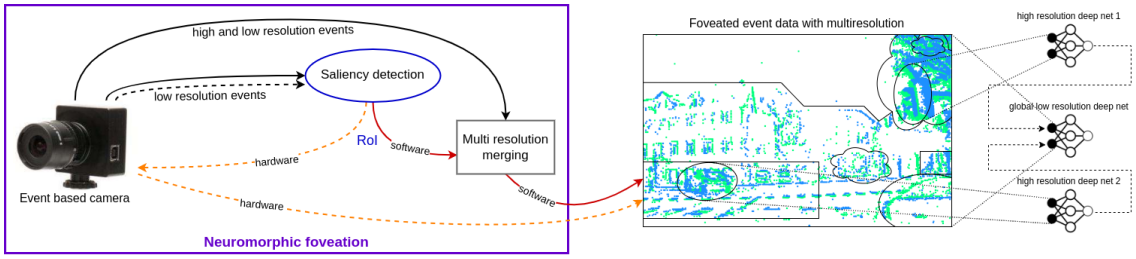


Figure 8.1 – Overview of the neuromorphic foveation concept as defined in this work. The results presented below correspond to the software neuromorphic foveation (red full arrows) as a validation for a future hardware neuromorphic foveation (orange dotted arrows). The multi-resolution event sample used as an example of the output of such a process is extracted from the DDD17 dataset (Binas et al., 2017).

To demonstrate the interest in applying foveation to event data, we define in this work the concept of neuromorphic foveation (see Fig. 8.1), which consists of a feedback loop between an event camera and a neuromorphic saliency detector, merging events at multiple resolutions according to the detected RoIs. The detection of such RoIs relies on the visual attention mechanism described in Chapter 7.2: a set of coordinates in the event sensor is considered interesting if it is strongly activated in close temporal and spatial proximity. Such a process should be allowed by a foveated DVS as described in (Serrano-Gotarredona et al., 2022) ; however, as this sensor is not yet available at the time of writing, we validate here the neuromorphic foveation concept with a first software implementation (red pathway in Fig. 8.1) before extending this feedback loop system to hardware (orange dotted pathway) in future works.

With the help of APROVIS3D collaborators, we studied the respective evolution of the amount of event data processed in a computer vision task and its accuracy when software neuromorphic foveation as described above is applied. In order to simulate the foveation, the event data will be processed at a higher or lower resolution, depending on the relevance of the spatial regions in the image at different coordinates. The proposed model goes beyond biology by allowing multiple RoIs of arbitrary size and shape.

The following sections establish a detailed outline of the different hardware and mechanisms involved in this work ; a complete description of the software neuromorphic foveation methodology ; and the experimental evaluation procedure. Our contributions are the following:

— A complete description of the binary foveation process and the different hyperparameters to facilitate the reproducibility of our neuromorphic foveation mechanism ;

— A complete comparison of the semantic segmentation performance between the original, reduced and foveated DDD17 dataset using various spatial event data downscaling, with results provided by PhD student Dalia Hareb (i3S) ;
— The evaluation of the neuromorphic foveation on a second computer vision task, i.e. the classification of the DVS 128 Gesture dataset, with results provided by Antoine Grimaldi (INT) ;
— An additional comparison of the foveation to another data pre-processing, where we study the semantic segmentation and classification performance on selected RoIs ;
— A study of the overall qualitative and quantitative aspects of saliency detection.

To the best of our knowledge, this work and our results are the first to show that foveation offers a significantly better compromise between quantity and quality of the information than high or low resolution, especially on fragmented datasets. Furthermore, we demonstrate that our saliency detector is specifically efficient on data reduced using the *event count* method.

Our code is implemented in Python using the PyNN library (Davison et al., 2009) *.
The results presented in this chapter were made publicly available on three occasions: a short abstract defining the concept and methodology was presented during the French national conference ORASIS 2021 (Gruel, Martinet, Linares-Barranco, & Serrano-Gotarredona, 2021), an extended abstract with a first proof of concept was featured in the CVPR 2022 workshop "NeuroVision: What can computer vision learn from visual neuroscience ?" (Gruel, Hareb, Martinet, Linares-Barranco, & Serrano-Gotarredona, 2022). A journal paper was also submitted to the Biological Cybernetics special issue "What Can Computer Vision Learn from Visual Neuroscience ?" (Gruel, Hareb, Grimaldi, et al., 2022).

8.2 Methodology — Binary foveation

In this work, we consider the foveation process akin to the combination of a sample's events in high resolution and low resolution using a mask, as presented by Fig. 8.2. This binary combination is a software simplification of the neuromorphic foveation concept as described above and discriminates the fovea (events in high resolution — purple region in Fig. 8.2) from the retinal periphery (low resolution ; i.e. spatially downscaled — yellow region in Fig. 8.2). The RoI is detected using the saliency detector described in Section 7.2 ; it is thus assimilated to the fovea. The saliency detection and binary foveation are performed using the hyperparameters described in Tab. 8.1.

As explained above, the software implementation of neuromorphic foveation used in this work is akin to a binary foveation process merging event data of two different resolutions together. As it is more difficult to produce higher resolution data of an original dataset, we chose to use four of the spatial event downscaling methods described in Section 6: the spatial funnelling, the event count and the linear and cubic log luminance. In order to remain consistent with a future conversion of this software foveation model into hardware, where the saliency is detected on aggregated pixels before refining the resolution at the relevant areas in order to minimise the bandwidth, we follow the process described in Fig. 8.2. The original dataset, corresponding thereafter to the denomina-

∗. The Python code is publicly available online at github.com/amygruel/FoveationStakes_DVS/

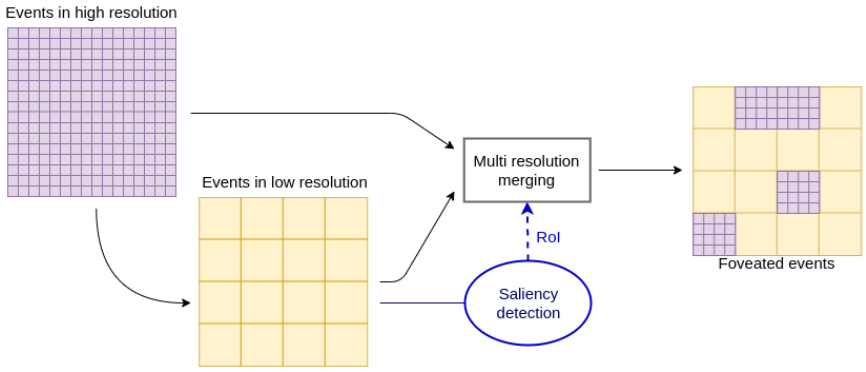


Figure 8.2 – Binary foveation of events using the corresponding high and low resolution (spatially downscaled by a dividing $factor$) and based on a known RoI.

tion "high resolution", is spatially reduced before being given as input to the saliency detector. A subsequent "multi-resolution merging" process then combines both datasets into foveated event data according to the detected RoIs.

Parameter	Values
SNN model	
Resting membrane potential	$-65mV$
Reset membrane potential	$-100mV$
Neuronal threshold	$-25mV$
Membrane time constant	$2.5ms$
Refractory period	$0.1ms$
Excitatory decay time	$5ms$
Inhibitory decay time	$5ms$
Input to saliency synapse	
S	5
ω_{init}	1
ω_{min}	0.7
$\Delta\omega$	0.01
t_d	100
Exponential WTA	
ω_{max}	0.02
w	$\frac{\text{sensor width}}{S}$
h	$\frac{\text{sensor height}}{S}$

TABLE 8.1 – Hyperparameters used to implement the saliency detection model (see Eq. 7.1 and 7.2).

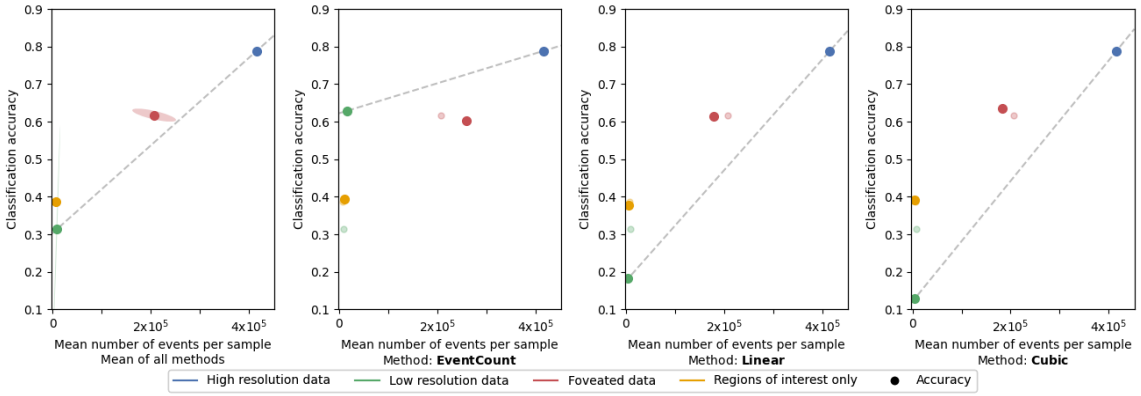


Figure 8.3 – Classification performance on DVS 128 Gesture according to the number of events in the dataset in high resolution (in blue), low resolution (in green), with only RoIs (in yellow) and after foveation (in red). The dashed line connects the results of high and low resolutions ; an ideal result would lie above this line, thus achieving a better trade-off between number of events and performance than the original and reduced data. The subplot on the left depicts the mean values and the corresponding confidence ellipsis ; it shows that the foveation applied to classification leads on average to a good trade-off between number of events and performance since the resulting accuracy lies in the upper left area. The mean values are plotted as an overlay in the subsequent subplots to assess the quality of the foveation and the reduction compared to the others. The trade-off between classification performance and number of events per sample is overall better for the foveated data, especially using the log-luminance reconstruction methods.

8.3 Experimental validation

To validate our proposed model, we apply two traditional computer vision tasks, semantic segmentation and classification, to foveated and spatially reduced datasets. All datasets are spatially downscaled by a dividing factor of 4. This section describes the datasets and the different models used to perform such tasks, as well as the comparative results. As presented in Tab. 3 and Fig. C.4 (in Annexes), the event data's properties were compared for sample in high resolution (original dataset), low resolution (spatially downscaled with factor 4) and foveated (binary combination of the previous two).

8.3.1 Performance on a classification task

We test the impact of the neuromorphic foveation on a classification task, which assigns a label to each sample of a dataset, using the HOTS model (see Section 4.3.1.3). Fig. 8.3 presents a comparison between the different versions of the DVS 128 Gesture dataset (Amir et al., 2017), i.e. in high (blue marks) and low (green marks) resolutions, after RoIs selection (yellow marks) and after foveation (red marks), according to the classification model *HOTS*'s performance. Fig. 8.3 depicts the trade-off between the accuracy, the MIoU and the mean number per sample for each reduction method — the first subplot corresponding to the mean and confidence ellipse for all methods combined. We compare the classification performance on the foveated dataset and on the events belonging solely to the detected RoIs in order to validate the interest of foveation: indeed, the RoIs contain only the most interesting information in the visual scene (according to our defini-

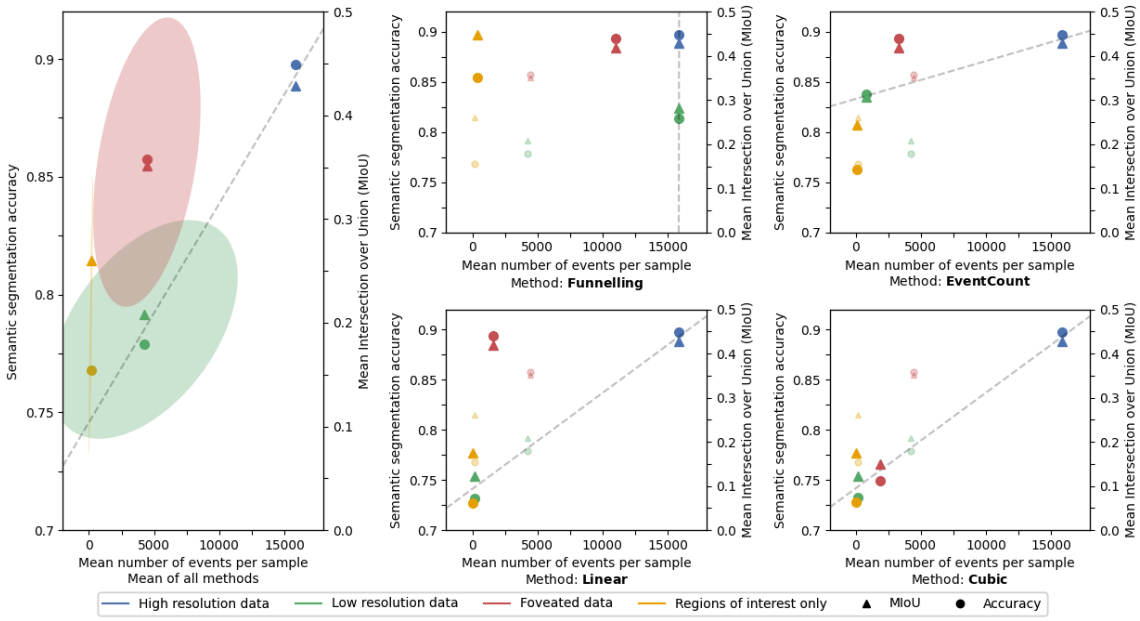


Figure 8.4 – Semantic segmentation performance according to the number of events in the dataset in high resolution (in blue), low resolution (in green), with only RoIs (in yellow) and after foveation (in red). The dashed line connects the results of high and low resolutions ; an ideal result would lie above this line, thus achieving a better trade-off between number of events and performance than the original and reduced data. The subplot on the left depicts the mean values and the corresponding confidence ellipsis ; it shows that foveated data managed to keep on average the same performance as the original dataset while decreasing by two-thirds the mean number of events to reach the low resolution's value. The mean values are plotted as an overlay in the subsequent subplots to assess the quality of the foveation and the reduction compared to the others. This highlights the clear advantage of using the methods *event count* and *linear* compared to *funnelling*.

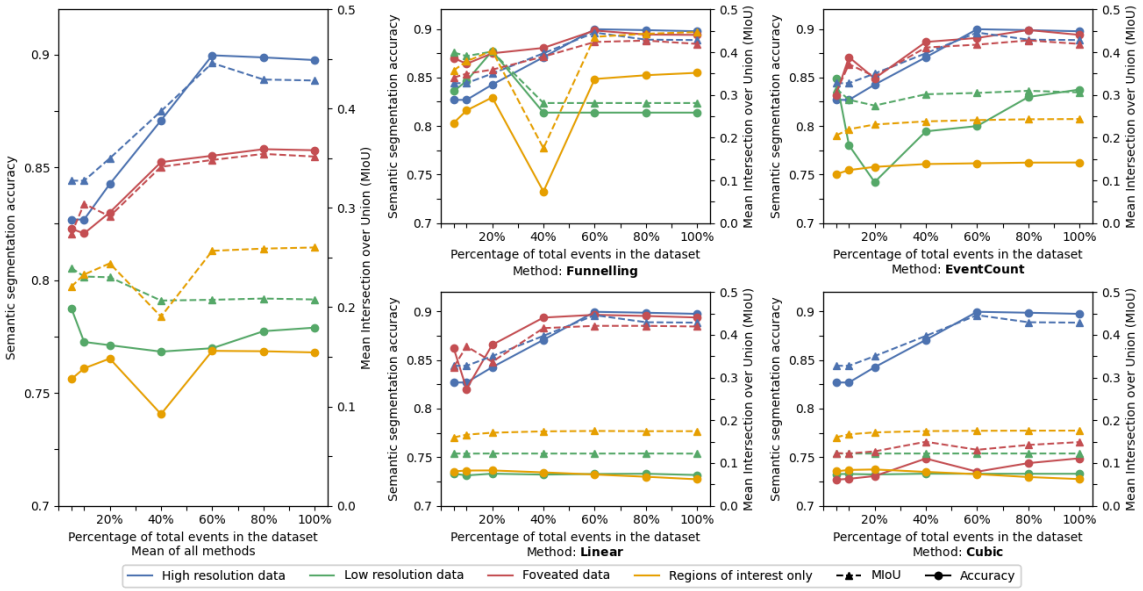


Figure 8.5 – Semantic segmentation performance evolution according to the percentage of total events in the dataset after processing for the event data in high resolution (in blue), low resolution (in green), only from RoIs (in yellow) and after foveation (in red). The different subsets are obtained by downscaling the respective datasets using the stochastic structural downscaling method for $p \in [0.2, 0.4, 0.6, 0.8, 1]$. The subplot on the left depicts the mean values and how all four types of data behave similarly. According to the subsequent subplots, it is interesting to note that, except for the *cubic* method, the foveated data overall outperforms the high-resolution data from a 60% decrease and downwards. The *funnelling* method is the one maintaining the highest accuracy for the longest, either in the foveated or reduced dataset.

tion of saliency) but might miss contextual information helping to classify the data correctly. For example, in DVS 128 Gesture one might need to know about the subject shape to understand that the moving OoI detected by the model is the arm. Foveation however keeps the information provided by the *non-interesting* regions but in a reduced format, thus providing interesting information at a high resolution and contextual, less interesting information at a lesser resolution.

8.3.2 Performance on a segmentation task

To validate the neuromorphic foveation, we evaluate it using the semantic segmentation model Ev-SegNet (see Section 3.3.3). Fig. 8.4 and 8.5 present a comparison between the different versions of the DDD17 dataset (Binas et al., 2017), i.e. in high (blue marks) and low (green marks) resolutions, after RoIs selection (yellow marks) and after foveation (red marks), according to the semantic segmentation model *Ev-SegNet*'s performance. One could wonder how a semantic segmentation model would behave on data containing solely events within the identified RoIs (yellow marks in Fig. 8.4 and 8.5). Thanks to a study performed by Dalia, we know that Ev-SegNet assigns the class 0 (for background and sky) to segments with no events ; in the case of RoIs selection, it means that most of the scene is labelled as background and the other classes only appear in the RoIs, which explains the low performance of semantic segmentation on data post RoIs selection (Fig. 8.4) compared to the classification performance observed in Fig. 8.3.

Fig. 8.4 depicts the trade-off between the accuracy, the MIoU and the mean number per sample for each reduction method — the first subplot corresponding to the mean and confidence ellipse for all methods combined. Fig. 8.5 shows the evolution of the segmentation performance for an increasingly sub-sampled input dataset for each pre-process. Each pre-process including spatial reduction (i.e. low resolution and foveated data) is detailed according to the spatial downscaling method used. Each dataset is reduced using the structural stochastic temporal downscaling method introduced in Sec. 6.3.2, with $p \in [0.2, 0.4, 0.6, 0.8, 1]$ corresponding to the values in the x-axis of Fig. 8.5.

In those first two figures, the foveation is obtained by detecting the saliency on the data downscaled using the specified method, then merging it with the dataset reduced using the same method. This allows us to stay as close as possible to the hardware foveation design by simulating the feedback loop by using twice the same reduced data. Similarly to the classification results, we compare the semantic segmentation performance on the foveated dataset and on the events belonging to the detected RoIs in order to demonstrate the importance of keeping the information provided by the non-interesting regions but in a reduced format.

Fig. 8.6 studies the qualitative aspect of saliency detection by presenting the semantic segmentation according to the mean number of events per sample. Each foveation method, i.e. each method producing data on which the saliency was detected, is indicated in different colours. The average values for all datasets foveated using one method are presented in opaque, while each specific dataset's performance and number of events are overlayed.

As we want to maximise the performance while minimising the number of events, the goal to be reached is situated in the upper left corner of this plot. *Event count* emerges as the method for the most efficient salience detection.

8.3.3 Quantitative assessment of the saliency detection

Fig. 8.7 presents the spatial and temporal density according to the mean number of RoI detected per method to discuss the quantitative aspect of saliency detection.

The temporal density D_t corresponds to the activation probability of pixels averaged over the whole sensor and is defined in Eq. 6.5 (in Chapter 6).

A contrario, the spatial density D_s is the activation probability of the whole sensor over a limited time-window averaged over the temporal length of the sample, as described in Eq. 8.1.

$$D_s = \frac{\sum_{t=t_{window}}^{T} P_{[t-t_w,t]}}{N_w} \tag{8.1}$$

with t_w and T respectively the length of the time window and the length of the sample in time. The results in Fig. 8.7 are presented for $t_w = 50\mu s$. N_w corresponds to the number of successive time windows in the sample, as presented in Eq. 8.2.

$$N_w = \lceil \frac{T}{t_w} \rceil \tag{8.2}$$

The activation probability $P_{[t-t_w,t]}$ is calculated as the ratio between the number of pixels activated during the time-window $[t - t_w, t]$ and the overall number of pixels in the sensor:

$$P_{[t-t_w,t]} = \frac{\sum_{T=t-t_w}^{t} \sum_{x=1}^{w} \sum_{y=1}^{h} \delta(t_x, t) \cdot \delta(t_y, t)}{w \cdot h} \tag{8.3}$$

with w and h respectively the width and height of the sensor, t_x and t_y the timestamp of any events occurring at coordinates (x, y) and δ the Kronecker delta function, which returns 1 if the variables are equal, and 0 otherwise.

8.4 Discussion

To validate our initial hypothesis, the foveation would have to produce a number of events significantly closer to the low resolution while allowing for a performance closer to the high resolution. In other terms, the foveated results should be above the dotted grey line on Fig. 8.3 and Fig. 8.4. We do observe a striking decrease in the number of events between pre- (high resolution) and post-processing (low resolution and foveation) of the dataset. Concerning the DVS128 Gesture dataset and its classification performance (Fig. 8.3), the average number of events per sample drops to 1% of the original metric after reduction and to 50% after foveation, while the performance decreases respectively by 63% and only by 25%. A similar behaviour is found with the DDD17 dataset (Fig. 8.4): on average, the spatial downscaling and the foveation keep 30% of the original events. The most important drop of events is seen with the *linear* method: the reduced data drops to only 1% of the original size, and the foveated data to 10%. Similarly, the foveation's semantic segmentation performance is averaged over all methods and is remarkably close to the high resolution's performance.

Fig. 8.4 pictures an outlier, the *cubic* method. The trade-off between performance and data size displayed by the foveation is not as good as in other methods; this can be explained by the fact that the method produces too sparse data to offer a coherent saliency detection.

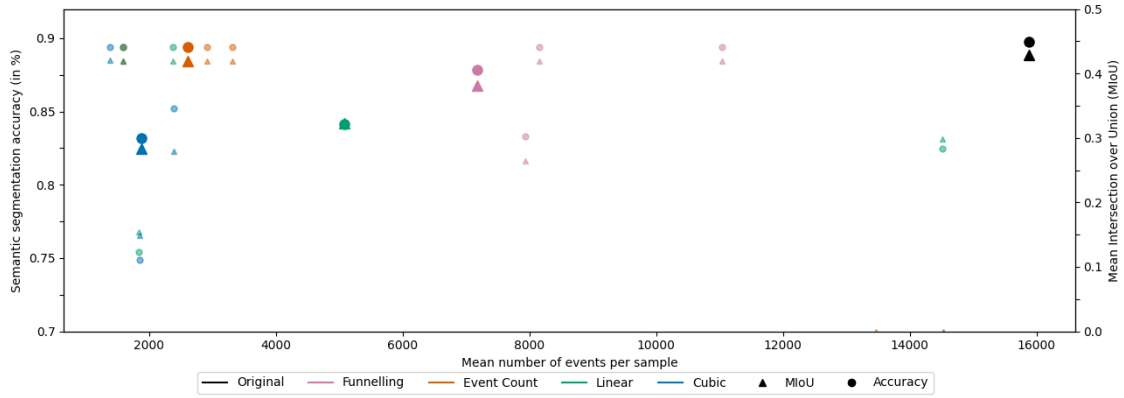


Figure 8.6 – Qualitative study of the saliency detection, based on the semantic segmentation performance. The plot depicts the average semantic segmentation performance according to the spatial downscaling method used to generate the data. The overlayed dots correspond to different reduction methods (with no indication of the corresponding methods) with which the multi-resolution merging was achieved using the RoIs detected with one specific method. The goal is to minimise the mean number of events per sample while increasing the performance, thus best results are close to the subplot's upper left corner. Foveated data using RoIs detected on *event count* is the closest to this goal.

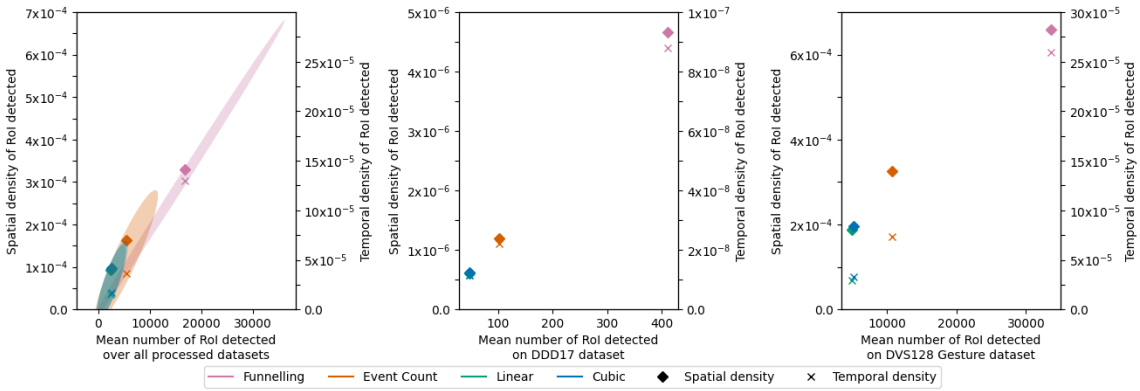


Figure 8.7 – Quantitative study of the saliency detection, by comparing the RoIs' spatial and temporal density according to the mean number of RoIs detected per sample. While there is neither good nor bad result in this graph, it shows that the *funnelling* method yields a significantly higher temporal and spatial density, as well as the highest mean number of RoIs detected.

Additionally, we compare the performance of semantic segmentation and classification on foveated events to events belonging in the detected RoIs, pictured in yellow in Fig. 8.3, 8.4 and 8.5. For both tasks, the performance shows a significant decrease when achieved on RoIs events. The ratio of segmentation accuracy and MIoU to mean number of events is similar to the one obtained on reduced data, if not worse for the *event count* method. The classification performance does not show such an intense decline but is still significantly further away from the ideal compromise than that obtained on the foveated data. This difference is explained by the intrinsic distinction between the datasets and tasks: labelling a gesture in DVS 128 Gesture can be done by identifying the

movement of the hands, i.e. by following the trajectory of the RoIs; while segmenting a driving scene requires more information spread out on the whole scene and cannot rely solely on the data provided in the RoIs, although these greatly facilitate the task.

All in all, those observations combined do confirm our core thesis, that is, that neuromorphic foveation leads to the best trade-off between information quantity and quality, at least on a software level. Neuromorphic foveation is more relevant than attentional filtering to preserve event data while minimising its size.

Furthermore, it is interesting to note that when comparing the proportional decrease of the number of events in the dataset post-process in Fig. 8.5 while all three types of data show the same behaviour, the averaged foveated data outperforms the high-resolution data from a 60% decrease and downwards. This is explained by the fact that the majority of events kept in the foveated dataset provide relevant information to the semantic segmentation model, while a significant part of the events in the original dataset is not as useful.

The less important drop of the data size when applying foveation to the classification of DVS 128 Gesture (Fig. 8.3) compared to the one presented in Fig. 8.4 is explained by the inner properties of the DVS 128 Gesture dataset: the lower spatial density of this dataset compared to DDD17, due to a static setting and a non-moving camera, leads to the detection of more densely aggregated RoIs already containing most of the sample's event data. In other words, the principal OoI in DVS 128 Gesture is the hand, which will be the main subject of saliency detection due to its constant movement. This is confirmed in Fig. C.4 where the high resolution is only visible on the hand in the foveated sample. As the movement of the hand is the main cause of event production while recording the scene, the corresponding region contains the most events — thus the foveation on the DVS 128 Gesture dataset does not decrease the data size as much as when applied to a moving camera recording.

This theoretical reasoning is confirmed by the results presented in Fig. 8.7: the mean number of RoIs detected on DVS 128 Gesture and its corresponding temporal and spatial densities (right sub-graph) are all significantly greater than those detected on DDD17 (middle sub-graph). Fig. 8.7 also highlights the important quantitative variations of the saliency detection according to the foveation method used: *funnelling* produces a great amount of RoIs, due to its lack of event drop; while *linear* detects the least saliency. The choice of the method used to detect the salience is not trivial, each one having its interests and its disadvantages. One must thus take time to think upstream about the physical properties of the dataset and the desired intensity of the foveation. All in all, when applying the foveation to the classification of DVS 128 Gesture and semantic segmentation of DDD17, *event count* seems like a good trade-off between the three proposed methods.

The qualitative study of the saliency detection displayed in Fig. 8.6 using different methods (i.e. detecting saliency on data reduced using those methods) reinforces the interest in using the *event count* method for foveation. Indeed this figure highlights the overall advantage of *event count*, as its use leads to a minimised data quantity for a maximised semantic segmentation performance.

8.5 Conclusion

In this work, we demonstrate the stakes of foveation applied to event data for semantic segmentation and classification. Such a strategy does concurrently preserve the accuracy of event data processing and greatly reduces the amount of data needed for the task.

To meet our initial goal of finding the ideal process to reduce the number of events while maintaining the quality of the information transmitted, a final aspect can be discussed here to complete the results presented above: that of the generation time. Indeed, the spatial reduction of events as well as the neuromorphic detection of salience requires a non-negligible processing time. The first has been discussed in Chapter 6 and depends strongly on the selected reduction method. The second varies with the hist architecture used. Indeed, the SNN model can be run either on CPU, using an SNN simulator such as PyNN (Davison et al., 2009), or on GPU with adapted simulators such as Norse (Pehle & Pedersen, 2021). In both cases, the simulation time increases with the number of neurons in the model and the size of the input data. A third option is to use neuromorphic chips, such as Human Brain Project's SpiNNaker (S. Furber & Bogdan, 2020) or Loihi (Davies et al., 2018), which enable fast and low power simulations and which simulation time only depends on the input data size. All in all, it is very unlikely that these two processes (reduction and saliency detection) combined can be done in real-time.

However, we seek to convert this software neuromorphic foveation process into a hardware implementation using the foveated sensor introduced by (Serrano-Gotarredona et al., 2022). With such a sensor that implements spatial reduction and foveation electronically, we can disregard the generation time.

Further work will implement the hardware neuromorphic foveation feedback process by concurrently using the foveated DVS presented in (Serrano-Gotarredona et al., 2022) and neuromorphic hardware, e.g. SpiNNaker, in order to record foveated event data in an embedded fashion.

CHAPTER 9

Towards embedded applications — First steps and perspectives

Now that we have demonstrated the interest of using various approaches to pre-process event data for efficient embedded processing in previous chapters, we propose in this chapter to study original embedded applications. Two of the tasks addressed below are motivated by our involvement in the APROVIS3D project. The target use cases of this project are 1) the detection of the coastline by an unmanned aerial vehicle flying around an island of the Mediterranean Sea to allow for autonomous navigation and 2) the 3D reconstruction of this coastline. We propose various approaches to tackle these axes of research and present some preliminary results. We also take an interest in embedded lip-reading, an advantageous neuromorphic task to demonstrate the criticality of the time dimension of event videos when properly exploited.
The perspectives of this work include the development of a comprehensive dataset for coastline detection, the implementation of a demonstrator for embedded 3D reconstruction, the development and extensive experimental validation of a model of weight and delay learning and the implementation of SpiNN-3 of an existing unsupervised delay learning model.

9.1 Introduction 163
9.2 Coastline detection 164
9.2.1 Adapting an ANN classifier into an embedded SNN classifier 164
9.2.1.1 Input data 165
9.2.1.2 Adaptation on SpiNNaker 167
9.2.2 Delay learning 167
9.2.2.1 Metrics for the assessment of the training phase . 168
9.2.2.2 Classification performance 169
9.2.2.3 Tentative model of delay learning 173
9.2.2.4 Perspectives 174
9.2.3 Line detection 175
9.2.3.1 Horizontal and vertical lines 175
9.2.3.2 Diagonal lines 176
9.3 3D reconstruction 177
9.3.1 Embedding the model on SpiNN-3 177
9.3.2 Stereo-matching performance 178
9.3.2.1 Performance on Dikov's datasets 179
9.3.2.2 Performance on DSEC 179
9.3.3 Live demonstrator 179
9.4 End-to-end neuromorphic lip reading 180
9.4.1 i3S dataset 181
9.4.1.1 Desired features 181
9.4.1.2 Dataset acquisition 182
9.4.1.3 Recorded data postprocessing 183
9.4.2 Neuromorphic lip reading model 183
9.4.2.1 Event data preprocessing 183
9.4.2.2 Topology exploration 184
9.4.3 Experimental validation 185
9.4.3.1 Experimental setup 186
9.4.3.2 Identification of the best topology 186
9.4.3.3 Ablation study and comparison to state-of-the-art 186
9.4.4 Limitations and perspectives 186
9.5 Conclusion 188

9.1 Introduction

In the Contributions part of this manuscript, previous chapters focused on various approaches we implemented during this PhD to pre-process event data using SNN. We aimed to optimise the performance of embedded computer vision tasks by minimising the data size while maximising the information conveyed. Now that we demonstrated the benefits of the three proposed methodologies, i.e. event data downscaling in Chapter 6, visual attention in Chapter 7) and neuromorphic foveation in Chapter 8, we consider the embedded processing of event data applied to various applications.

Event data interfaced with an SNN model on embedded neuromorphic hardware allow for a large panel of energy-efficient, low-memory applications (due to both hardware architectures and intrinsic data characteristics — see Chapter 4). We already considered two common tasks in previous chapters, although it was for the purpose of validating various pre-processing methods and not meant to be embedded:

— the classification of either gestures or numbers, respectively applied to the datasets DVS 128 Gesture (see Sec. 3.3.1) and N-MNIST (see Sec. 3.3.2). We used three published SNN models: the PLIF and SLAYER benchmark classifiers as well as the HOTS model (see Sec. 4.3.1).

— the semantic segmentation of a driving scene, applied to the DDD17 dataset using Ev-SegNet (see Sec. 3.3.3 and 3.3.3).

During this work, we took an interest in two additional tasks for which the literature is less extensive. Through the supervision of interns and collaborations with other members of the APROVIS3D consortium, we have produced some first results that are described in this chapter.

The first application we address is motivated by a use case of APROVIS3D. Even though a high temporal resolution is not required for this task, it serves as a support for a first approach to exploiting the natural match between SNN and event data. As described above, this project aims to implement a neuromorphic architecture embedded in an unmanned aerial vehicle (UAV) flying over the Mediterranean Sea, which performs coastline detection for the autonomous navigation of the drone. Our collaborators at NTUA in Athens, Greece, have recorded RGB and event videos of the beach and the Mediterranean Sea as seen from a UAV, at different heights and with different speeds (see the example given in Fig. 9.1). Additionally, at UCA, we similarly recorded RGB videos of the beach and the Mediterranean Sea that we later translated into events to complete this dataset, which we will use as input to our approaches.

To tackle this task, we propose three methodologies and describe the first results below. A first approach would be to translate the ANN classifier HOTS to an embedded SNN for sea-ground segmentation: HOTS would differentiate between sea and ground patches in the original data, thus allowing to extrapolate the coastline. The "patches" are sampled and extracted from the latticed recorded data. This approach is innovative, but may be difficult to generalise: it depends on the environment and is based on certain conditions (rough seas, more movements in the water than in potential vegetation, etc). It is currently the object of a work group between NTUA, UCA and INT (in Marseille, France) in order to implement a demonstrator by the end of the APROVIS3D project (September 2023).

A second approach uses a delay learning model to differentiate between the ground and the sea by the movement of the events: those recorded on the ground will correspond to the ones generated by the ego-motion of the embedded camera only, while the additional motion of waves will disrupt

the movement of those recorded on the sea. The implementation and application of the delay learning model introduced in (Nadafian & Ganjtabesh, 2020) have been the subject of multiple internships supervised during this PhD. We also propose as a perspective a potential learning rule inspired by the STDP and applied to delay — this proposition was presented as a poster during the French national conference JOBIM 2023 (Gruel & Martinet, 2022). Additionally, a complete review of "Precise Spiking Motifs in Neurobiological and Neuromorphic Data" was published in the MDPI - Brain Sciences journal (Grimaldi et al., 2022).

The third and final approach we propose to tackle coastline detection is a simple line detection model. This would allow us to remove the intermediate step of classifying patches into sea and ground and directly detect the coastline. We present here some first promising results on simple synthetic event data.

In this chapter, we also present various works implementing tentative first steps to answering the second task targeted by the project, i.e. the 3D reconstruction of the Mediterranean coastline. Two interns, Ms Huiyu Han and M. Noah Candaele, supervised during this PhD worked on this problem over a total period of 7 months. We present below a summary of the work accomplished during this period.

Finally, we address a third embedded application: neuromorphic lip-reading. During this PhD, we supervised two grouped internships on this subject, leading to the recording of an event-based lip-reading dataset and implementation of the first-ever end-to-end neuromorphic model for embedded lip-reading, which could directly benefit the development of portable assistive devices or surveillance equipment. This model and results presented below were made publicly available in an article presented during the CVPR 2023 Workshop on Event-based Vision (Bulzomi, Schweiker, Gruel, & Martinet, 2023).

9.2 Coastline detection

As presented above, the detection of the Mediterranean coastline is the main application task targeted by the APROVIS3D consortium and a demonstrator is to be produced by the end of the project. The following section describes the three lines of research conducted in order to answer this problem.

9.2.1 Adapting an ANN classifier into an embedded SNN classifier

A first attempt to solve this task is carried out by a dedicated work group composed of APROVIS3D members from NTUA in Athens, Greece, from INT in Marseille, France and from UCA.

At NTUA, PhD student Sotiris Aspragkathos and Prof. George C. Karras record event data and RGB videos of the sea and the coastline ground, separated and mixed. Once the data is acquired, at UCA, we handle the verification of the provided videos' quality and translate their formats into a more universal format. This modified dataset is then forwarded to PhD student Antoine Grimaldi and Prof. Laurent Perrinet at INT. Having implemented the HOTS classifier from previous work, they update this architecture to efficiently differentiate between patches of sea and ground.

Once it is properly adapted to this task, we at UCA will take care of implementing the HOTS network with the hyperparameters learned during training on the neuromorphic hardware SpiNN-3. The final topology will finally be embedded on the UAV set up by NTUA.

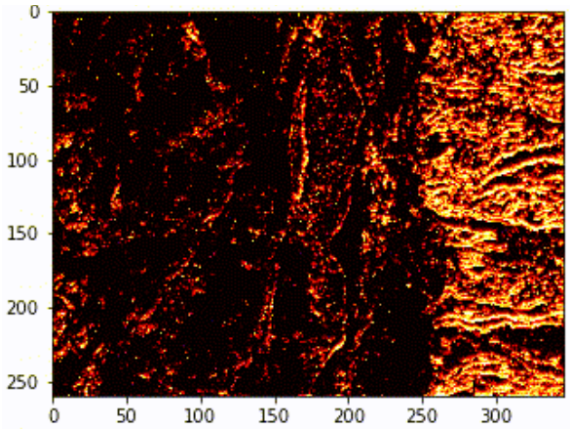


Figure 9.1 – Example of accumulated synthetic events obtained from the recording of the Mediterranean coastline by an RGB camera onboard a drone.

9.2.1.1 Input data

NTUA records event and RGB videos using an embedded DVS sensor provided by IMSE, as well as a CSL's DAVIS346 mono camera. This data could only be saved as an `AEDAT2` file, using the object structure described in Tab. 4 in Annexes.

However, this format is no longer readable in Python because the associated libraries are no longer supported. Additionally, the HOTS implementation used by INT takes input data according to the Tonic format (Lenz et al., 2021). This library stores and handles events according to a specific object structure, with specific intrinsic modules and attributes described below.

We thus implemented a pipeline to transform the `AEDAT2` files into `NPY` files. Additionally, we also developed a pipeline to produce `NPY` files from RGB videos recorded in a similar context. This allows us to comparatively analyse the event data obtained from the DVS sensor and the standard RGB camera.

Pipeline - event data, from AEDAT2 to NPY As expressed previously, the `AEDAT` format has slowly evolved from version 2 to 4 over the last few years and current versions of Python libraries can no longer handle the `AEDAT2` format. After some research, we found the AedatTools functions, implemented in Matlab and made publicly available by a Github user (qiaokaki, 2017). We leveraged these tools to implement a Matlab script translating an `AEDAT2` file into a `CSV` file. A second Python script is then used to transform this `CSV` file into a structured Numpy array, saved as an `NPY` file. The two scripts described above can be found in the Annexes (Algorithms .1 and .2).

Pipeline - RGB data to event data When recording event data from various UAV flights, NTUA also recorded and made available to us corresponding RGB videos. In addition, the early recorded videos were not suitable for learning sea and ground patches only as these videos indiscriminately mixed the two scenes. While waiting for new event data to be recorded by NTUA, we recorded with the help of the robotic team of the I3S laboratory, some sea-only and ground-only scenes using a RGB camera. We then translated these videos into event data using the "Video-to-events" library (D. Gehrig et al., 2020).

This synthetic dataset, although imperfect in its synchronous aspect, allowed us 1) to have data on which to begin the HOTS training and 2) to compare these "ideal" data to the data recorded in the field by the embedded DVS sensor (see Sec. 9.2.1.1).

Tonic dataset structure Tonic wraps event data according to a specific structure (Lenz, 2019): an object is created for each dataset, inspired by TorchVision's dataset module (Linux Fundation, 2017). It requires methods to initialise the object, get a specific element of the dataset and output its length. The events from one sample of the dataset are stored as a structured Numpy array of shape $(N, 4)$, with N the number of events and the four channels corresponding to the x and y coordinates, timestamp and polarity of each event.

We implemented our own wrapper to allow for the HOTS model to easily use the event data recorded by NTUA as input. A custom module was added to those required by Tonic: this function splits the input stream of events into patches defined spatially and temporally, themselves split into training and testing samples according to a user-defined ratio.

This wrapper can be applied to event data either recorded by NTUA's DVS sensor or produced from RGB videos, as long as the input files are of the format `NPY` or `NPZ`.

Validation of produced datasets Once both the DVS recordings and synthetic event data were wrapped according to the Tonic structure, we analysed both datasets to assess the quality of the data. However, the analysis of the first dataset provided by NTUA displayed many incoherences such as long periods of time (up to 10s) with no events and the absence of activation of certain pixels over the whole recording. According to Prof. Teresa Serrano-Gotarredona (a microelectronic researcher at IMSE, Spain and member of the APROVIS3D consortium), this behaviour is explained by the presence of the shadow of the UAV helix in the DVS sensor's field of view, which causes a high flow of events. The event rate produced by the DVS sensor has an upper bound ; additionally, the FPGA board handling the recording has a maximum speed due to the time needed to put a timestamp to the events, temporarily store them and send them to through the USB bus once the buffer is full. Those three elements combined lead to the saturation of the camera and intermittent communication of the events: the intermittent activation of rows of pixels is due to the delay in transmission of the output spikes ; the time periods of inactivation are caused by the higher rate of buffer refills than FPGA timestamping and communication through the USB bus, causing event loss.

Following this interpretation, Sotiris Aspragkathos recorded new data with an event rate lowered by reducing the sensor sensitivity and stimulus speed and contrast. After analysis, this new dataset is deemed adequate to be used as a support to train HOTS and perform coastline navigation.

The different datasets and analysis described above can be found in Annexes, in Sec. E. The first dataset recorded using a DVS sensor by NTUA and presenting issues is referred to as "NTUA_DVS_2022", and the second one recorded in similar conditions is referred to as "NTUA_DVS_2023". The dataset recorded by an RGB camera by UCA then translated into events is referred to as "UCA_synthetic_2022".

Perspectives We aim to record and make available to the public a more complete and improved dataset for the coastline segmentation task, from which we ultimately wish to infer segmentation offline. The data would be recorded using a more standard DVS (such as a DAVIS camera, whose

characteristics will be specified when the dataset is opened to the public). We established the following requirements to accomplish this task:

— at least three levels of UAV altitude (for example, 5, 10 and 15 meters — knowing that the UAV does not usually fly higher than 20 meters in the project use case);
— at least 3 levels of UAV speed (knowing that the UAV does not usually fly more than 5 m/s in the project use case);
— different textures: sea, sand, grass, bushes, rocky coast, etc;

We would record three one-minute-long samples for each combination of requirements described above, thus collecting a total of at least 135 samples in the complete dataset. All samples would be labelled with the values corresponding to each requirement described above. We believe that such additions to the dataset would make segmentation more adaptive.

9.2.1.2 Adaptation on SpiNNaker

As HOTS has a relatively simple architecture, once trained to detect patches of ground and sea, this model will easily be implementable on an embedded SpiNN-3, using the Python library sPyNNaker (see Sec. 2.2.2). Each set of hyperparameters learned offline will be set to enable online inference, thus automatic coastline detection and autonomous navigation of the drone.

9.2.2 Delay learning

A second approach to coastline detection would be to classify smaller samples of the scene captured by the sensor according to the movement of the recorded events. Indeed, depending on whether the events recorded correspond to the ground or to the sea, their movement is quite different: as can be seen in Fig. 9.3, the events corresponding to the ground are generated by the sensor's ego-motion (Chapel & Bouwmans, 2020) thus follow a vertical, north-south motion; however the movement of the events corresponding to the sea is the sum between the movement generated by the ego-motion (vertical motion) and by the waves (horizontal motion when following the coastline). We thus propose to classify those samples using delay learning which performs particularly well for spatiotemporal pattern recognition (see Fig.9.2, 9.3 and 9.4).

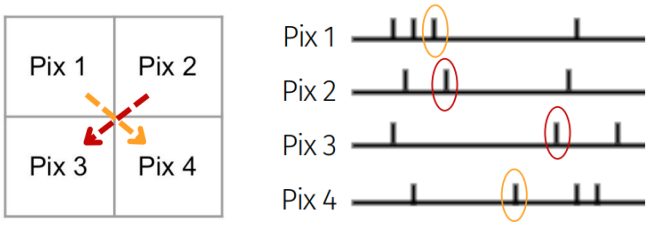


Figure 9.2 – Delay learning changes the delay of the electrical spike journey in the pre-synaptic neuron's axon so that it recognises specific patterns in the visual input data.

To this end, we supervised nine interns who worked on the re-implementation and adaptation to real-life data of the bio-plausible unsupervised delay learning model introduced in (Nadafian & Ganjtabesh, 2020) (see Sec. 5.3.2):

— M. Hugo Bulzomi, a 1st-year student in UCA master's degree in computer science at the time, re-implemented the authors' model using PyNN, generated new synthetic input data

and produced some preliminary results. His internship conducted intermittently with his courses, spanned the duration of the semester of Fall 2021.

— M. Sami Benyahia, Ms Carla Guerrero, M. Raphaël Julien and M. Quentin Mérilleau, 1st-year students in UCA master's degree in computer science, worked on the implementation of the model on SpiNN-3 — unfortunately without much result due to the high level of complexity of this project. Their internship was conducted in the semester of Spring 2022, intermittently with their courses.

— M. Kévin Constantin, a 5th-year engineering student in computer science, reused Hugo's code and made it more modular so that the number of pattern motions to be classified can be easily modified. Additionally, he developed tools to visualise the evolution of the weights and delays, as well as implemented metrics to asses the training phase efficiency. His internship conducted intermittently with his courses, spanned the duration of the semester of Fall 2022.

— M. Soussoh Kevin Egbohou, M. Gabriel Reynes and M. William Fernandes, 2nd-year work-study students in engineering, worked on adapting the model to real-life data as well as the implementation of a real-time performance indicator. Their internship was conducted during the semester of Spring 2023, intermittently with their courses as well as their time in enterprise.

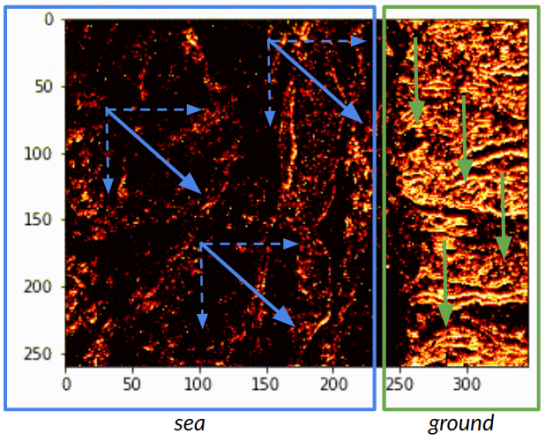


Figure 9.3 – Overall movements of the synthetic events obtained from the recording of the Mediterranean coastline by an RGB camera embedded on a drone. The top of the sensor corresponds to the direction of the UAV.

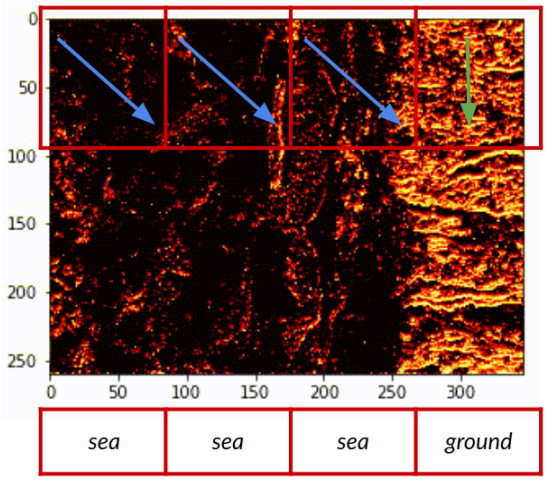


Figure 9.4 – Overall movements of the synthetic events obtained from the recording of the Mediterranean coastline by an RGB camera embedded on a drone. The top of the sensor corresponds to the direction of the UAV.

9.2.2.1 Metrics for the assessment of the training phase

Within the framework of his internship, Hugo implemented a first index to assess the homogeneity of the classification performed by the model; the Gini index, measuring the statistical dispersion i.e. the "purity" of the clusters produced by the classification, where each cluster should only gather similar input patterns. It is calculated as follows:

$$Gini(cluster) = 1 - \sum_{j=1}^{n} (p(j/cluster))^2 \tag{9.1}$$

where n is the number of classes (here the number of patterns to classify) and $p(j/cluster)$ is the frequency of class j in the cluster. The closer the Gini index is to 0, the "purer" is the cluster. We assess the final performance of the network as the average Gini index over each cluster populated by the network.

In a second phase, Kévin implemented three additional metrics to assess the convergence of the convolution kernels towards the recognition of input patterns: the precision, the recall and the F1-Score (see Eq. 9.2).

$$\begin{aligned} precision &= \frac{TP}{TP + FP} \\ recall &= \frac{TP}{TP + FN} \\ F1_{\text{Score}} &= 2 \times \frac{precision \times recall}{precision + recall} \end{aligned} \tag{9.2}$$

where TP corresponds to the true positive events, i.e. when a spike takes place when it is meant to happen; FP corresponds to the false negative events, i.e. when a convolution layer spikes for an input pattern it is not specialised for; FN corresponds to the false negative events, i.e. when a convolution layer does not spike for an input pattern it should recognise. The precision thus evaluates whether the actual spike activity matches with what it should have been, and the recall evaluates whether all the spikes that should have happened did happen. The overall performance of the network is therefore evaluated by comparing the expected activity (i.e. the input spikes) to the obtained activity (i.e. the output spikes) for each convolution layer.

The metrics described in Eq. 9.2 require the initial step of linking each convolution layer to a specific input pattern. A layer is specialised once its delays and weights match an input pattern, with delays 1) not homogeneous but progressively increasing and 2) significantly lower than the delays of neurons external to this pattern.

9.2.2.2 Classification performance

Four convolutions trained on noiseless data Using the four initial convolution layers, we train the model on non-noisy synthetic input data after a long phase of manual tuning of various hyperparameters. We first test the trained model non-noisy data; we obtain the features presented in Fig. 9.5a, which are quite similar to the results obtained by the authors. Each layer is specialised, with a perfect clusterisation according to the Gini index presented in Table 9.1a. Secondly, we test this trained model on noisy input data: in this case, the average Gini index increases to 0.24 (see Table 9.1b) and the clusters are less homogeneous, while still acceptable.

Four convolutions trained on noisy data We wish now to assess the classification performance of this model when trained on noisy input data, similarly to the experiments conducted in (Nadafian & Ganjtabesh, 2020). When tested on noiseless data, the results show a less marked specialisation than when trained on noiseless data. Additionally, the convolution layers B and C in Fig. 9.5b seem to have specialised for a similar motion. According to Table 9.2a, two layers overpower the others

	Diagonal A	Diagonal B	Diagonal C	Diagonal D	Gini
Cluster 1	25	0	0	0	0
Cluster 2	0	25	0	0	0
Cluster 3	0	0	25	0	0
Cluster 4	0	0	0	25	0

(a) Without noise

	Diagonal A	Diagonal B	Diagonal C	Diagonal D	Gini
Cluster 1	15	1	0	0	0.12
Cluster 2	4	22	0	2	0.36
Cluster 3	6	2	25	3	0.48
Cluster 4	0	0	0	20	0

(b) With noise

TABLE 9.1 – Number of each diagonal pattern per cluster and corresponding Gini index, with a network trained on noiseless data and tested on noisy and non-noisy input data.

and seem to recognise the same trajectory. Despite those issues, the average Gini index reaches 0.25. Finally, we test this model on noisy input data and observe that the clustering has indeed suffered from the specialisation defects observed in the convolution filters, with an average Gini index of 0.605 (see Table 9.2b).

	Diagonal A	Diagonal B	Diagonal C	Diagonal D	Gini
Cluster 1	25	25	0	0	0.5
Cluster 2	0	0	0	0	0
Cluster 3	0	0	0	0	0
Cluster 4	0	0	25	25	0.5

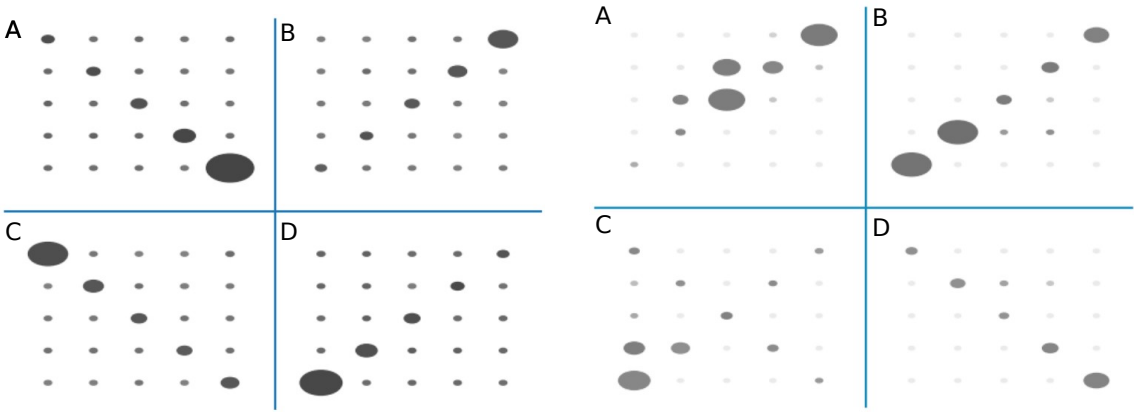
(a) Without noise

	Diagonal A	Diagonal B	Diagonal C	Diagonal D	Gini
Cluster 1	17	13	0	1	0.52
Cluster 2	6	6	8	10	0.74
Cluster 3	2	6	2	5	0.69
Cluster 4	0	0	15	9	0.47

(b) With noise

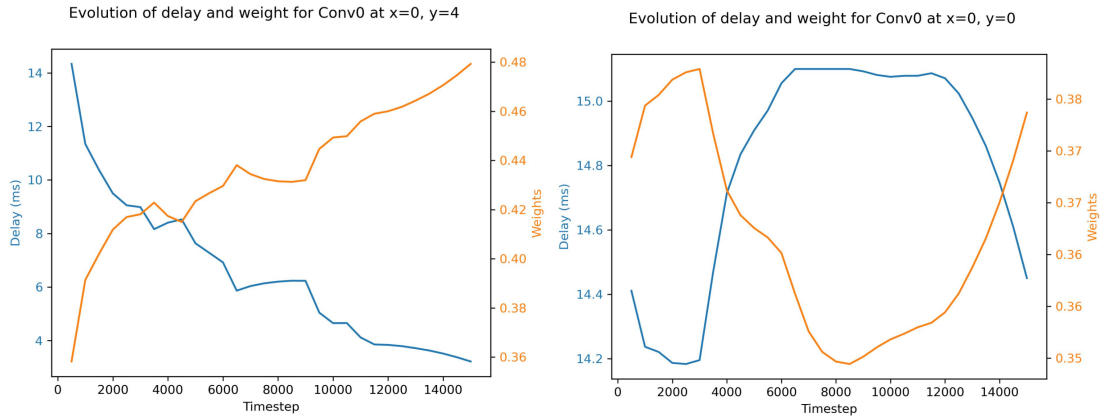
TABLE 9.2 – Number of each diagonal pattern per cluster and corresponding Gini index, with a network trained on noisy data and tested on noisy and non-noisy input data.

Varying number of convolutions trained on noiseless data As stated precedently, while Hugo used an approach to visualise the learnt features similar to the authors, one goal of Kévin's internship was to develop new visualisation tools to assess the quality of the training: we can now visualise the actual evolution of the weight and delay values over the training phase (see Fig. 9.6). Additionally, we updated Hugo's code so that the model can be more modular and one can easily



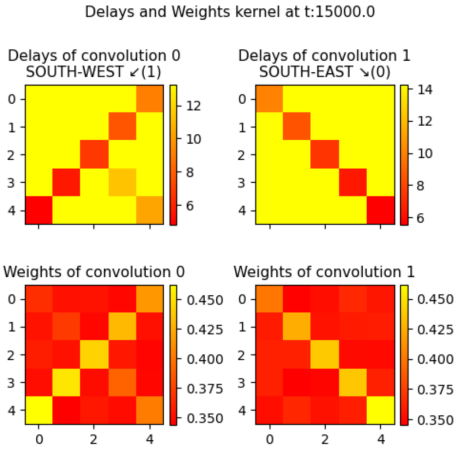
(a) Without noise (b) With noise

Figure 9.5 – Features learned by the delay learning model implemented in (Nadafian & Ganjtabesh, 2020), without or with additional noise in the input data used during training. Similarly to Fig. 5.8, the size and intensity of each circle represent respectively the delay and weight of its corresponding synapse, where the bigger circle illustrates lower synaptic delay and the colour black indicates the higher synaptic weight. On features learned on non-noisy data (left), the diagonal A corresponds to a south-east motion, B to a north-east motion, C to a north-west motion and D to a south-west motion. Figure from (Bulzomi, 2022).

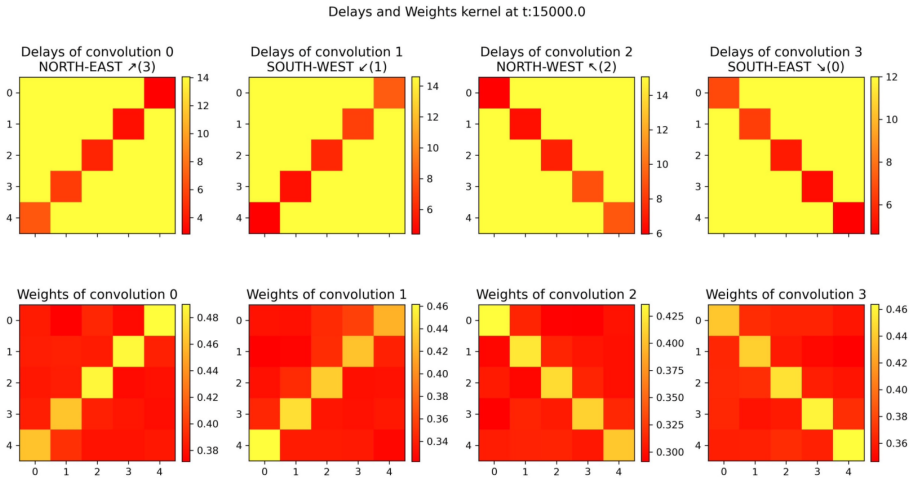


(a) Pixel inside the pattern (b) Pixel outside the pattern

Figure 9.6 – Evolution of the features learned over time by the delay learning model implemented in (Nadafian & Ganjtabesh, 2020), trained on noiseless input data. The delays are displayed in blue and weights in orange. The figure on the left presents the evolution of the delays and weights of the kernel corresponding to the sensor pixel of coordinates $(x = 0, y = 4)$, i.e. within the pattern ; while the figure on the right presents the evolution of the delays and weights of the kernel corresponding to the sensor pixel of coordinates $(x = 0, y = 0)$, i.e. outside the pattern. Figure from (Constantin, 2023).



(a) Two convolution layers



(b) Four convolution layers

Figure 9.7 – Features learned at $t = 15,000\mu s$ by the delay learning model implemented in (Nadafian & Ganjtabesh, 2020), without any additional noise in the input data used during training and using two (top) or four (bottom) convolution layers. The delays and weights are displayed separately: the first line corresponds to the delays, the second to the weights and each column to one convolution layer. The pattern learned by each layer is identified above each column. Figure from (Constantin, 2023).

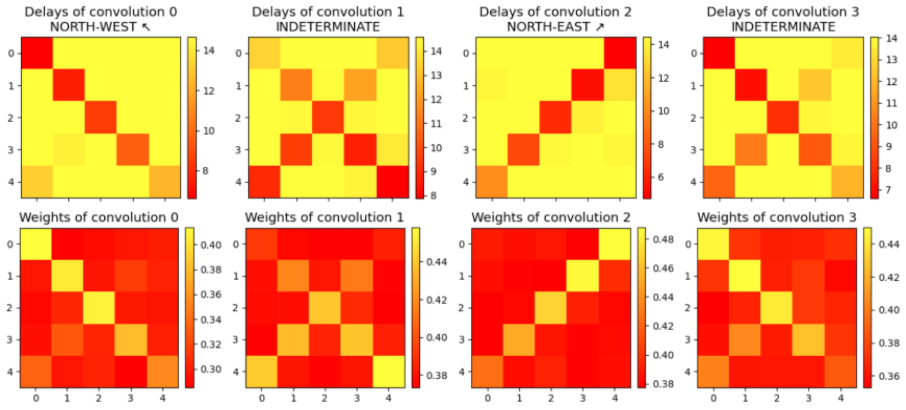
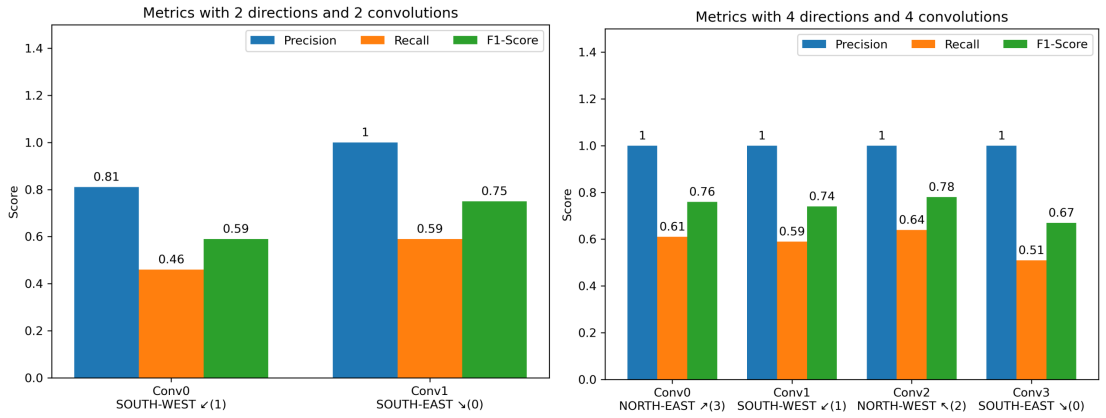


Figure 9.8 – Issue in the identification of patterns based on delay learning.

modify the number of convolution layers, as can be seen in Fig. 9.7. The variation in the number of convolution layers does not seem to influence the quality of learning according to the metrics displayed in Fig. 9.9; the accuracy is significantly high, meaning that the network is almost never wrong when predicting a given motion in the input data. However, the model quite often fails to recognise an existing motion, with a probability close to 0.5 according to the recall value.



(a) Two convolution layers (b) Four convolution layers

Figure 9.9 – Metrics (described in Eq. 9.2) to assess the quality of the classification performed by the features learned by the delay learning model implemented in (Nadafian & Ganjtabesh, 2020), without any additional noise in the input data used during training and using two (left) or four (right) convolution layers (corresponding to the features displayed in Fig. 9.7). The precision is displayed in blue, the recall in orange and the F1-Score in green. Figure from (Constantin, 2023).

Defective lateral inhibition While showing a certain learning deficiency, the results presented in Tables 9.2a and 9.2b display a certain coherence: diagonals A and B, as well as C and D, seem to respectively produce similar clusters of good quality, specialising for the same trajectory independently of the motion direction. This behaviour of double specialisation can also be observed on noiseless data in certain simulation runs, such as in Fig. 9.8.

It seems that it is common for two layers to be occluded: a layer specialised for one motion has weights correctly adjusted for its trajectory; only the delays allow us to differentiate between the direction of this motion. This phenomenon is supposed to be avoided thanks to a lateral inhibition mechanism that appears to be defective in this architecture.

This issue could be addressed by implementing a more efficient lateral inhibition mechanism or by adjusting the parameters according to the number of convolution layers, the window size, the event density...

9.2.2.3 Tentative model of delay learning

After studying the neuromorphic state-of-the-art and especially the bio-plausible model introduced in (Nadafian & Ganjtabesh, 2020), as well as investigating the potential biological explanations for delay learning in (Grimaldi et al., 2022), we propose here a tentative learning rule for simultaneous weight and delay learning inspired from the traditional STDP learning rule, for recognition of spatiotemporal patterns. Here the weight would follow a traditional STDP rule (see

Sec. 2.1.4.3) and the delay would be updated over time according to the following equation:

$$\begin{aligned} \Delta\delta &= \begin{cases} -\alpha \sin(\frac{\Delta t}{\tau}) & \text{if} \Delta t \in [-\tau\pi; \tau\pi] \\ 0 & \text{otherwise} \end{cases} \\ \text{with } \Delta t &= \Delta t_{pre} - t_{post} \\ \text{and } \Delta t_{pre} &= t_{spikeA} - t_{spikeB} \\ &= t_{activationA} + \delta_A - t_{activationB} - \delta_B \end{aligned} \tag{9.3}$$

where δ is the delay of the last spiking neuron between the pre-synaptic neurons A and B, τ is the time constant of the learning rule and α is the learning rate. When τ increases, the time window of the learning rule increases and the learning is applied to neurons whose spikes are farther apart (and *vice versa*). When α increases, the value of the $\Delta\delta$ extrema increases and the learning is steeper. The behaviour of $\Delta\delta$ and $\Delta\omega$ are illustrated in Fig. 9.10.

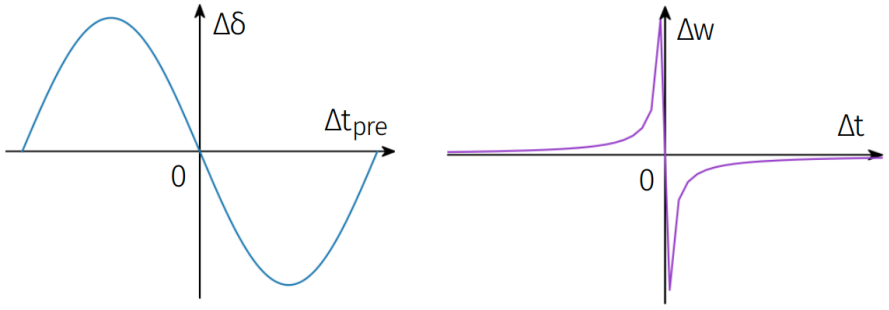


Figure 9.10 – Behaviour of the tentative model of weight and delay learning we propose in this work. The plot on the left presents the evolution of the delay according to Eq. 9.3, and the plot on the right the behaviour implemented by a traditional STDP.

9.2.2.4 Perspectives

Our work on delay learning is still in progress, and several avenues are open to us to improve this model and apply it to our coastline detection application. Concerning the re-implementation of the model of (Nadafian & Ganjtabesh, 2020), a first step would be to achieve significantly accurate results on noisy input data, with or without noise in the training dataset, to more easily adapt the network to real-life data. A second would be to increase the number of patterns to be detected: the network performs well for 4 directions, but an application to real-life data would require the distinction between at least 8 directions. However such an experiment necessitates increasing the kernel size, thus asking to increase the computational capacities of the support machine.

Once those two steps are achieved, we could hope to apply this model to samples extracted from latticed real-life event data, either by inferring motions from the features learnt on synthetic data or by training our model directly on real-life data.

Finally, we aim to implement this model on an embedded SpiNN-3 board to achieve online classification to ultimately perform coastline detection and autonomous navigation.

The last two steps could similarly be followed with our original weight and delay learning rules, provided additional efforts have been lead towards the implementation and experimental validation of this model.

9.2.3 Line detection

As a third and final approach to coastline detection, we developed a small SNN for line detection, easily embedded on a SpiNN-3 to interact with an event camera. As this model is still under development, we have only produced a few first results which we present in the paragraph below.

9.2.3.1 Horizontal and vertical lines

The first network we implemented allows for the detection of vertical and horizontal lines, without (first version - see Fig. 9.11) or with (second version — see Fig.9.12) the position in the sensor of the input lines.

For an input sensor of $n \times n$, the first version of this network requires $2(n+1)$ neurons and $2(n^2+n)$ excitatory synaptic connections. Each input neuron is connected to one neuron in the vertical line detector and one in the horizontal line detector according to a low weight ω so that the latter is activated only if all the neurons of the same (vertical or horizontal) line are activated. Each of those two detectors of n neurons, arranged in linear ways according to a 2D orthogonal reference frame, are then respectively connected according to a strong weight ω to a neuron playing the role of overall supervisor. If one of the two neurons is activated, at least one line of the corresponding type is detected.

In our case, we need the detection of the coastline to allow for the autonomous navigation of the drone, therefore we need to know the position of each detected line. We implement a second version presented in Fig. 9.12 where the supervisor neurons from the first version are removed, and a WTA mechanism is added to the vertical and horizontal line detectors to obtain the exact position of the strongest line detected in the visual scene (as we wish to detect only the coastline). We obtain a network of $2n$ neurons and $2n^2 - n$ synaptic connections, of which n^2 are excitatory and $n^2 - n$ inhibitory (WTA connections).

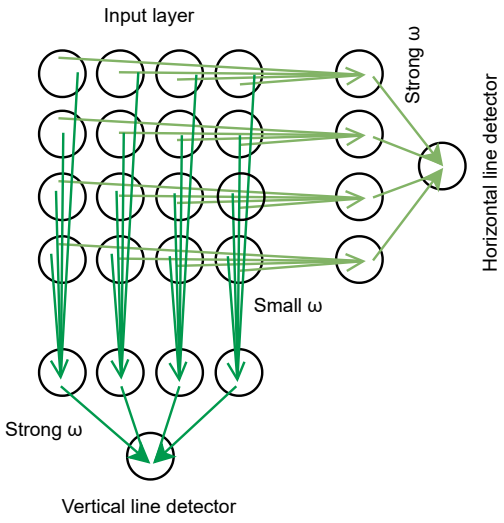


Figure 9.11 – SNN architecture for vertical and horizontal line detection, with no knowledge of the line position.

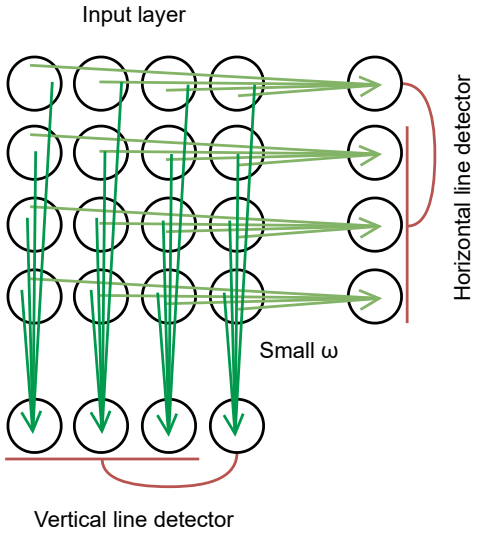


Figure 9.12 – SNN architecture for vertical and horizontal line detection, with knowledge of the line position.

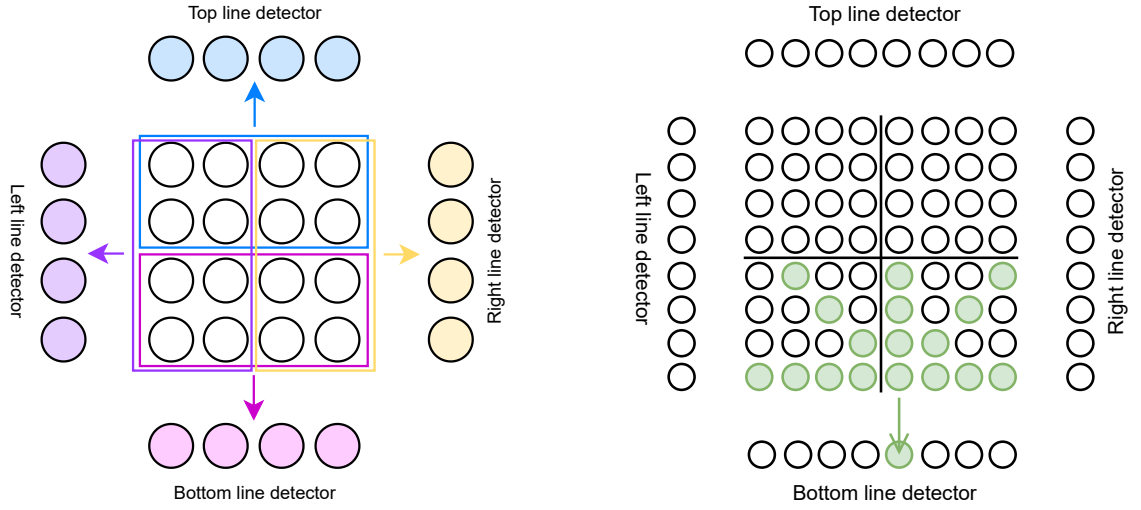
We produced some initial promising results on static and moving lines, randomly oriented and positioned in the scene and with or without noise. The corresponding plots can be found in Sec. E.

9.2.3.2 Diagonal lines

The networks described above are light and perform well, but do not allow for the detection of diagonal lines, thus are not applicable for coastline detection. We propose here a second architecture extending the first one to detect diagonal lines.

As seen in Fig.9.13a, instead of two detectors specialised either for vertical or horizontal lines, we propose four detectors, each detecting lines on half of the sensor: top detector for the upper part, bottom detector for the lower part, left detector for the left part and right detector for the right part. Each neuron of each part of the sensor is connected to at least one neuron of the corresponding detector. One example of the pattern of connections is given for the lower part, for one neuron of the bottom detector, in Fig. 9.13b: the neurons on a line starting from the border neuron at the same index as the detector neuron, and expanding according to a varying degree $alpha$ so that a majority of line orientations is covered, are excitedly connected to the corresponding neuron in the detector.

This network theoretically allows for the detection of any line in the sensor, whether it is diagonal, vertical or horizontal (see examples given in Fig. 9.14), using only $4n$ neurons for an input sensor of dimension $n \times n$ neurons.



(a) Architecture of the model

(b) Pattern of connections

Figure 9.13 – SNN overall architecture (left) and pattern of connections for one detector (left) for the detection of diagonal lines.

Perspectives However different experiments demonstrated that the current pattern of excitatory connections is not sufficient to assure the activation of one detector by a line ending on the corresponding sensor border. Indeed, in the case of the connectivity presented in Fig. 9.13b, a horizontal line crossing the lower part of the sensor would activate the corresponding neuron in the bot-

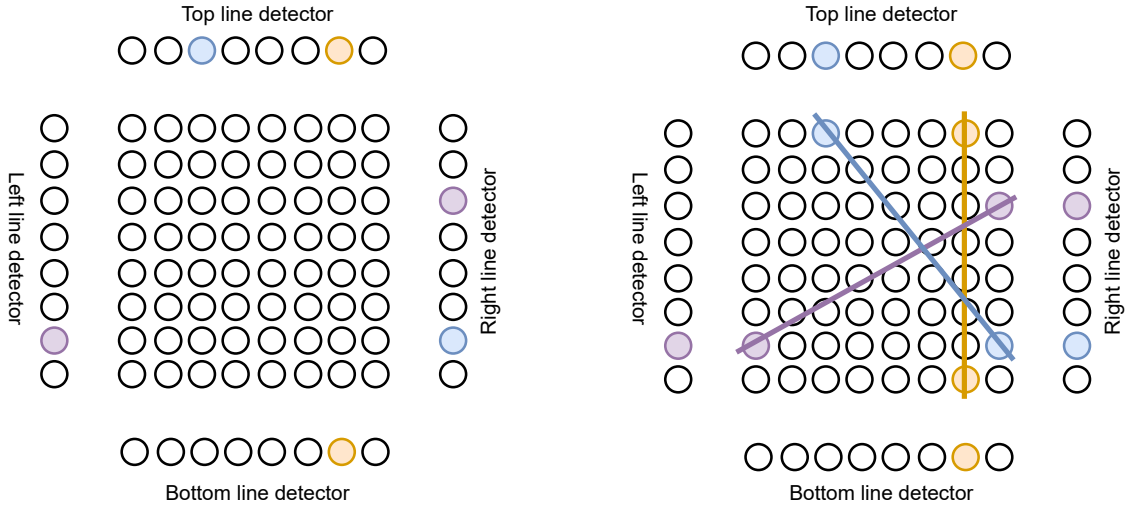


Figure 9.14 – Three examples of line reconstruction from various detector's neurons activations. Two neurons are activated in two of the detectors, thus giving the coordinates of the intersection points of the detected lines with the edges of the sensor.

tom detector; actually, as this pattern is similar to all neurons of the bottom detector, it would be entirely activated thus distorting the results.

We are currently working on implementing a lateral inhibition mechanism that would ensure that a detector neuron is activated if and only if all the neurons on the same line are activated.

9.3 3D reconstruction

As stated in the introduction of this chapter, the APROVIS3D project has a use case of 3D reconstruction of a visual scene from the stereo matching between the recording of two event cameras, using an SNN architecture. We investigated this question *via* the supervision of two internships:

— Ms Huiyu Han, a 5th-year engineering student in applied mathematics and modelling at the time, conducted a thorough state-of-the-art study (see Sec. 4.3.2), reimplemented the spiking stereo-matching method introduced by (Dikov et al., 2017) on SpiNN-3 (see Sec. 2.2.2) and adapted to be tested on real data, i.e. the DSEC dataset (see Sec. 3.3.4). Her internship lasted from March to September 2021.

— M. Noah Candaele, a 1st-year student in a master's degree in computer science, initiated the implementation of a demonstrator interfacing SpiNN and event-driven camera for real-time stereo matching. His internship conducted intermittently with his courses, spanned the duration of the semester of Fall 2022.

9.3.1 Embedding the model on SpiNN-3

(Dikov et al., 2017) originally reduced the size of their spiking stereo-matching model so that it can be embedded on the neuromorphic set-up they had available: indeed, to take as input the data produced by a sensor of size 128×128 with a disparity range of $[0, 127]$, their network is

composed of around 2^{20} cells i.e. $3.17M$ neurons (see Eq. 4.5, where we do not consider the retina input as the corresponding neurons are not simulated *stricto senso* by the SpiNNaker boards), which requires 15 SpiNN-5 boards. The authors filter the input event and reduce the network size so that $x_{max} = y_{max} = 106$ and $d_{max} = 32$. With those new parameters, the network contains only $315K$ neural micro-ensembles i.e. $945K$ neurons and can be run on the authors' locally available system of 6 interconnected SpiNN-5 boards.

In the framework of Huiyu's internship, our main goal was to further reduce this network and adapt it so that it can fit on a unique embedded SpiNN-3 board. In order to achieve this goal, our first approach was to calculate the maximum number of parameters such as $x_{max} = y_{max} = d_{max} + 1$ and $N_n < 18K$ (see Sec. 2.2.2). However, with the set of parameters obtained ($x_{max} = y_{max} = 22$ and $d_{max} = 21$) the board produces an unexpected resource error.

Indeed, this issue can be explained as follows: the SpiNNaker board places each population on a separate core, thus allowing for a maximum of 63 populations on SpiNN-3. The network structure originally implemented in (Dikov et al., 2017) initialises one population per vertical column of blocker neurons and per vertical column of collector neurons ("columns" referring here to neurons sharing the same x coordinate). The number of populations can thus be calculated as twice the number of neural micro-ensembles n in each layer (see Eq. 4.3), with y_{max} the number of neurons in each collector population and $2x_{max}$ the number of neurons in each blocker population (since there are two blockers for each collector). With the current set of parameters, we have 552 populations that cannot be embedded on our SpiNN-3's 63 cores ; therefore we reduced our set of parameters to $x_{max} = y_{max} = 7$ and $d_{max} = 6$ to implement the original network structure on SpiNN-3.

However, the setup described above is not optimal: we only place 7 neurons per population/core when each core could fit up to 256 neurons. We thus adjusted the network structure to place more neurons per population: our new structure initialises one population per horizontal rank of blocker neurons and per horizontal layer of collector neurons ("layer" referring here to neurons sharing the same y coordinate). The number of collector populations is now y_{max} and the number of blocker populations $2y_{max}$, for a total of $3y_{max}$ populations. Each population contains a number of neurons that equals the total number of neural micro-ensemble N_{nme} described in Eq. 4.3. This new structure allows us to take input of size $x_{max} = y_{max} = 21$ and $d_{max} = 20$ while maintaining the original pattern of excitatory and inhibitory connections.

A second approach to embed (Dikov et al., 2017)'s stereo-matching network on a SpiNN-3 was to spatially reduce the input data instead of cropping it so that it fits the parameters determined above.

9.3.2 Stereo-matching performance

We present below some initial results with both approaches on two datasets (two fans and pendulum) provided by (Dikov et al., 2017) to reproduce their results and more innovatively on the DSEC dataset (see Sec. 3.3.4).

9.3.2.1 Performance on Dikov's datasets

The authors provide two datasets, presented in Section 4.3.2.2, and corresponding results that we wish to reproduce with our adaptations.

We present below a comparison between the results obtained with the original network as implemented in (Dikov et al., 2017) and those obtained on EBRAINS after either downscaling the input data by 2 or on SpiNN-3 after downscaling and cropping the input data. The spatial downscaling is achieved using the event count method described in Sec. 6.2.2.1 using a factor 3. Table 9 (in Annexes) presents the performance of cooperative stereo-matching obtained on the "two fans" dataset, where the network activity is quasi-stationary in terms of disparities despite the high rotation speed of the blades thus produces good performance thanks to the reduced number of event. However, Table 10 (in Annexes) presents similar results obtained on the "pendulum" dataset, where the constant evolution of disparity produces a significant number of events thus asking for more resources (as proves the high accuracy obtained on EBRAINS compared to SpiNN-3).

9.3.2.2 Performance on DSEC

Finally, we adapted this cooperative stereo-matching network to a more complicated, real-world driving dataset: DSEC (see Sec. 3.3.4). As the spatial resolution is too important for event Manchester's 1 million core large scale SpiNNaker-1 system (available to us *via* EBRAINS), we spatially downscaled by 5 the first $300ms$ of one sample of this dataset. The initial disparity map we obtain is particularly sparse because the pixels spike less often than the network's simulation timesteps, thus not providing enough information at each timestep. To overcome this issue, each non-activated pixel is given the last computed disparity value at each timestep. Table 11 presents the disparity maps computed from ground truth and from the stereo-matching network results, with and without solving the non-activation issue mentioned above. We explain the surprisingly unsatisfying network results by the high speed of change of disparity at each pixel, compared to the dataset provided by the authors where each pixel's disparity changes much less often if at all. Additionally, we suspect that the parameters used in (Dikov et al., 2017) are not quite so generalisable and that a solution to this problem would be to find a method to adjust the parameters to this more complex dataset.

9.3.3 Live demonstrator

The main goal of Noah's tutorship was to take up the work done during Huiyu's internship and from there, create a live demonstrator interfacing the cooperative stereo-matching network implemented on SpiNN-3 with two locally available event cameras. A stereo event stream is recorded using two Prophesee GEN3.1 cameras, calibrated so that they record the same visual scene from two slightly different angles but synchronised in terms of focus, distortion, rotation, etc. If the calibration is not done correctly, the images acquired will not be aligned and the depth information will not be reliable.

The current implementation of sPyNNaker (see Sec. 2.2.2) provides different methods to retrieve records from an external camera, that is then stored in the chips' SDRAM to ensure that it is available for future access. However, sPyNNaker does not currently provide any out-of-the-box support for real-time interactions with simulations (such as interacting with an external device and receiving new data while a simulation is running), except the less-than-ideal solution of launching multiple successive runs and retrieving information in between. We overcome this problem using

the `external_devices` module: it provides capabilities for real-time injection and retrieval of spikes during a running PyNN simulation, thus allowing for control over the simulation. This live input/output interface creates a database that is subsequently used by external receivers and/or senders to identify recently activated neurons. The overall behaviour adopted by this module for communication between a SpiNNaker board and an external device is presented in Fig. 9.15.

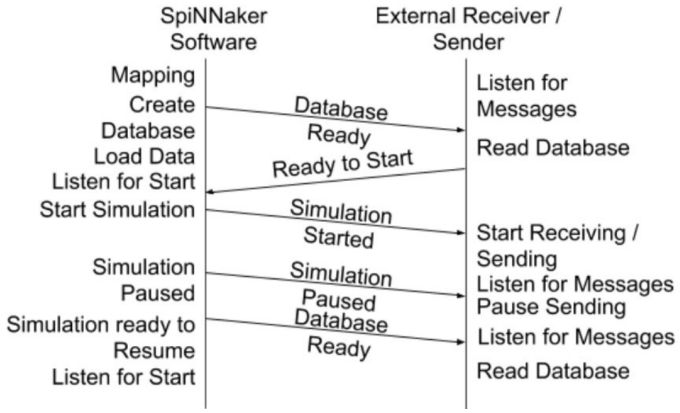


Figure 9.15 – Communication between the SpiNNaker board and an external device as handled by the `external_devices` module, from (Manchester, 2014).

The SpiNNaker board can thus receive event data in real-time but requires that the population of input neurons be implemented using the specific population type `SpikeInjectors`, which are sent using the `send_spike()` method (see Algo. .3 and .4 in Annexes). On the output side, the spikes can be extracted in real-time, at each timestep, using a "callback" function similar to a "listener" (see Algo. .5 in Annexes).

All in all, the module and population types described above could theoretically allow for the interfacing between our cooperative stereo-matching network on SpiNN-3 and our two event cameras to visualise the disparity map obtained from recorded data in real-time. For now, only a first step has been achieved: the real-time visualisation of the disparity map obtained from pre-recorded data, i.e. the "two fans" dataset (see Fig. G.7 in Annexes).

9.4 End-to-end neuromorphic lip reading

Human speech perception is intrinsically a multi-modal task since speech production requires the speaker to move the lips, producing visual cues in addition to auditory information. Lip reading consists in visually interpreting the movements of the lips to understand speech without the use of sound. The prospect of systems capable of reading lips is attractive for a variety of real-life applications, like assistive devices for persons with disabilities, security and surveillance applications, or automatic transcription of video content. It can either complement an audio-based speech recognition system or replace it when sound is not available. During this PhD, we supervised students in UCA master's degree in computer sciences on the subject of neuromorphic lip reading:

— M. Raphaël Pietrzak and Ms Juliette Sabatier (semester of Fall 2021) recorded and pre-processed a new event-based dataset for the classification of lip motions.

— M. Marcel Schweiker (semester of Fall 2022) re-implemented the ANN lip-reading model introduced in (Tan et al., 2022) and worked on its application to the i3S dataset, recorded by Raphaël and Juliette.
— M. Hugo Bulzomi (semester of Fall 2022) developed an SNN model for lip-reading, applied to the I3S dataset as well as to the event-based dataset introduced in (Tan et al., 2022).

We believe that the recording of an event-based lip-reading dataset and implementation of an end-to-end neuromorphic lip-reading system could directly benefit the development of portable assistive devices or surveillance equipment. As of now, SNNs are yet to be applied to automatic lip reading despite their attractiveness, supported by the arguments listed above. This innovative work is therefore the first to exploit its advantages for the specific task of lip-reading on event data.

9.4.1 i3S dataset

The goal of Raphaël and Juliette's internship initially was to develop and record an event-based dataset addressing a specific task that could not be achieved by solely considering events accumulated into a frame. Indeed, sometimes the simple observation of such a frame easily allows any ANN to determine the class without having to take into account the sequence of the frames (see example presented by Fig. 9.16). All in all, our motivation to propose such an internship was to reinforce the significant interest in using SNNs to process event data due to the temporal aspect of their learning.

We decided to focus on a dataset allowing for the classification of precise movements, where the time value of event videos would be crucial and properly exploited. We finally settled on registering lip movements to perform a potential reading on the lips, as no similar dataset existed at that time (DVS-Lip was later published in June 2022, see Sec. 3.3.5). Compared to other visual tasks, automatic lip reading deals with especially subtle movements with very precise timing and is very sensitive to noise and other environmental factors. A model for automatic lip reading thus needs to excel at extracting both spatial and temporal information.

All in all, the dataset we recorded, which will be referred to as the "i3S dataset"* in the remainder, contains 45 words recorded from 7 participants with 3 repetitions of each word per participant, for a total of 945 samples representing 11 GB of data.

9.4.1.1 Desired features

The i3S dataset focuses only on lip movements and has a wide variety of words/sounds in addition to profiles whose characteristics differentiate them from each other.

We identified different features necessary to obtain a complete and useful dataset:
— Similar recording conditions: it is necessary that the conditions (position of the face, lighting, setting, etc) be identical so that learning is not distorted by unintended elements such as noise or an element of a class that stands out from the others due to an external factor.
— Subjects' diversity: as this dataset records people, it is necessary to have diversity in the profiles in order not to introduce bias during learning. Recorded subjects were selected according to diversity in the following criteria: face shape, gender, and facial hair (hairless,

∗. We refer to this dataset as the "i3S dataset" and not the "UCA dataset" since it was recorded with funding from CNRS and not UCA.

beard, moustache, etc). We deemed the last criterion particularly important as it can hide part of the mouth and change the appearance of the movement to the camera.
— High number and variety of classes: the more different scenarios and repetitions of the same scenario a dataset contains, the more complete and precise it will be in its learning. The i3S dataset is thus composed of 45 words with 2 to 4 syllables, each containing at least one sound associated with each element of the French phonetic alphabet. The complete list of words can be found in Sec. H.1 in Annexes.
— Ease of use of the format: the format in which the data is saved should correspond to event data while being as easy as possible to use.

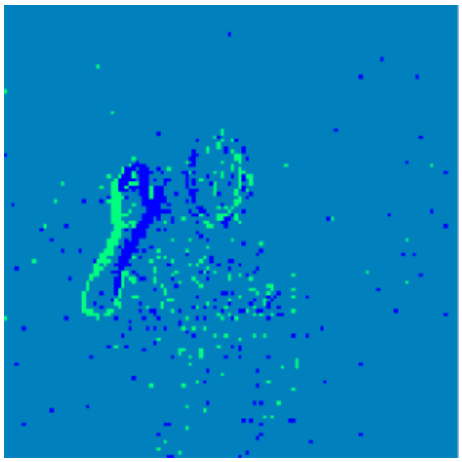


Figure 9.16 – Accumulated events from a DVS 128 Gesture sample. Here, one can easily guess the DVS 128 Gesture class presented: the polarity (green dots for positive events, blue for negative events) hints at movement going from right to left, and the lesser number of negative pixels at the bottom of the forearm suggests a rotational movement.

9.4.1.2 Dataset acquisition

The data was recorded using the EVK1 - Gen. 3.1 VGA camera from Prophesee, provided with the corresponding software Metavision and an extensive API to process the data. The event data is saved in raw format with "EVT2" encoding.

Only the bottom of the subjects' faces is recorded to remove any irrelevant disruptive information and allow for the anonymisation of people who have agreed to be registered for the dataset. To ensure the legality of the dataset, UCA's legal department assures that the GDPR does not apply given the tight framing on the lower part of the face and the retention of no personal data, as long as the participants sign a short legal form.

Each participant was asked to say each word three times to get different lip movements on the same word. The camera was placed at the level of their mouths and, using our instructions and the feedback they could see on our computer screen, they positioned themselves so that the bottom of their nose was visible at the top of the event video, about $10cm$ away from the lens. To stabilise themselves, the participants put both hands on the table. The lip movements for one word were performed $1s$ after the beginning of the recording, and an additional $1s$ was added before its stop.

9.4.1.3 Recorded data postprocessing

The camera parameters used to record the data were not optimal and did not totally prevent the formation of noise in the events. To solve this issue, the data was subsequently converted into Numpy arrays and then processed using the denoising function provided by the Tonic library (Lenz et al., 2021).

The operation of this function is simple: if an event occurs but no other event is detected within a pixel radius for a period of time given in the parameter, it is removed. Thus, with the right time parameter, the isolated events will be deleted and those participating in the movement kept.

Once processed, each sample is saved into a file tree structure adapted to the classification of spoken words: each repertory corresponds to one class, i.e. one word. Each class is subdivided into subclasses under the denomination "userX_Y", with X the subject number and Y their physical particularity (beard, moustache, etc). Each subclass contains the corresponding `RAW`, `BIAS` and `NPY` files, named "recording_Z.format" with Z the recording order. One example of such a tree file can be found in Sec. H.2 in Annexes.

9.4.2 Neuromorphic lip reading model

Under our supervision during the semester of Fall 2022, the intern Hugo Bulzomi proposed a neuromorphic model for lip reading taking as input the data produced by an event-based sensor capturing lip motion. The model classifies short sequences of events into word categories using an SNN architecture, thus leveraging the advantages of neuromorphic computing, including energy efficiency and low latency, making it suitable for real-time embedded scenarios. His work draws inspiration from Tan et al. (Tan et al., 2022), who achieved state-of-the-art results with an original DNN architecture for an automatic lip reading task on event data (see Sec. 3.3.5). Additionally, the model takes as input the DVS-Lip dataset Tan et al. introduced in (Tan et al., 2022). To the best of our knowledge, the model described below is the first proposal of an end-to-end neuromorphic lip reading model.

9.4.2.1 Event data preprocessing

As we opted to use surrogate gradient descent to train this lip reading model, it became necessary to convert our data to a synchronous form. Finding efficient ways to represent event data is a difficult problem subject of many prior studies. Since we target the DVS-Lip dataset, we chose to use a method similar to the one described by (Tan et al., 2022). In their study, the authors converted the asynchronous events into a 3-dimensional array, i.e. a voxel grid. Eq. 9.4 and 9.5 from (Tan et al., 2022) show how we can create a voxel grid of a specific length T where the polarity of each event is spread through the two closest spatiotemporal voxels:

$$t_k^* = \frac{T-1}{t_N - t_1}(t_k - t_1) \tag{9.4}$$

$$V(t,y,x) = \sum_k p_k max(0, 1 - |t - t_k^*|) \tag{9.5}$$

where T is the number of frames we want to use (i.e. the time resolution of our grid), and t_x is the timestamp of the x^{th} event from the original video. The event data are discretised accordingly into a 3-dimensional grid of shapes (t, x, y).

(Tan et al., 2022) uses two different values for T: 30 for the low-rate branch of their network and 210 for their high-rate branch. However, the individual performances of each branch when not combined are very similar (accuracy of 69.57% and 69.49% respectively). Only when both are combined in a multi-grained network along with message flow module blocks does the overall accuracy increase to 72.10%. After some trials, we manually set $T = 30$ since using a higher value would tremendously slow the training processes for little improvement, and a lower value leads to an accuracy collapse.

9.4.2.2 Topology exploration

To the best of our knowledge, no prior work uses SNNs to classify dynamic scenes for lip reading. Moreover, the literature on dynamic classification with SNNs is scarce. The closest studies to our work used the DVS-Gesture dataset introduced by (Amir et al., 2017) along with reasonably simple topologies and different variations, like in (Yao et al., 2021). The DVS-Gesture dataset itself is comparatively easy to classify, we hope to contribute to the area of neuromorphic computer vision by showing that more difficult problems can be tackled with SNNs.

We tested several topologies of different levels of complexity. Fig. 9.17 presents two simple SNN topologies that we used; we will refer to those models as SNN1 and SNN2. SNN1 takes inspiration from (Yao et al., 2021), who originally designed this topology to classify DVS-Gesture; while SNN2 takes inspiration from (L. Zhu et al., 2022), where the authors use a spiking encoder-decoder architecture for event-video reconstruction. SNN2 is the encoder part of their model, where the residual layers have been replaced by two more convolution layers, yielding a higher accuracy. Batch normalisation layers have been added after each convolution since such layers have been shown to considerably facilitate the learning process (Cordone, Miramond, & Thierion, 2022).

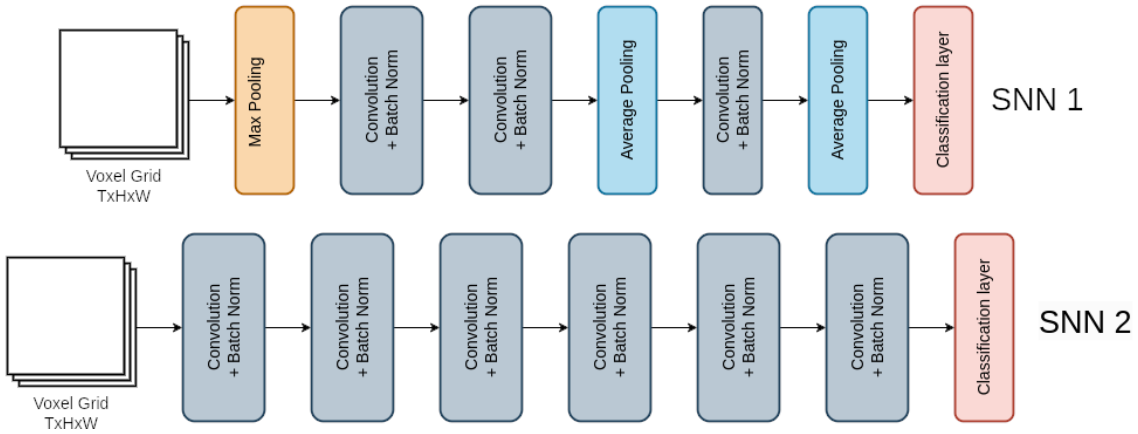


Figure 9.17 – Architecture of two lip reading SNN models.

Along with this simple architecture, we designed a spiking equivalent of the low-rate branch of MSTP. AS MSTP uses ResNet (He et al., 2016) as the backbone, we first implemented a spiking ResNet backbone (similarly to (Fang, Yu, Chen, Huang, et al., 2021)) and then applied the MSTP architecture. The topology of our spiking MSTP low-rate branch is presented in Fig. 9.18.

One key difference between our spiking adaptation of MSTP and the original model lies in the GRU layer employed near the architecture's output. RNNs are generally employed when informa-

tion has to be extracted from temporal sequences. Their output depends on both their current input as well as their hidden state. More particularly, GRU allows each neuron to learn how much of the previous information needs to be forgotten and how much of the current new input should be memorised (Chung, Gulcehre, Cho, & Bengio, 2014). Though a spiking LSTM layer (adaptable into a spiking GRU) has been proposed in (Lotfi Rezaabad & Vishwanath, 2020), we found it to give very underwhelming results in our case: our network exhibited a very slow and inefficient learning process, especially in the three original layers of bidirectional GRU used in MSTP. We ultimately decided to try to find alternatives for extracting temporal information to replace this GRU layer.

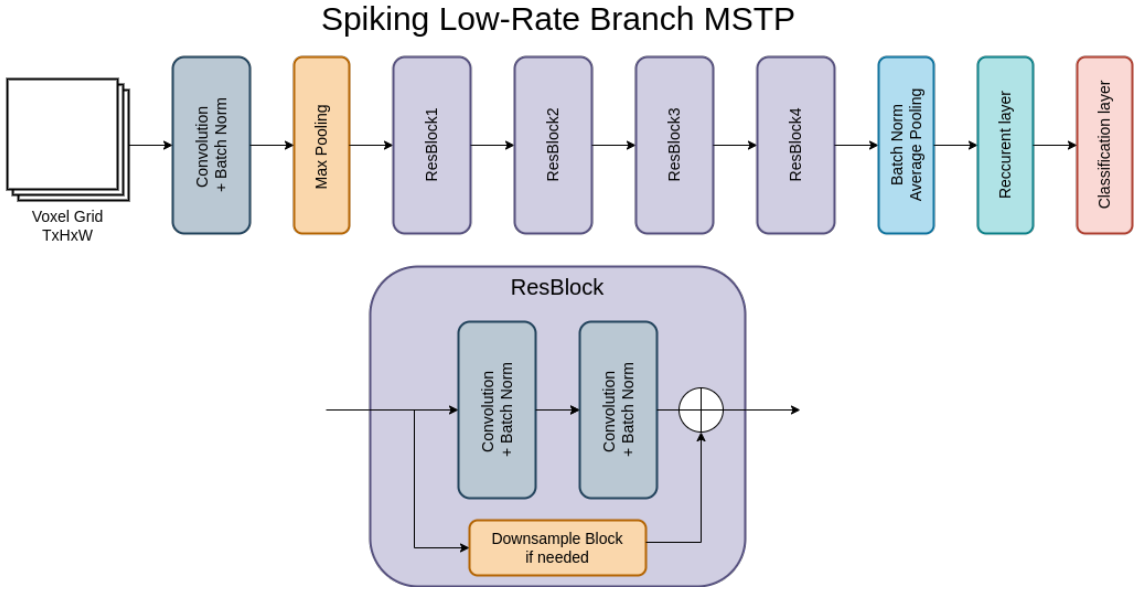


Figure 9.18 – Architecture of the overall spiking low-rate branch MSTP topology along with the associated ResBlock.

Table 9.3 presents the parameters used to implement the various SNN lip reading models using the PLIF neuronal model (Fang, Yu, Chen, Masquelier, et al., 2021).

Parameters	Values
Initial τ	2.0
Activation threshold	$1.0V$
Reset potential	$0.0V$

TABLE 9.3 – Neurons parameters of the lip-reading SNN model.

9.4.3 Experimental validation

Experimental results demonstrate the effectiveness of the proposed model in lip reading, achieving high accuracy in word classification.

9.4.3.1 Experimental setup

Experiments with SNNs were performed on the DVS-Lip dataset using 30 timesteps ($T = 30$ in Eq. 9.4) to help us identify the most promising spiking topology before comparing MSTP to this optimised SNN. The different models were implemented using the SpikingJelly library and the Adam optimiser with a learning rate of $1e^{-3}$, and a cosine annealing scheduler to adjust the learning rate during training (similarly to (Cordone et al., 2022)). All experiments were performed on a laptop with an Intel Core i9-12950HX CPU (2.5 GHz x 16), 62,5 GB RAM, with an NVIDIA RTX A5500 laptop GPU with 16 GB of VRAM.

9.4.3.2 Identification of the best topology

This paragraph presents the classification accuracy of the DVS-Lip dataset obtained with the three SNN models described above.

The choice of the surrogate function of the gradient descent can vastly influence how the networks perform. We tested three of the main existing surrogate functions: piecewise quadratic, ATan, and the Gaussian error surrogate function (Erf). Table 9.4 shows preliminary classification results on a subset of DVS-Lip, using the three functions mentioned above. As Erf's performance is slightly better than ATan and Piecewise's, we use this function in subsequent experiments.

Tab. 9.5 presents the performance on the whole DVS-Lip dataset of the three architectures described above. This second batch of experiments shows that the spiking MSTP obtains significantly better results than the other models (SNN1 and SNN2).

9.4.3.3 Ablation study and comparison to state-of-the-art

Up until this point, our Spiking MSTP did not use any recurrent layer and the original GRU was simply replaced by a spiking Fully Connected layer. Table 9.6 presents the results of different experiments with possible GRU replacement in order to identify whether we could reach a higher performance thanks to other temporal information extraction methods. The results, including the accuracy of the original MSTP published in (Tan et al., 2022), show that significant performance growth can be gained by using stateful synapses either as a spiking replacement for the GRU layer.

According to Table 9.6, the existing MSTP still outperforms our best SNN model. However, we observe in Table 9.7 that our model reaches 83% of the existing MSTP accuracy but requires an amount of memory 5 times smaller than the one used by the MSTP model. Furthermore, the main topological difference between our model and the low-rate branch of MSTP is the absence of the three layers of bidirectional GRU. If removed, the low-rate branch of the existing MSTP performs slightly worse than our Spiking MSTP with similar memory size. All in all, our model seems to be a promising option for embedded lip reading: thanks to its spiking nature, its lower accuracy makes a good trade-off for a more energy-efficient model.

9.4.4 Limitations and perspectives

Our lip reading model is amongst the first advanced deep spiking models applied to such a challenging task and manages to get promising results while keeping a relatively small size. Although the accuracy of the proposed model is lower than the current state-of-the-art, we demonstrate that SNNs can be used for complex video classification tasks since lip reading remains

Models	Activation function	Accuracy
SNN1	**Erf**	**0.546**
SNN1	Piecewise	0.531
SNN1	ATan	0.534

TABLE 9.4 – SNN1 accuracy on a subset of 10 classes from DVS-Lip, using different surrogate functions. The classes correspond to the words: allow, allowed, America, American, benefit, benefits, billion, called, challenge, and change.

Models	Variation	Accuracy
SNN1	Base model	0.395
SNN2	Base model	0.514
Spiking MSTP	No GRU	**0.522**

TABLE 9.5 – SNN results on the entire DVS-Lip dataset.

Model	Model	Accuracy
MSTP	**Original ANN (Tan et al., 2022)**	**0.721**
Spiking MSTP	Simple fc layer	0.522
Spiking MSTP	Spiking bi-GRU	0.463
Spiking MSTP	Spiking recurrent layer	0.476
Spiking MSTP	**Stateful synapse**	**0.602**

TABLE 9.6 – Spiking MSTP results on the entire DVS-Lip dataset, trying different substitutions for the 3-layer bidirectional GRU in the classification part.

Model	Accuracy	Size
Event Clouds MSTP	**0.721**	241.5MB
MSTP low branch without GRU	0.591	**47MB**
SNN1	0.395	**26.7MB**
SNN2	0.514	88.9MB
Spiking MSTP	**0.602**	47MB

TABLE 9.7 – Size and accuracy comparison between our models and the state-of-the-art.

very challenging even for humans. Moreover, this work provides valuable insights for future studies in this area. We demonstrate that surrogate gradient descent does provide a worthwhile option for supervised training of SNNs, thus giving our spiking MSTP the potential to have competitive results with those of regular deep ANNs.

Further experiments to improve the data pre-processing and the SNN model itself may, in the near future, allow a state-of-the-art model to be achieved. In the absence of published deep and complex SNNs for similar tasks, we hope to provide a spiking baseline for future work in this area.

We believe that our model can still be improved, especially regarding the replacement of the GRU layer ; even though our stateful synapses allowed us to break the 60% accuracy line, we believe it to be the cause of the performance gap between MSTP and this model and can be bridged by identifying more efficient ways to extract information from the temporal component of the data. Furthermore, even though we believe surrogate gradient descent is the current best training method for supervised learning with SNNs, it brings many of the regular ANNs' limitations to SNNs while performing a lot of approximations during training. We thus hope for the development of other training methods in the future, allowing us to tap into the full potential of SNNs.

Finally, we wish to compare the results described above, obtained on the DVS-Lip dataset, to new experiments conducted on the cleaned-up i3S dataset.

9.5 Conclusion

This chapter thus concludes the Contributions part of this manuscript. We detailed the different contributions we initiated and the related perspectives:

— the implementation of three approaches to tackle APROVIS3D's task of coastline detection to allow the autonomous navigation of a drone ;
— the extension of existing work to perform embedded 3D reconstruction using a stereo event stream ;
— the development of a model of end-to-end neuromorphic lip-reading with the associated recording of an event-based dataset.

Each of these axes of research has potential follow-ups, described in detail in this chapter. These include the development of a comprehensive dataset for coastline detection, the implementation of a demonstrator for embedded and self-sufficient 3D reconstruction, the development and extensive experimental validation of a model of weight and delay learning and the implementation of SpiNN-3 of an existing unsupervised delay learning model.

CHAPTER 10

Conclusion

As demonstrated multiple times in this manuscript, the embedded combined use of SNNs and event cameras is significantly attractive regarding biological inspiration, reduced energy consumption, decision latency, and memory use. It is gaining momentum in the field of embedded computer vision. However, their novelty leaves room for many improvements in terms of optimal preprocessing of data as well as how this data is processed, making the most of the singularity of these scientific concepts.

10.1 Summary of contributions

Within this PhD, we identified several challenges related to this broad research topic, which we tackled through two main questions:

1. How can we optimise the trade-off between quantity and quality of the visual information as a preprocess for embedded data processing ?
2. How can we exploit the relevance of the SNN to process the specific chronology of event data in an embedded context ?

We answered the first axis of research with three potential solutions: event data could either 1) be spatially or temporally reduced, either online or offline ; 2) keep only salient elements and discard the rest ; 3) be foveated, as a bio-plausible compromise between the two previous solutions. We compared qualitatively and quantitatively the event data obtained after each preprocessing method to assess whether the trade-off between the amount of data (i.e. events) kept *versus* the relevance of information kept is optimal. We also evaluated the computation time of each method, as their ultimate goal is to be embedded and able to run online.

The first solution, i.e. event data reduction, was implemented and experimentally assessed as ten different downscaling methods, of which six influence the spatial resolution and four the temporal resolution. We identified preferable methods for runtime, post-processing classification accuracy and the number of events: SNN pooling for spatial downscaling and stochastic structural for temporal downscaling. However, we highlighted the potential limitations of event data downscaling. There is a high variation in optimising the trade-off mentioned earlier, which strongly depends on the chosen method, dividing factor and application task. Additionally, we provided insights into the human perception of gesture classification using event data, thanks to an original human performance assessment experiment.

As a second solution to this first axis of research, we proposed the detection and selection of salient elements in the event scene thanks to a bio-inspired visual attention mechanism. We introduced a neuromorphic model for visual attention on event data, with multiple variations enabling

the selection of one or multiple OoIs in the visual scene. Based on event density, this adaptative model relies solely on intrinsic SNN dynamics and requires no training, merely a fine-tuning phase for a new type of data. It was designed on a CPU and deployed on the neuromorphic platforms Loihi, SpiNNaker and Loihi. Using this model, we managed to reduce the spatial repartition of events to 20% of the original one while maintaining a classification performance at 70% of the original one, with a detection latency under $15ms$. However, the simultaneous salient detection is only effective at most 50% of the time ; this is not efficient enough for an embedded model, whose real-life applications favour a significantly lower margin of error.

This leads us to our third and last solution: neuromorphic foveation, which corresponds to a feedback loop between a foveated event camera and a neuromorphic saliency detector, merging events at multiple resolutions according to the salient regions detected in the visual scene. This solution is ultimately to be the subject of a hardware implementation ; however, in the context of this PhD, we have implemented a first software version to begin the experimental validation of such a concept. We have designed a "binary" version of this concept, akin to the combination of the events of each sample in high resolution and low resolution using a mask. Using the binary implementation, we applied neuromorphic foveation to two use cases, gesture classification and semantic segmentation of a driving scene. We demonstrated the significant benefits of this method for optimising the trade-off between event quantity and quality. Indeed, in the embedded scenario, we managed to keep the same performance as with the original dataset on average while decreasing the mean number of events by two-thirds to reach the low-resolution value. Additionally, we have experimentally shown that most events kept in the foveated dataset provide relevant information while a significant part of the events in the original dataset is not as valuable but still provides interesting contextual information specifically relevant for semantic segmentation.

Overall, the results obtained during this PhD demonstrate that neuromorphic foveation leads to a good trade-off between information quantity and quality, at least on a software level.

We have undertaken several lines of research to address our second problem introduced earlier while remaining within the framework of the APROVIS3D project. To exploit the relevance of SNNs to process the particular temporal aspect of event data, we identified three embedded application tasks relevant to this issue and to APROVIS3D: coastline detection, 3D reconstruction and neuromorphic lip reading.

We have laid the foundations for three possible approaches to the first task: ANN-to-SNN conversion, adaptative line detection and delay learning. While the two initial approaches fail to exploit the event data temporality and the task could have been solved using frames instead of single events, the last one is auspicious by its bio-plausibility. Using delay learning, the events are segmented not according to the global object they form but thanks to their specific spatiotemporal patterns.

The second task calls for the implementation of bio-inspired stereo-matching models. We implemented the first steps towards a live demonstrator for neuromorphic 3D reconstruction based on existing work adapted to run online on a SpiNNaker board. Temporal timing of events and spikes also intervene in this model: each network cell encodes a belief in matching a heterolateral pair of pixels, relying on the temporal timing of the stereo activation to prevent homolateral matching.

The third task, neuromorphic lip reading, falls further away from the APROVIS3D spectrum but highly benefits from the SNN's intrinsic temporal processing and event data's crucial chronology. We implemented an SNN architecture requiring event data as input — as a similar task could not easily be solved using frames instead of individual events — to perform lip reading. It

leverages the advantages of neuromorphic computing, including energy efficiency and low latency, which makes it suitable for real-time embedded scenarios and produces promising results while keeping a relatively small size.

10.2 Perspectives

The results obtained during this PhD open the door to a multitude of potential developments, aiming to optimise embedded neuromorphic computer vision in the longer or shorter term.

10.2.1 Short term perspectives

Event data downscaling We believe that the subject of event data downscaling is quite promising. Indeed, other researchers started tackling this issue as well following the publication of our works (Rizzo, Schuman, & Plank, 2023). On our side, we believe we are still missing a final assessment: the complete evaluation of real-time data downscaling with the newer temporal methods. The diverse combinations of spatial and temporal methods could also bring a new dimension to the research field of event data downscaling.

Additionally, we believe that it would be highly beneficial for the scientific community to gain easier access to the methods we implemented and validated. They are currently available online on a GitHub repository ; one way to make them more accessible overall would be to integrate them into the Tonic library (Lenz et al., 2021), which positively encourages collaborative involvement and outsider contributions. This library already proposes an alternative version of the temporal and spatial funnelling methods (with no removal of duplicates) ; it could only benefit from the additional implementation of the eight methods we introduced.

Simultaneous saliency detection of multiple OoIs We believe some SNN architectures implemented during this PhD can be improved. The simultaneous saliency detection model for multiple OoIs requires further parameter tuning to increase the multi-OoIs detection performance beyond the current maximum of 50%. Additionally, this model was solely assessed for the simultaneous detection of two OoIs while it could theoretically reach a greater number ; this should be the subject of further experiments on more complex datasets and scenarios with a greater number of salient objects. Finally, it should be applied to more complex scenarios (such as driving scenes) to assess its viability on an embedded system.

A second possible improvement is to update this model to enable an adaptative number of OoIs to be detected. In this ideal model, the number of output layers could be adapted to the number of OoIs present in the visual scene, i.e. the number of regions identified by the saliency detector, *via* a feedback regulation: for example, the activity of the n_{th} output layer would give feedback on the existence of a n_{th} OoI in the current scene, allowing the network to add or remove output layers appropriately. Such an implementation would require neurogenesis and synaptogenesis (i.e. dynamic instantiation of neurons and synapses), which is currently unavailable on most SNN simulators and neuromorphic hardware ; however, we could mimic this bio-inspired behaviour by initiating multiple output layers and inhibiting them according to their use. Additionally, the framework behind Kraken (Di Mauro et al., 2022) may allow such a behaviour to be implemented, even in an embedded scenario. This possibility is still to be further confirmed with experimental trials.

The scientific community is producing more and more results demonstrating the benefits of deploying SNN architectures on hardware. This will both facilitate on-board implementation and interfacing with event cameras as well as demonstrate the advantage of these neural networks over standard ANNs, which are still widely favoured in computer vision nowadays. Therefore, we wish to validate our SNN architectures, particularly our visual attention model variations, on neuromorphic hardware — further than the work begun in (Bulzomi, Gruel, et al., 2023). These models could be deployed on SpiNNaker (S. B. Furber et al., 2013), which has the advantages over Loihi of being academic, thus more accessible for students, and able to implement dynamic adaptation of parameters mid-run.

Delay learning In this thesis, we proposed 1) an extension of the delay learning model introduced by (Nadafian & Ganjtabesh, 2020) and 2) the first tentative steps towards a learning rule for simultaneous weight and delay learning inspired by the traditional STDP learning rule. Both those works are ongoing and are to be improved in order to promote delay learning, a mechanism we are convinced could be a significant contribution to the neuromorphic community.

Among others, the demonstration of SpiNNaker's capacity to simulate such a model would further support our argument ; as well as its application to a real-world use case. Indeed, we aim to apply delay learning to the coastline detection task targeted by APROVIS3D ; their deployment on software would ease their embedded application.

10.2.2 Medium term perspectives

Neuromorphic foveation In this work, we assessed the neuromorphic foveation concept by implementing a binary version (with only two resolutions) on software. However, this work draws inspiration from a hardware concept introduced in (Serrano-Gotarredona et al., 2022), not yet available for experimental use due to the sanitary crisis. This foveated event camera we wish to emulate is more flexible than our binary implementation. A more thorough conceptualisation of the neuromorphic foveation concept is required to achieve a more realistic implementation. We could implement an adaptative number of resolutions, where the spatial dividing factor depends on the activity in the saliency detector (see Chapter 8): the higher the spike rate is, the more interesting the detected object is and the higher the resolution must be ; in a region of the sensor, the dividing factor would thus be inversely proportional to the corresponding spike rate. This nested foveation model would output data with several levels of finesse, starting from the lower resolution and then refining salient areas in overlapping sub-regions.

Now that the software implementation has been validated, we wish to integrate it on the foveated event camera created by IMSE (Serrano-Gotarredona et al., 2022). Indeed, this hardware deployment would enable real-time neuromorphic foveation thanks to the feedback loop provided by the saliency model deployed on an interfaced SpiNN-3.

Coastline detection As part of the APROVIS3D work group dedicated to embedded coastline detection, we still have to deploy a complete live demonstrator of an embedded SNN model applied to a live event stream and able to perform online coastline detection. This task is to be achieved in collaboration with INT and NTUA.

We introduced in this thesis a tentative lightweight SNN that could similarly solve this task. We believe this line of research would gain from being pursued further.

10.2.3 Long term perspectives

Saliency mechanism We implemented a bottom-up saliency mechanism applied to event data in this thesis. We arbitrarily defined salient elements as regions in the sensor where events take place close in time and space. However, there exist many different ways to define saliency in the literature, even limiting ourselves to bottom-up visual attention. We believe it would be highly beneficial for the neuromorphic community to further research ways to implement visual attention in event data using SNNs, since, as we demonstrated in this thesis, this is a promising challenge for limiting the processing of event data to interesting elements of the scene.

Concerning the simultaneous detection of multiple OoIs: our proposed model adds to the complexity of the network by implementing a new layer for each new OoI to be detected. This strays farther from biology, where the human eye detects and processes one OoI at a time, quickly alternating between salient elements using saccades. SNNs implementing this mechanism would allow for the successive detection and processing of each OoI in the scene ; once processed, the OoI is discarded using a temporal WTA mechanism, and the network outputs another OoI. Such a mechanism could easily handle a dynamic scene by successively processing OoIs as they come while being more bio-plausible. However, this research direction raises questions at a higher level: in the neuromorphic visual attention field of research, should we aim to stay close to biology and benefit from its efficiency ? Or take advantage of the computational power of neuromorphic hardware which allows, among other things, massively parallel processing ?

10.3 Next steps

Overall, we believe we provided results with a significant impact on embedded neuromorphic computing, especially concerning the optimisation of data preprocessing. Concrete proof of this can be found in the fact that the work carried out during this thesis on visual attention led to another PhD thesis, the one of Mr Hugo Bulzomi, with an application in autonomous driving. Indeed, Hugo will start a CIFRE PhD in Autumn 2023 in collaboration with i3S and IMRA Europe (a branch of Aisin) ; we co-authored the first work on his thesis subject, published in MVA 2023 (Bulzomi, Gruel, et al., 2023).

We hope to have made a valuable contribution to the exploitation of the unique advantages of combining two innovative concepts that are SNNs and event-based cameras. We believe in the usefulness of neuromorphic computing for the future of AI, thanks to its low memory and energy consumption as well as its high adaptability. This field of research benefits greatly from the interdisciplinary nature of the researchers involved, ranging from biology and computer science to microelectronics, robotics and mathematics. This field of research benefits greatly from the interdisciplinary nature of the researchers involved, ranging from biology and computer science to microelectronics, robotics and mathematics. As a result, scientific knowledge is being shared, leading to a boom in neuromorphic research.

Bibliography

References

Abeywickrama, H. V., Jayawickrama, B. A., He, Y., & Dutkiewicz, E. (2018). Empirical power consumption model for uavs. *IEEE Vehicular Technology Conference*. doi: 10.1109/vtcfall.2018.8690666

Acharya, J., Padala, V., & Basu, A. (2019). Spiking neural network based region proposal networks for neuromorphic vision sensors. In *2019 ieee international symposium on circuits and systems (iscas)* (pp. 1–5).

Afshar, S., Nicholson, A. P., van Schaik, A., & Cohen, G. (2020). Event-based object detection and tracking for space situational awareness. *IEEE Sensors Journal*, *20*(24), 15117-15132. doi: 10.1109/JSEN.2020.3009687

Afshar, S., Tapson, T. J. H. J., van Schaik, A., & Cohen, G. (2019). Investigation of Event-Based Surfaces for High-Speed Detection, Unsupervised Feature Extraction, and Object Recognition. *Frontiers in Neuroscience*.

Akopyan, F., Sawada, J., Cassidy, A., Alvarez-Icaza, R., Arthur, J., Merolla, P., ... Modha, D. S. (2015). Truenorth: Design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, *34*(10), 1537-1557. doi: 10.1109/TCAD.2015.2474396

Alakuijala, J., Farruggia, A., Ferragina, P., Kliuchnikov, E., Obryk, R., Szabadka, Z., & Vandevenne, L. (2018, 12). Brotli: A general-purpose data compressor. *ACM Transactions on Information Systems*, *37*, 1-30. doi: 10.1145/3231935

Almatrafi, M., Baldwin, R., Aizawa, K., & Hirakawa, K. (2020). Distance surface for event-based optical flow. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *42*(7), 1547–1556. doi: 10.1109/tpami.2020.2986748

Alnajjar, F., Zin, I. B. M., & Murase, K. (2008). A spiking neural network with dynamic memory for a real autonomous mobile robot in dynamic environment. In *2008 ieee international joint conference on neural networks (ieee world congress on computational intelligence)* (p. 2207-2213). doi: 10.1109/IJCNN.2008.4634103

Alom, M. Z., Taha, T. M., Yakopcic, C., Westberg, S., Sidike, P., Nasrin, M. S., ... Asari, V. K. (2018). The history began from AlexNet: A comprehensive survey on deep learning approaches. *CoRR*, *abs/1803.01164*. Consulté sur http://arxiv.org/abs/1803.01164

Alonso, I., & Murillo, A. (2019). EV-SegNet: Semantic Segmentation for Event-based Cameras. *CVPR W.*.

Ambroise, M., Levi, T., Joucla, S., Yvert, B., & Saïghi, S. (2013, 11). Real-time biomimetic central pattern generators in an fpga for hybrid experiments. *Frontiers in neuroscience*, *7*, 215. doi: 10.3389/fnins.2013.00215

Amir, A., Taba, B., Berg, D., Melano, T., McKinstry, J., Di Nolfo, C., ... Modha, D. (2017, juillet). A Low Power, Fully Event-Based Gesture Recognition System. In *Computer Vision and Pattern Recognition (CVPR)* (pp. 7388–7397). Honolulu, HI : IEEE. Consulté le 2021-02-23, sur https://ieeexplore.ieee.org/document/8100264/ doi: 10.1109/CVPR.2017.781

Anderson, J. R. (2005). *Cognitive Psychology and Its Implications*.

Araujo, H., & Dias, J. (1997). An introduction to the log-polar mapping. *Proceedings II Workshop on Cybernetic Vision*(1), 139–144. Consulté sur http://ieeexplore.ieee.org/document/629454/ doi: 10.1109/CYBVIS.1996.629454

Aydemir, B., Hoffstetter, L., Zhang, T., Salzmann, M., & Süsstrunk, S. (2023). Tempsal - uncovering temporal information for deep saliency prediction. *CoRR*, *abs/2301.02315*. Consulté sur https://doi.org/10.48550/arXiv.2301.02315 doi: 10.48550/arXiv.2301.02315

Baldwin, R. W., Almatrafi, M., Asari, V. K., & Hirakawa, K. (2020). Event probability mask (EPM) and event denoising convolutional neural network (edncnn) for neuromorphic cameras. *CoRR*, *abs/2003.08282*. Consulté sur https://arxiv.org/abs/2003.08282

Banerjee, S., Wang, Z. W., Chopp, H. H., Cossairt, O., & Katsaggelos, A. K. (2021). Lossy event compression based on image-derived quad trees and poisson disk sampling. In *2021 ieee international conference on image processing (icip)* (p. 2154-2158). doi: 10.1109/ICIP42928.2021.9506546

Bear, M. F., Connors, B. W., & Paradiso, M. A. (2007). The Human Eye. In *Neurosciences, Exploring the brain* (3e éd.).

Bekolay, T., Bergstra, J., Hunsberger, E., Dewolf, T., Stewart, T., Rasmussen, D., ... Eliasmith, C. (2014, 01). Nengo: A python tool for building large-scale functional brain models. *Frontiers in neuroinformatics*, *7*, 48. doi: 10.3389/fninf.2013.00048

Benjamin, B. V., Gao, P., McQuinn, E., Choudhary, S., Chandrasekaran, A. R., Bussat, J.-M., ... Boahen, K. (2014). Neurogrid: A mixed-analog-digital multichip system for large-scale neural simulations. *Proceedings of the IEEE*, *102*(5), 699-716. doi: 10.1109/JPROC.2014.2313565

Benosman, R., Clercq, C., Lagorce, X., Ieng, S.-H., & Bartolozzi, C. (2014). Event-based visual flow. *IEEE Transactions on Neural Networks and Learning Systems*, *25*(2), 407-417. doi: 10.1109/TNNLS.2013.2273537

Bernert, M., & Yvert, B. (2019, octobre). An Attention-Based Spiking Neural Network for Unsupervised Spike-Sorting. *International Journal of Neural Systems*, *29*(08), 1850059. Consulté le 2021-02-08, sur https://www.worldscientific.com/doi/abs/10.1142/S0129065718500594 doi: 10.1142/S0129065718500594

Bi, Y., Chadha, A., Abbas, A., Bourtsoulatze, E., & Andreopoulos, Y. (2019). Graph-based object classification for neuromorphic vision sensing. In *2019 ieee/cvf international conference on computer vision (iccv)* (p. 491-501). doi: 10.1109/ICCV.2019.00058

Bichot, N., Heard, M., DeGennaro, E., & Desimone, R. (2015, novembre). A source for feature based attention in the prefrontal cortex. *Neuron*, *88*. doi: 10.1016/j.neuron.2015.10.001

Biglari, A., & Tang, W. (2023, Feb). A review of embedded machine learning based on hardware, application, and sensing scheme. *Sensors*, *23*(4), 2131. Consulté sur http://dx.doi.org/10.3390/s23042131 doi: 10.3390/s23042131

Binas, J., Neil, D., Liu, S.-C., & Delbruck, T. (2017). DDD17: End-To-End DAVIS Driving Dataset. *arXiv:1711.01458 [cs]*.

Birkoben, T., Winterfeld, H., Fichtner, S., Petraru, A., & Kohlstedt, H. (2020, 10). A spiking and adapting tactile sensor for neuromorphic applications. *Scientific Reports*, *10*. doi: 10.1038/s41598-020-74219-1

Blalock, D. W., Madden, S., & Guttag, J. V. (2018). Sprintz: Time series compression for the internet of things. *CoRR*, *abs/1808.02515*. Consulté sur http://arxiv.org/abs/1808.02515

Bogdan, P., García, G., Davidson, S., Hopkins, M., James, R., & Furber, S. (2019). Event-based computation: Unsupervised elementary motion decomposition. In *Emerging technology conference.*

Bohté, S. M., Kok, J. N., & Poutré, H. L. (2000). Error-backpropagation in temporally encoded networks of spiking neurons. *Neurocomputing*, *48*, 17-37.

Borji, A., & Itti, L. (2013). State-of-the-art in visual attention modeling. *IEEE PAMI*, *35*(1), 185-207. doi: 10.1109/TPAMI.2012.89

Bouvier, M., Valentian, A., Mesquida, T., Rummens, F., Reyboz, M., Vianello, E., & Beigné, E. (2020). Spiking neural networks hardware implementations and challenges: a survey. *CoRR*, *abs/2005.01467*. Consulté sur https://arxiv.org/abs/2005.01467

Bryner, S., Gallego, G., Rebecq, H., & Scaramuzza, D. (2019). Event-based, Direct Camera Tracking from a Photometric 3D Map using Nonlinear Optimization. In *IEEE int. conf. robot. autom. (icra).*

Bulzomi, H. (2022). *Synaptic delays for temporal pattern recognition* [Rapport de tutorat de 14 jours — M1 Informatique].

Calabrese, E., Taverni, G., Awai Easthope, C., Skriabine, S., Corradi, F., Longinotti, L., ... Delbruck, T. (2019, June). DHP19: Dynamic Vision Sensor 3D Human Pose Dataset. In *The ieee conference on computer vision and pattern recognition (cvpr) workshops.*

Cannici, M., Ciccone, M., Romanoni, A., & Matteucci, M. (2018, novembre). Attention Mechanisms for Object Recognition with Event-Based Cameras. *arXiv*.

Chan, V., Liu, S.-C., & van Schaik, A. (2007). Aer ear: A matched silicon cochlea pair with address event representation interface. *IEEE Transactions on Circuits and Systems I: Regular Papers*, *54*(1), 48-59. doi: 10.1109/TCSI.2006.887979

Chapel, M., & Bouwmans, T. (2020). Moving objects detection with a moving camera: A comprehensive review. *CoRR*, *abs/2001.05238*. Consulté sur https://arxiv.org/abs/2001.05238

Chen, S., & Guo, M. (2019). Live demonstration: Celex-v: A 1m pixel multi-mode event-based sensor. In *Cvprw.* doi: 10.1109/CVPRW.2019.00214

Chen, X., Liang, C., Huang, D., Real, E., Wang, K., Liu, Y., ... Le, Q. V. (2023). Symbolic discovery of optimization algorithms. *ArXiv*, *abs/2302.06675*.

Cheng, W., Luo, H., Yang, W., Yu, L., Chen, S., & Li, W. (2019). DET: A High-resolution DVS Dataset for Lane Extraction. In *Proc. ieee conf. comput. vis. pattern recog. workshops.*

Chevallier, S., Cuperlier, N., & Gaussier, P. (2010, septembre). Efficient neural models for visual attention. In *Computer Vision and Graphics.* doi: 10.1007/978-3-642-15910-7

Chevallier, S., Paugam-Moisy, H., & Lemaître, F. (2005, février). Distributed processing for modelling real-time multimodal perception in a virtual robot. In *PDCN'2005*.

Chevallier, S., & Tarroux, P. (2008, mai). Covert Attention with a Spiking Neural Network. In *Intl Conference on Computer Vision Systems*. doi: 10.1007/978-3-540-79547-6_6

Chik, D., Borisyuk, R., & Kazanovich, Y. (2009, septembre). Selective attention model with spiking elements. *Neural Networks*. doi: 10.1016/j.neunet.2009.02.002

Chin, T.-J., Bagchi, S., Eriksson, A., & Schaik, A. V. (2019). Star tracking using an event camera. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. doi: 10.1109/cvprw.2019.00208

Christensen, D. V., Dittmann, R., Linares-Barranco, B., Sebastian, A., Gallo, M. L., Redaelli, A., ... Pryds, N. (2022, may). 2022 roadmap on neuromorphic computing and engineering. *Neuromorphic Computing and Engineering*, *2*(2), 022501. Consulté sur https://doi.org/10.1088%2F2634-4386%2Fac4a83 doi: 10.1088/2634-4386/ac4a83

Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Cohen, G., Afshar, S., van Schaik, A., Wabnitz, A., Bessell, T., Rutten, M., & Morreale, B. (2017, 11). Event-based sensing for space situational awareness..

Constantin, K. (2023). *Synaptic delays for temporal pattern recognition* [Rapport de TER — M1 Informatique]. Consulté le 2023/07/06, sur https://github.com/KevayneCst/TER/blob/main/REPORT_SynapticDelaysForTemporalPatternRecognition.pdf

Cordone, L., Miramond, B., & Ferrante, S. (2021, juillet). Learning from Event Cameras with Sparse Spiking Convolutional Neural Networks. In IEEE (Ed.), *International Joint Conference On Neural Networks 2021 (IJCNN 2021)* (p. 8). Conférence virtuelle, China. Consulté sur https://hal.archives-ouvertes.fr/hal-03208690

Cordone, L., Miramond, B., & Thierion, P. (2022). Object detection with spiking neural networks on automotive event data. *arXiv preprint arXiv:2205.04339*.

Cox, D., & Dean, T. (2014). Neural networks and neuroscience-inspired computer vision. *Current Biology*, *24*(18), R921-R929. Consulté sur https://www.sciencedirect.com/science/article/pii/S0960982214010392 doi: https://doi.org/10.1016/j.cub.2014.08.026

Dabane, G., Perrinet, L. U., & Daucé, E. (2022). What you see is what you transform: Foveated spatial transformers as a bio-inspired attention mechanism. In *Ijcnn 2022 : International joint conference on neural networks*. Consulté le 2022-05-11, sur https://www.techrxiv.org/articles/preprint/What_You_See_Is_What_You_Transform_Foveated_Spatial_Transformers_as_a_bio-inspired_attention_mechanism/16550391/1 doi: 10.36227/techrxiv.16550391.v1

Dai, Z., Liu, H., Le, Q. V., & Tan, M. (2021). Coatnet: Marrying convolution and attention for all data sizes. *CoRR*, *abs/2106.04803*. Consulté sur https://arxiv.org/abs/2106.04803

D'Angelo, G., Janotte, E., Schoepe, T., O'Keeffe, J., Milde, M., Chicca, E., & Bartolozzi, C. (2020, 05). Event-based eccentric motion detection exploiting time difference encoding. *Frontiers in Neuroscience*, *14*, 451. doi: 10.3389/fnins.2020.00451

Daucé, E., Albiges, P., & Perrinet, L. U. (2020). A dual foveal-peripheral visual processing model implements efficient saccade selection. *Journal of Vision*, *20*(8), 22–22. Consulté le 2021-07-02, sur https://jov.arvojournals.org/article.aspx?articleid=2770680 doi: 10.1167/jov.20.8.22

Daucé, E., & Perrinet, L. (2020). Visual Search as Active Inference. In T. Verbelen, P. Lanillos, C. L. Buckley, & C. De Boom (Eds.), *Active Inference* (pp. 165–178). Springer International Publishing. doi: 10.1007/978-3-030-64919-7_17

Davies, M., Srinivasa, N., Lin, T.-H., Chinya, G., Cao, Y., Choday, S. H., ... Wang, H. (2018). Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro*, *38*(1), 82-99. doi: 10.1109/MM.2018.112130359

Davies, M., Wild, A., Orchard, G., Sandamirskaya, Y., Guerra, G. A. F., Joshi, P., ... Risbud, S. R. (2021). Advancing neuromorphic computing with loihi: A survey of results and outlook. *Proceedings of the IEEE*, *109*.

Davison, A. P., Brüderle, D., Eppler, J. M., Kremkow, J., Muller, E., Pecevski, D., ... Yger, P. (2009, Jan). Pynn: a common interface for neuronal network simulators. *Frontiers in Neuroinformatics*, *0*. doi: 10.3389/neuro.11.011.2008

Debat, G., Chauhan, T., Cottereau, B. R., Masquelier, T., Paindavoine, M., & Baures, R. (2021, May). Event-based trajectory prediction using spiking neural networks. *Frontiers in Computational Neuroscience*, *15*. doi: 10.3389/fncom.2021.658764

Delbruck, T. (2008a). Frame-free dynamic digital vision. In *International symposium on secure-life electronics*. Consulté sur https://doi.org/10.5167/uzh-17620

Delbruck, T. (2008b). Frame-free dynamic digital vision. In *Proceedings of intl. symp. on secure-life electronics* (p. 21-26.). Tokyo, Japan.

Delbrück, T., Graca, R., & Paluch, M. (2021). Feedback control of event cameras. *CoRR*, *abs/2105.00409*. Consulté sur https://arxiv.org/abs/2105.00409

Delorme, A., Perrinet, L. U., Thorpe, S. J., & Samuelides, M. (2001). Network of integrate-and-fire neurons using rank order coding b: spike timing dependant plasticity and emergence of orientation selectivity. *Neurocomputing*, *38–40*(1–4), 539–45. Consulté sur http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.18.4990 doi: 10.1.1.18.4990

De Micco, L., Vargas, F. L., & Fierens, P. I. (2020). A literature review on embedded systems. *IEEE Latin America Transactions*, *18*(02), 188-205. doi: 10.1109/TLA.2020.9085271

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 ieee conference on computer vision and pattern recognition* (pp. 248–255).

Desislavov, R., Martínez-Plumed, F., & Hernández-Orallo, J. (2021). Compute and energy consumption trends in deep learning inference. *CoRR*, *abs/2109.05472*. Consulté sur https://arxiv.org/abs/2109.05472

de Tournemire, P., Nitti, D., Perot, E., Migliore, D., & Sironi, A. (2020). A large scale event-based detection dataset for automotive. *CoRR*, *abs/2001.08499*. Consulté sur https://arxiv.org/abs/2001.08499

Dikov, G., Firouzi, M., Röhrbein, F., Conradt, J., & Richter, C. (2017, 07). Spiking cooperative stereo-matching at 2 ms latency with neuromorphic hardware. In (p. 119-137). doi: 10.1007/978-3-319-63537-8_11

Dilmaghani, M. S., Shariff, W., Ryan, C., Lemley, J., & Corcoran, P. (2022). *Control and evaluation of event cameras output sharpness via bias.*

Di Mauro, A., Scherer, M., Mas, J. F., Bougenot, B., Magno, M., & Benini, L. (2021). Flydvs: An event-driven wireless ultra-low power visual sensor node. In *2021 design, automation & test in europe conference & exhibition (date)* (pp. 1851–1854).

Di Mauro, A., Scherer, M., Rossi, D., & Benini, L. (2022). Kraken: A direct event/frame-based multi-sensor fusion soc for ultra-efficient visual processing in nano-uavs. In *Hcs* (p. 1-19). doi: 10.1109/HCS55958.2022.9895621

Ding, S., Chen, J., Wang, Y., Kang, Y., Song, W., Cheng, J., & Cao, Y. (2023). E-mlb: Multilevel benchmark for event-based camera denoising. *IEEE Transactions on Multimedia*, 1-12. doi: 10.1109/TMM.2023.3260638

D'Angelo, G., Perrett, A., Iacono, M., Furber, S., & Bartolozzi, C. (2022). Event driven bio-inspired attentive system for the icub humanoid robot on spinnaker. *Neuromorphic Computing and Engineering*.

Ercsey-Ravasz, M., Markov, N., Lamy, C., Essen, D., Knoblauch, K., Toroczkai, Z., & Kennedy, H. (2013, 10). A predictive network model of cerebral cortical connectivity based on a distance rule. *Neuron*, *80*, 184-197. doi: 10.1016/j.neuron.2013.07.036

Eriksen, C. W., & St. James, J. D. (1986, juillet). Visual attention within and around the field of focal attention: A zoom lens model. *Perception & Psychophysics*, *40*(4), 225–240. Consulté le 2021-01-20, sur https://doi.org/10.3758/BF03211502 doi: 10.3758/BF03211502

Eshraghian, J. K., Ward, M., Neftci, E., Wang, X., Lenz, G., Dwivedi, G., ... Lu, W. D. (2021). Training spiking neural networks using lessons from deep learning. *arXiv preprint arXiv:2109.12894*.

Esser, S. K., Merolla, P. A., Arthur, J. V., Cassidy, A. S., Appuswamy, R., Andreopoulos, A., ... Modha, D. S. (2016). Convolutional networks for fast, energy-efficient neuromorphic computing. *CoRR*, *abs/1603.08270*. Consulté sur http://arxiv.org/abs/1603.08270

Falez, P., Tirilly, P., Bilasco, I. M., Devienne, P., & Boulet, P. (2019). Unsupervised visual feature learning with spike-timing-dependent plasticity: How far are we from traditional feature learning approaches ? *CoRR*, *abs/1901.04392*. Consulté sur http://arxiv.org/abs/1901.04392

Fang, W., Yu, Z., Chen, Y., Huang, T., Masquelier, T., & Tian, Y. (2021). Deep residual learning in spiking neural networks. *Advances in Neural Information Processing Systems*, *34*, 21056–21069.

Fang, W., Yu, Z., Chen, Y., Masquelier, T., Huang, T., & Tian, Y. (2021, October). Incorporating learnable membrane time constant to enhance learning of spiking neural networks. In *International conference on computer vision (iccv)* (p. 2661-2671).

Feng, Y., Lv, H., Liu, H., Zhang, Y., Xiao, Y., & Han, C. (2020). Event density based denoising method for dynamic vision sensor. *Applied Sciences*, *10*(6). doi: 10.3390/app10062024

Fidjeland, A., Roesch, E., Shanahan, M., & Luk, W. (2009, 07). Nemo: A platform for neural modelling of spiking neurons using gpus. In (p. 137-144). doi: 10.1109/ASAP.2009.24

Fields, R. (2015). A new mechanism of nervous system plasticity: activity-dependent myelination. *Nature Reviews Neuroscience*, *16*. doi: 10.1038/nrn4023

Finateu, T., Niwa, A., Matolin, D., Tsuchimoto, K., Mascheroni, A., Reynaud, E., ... Posch, C. (2020). 5.10 a 1280×720 back-illuminated stacked temporal contrast event-based vision sensor with 4.86µm pixels, 1.066geps readout, programmable event-rate controller and compressive data-formatting pipeline. In *2020 ieee international solid-state circuits conference - (isscc)* (p. 112-114). doi: 10.1109/ISSCC19947.2020.9063149

Franceschini, N., Ruffier, F., & Serres, J. (2009, août). Optic Flow Based Autopilots: Speed Control and Obstacle Avoidance. In *Flying Insects and Robots* (p. 29-50). Springer Berlin Heidelberg. Consulté sur https://amu.hal.science/hal-02294508 doi: 10.1007/978-3-540-89393-6_3

Furber, S., & Bogdan, P. (2020). *Spinnaker - a spiking neural network architecture*. NOW Publishers INC.

Furber, S. B., John Bainbridge, W., Mike Cumpstey, J., & Temple, S. (2004). Sparse distributed memory using n-of-m codes. *Neural Networks*, *17*(10), 1437-1451. Consulté sur https://www.sciencedirect.com/science/article/pii/S0893608004001443 doi: https://doi.org/10.1016/j.neunet.2004.07.003

Furber, S. B., Lester, D. R., Plana, L. A., Garside, J. D., Painkras, E., Temple, S., & Brown, A. D. (2013). Overview of the spinnaker system architecture. *IEEE Transactions on Computers*, *62*(12), 2454-2467. doi: 10.1109/TC.2012.142

Gallego, G., Delbruck, T., Orchard, G., & al. (2020). Event-based vision: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Gallego, G., Lund, J. E. A., Mueggler, E., Rebecq, H., Delbruck, T., & Scaramuzza, D. (2018, octobre). Event-based, 6-DOF camera tracking from photometric depth maps. *IEEE Trans. Pattern Anal. Mach. Intell.*, *40*(10), 2402–2412. doi: 10.1109/TPAMI.2017.2769655

Gallego, G., Rebecq, H., & Scaramuzza, D. (2018). A unifying contrast maximization framework for event cameras, with applications to motion, depth, and optical flow estimation. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 3867-3876. Consulté sur https://api.semanticscholar.org/CorpusID:4597042

Gallego, G., & Scaramuzza, D. (2017). Accurate angular velocity estimation with an event camera. *IEEE Robotics and Automation Letters*, *2*, 632-639. Consulté sur https://api.semanticscholar.org/CorpusID:3328976

Gehrig, D., Gehrig, M., Hidalgo-Carrió, J., & Scaramuzza, D. (2020, June). Video to events: Recycling video datasets for event cameras. In *Ieee conf. comput. vis. pattern recog. (cvpr)*.

Gehrig, D., & Scaramuzza, D. (2022). Are high-resolution cameras really needed ? *arXiv*.

Gehrig, M., Aarents, W., Gehrig, D., & Scaramuzza, D. (2021). Dsec: A stereo event camera dataset for driving scenarios. *IEEE Robotics and Automation Letters*. doi: 10.1109/LRA.2021.3068942

Geirhos, R., Jacobsen, J.-H., Michaelis, C., Zemel, R., Brendel, W., Bethge, M., & Wichmann, F. A. (2020, nov). Shortcut learning in deep neural networks. *Nature Machine Intelligence*, *2*(11), 665–673. Consulté sur https://doi.org/10.1038%2Fs42256-020-00257-z doi: 10.1038/s42256-020-00257-z

Geisler, W. S., & Perry, J. S. (1998). Real-time foveated multiresolution system for low-bandwidth video communication. In B. E. Rogowitz & T. N. Pappas (Eds.), *Human vision and*

electronic imaging iii (Vol. 3299, pp. 294 – 305). SPIE. Consulté sur https://doi.org/10.1117/12.320120 doi: 10.1117/12.320120

Gerstner, W., Kempter, R., van Hemmen, L., & Wagner, H. (1996). A neuronal learning rule for sub-millisecond temporal coding. *Nature*, *383*. doi: 10.1038/383076a0

Gerstner, W., Ritz, R., & van Hemmen, L. (1993, 10). Why spikes ? hebbian learning and retrieval of time-resolved excitation patterns. *Biological Cybernetics*, *69*, 503-515. doi: 10.1007/BF00199450

Gewaltig, M.-O., & Diesmann, M. (2007). Nest (neural simulation tool). *Scholarpedia*, *2*(4), 1430.

Ghosh, S., D'Angelo, G., Glover, A., Iacono, M., Niebur, E., & Bartolozzi, C. (2022, May). Event-driven proto-object based saliency in 3d space to attract a robot's attention. *Scientific reports*, *12*(1), 7645. Consulté sur https://europepmc.org/articles/PMC9090933 doi: 10.1038/s41598-022-11723-6

Glover, A., & Bartolozzi, C. (2016). Event-driven ball detection and gaze fixation in clutter. *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. doi: 10.1109/iros.2016.7759345

Gregor, K., Danihelka, I., Graves, A., & Wierstra, D. (2015). DRAW: A recurrent neural network for image generation. *CoRR*, *abs/1502.04623*. Consulté sur http://arxiv.org/abs/1502.04623

Grimaldi, A., Besnainou, C., Ladret, H., & Perrinet, L. U. (2022). Learning heterogeneous delays of spiking neurons for motion detection. In *Proceedings of icip 2022*. Consulté sur https://ieeexplore.ieee.org/document/9897394/ doi: 10.1109/ICIP46576.2022.9897394

Grimaldi, A., Boutin, V., Ieng, S.-H., Benosman, R., & Perrinet, L. (2022). *A robust event-driven approach to always-on object recognition.* Consulté sur https://www.techrxiv.org/articles/preprint/A_robust_event-driven_approach_to_always-on_object_recognition/18003077 doi: 10.36227/techrxiv.18003077.v1

Grimaldi, A., Boutin, V., Ieng, S.-H., Benosman, R., & Perrinet, L. U. (2023). A robust event-driven approach to always-on object recognition. *Submitted*.

Grimaldi, A., & Perrinet, L. U. (2023). Learning heterogeneous delays in a layer of spiking neurons for fast motion detection. *Submitted to Biological Cybernetics*.

Gruel, A., Martinet, J., Serrano-Gotarredona, T., & Linares-Barranco, B. (2022). Event data downscaling for embedded computer vision. In *Visapp*.

Guo, M., Huang, J., & Chen, S. (2017). Live demonstration: A 768 × 640 pixels 200meps dynamic vision sensor. In *2017 ieee international symposium on circuits and systems (iscas)* (p. 1-1). doi: 10.1109/ISCAS.2017.8050397

Guo, S., & Delbruck, T. (2022, février). Low cost and latency event camera background activity denoising. *IEEE Trans. Pattern Anal. Mach. Intell., PP*. Consulté sur http://dx.doi.org/10.1109/TPAMI.2022.3152999 doi: 10.1109/TPAMI.2022.3152999

Guo, Y.-Z., Tong, X.-Y., Chen, Y., Zhang, L., Liu, X., Ma, Z., & Huang, X. (2022). Recdis-snn: Rectifying membrane potential distribution for directly training spiking neural networks. *Computer Vision and Pattern Recognition*. doi: 10.1109/cvpr52688.2022.00042

Gütig, R., & Sompolinsky, H. (2005, 01). The tempotron: a neuron that learns spike-timing based decisions. *Reviews in the neurosciences*, *16*, S27-S27.

Hagenaars, J., Paredes-Valles, F., & de Croon, G. (2021). Self-supervised learning of event-based optical flow with spiking neural networks. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, & J. W. Vaughan (Eds.), *Advances in neural information processing systems* (Vol. 34, pp. 7167–7179). Curran Associates, Inc. Consulté sur https://proceedings.neurips.cc/paper/2021/file/39d4b545fb02556829aab1db805021c3-Paper.pdf

Hao, Q., Tao, Y., Cao, J., Tang, M., Cheng, Y., Zhou, D., ... Cui, H. (2021). Retina-like Imaging and Its Applications: A Brief Review. *Applied Sciences*, *11*(15), 7058. Consulté le 2021-10-17, sur https://www.mdpi.com/2076-3417/11/15/7058 doi: 10.3390/app11157058

Hazan, H., Saunders, D. J., Khan, H., Patel, D., Sanghavi, D. T., Siegelmann, H. T., & Kozma, R. (2018). Bindsnet: A machine learning-oriented spiking neural networks library in python. *Frontiers in Neuroinformatics*, *12*, 89. Consulté sur https://www.frontiersin.org/article/10.3389/fninf.2018.00089 doi: 10.3389/fninf.2018.00089

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 770–778).

Hebb, D. (1949). The organization of behavior: A neuropsychological theory. *Journal of the American Medical Association*, *143*(12). doi: 10.1001/jama.1950.02910470083028

Hines, M., & Carnevale, N. (2001, 05). Neuron: a tool for neuroscientists. *The Neuroscientist : a review journal bringing neurobiology, neurology and psychiatry*, *7*, 123-35. doi: 10.1177/107385840100700207

Hodgkin, A. L., & Huxley, A. F. (1952). Currents carried by sodium and potassium ions through the membrane of the giant axon of loligo. *The Journal of physiology*, *116*(4), 449–472.

Hu, Y. (2016). Generation of Benchmarks for Visual Recognition with Spiking Neural Networks. In *Nsc short project report* (Vol. 10, p. 405). Switzerland.

Hu, Y., Binas, J., Neil, D., Liu, S.-C., & Delbruck, T. (2020, September). DDD20 End-to-End Event Camera Driving Dataset: Fusing Frames and Events with Deep Learning for Improved Steering Prediction. In *Special session beyond traditional sensing for intelligent transportation, the 23rd ieee international conference on intelligent transportation systems.* Rhodes, Greece.

Hu, Y., Liu, H., Pfeiffer, M., & Delbruck, T. (2016). Dvs benchmark datasets for object tracking, action recognition and object recognition. In *Frontiers in neuromorphic engineering* (Vol. 10, p. 405).

Huang, C. (2021). Event-based timestamp image encoding network for human action recognition and anticipation. In *2021 international joint conference on neural networks (ijcnn)* (p. 1-9).

Hüning, H., Glünder, H., & Palm, G. (1998). Synaptic delay learning in pulse-coupled neurons. *Neural Computation*, *10*(3), 555-565. doi: 10.1162/089976698300017665

Iacono, M., D'Angelo, G., Glover, A., Tikhanoff, V., Niebur, E., & Bartolozzi, C. (2019). Proto-object based saliency for event-driven cameras. In *Iros* (p. 805-812). doi: 10.1109/IROS40897.2019.8967943

Indiveri, G., & Sandamirskaya, Y. (2019). The importance of space and time for signal processing in neuromorphic agents: The challenge of developing low-power, autonomous agents that interact with the environment. *IEEE Signal Processing Magazine*, *36*, 16-28. Consulté sur https://api.semanticscholar.org/CorpusID:207832248

Itti, L., & Koch, C. (2001, mars). Computational modelling of visual attention. *Nature reviews. Neuroscience*. doi: 10.1038/35058500

Izhikevich, E. (2006, 03). Polychronization: Computation with spikes. *Neural computation*, *18*, 245-82. doi: 10.1162/089976606775093882

Izhikevich, E. M. (2004, Sep). Which model to use for cortical spiking neurons ? *IEEE transactions on neural networks*, *15*(5), 1063–1070. doi: 10.1109/TNN.2004.832719

James, W. (1890). *The principles of psychology*. H. Holt and Co.

Javanshir, A., Nguyen, T. T., Mahmud, M. A. P., & Kouzani, A. Z. (2022, 05). Advancements in Algorithms and Neuromorphic Hardware for Spiking Neural Networks. *Neural Computation*, *34*(6), 1289-1328. Consulté sur https://doi.org/10.1162/neco_a_01499 doi: 10.1162/neco_a_01499

Jetley, S., Lord, N. A., Lee, N., & Torr, P. H. S. (2018, avril). Learn To Pay Attention. *arXiv*.

Jonides, J. (1983, avril). Further toward a model of the Mind's eye's movement. *Bulletin of the Psychonomic Society*, *21*(4), 247–250. Consulté le 2021-01-19, sur https://doi.org/10.3758/BF03334699 doi: 10.3758/BF03334699

Jérémie, J.-N., Daucé, E., & Perrinet, L. U. (2023). Retinotopy improves the categorisation and localisation of visual objects in cnns. *In preparation*.

Katayama, K., Yano, M., & Horiguchi, T. (2004, novembre). Neural network model of selective visual attention using Hodgkin-Huxley equation. *Biological Cybernetics*. doi: 10.1007/s00422-004-0504-4

Kerr, R. R., Burkitt, A. N., Thomas, D. A., Gilson, M., & Grayden, D. B. (2013, 02). Delay selection by spike-timing-dependent plasticity in recurrent networks of spiking neurons receiving oscillatory inputs. *PLOS Computational Biology*, *9*(2), 1-19. Consulté sur https://doi.org/10.1371/journal.pcbi.1002897 doi: 10.1371/journal.pcbi.1002897

Khan, N., Iqbal, K., & Martini, M. G. (2020). Lossless compression of data from static and mobile dynamic vision sensors-performance and trade-offs. *IEEE Access*, *8*, 103149-103163. doi: 10.1109/ACCESS.2020.2996661

Khan, N., Iqbal, K., & Martini, M. G. (2021). Time-aggregation-based lossless video encoding for neuromorphic vision sensor data. *IEEE Internet of Things Journal*, *8*(1), 596-609. doi: 10.1109/JIOT.2020.3007866

Kheradpisheh, S. R., & Masquelier, T. (2019). S4NN: temporal backpropagation for spiking neural networks with one spike per neuron. *CoRR*, *abs/1910.09495*. Consulté sur http://arxiv.org/abs/1910.09495

Kim, H., Leutenegger, S., & Davison, A. J. (2016). Real-time 3d reconstruction and 6-dof tracking with an event camera. *Computer Vision – ECCV 2016 Lecture Notes in Computer Science*, 349–364. doi: 10.1007/978-3-319-46466-4_21

Knight, J. C., Komissarov, A., & Nowotny, T. (2021). Pygenn: A python library for gpu-enhanced neural networks. *Frontiers in Neuroinformatics*, *15*.

Knudsen, E. I. (2018, novembre). Neural Circuits That Mediate Selective Attention: A Comparative Perspective. *Trends in Neurosciences*. doi: 10.1016/j.tins.2018.06.006

Kogler, J., Sulzbachner, C., & Kubinger, W. (2009). Bio-inspired stereo vision system with silicon retina imagers. In *International conference on virtual storytelling*. Consulté sur https://api.semanticscholar.org/CorpusID:27059477

Koickal, T. J., Hamilton, A., Tan, S. L., Covington, J. A., Gardner, J. W., & Pearce, T. C. (2007). Analog vlsi circuit implementation of an adaptive neuromorphic olfaction chip. *IEEE Transactions on Circuits and Systems I: Regular Papers*, *54*(1), 60-73. doi: 10.1109/TCSI.2006.888677

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In F. Pereira, C. Burges, L. Bottou, & K. Weinberger (Eds.), *Advances in neural information processing systems* (Vol. 25). Curran Associates, Inc. Consulté sur https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

Kubendran, R., Paul, A., & Cauwenberghs, G. (2021). A 256x256 6.3pj/pixel-event query-driven dynamic vision sensor with energy-conserving row-parallel event scanning. In *2021 ieee custom integrated circuits conference (cicc)* (p. 1-2). doi: 10.1109/CICC51472.2021.9431446

Kubke, M. F., & Carr, C. E. (2006). Morphological variation in the nucleus laminaris of birds. *International Journal of Comparative Psychology*, *19*(1). doi: 10.46867/ijcp.2006.19.01.03

Kundu, S., Datta, G., Pedram, M., & Beerel, P. A. (2021). Spike-thrift: Towards energy-efficient deep spiking neural networks by limiting spiking activity via attention-guided compression. *WACV*.

LaBerge, D., & Brown, V. (1989). Theory of attentional operations in shape identification. *Psychological Review*. Consulté le 2021-01-20, sur https://doi.apa.org/doiLanding?doi=10.1037%2F0033-295X.96.1.101

Lagorce, X., Ieng, S.-H., Clady, X., Pfeiffer, M., & Benosman, R. (2015, 02). Spatiotemporal features for asynchronous event-based data. *Front. Neurosci. - Neuromorphic Engineering*. doi: 10.3389/fnins.2015.00046

Lagorce, X., Orchard, G., Galluppi, F., Shi, B. E., & Benosman, R. B. (2016). Hots: a hierarchy of event-based time-surfaces for pattern recognition. *IEEE transactions on pattern analysis and machine intelligence*, *39*(7), 1346–1359.

Land, M. F. (2018). *Eyes to See: The Astonishing Variety of Vision in Nature*. Oxford University Press.

Lapicque, L. (1907). Recherches quantitatives sur l'excitation electrique des nerfs traitee comme une polarization. *Journal de Physiologie et de Pathologie Generalej*, *9*, 620–635.

LeBow, N., Rueckauer, B., Sun, P., Rovira, M., Jiménez-Jorquera, C., Liu, S., & Margarit-Taulé, J. (2021). Real-time edge neuromorphic tasting from chemical microsensor arrays. *Frontiers in Neuroscience*, *15*. Consulté sur https://www.frontiersin.org/articles/10.3389/fnins.2021.771480/full doi: 10.3389/fnins.2021.771480

Le Callet, P., & Niebur, E. (2013, septembre). Visual Attention and Applications in Multimedia Technologies. *Proceedings of the IEEE.*. doi: 10.1109/JPROC.2013.2265801

LeCun, Y., Bengio, Y., & Hinton, G. (2015, 05). Deep learning. *Nature*, *521*, 436-44. doi: 10.1038/nature14539

Lecun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998, 12). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*, 2278 - 2324. doi: 10.1109/5.726791

Lee, J., Delbrück, T., & Pfeiffer, M. (2016). Training deep spiking neural networks using backpropagation. *CoRR*, *abs/1608.08782*. Consulté sur http://arxiv.org/abs/1608.08782

Lele, A. S., Fang, Y., Anwar, A., & Raychowdhury, A. (2022). Bio-mimetic high-speed target localization with fused frame and event vision for edge application. *Frontiers in Neuroscience*, *16*. doi: 10.3389/fnins.2022.1010302

Leonard, R. G., & Doddington, G. (1993). Tidigits. Philadelphia: Linguistic Data Consortium. Consulté sur https://catalog.ldc.upenn.edu/LDC93S10

Li, C., Longinotti, L., Corradi, F., & Delbruck, T. (2019). A 132 by 104 10 micro m-pixel 250 micro w 1kefps dynamic vision sensor with pixel-parallel noise and spatial redundancy suppression. In *2019 symposium on vlsi circuits* (p. C216-C217). doi: 10.23919/VLSIC.2019.8778050

Li, H., Liu, H., Ji, X., Li, G., & Shi, L. (s. d.). CIFAR10-DVS: An Event-Stream Dataset for Object Classification. *Front. Neurosci.*, *11*, 309. doi: 10.3389/fnins.2017.00309

Liao, X., Vasilakos, A. V., & He, Y. (2017). Small-world human brain networks: Perspectives and challenges. *Neuroscience and Biobehavioral Reviews*, *77*, 286-300. Consulté sur https://www.sciencedirect.com/science/article/pii/S0149763416307849 doi: https://doi.org/10.1016/j.neubiorev.2017.03.018

Lichtsteiner, P., Posch, C., & Delbruck, T. (2008a). A 128x128 120 db 15 us latency asynchronous temporal contrast vision sensor. *IEEE Journal of Solid-State Circuits*. doi: 10.1109/JSSC.2007.914337

Lichtsteiner, P., Posch, C., & Delbruck, T. (2008b). A 128 x 128 120 dB 15 µs Latency Asynchronous Temporal Contrast Vision Sensor. *IEEE Journal of Solid-State Circuits*, *43*, 566-576. doi: 10.1109/JSSC.2007.914337

Liu, M., & Delbrück, T. (2018). Adaptive time-slice block-matching optical flow algorithm for dynamic vision sensors. In *British machine vision conference*. Consulté sur https://api.semanticscholar.org/CorpusID:52283776

Liu, M., & Delbruck, T. (2022). Edflow: Event driven optical flow camera with keypoint detection and adaptive block matching. *IEEE Transactions on Circuits and Systems for Video Technology*, *32*(9), 5776-5789. doi: 10.1109/TCSVT.2022.3156653

Lotfi Rezaabad, A., & Vishwanath, S. (2020). Long short-term memory spiking networks and their applications. In *International conference on neuromorphic systems 2020* (pp. 1–9).

Lungu, I.-A., Corradi, F., & Delbruck, T. (2017). Live Demonstration: Convolutional Neural Network Driven by Dynamic Vision Sensor Playing RoShamBo. In *2017 ieee symposium on circuits and systems (iscas 2017)*. Baltimore, MD, USA.

Lyon, R. F., & Mead, C. (1988). An analog electronic cochlea. *IEEE Trans. Acoust. Speech Signal Process.*, *36*, 1119-1134. Consulté sur https://api.semanticscholar.org/CorpusID:17699726

Maass, W. (1997). Networks of spiking neurons: The third generation of neural network models. *Neural Networks*, *10*(9), 1659-1671. Consulté sur https://www.sciencedirect.com/science/article/pii/S0893608097000117 doi: https://doi.org/10.1016/S0893-6080(97)00011-7

Macdonald, F. L. A., Lepora, N. F., Conradt, J., & Ward-Cherrier, B. (2022). Neuromorphic tactile edge orientation classification in an unsupervised spiking neural network. *Sensors*, *22*(18). Consulté sur https://www.mdpi.com/1424-8220/22/18/6998 doi: 10.3390/s22186998

Mahajan, D., Girshick, R., Ramanathan, V., He, K., Paluri, M., Li, Y., ... van der Maaten, L. (2018, September). Exploring the limits of weakly supervised pretraining. In *Proceedings of the european conference on computer vision (eccv)*.

Mahowald, M., & Mead, C. (1991). The Silicon Retina. *Sci. American*.

Maqueda, A. I., Loquercio, A., Gallego, G., Garcıa, N., & Scaramuzza, D. (2018). Event-based vision meets deep learning on steering prediction for self-driving cars. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 5419–5427).

Marr, D., & Poggio, T. (1976). Cooperative computation of stereo disparity. *Science*, *194*(4262), 283-287. Consulté sur https://www.science.org/doi/abs/10.1126/science.968482 doi: 10.1126/science.968482

Martinet, J., Lablack, A., Lew, S., & Djeraba, C. (2009). Gaze based quality assessment of visual media understanding. In *Ieee pacific-rim symposium on image and video technology-cvim'09*.

Masquelier, T. (2021, Apr). Back-propagation now works in spiking neural networks! *ERCIM NEWS 152 - Special theme: Brain-inspired Computing*, 11–12. Consulté sur https://ercim-news.ercim.eu/en125/special/back-propagation-now-works-in-spiking-neural-networks

Masquelier, T., & Thorpe, S. J. (2007, 02). Unsupervised learning of visual features through spike timing dependent plasticity. *PLOS Computational Biology*, *3*(2), 1-11. Consulté sur https://doi.org/10.1371/journal.pcbi.0030031 doi: 10.1371/journal.pcbi.0030031

Mathôt, S., Schreij, D., & Theeuwes, J. (2012). Opensesame: An open-source, graphical experiment builder for the social sciences. In *Behavior research methods 44(2)* (p. 314-324). doi: 10.3758/s13428-011-0168-7

Matsubara, T. (2017). Spike timing-dependent conduction delay learning model classifying spatio-temporal spike patterns. In *2017 international joint conference on neural networks (ijcnn)* (p. 1831-1839). doi: 10.1109/IJCNN.2017.7966073

Meister, M., & Berry, M. J. (1999). The neural code of the retina. *neuron*, *22*(3), 435–450.

Miao, S., Chen, G., Ning, X., Zi, Y., Ren, K., Bing, Z., & Knoll, A. C. (2019). Neuromorphic benchmark datasets for pedestrian detection, action recognition, and fall detection. *Frontiers in neurorobotics*, *13*, 38.

Mitrokhin, A., Fermuller, C., Parameshwara, C., & Aloimonos, Y. (2018). Event-based Moving Object Detection and Tracking. *arXiv preprint arXiv:1803.04523*.

Mitrokhin, A., Hua, Z., Fermüller, C., & Aloimonos, Y. (2020, June). Learning visual motion segmentation using event surfaces. In *2020 ieee/cvf conference on computer vision and pattern recognition (cvpr)* (p. 14402-14411). Consulté sur https://openaccess.thecvf.com/content_CVPR_2020/papers/Mitrokhin_Learning_Visual_Motion_Segmentation_Using_Event_Surfaces_CVPR_2020_paper.pdf doi: 10.1109/CVPR42600.2020.01442

Mitrokhin, A., Ye, C., Fermüller, C., Aloimonos, Y., & Delbruck, T. (2019, Nov). Ev-imo: Motion segmentation dataset and learning pipeline for event cameras. In *2019 ieee/rsj international conference on intelligent robots and systems (iros)* (p. 6105-6112). Consulté sur http://arxiv.org/abs/1903.07520 doi: 10.1109/IROS40897.2019.8968520

Moeys, D. P., Corradi, F., E. Kerr, P. V., Das, G., Neil, D., Kerr, D., & Delbrück, T. (2016). "Steering a predator robot using a mixed frame/event-driven convolutional neural network". In *2016 second international conference on event-based control, communication, and signal processing (ebccsp)* (p. 1-8).

Moeys, D. P., Neil, D., Corradi, F., Kerr, E., Vance, P., Das, G., & et al. (2018). PRED18: Dataset and further experiments with DAVIS event camera in predator-prey robot chasing. In *Ebccsp 2018*.

Molin, J., Thakur, C., Niebur, E., & Etienne-Cummings, R. (2021). A neuromorphic proto-object based dynamic visual saliency model with a hybrid fpga implementation. In *Tbiocas* (Vol. 15, p. 580–594).

Moore, T., & Zirnsak, M. (2017, janvier). Neural Mechanisms of Selective Visual Attention. *Annual Review of Psychology*, *68*(47-72). doi: 10.1146/annurev-psych-122414-033400

Moosmann, F., Larlus, D., & Jurie, F. (2006). Learning saliency maps for object categorization..

Moradi, S., Qiao, N., Stefanini, F., & Indiveri, G. (2018, feb). A scalable multicore architecture with heterogeneous memory structures for dynamic neuromorphic asynchronous processors (DYNAPs). *IEEE Transactions on Biomedical Circuits and Systems*, *12*(1), 106–122. Consulté sur https://doi.org/10.1109%2Ftbcas.2017.2759700 doi: 10.1109/tbcas.2017.2759700

Mostafa, H. (2016). Supervised learning based on temporal coding in spiking neural networks. *CoRR*, *abs/1606.08165*. Consulté sur http://arxiv.org/abs/1606.08165

Mueggler, E., Gallego, G., Rebecq, H., & Scaramuzza, D. (2018). Continuous-time visual-inertial odometry for event cameras. *IEEE Transactions on Robotics*, *34*(6), 1425-1440. doi: 10.1109/TRO.2018.2858287

Nadafian, A., & Ganjtabesh, M. (2020). Bio-plausible unsupervised delay learning for extracting temporal features in spiking neural networks. In *arxiv cs.ne*.

Nageswaran, J. M., Dutt, N., Krichmar, J. L., Nicolau, A., & Veidenbaum, A. V. (2009). A configurable simulation environment for the efficient simulation of large-scale spiking neural networks on graphics processors. *Neural Networks*, *22*(5), 791-800. Consulté sur https://www.sciencedirect.com/science/article/pii/S0893608009001373 (Advances in Neural Networks Research: IJCNN2009) doi: https://doi.org/10.1016/j.neunet.2009.06.028

Neftci, E., Mostafa, H., & Zenke, F. (2019, 11). Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks. *IEEE Signal Processing Magazine*, *36*, 51-63. doi: 10.1109/MSP.2019.2931595

Orchard, G., Benosman, R., Etienne-Cummings, R., & Thakor, N. V. (2013). A spiking neural network architecture for visual motion estimation. *2013 IEEE Biomedical Circuits and Systems Conference (BioCAS)*. doi: 10.1109/biocas.2013.6679698

Orchard, G., Jayawant, A., Cohen, G. K., & Thakor, N. (2015). Converting static image datasets to spiking neuromorphic datasets using saccades. *Frontiers in Neuroscience*, *9*, 437. Consulté sur https://www.frontiersin.org/article/10.3389/fnins.2015.00437 doi: 10.3389/fnins.2015.00437

Osswald, M., Ieng, S.-H., Benosman, R., & Indiveri, G. (2017, 01). A spiking neural network model of 3d perception for event-based neuromorphic stereo vision systems. *Scientific Reports*, *7*, 40703. doi: 10.1038/srep40703

Oudjail, V., & Martinet, J. (2019, février). Bio-inspired event-based motion analysis with spiking neural networks. In *Visapp*. Consulté sur https://hal.archives-ouvertes.fr/hal-01940917

Padala, V., Basu, A., & Orchard, G. (2018, 03). A noise filtering algorithm for event-based asynchronous change detection image sensors on truenorth and its implementation on truenorth. *Frontiers in Neuroscience*, *12*, 118. doi: 10.3389/fnins.2018.00118

Panzeri, S., Janotte, E., Pequeño-Zurro, A., Bonato, J., & Bartolozzi, C. (2023, 01). Constraints on the design of neuromorphic circuits set by the properties of neural population codes. *Neuromorphic Computing and Engineering*, *3*. doi: 10.1088/2634-4386/acaf9c

Parameshwara, C. M., Li, S., Fermüller, C., Sanket, N. J., Evanusa, M. S., & Aloimonos, Y. (2021). SpikeMS: Deep spiking neural network for motion segmentation. In *2021 ieee/rsj international conference on intelligent robots and systems (iros)* (p. 3414-3420). doi: 10.1109/IROS51168.2021.9636506

Paredes-Vallés, F., Hagenaars, J. J., & de Croon, G. (2021). Self-supervised learning of event-based optical flow with spiking neural networks. *CoRR*, *abs/2106.01862*. Consulté sur https://arxiv.org/abs/2106.01862

Paredes-Valles, F., Scheper, K. Y. W., & De Croon, G. C. H. E. (2020). Unsupervised learning of a hierarchical spiking neural network for optical flow estimation: From events to global motion perception. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *42*(8), 2051-2064.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., ... Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems*. Curran Associates, Inc. Consulté sur http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

Patterson, D. A., Gonzalez, J., Le, Q. V., Liang, C., Munguia, L., Rothchild, D., ... Dean, J. (2021). Carbon emissions and large neural network training. *CoRR*, *abs/2104.10350*. Consulté sur https://arxiv.org/abs/2104.10350

Paugam-Moisy, H., & Bohte, S. M. (2012, septembre). Computing with Spiking Neuron Networks. In G. Rozenberg, T. Back, & J. Kok (Eds.), *Handbook of Natural Computing* (p. 335-376). Springer-Verlag. Consulté sur https://hal.archives-ouvertes.fr/hal-01587781 doi: 10.1007/978-3-540-92910-9_10

Paugam-Moisy, H., Martinez, R., & Bengio, S. (2008). Delay learning and polychronization for reservoir computing. *Neurocomp.*, *71*.

Pehle, C., & Pedersen, J. E. (2021, janvier). *Norse - A deep learning library for spiking neural networks*. Zenodo. Consulté sur https://doi.org/10.5281/zenodo.4422025 (Documentation: https://norse.ai/docs/) doi: 10.5281/zenodo.4422025

Peng, J., Tian, L., Jia, X., Guo, H., Xu, Y., Xie, D., ... Wang, Y. (2019). Multi-task adas system on fpga. In *Aicas*. doi: 10.1109/AICAS.2019.8771615

Perrinet, L. U. (2010). Role of homeostasis in learning sparse representations. *Neural Computation*, *22*(7), 1812–36. Consulté sur https://doi.org/10.1162/neco.2010.05-08-795 doi: 10.1162/neco.2010.05-08-795

Perrinet, L. U., & Samuelides, M. (2002). Coherence detection in a spiking neuron via hebbian learning. *Neurocomputing*, *44–46*(C), 817–22. Consulté sur http://dx.doi.org/10.1016/S0925-2312(02)00374-0 doi: 10.1016/S0925-2312(02)00374-0

Perrinet, L. U., Samuelides, M., & Thorpe, S. J. (2004). Sparse spike coding in an asynchronous feed-forward multi-layer neural network using matching pursuit. *Neurocomputing*, *57*, 125–134. Consulté sur http://dx.doi.org/10.1016/j.neucom.2004.01.010 (Special issue: New Aspects in Neurocomputing: 10th European Symposium on Artificial Neural Networks 2002 - Edited by T. Villmann) doi: 10.1016/j.neucom.2004.01.010

Pfeiffer, M., & Pfeil, T. (2018). Deep learning with spiking neurons: Opportunities and challenges. *Frontiers in Neuroscience*, *12*. Consulté sur https://www.frontiersin.org/articles/10.3389/fnins.2018.00774 doi: 10.3389/fnins.2018.00774

Ponulak, F., & Kasiński, A. (2009, 10). Supervised learning in spiking neural networks with resume: Sequence learning, classification, and spike shifting. *Neural computation*, *22*, 467-510. doi: 10.1162/neco.2009.11-08-901

Pramod, R., Katti, H., & Arun, S. (2022). Human peripheral blur is optimal for object recognition. *Vision Research*, *200*, 108083. Consulté sur https://www.sciencedirect.com/science/article/pii/S004269892200089X doi: https://doi.org/10.1016/j.visres.2022.108083

Pérez-Carrasco, J., Zhao, B., Serrano, C., Acha, B., Serrano-Gotarredona, T., Chen, S., & Linares-Barranco, B. (2013, 11). Mapping from frame-driven to frame-free event-driven vision systems by low-rate rate-coding and coincidence processing. application to feed forward convnets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *35*, 2706 - 2719. doi: 10.1109/TPAMI.2013.71

Qi, Y., Zhang, B., Taha, T. M., Chen, H., & Hasan, R. (2014). Fpga design of a multicore neuromorphic processing system. In *Naecon 2014 - ieee national aerospace and electronics conference* (p. 255-258). doi: 10.1109/NAECON.2014.7045812

Qiao, N., Mostafa, H., Corradi, F., Osswald, M., Stefanini, F., Sumislawska, D., & Indiveri, G. (2015, 04). A re-configurable on-line learning spiking neuromorphic processor comprising 256 neurons and 128k synapses. *Frontiers in Neuroscience*, *9*. doi: 10.3389/fnins.2015.00141

Rabbath, C., & Corriveau, D. (2019). A comparison of piecewise cubic hermite interpolating polynomials, cubic splines and piecewise linear functions for the approximation of projectile aerodynamics. *31st International Symposium on Ballistics*. doi: 10.12783/ballistics2019/33099

Raftopoulos, A. (2009). *Cognition and Perception: How Do Psychology and Neural Science Inform Philosophy ?* MIT Press. (Google-Books-ID: QykLHyWwLuQC)

Rançon, U., Cuadrado-Anibarro, J., Cottereau, B. R., & Masquelier, T. (2021). Stereospike: Depth learning with a spiking neural network. *CoRR*, *abs/2109.13751*. Consulté sur https://arxiv.org/abs/2109.13751

Rasamuel, M., Khacef, L., Rodriguez, L., & Miramond, B. (2019). Specialized visual sensor coupled to a dynamic neural field for embedded attentional process. *IEEE SAS*. doi: 10.1109/sas.2019.8705979

Rathi, N., & Roy, K. (2021). Diet-snn: A low-latency spiking neural network with direct input encoding and leakage and threshold optimization. *IEEE Transactions on Neural Networks and Learning Systems*. doi: 10.1109/tnnls.2021.3111897

Rebecq, H., Gallego, G., Mueggler, E., & Scaramuzza, D. (2018, 12). Emvs: Event-based multi-view stereo—3d reconstruction with an event camera in real-time. *International Journal of Computer Vision*, *126*. doi: 10.1007/s11263-017-1050-6

Rebecq, H., Horstschaefer, T., & Scaramuzza, D. (2017, 09). Real-time visual-inertial odometry for event cameras using keyframe-based nonlinear optimization.. doi: 10.5244/C.31.16

Rebecq, H., Ranftl, R., Koltun, V., & Scaramuzza, D. (2019a). Events-to-video: Bringing modern computer vision to event cameras. *IEEE Conference of Computer Vision and Pattern Recognition (CVPR)*. Consulté sur http://rpg.ifi.uzh.ch/docs/TPAMI19_Rebecq.pdf

Rebecq, H., Ranftl, R., Koltun, V., & Scaramuzza, D. (2019b). High speed and high dynamic range video with an event camera. *IEEE Trans. Pattern Anal. Mach. Intell. (T-PAMI)*. Consulté sur http://rpg.ifi.uzh.ch/docs/TPAMI19_Rebecq.pdf

Reichardt, W., & Rosenblith, W. A. (1961). Autocorrelation, a principle for evaluation of sensory information by the central nervous system. In *Sensory communication.*

Reinbacher, C., Munda, G., & Pock, T. (2017). Real-time panoramic tracking for event cameras. In *2017 ieee international conference on computational photography (iccp)* (p. 1-9). doi: 10.1109/ICCPHOT.2017.7951488

Renner, A., Evanusa, M., & Sandamirskaya, Y. (2019). Event-based attention and tracking on neuromorphic hardware. *IEEE CVPRW*. doi: 10.1109/cvprw.2019.00220

Risi, N., Aimar, A., Donati, E., Solinas, S., & Indiveri, G. (2020). A spike-based neuromorphic architecture of stereo vision. *Frontiers in Neurorobotics*, *14*. Consulté sur https://www.frontiersin.org/articles/10.3389/fnbot.2020.568283/abstract doi: 10.3389/fnbot.2020.568283

Rizzo, C., Schuman, C. D., & Plank, J. S. (2023). Neuromorphic downsampling of event-based camera output. In D. Kudithipudi, C. Frenkel, S. Cardwell, & J. B. Aimone (Eds.), *Neuro-inspired computational elements conference, nice2023, san antonio, tx, usa, april 11-14, 2023* (pp. 26–34). ACM. Consulté sur https://doi.org/10.1145/3584954.3584962 doi: 10.1145/3584954.3584962

Rolls, E., & Milward, T. (2000, 12). A model of invariant object recognition in the visual system: Learning rules, activation functions, lateral inhibition, and information-based performance measures. *Neural computation*, *12*, 2547-72. doi: 10.1162/089976600300014845

Romain, B., & Dan, G. (2008, 07). Brian: a simulator for spiking neural networks in python. *BMC Neuroscience*, *9*. doi: 10.1186/1471-2202-9-S1-P92

Roy, K., Jaiswal, A., & Panda, P. (2019). Towards spike-based machine intelligence with neuromorphic computing. *Nature*, *575*(7784), 607-617. Consulté sur https://EconPapers.repec.org/RePEc:nat:nature:v:575:y:2019:i:7784:d:10.1038_s41586-019-1677-2

Rueckauer, B., Hu, Y., Lungu, I., Pfeiffer, M., & Liu, S.-C. (2017). Conversion of continuous-valued deep networks to efficient event-driven networks for image classification. *Frontiers of Neuroscience*.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., ... Fei-Fei, L. (2014). Imagenet large scale visual recognition challenge. *CoRR*, *abs/1409.0575*. Consulté sur http://arxiv.org/abs/1409.0575

Russell, A., Mihalas, S., von der Heydt, R., Niebur, E., & Etienne-Cummings, R. (2013). A model of proto-object based saliency. *Vision research*, *94*. doi: 10.1016/j.visres.2013.10.005

Rückauer, B., & Delbruck, T. (2016). Evaluation of event-based algorithms for optical flow with ground-truth from inertial measurement sensor. In *Front. neurosci.* (Vol. 10, p. 176).

Sandamirskaya, Y. (2014). Dynamic neural fields as a step toward cognitive neuromorphic architectures. *Fr. in Neurosciences*, *7*. doi: 10.3389/fnins.2013.00276

Sarvaiya, J. N., Patnaik, S., & Bombaywala, S. (2009). Image registration using log-polar transform and phase correlation. In *IEEE Region 10 Annual International Conference, Proceedings/-TENCON* (pp. 1–5). doi: 10.1109/TENCON.2009.5396234

Scheerlinck, C., Rebecq, H., Gehrig, D., Barnes, N., Mahony, R., & Scaramuzza, D. (2020). Fast image reconstruction with an event camera. In *Proceedings of the ieee/cvf winter conference on applications of computer vision* (pp. 156–163).

Scheerlinck, C., Rebecq, H., Stoffregen, T., Barnes, N., Mahony, R., & Scaramuzza, D. (2019). Ced: Color event camera dataset. In *Ieee conference on computer vision and pattern recognition workshops (cvprw)*.

Schemmel, J., Brüderle, D., Grübl, A., Hock, M., Meier, K., & Millner, S. (2010). A wafer-scale neuromorphic hardware system for large-scale neural modeling. In *2010 ieee international symposium on circuits and systems (iscas)* (p. 1947-1950). doi: 10.1109/ISCAS.2010.5536970

Sekikawa, Y., Hara, K., & Saito, H. (2019, 12). Eventnet: Asynchronous recursive event processing. In *2019 ieee/cvf conference on computer vision and pattern recognition (cvpr)*.

Sengupta, A., Ye, Y., Wang, R., Liu, C., & Roy, K. (2018). Going deeper in spiking neural networks: VGG and residual architectures. *CoRR*, *abs/1802.02627*. Consulté sur http://arxiv.org/abs/1802.02627

Serrano-Gotarredona, T., Faramarzi, F., & Linares-Barranco, B. (2022). Electronically foveated dynamic vision sensor. In *2022 ieee international conference on omni-layer intelligent systems (coins)* (p. 1-5). doi: 10.1109/COINS54846.2022.9855009

Serrano-Gotarredona, T., & Linares-Barranco, B. (2013). A 128×128 1.5% contrast sensitivity 0.9% fpn 3 µs latency 4 mw asynchronous frame-free dynamic vision sensor using transimpedance preamplifiers. *IEEE Journal of Solid-State Circuits*, *48*(3), 827-838. doi: 10.1109/JSSC.2012.2230553

Shi, M., Zhang, T., & Zeng, Y. (2020). A Curiosity-Based Learning Method for Spiking Neural Networks. *Front. in Comp. Neuroscience*. doi: 10.3389/fncom.2020.00007

Shrestha, S. B., & Orchard, G. (2018). SLAYER: Spike layer error reassignment in time. In *Advances in neural information processing systems*. Curran Associates, Inc. Consulté sur http://papers.nips.cc/paper/7415-slayer-spike-layer-error-reassignment-in-time.pdf

Singla, D., Chatterjee, S., Ramapantulu, L., Ussa, A., Ramesh, B., & Basu, A. (2020). Hynna: Improved performance for neuromorphic vision sensor based surveillance using hybrid neural network architecture. In *2020 ieee international symposium on circuits and systems (iscas)* (pp. 1–5).

Sironi, A., Brambilla, M., Bourdis, N., Lagorce, X., & Benosman, R. (2018). HATS: Histograms of Averaged Time Surfaces for Robust Event-based Object Classification. In *Ieee conference on computer vision and pattern recognition (cvpr)*.

Skorka, O., & Joseph, D. (2011, 07). Toward a digital camera to rival the human eye. *Journal of Electronic Imaging - J ELECTRON IMAGING*, *20*. doi: 10.1117/1.3611015

Soltic, S., Wysoski, S. G., & Kasabov, N. K. (2008). Evolving spiking neural networks for taste recognition. In *2008 ieee international joint conference on neural networks (ieee world congress on computational intelligence)* (p. 2091-2097). doi: 10.1109/IJCNN.2008.4634085

Song, S., Zhu, Y., Hou, J., Zheng, Y., Huang, T., & Du, S. (2019). Improved convolutional neutral network based model for small visual object detection in autonomous driving. In *2019 ieee international conference on artificial intelligence circuits and systems (aicas)* (p. 179-183). doi: 10.1109/AICAS.2019.8771542

Sorbaro, M., Liu, Q., Bortone, M., & Sheik, S. (2019). Optimizing the energy consumption of spiking neural networks for neuromorphic applications. *CoRR*, *abs/1912.01268*. Consulté sur http://arxiv.org/abs/1912.01268

Srivastava, M., Hashimoto, T. B., & Liang, P. (2020). Robustness to spurious correlations via human annotations. *CoRR*, *abs/2007.06661*. Consulté sur https://arxiv.org/abs/2007.06661

Steffen, L., Reichard, D., Weinland, J., Kaiser, J., Roennau, A., & Dillmann, R. (2019). Neuromorphic Stereo Vision: A Survey of Bio-Inspired Sensors and Algorithms. *Frontiers in Neurorobotics*, *13*.

Stoffregen, T., Scheerlinck, C., Scaramuzza, D., Drummond, T., Barnes, N., Kleeman, L., & Mahony, R. E. (2020). How to train your event camera neural network. *CoRR*, *abs/2003.09078*. Consulté sur https://arxiv.org/abs/2003.09078

Stromatias, E., Galluppi, F., Patterson, C., & Furber, S. B. (2013). Power analysis of large-scale, real-time neural networks on spinnaker. *The 2013 International Joint Conference on Neural Networks (IJCNN)*, 1-8. Consulté sur https://api.semanticscholar.org/CorpusID:3690445

Suh, Y., Choi, S., Ito, M., Kim, J., Lee, Y., Seo, J., ... Park, Y. (2020). A 1280×960 dynamic vision sensor with a 4.95-micro m pixel pitch and motion artifact minimization. In *2020 ieee international symposium on circuits and systems (iscas)* (p. 1-5). doi: 10.1109/ISCAS45731.2020.9180436

Tan, G., Wang, Y., Han, H., Cao, Y., Wu, F., & Zha, Z.-J. (2022). Multi-grained spatio-temporal features perceived network for event-based lip-reading. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition* (pp. 20094–20103).

Tavanaei, A., Ghodrati, M., Kheradpisheh, S. R., Masquelier, T., & Maida, A. S. (2018). Deep learning in spiking neural networks. *CoRR*, *abs/1804.08150*. Consulté sur http://arxiv.org/abs/1804.08150

Tayarani-Najaran, M.-H., & Schmuker, M. (2021, Jan). Event-based sensing and signal processing in the visual, auditory, and olfactory domain: A review. *Frontiers in Neural Circuits*, *0*. doi: 10.3389/fncir.2021.610446

Thakur, C. S., Molin, J. L., Xiong, T., Zhang, J., Niebur, E., & Etienne-Cummings, R. (2017). Neuromorphic visual saliency implementation using stochastic computation. In *2017 ieee international symposium on circuits and systems (iscas)* (p. 1-4). doi: 10.1109/ISCAS.2017.8050868

Thorpe, S., Delorme, A., & Van Rullen, R. (2001, Sep). Spike-based strategies for rapid processing. *Neural Networks: The Official Journal of the International Neural Network Society*, *14*(6–7), 715–725. doi: 10.1016/s0893-6080(01)00083-1

Thorpe, S. J., & Fabre-Thorpe, M. (2001). Speed of processing in the human visual system. *Nature*, *381*(6582), 520–522. doi: 10.1038/381520a0

Traver, V. J., & Pla, F. (2003). Designing the Lattice for Log-Polar Images. *Discrete Geometry for Computer Imagery*, 164–173. Consulté sur http://link.springer.com/10.1007/978-3-540-39966-7_15 doi: 10.1007/978-3-540-39966-7_15

Treisman, A. M., & Gelade, G. (1980). A Feature-Integration Theory of Attention. *Cognitive Psychology*(12), 97–136.

Tsotsos, J. K. (1990). Analyzing vision at the complexity level. *Behavioral and Brain Sciences*, *13*(3). doi: 10.1017/S0140525X00079577

Vasudevan, A., Negri, P., Ielsi, C., Linares-Barranco, B., & Serrano-Gotarredona, T. (2022, 08). Sl-animals-dvs: event-driven sign language animals dataset. In (Vol. 25, p. 1-16). doi: 10.1007/s10044-021-01011-w

Vasudevan, A., Negri, P., Linares-Barranco, B., & Serrano-Gotarredona, T. (2020). Introduction and Analysis of an Event-Based Sign Language Dataset. In *15th ieee international conference on automatic face and gesture recognition (fg 2020) (fg)* (p. 441-448.). (doi: 10.1109/FG47880.2020.00069)

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, *30*. Consulté sur https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

Vidal, A. R., Rebecq, H., Horstschaefer, T., & Scaramuzza, D. (2018). Ultimate slam ? combining events, images, and imu for robust visual slam in hdr and high-speed scenarios. *IEEE Robotics and Automation Letters*, *3*(2), 994–1001. doi: 10.1109/lra.2018.2793357

Wang, J. (2022, 08). A review of spiking neural networks. *SHS Web of Conferences*, *144*, 03004. doi: 10.1051/shsconf/202214403004

Wang, J., Belatreche, A., Maguire, L., & McGinnity, M. (2010). Online versus offline learning for spiking neural networks: A review and new strategies. In *2010 ieee 9th international conference on cyberntic intelligent systems* (p. 1-6). doi: 10.1109/UKRICIS.2010.5898113

Wang, L., Mostafavi, I. M., Ho, Y.-S., & Yoon, K.-J. (2019). Event-based high dynamic range image and very high frame rate video generation using conditional generative adversarial networks. In *2019 ieee/cvf conference on computer vision and pattern recognition (cvpr)* (p. 10073-10082). doi: 10.1109/CVPR.2019.01032

Wang, S., Cheng, T. H., & Lim, M. (2023). Membrane potential distribution adjustment and parametric surrogate gradient in spiking neural networks. *arXiv.org*. doi: 10.48550/arxiv.2304.13289

Weerasinghe, M. M. A. (2022). *Neuromorphic spiking neural network algorithms for machine learning and pattern recognition* (PhD thesis). School of Engineering, Computer and Mathematical Sciences, Auckland University of Technology, Auckland, New Zealand. (Available at https://openrepository.aut.ac.nz/server/api/core/bitstreams/e5fc22c3-ee22-412c-81da-b1d4d58d1b0f/content)

Weikersdorfer, D., Adrian, D. B., Cremers, D., & Conradt, J. (2014). Event-based 3D SLAM with a depth-augmented dynamic vision sensor. In *International conference on robotics and automation*. Hong-Kong.

Weikersdorfer, D., & Conradt, J. (2012, 12). Event-based particle filtering for robot self-localization. In (p. 866-870). doi: 10.1109/ROBIO.2012.6491077

Wright, P. W., & Wiles, J. (2012). Learning transmission delays in spiking neural networks: A novel approach to sequence learning based on spike delay variance. In *The 2012 international joint conference on neural networks (ijcnn)* (p. 1-8). doi: 10.1109/IJCNN.2012.6252371

Wu, Q., McGinnity, T., Maguire, L., Cai, R., & Chen, M. (2013, septembre). A visual attention model based on hierarchical spiking neural networks. *Neurocomputing*. doi: 10.1016/j.neucom .2012.01.046

Wysoski, S., Benuskova, L., & Kasabov, N. (2006, 09). On-line learning with structural adaptation in a network of spiking neurons for visual pattern recognition. In (p. 61-70). doi: 10.1007/11840817_7

Yamazaki, K., Vo-Ho, V.-K., Bulsara, D., & Le, N. (2022, 06). Spiking neural networks and their applications: A review. *Brain sciences*, *12*. doi: 10.3390/brainsci12070863

Yang, L., Wang, Z., Zhang, S., Li, Y., Jiang, C., Sun, L., & Xu, W. (2023). Neuromorphic gustatory system with salt-taste perception, information processing, and excessive-intake warning capabilities. *Nano Letters*, *23*(1), 8-16. Consulté sur https://doi.org/10.1021/acs.nanolett.2c02775 (PMID: 36542842) doi: 10.1021/acs.nanolett.2c02775

Yao, M., Gao, H., Zhao, G., Wang, D., Lin, Y., Yang, Z., & Li, G. (2021). Temporal-wise attention spiking neural networks for event streams classification. In *Proceedings of the ieee/cvf international conference on computer vision* (pp. 10221–10230).

Yavuz, E., Turner, J., & Nowotny, T. (2016, 01). Genn: A code generation framework for accelerated brain simulations. *Scientific Reports*, *6*, 18854. doi: 10.1038/srep18854

Yousefzadeh, A., Serrano-Gotarredona, T., & Linares-Barranco, B. (2015). Fast pipeline 128×128 pixel spiking convolution core for event-driven vision processing in fpgas. In *2015 international conference on event-based control, communication, and signal processing (ebccsp)* (p. 1-8). doi: 10.1109/EBCCSP.2015.7300698

Zenke, F., & Gerstner, W. (2014). Limits to high-speed simulations of spiking neural networks using general-purpose computers. *Front Neuroinform*, *8*. Consulté sur http://journal.frontiersin.org/Journal/10.3389/fninf.2014.00076/abstract doi: 10 .3389/fninf.2014.00076

Zhang, J., Dong, B., Zhang, H., Ding, J., Heide, F., Yin, B., & Yang, X. (2022). Spiking transformers for event-based single object tracking. *Computer Vision and Pattern Recognition*. doi: 10.1109/cvpr52688.2022.00860

Zhang, L., Zhang, H., Zhu, C., Guo, S., Chen, J., & Wang, L. (2021). Fine-grained video deblurring with event camera. In J. Lokoč et al. (Eds.), *Multimedia modeling* (pp. 352–364). Cham : Springer International Publishing.

Zhang, W., & Linden, D. J. (2003). The other side of the engram: experience-driven changes in neuronal intrinsic excitability. *Nature Reviews Neuroscience*, *4*, 885-900.

Zhang, Z., Yezzi, A., & Gallego, G. (2021). Image reconstruction from events. why learn it ? *arXiv:2112.06242*.

Zhou, Z., Zhu, Y., He, C., Wang, Y., YAN, S., Tian, Y., & Yuan, L. (2023). Spikformer: When spiking neural network meets transformer. In *The eleventh international conference on learning representations*. Consulté sur https://openreview.net/forum?id=frE4fUwz_h

Zhu, A., Yuan, L., Chaney, K., & Daniilidis, K. (2019, 12). Unsupervised event-based learning of optical flow, depth, and egomotion. In *2019 ieee/cvf conference on computer vision and pattern recognition (cvpr)*.

Zhu, A. Z., Thakur, D., Ozaslan, T., Pfrommer, B., Kumar, V., & Daniilidis, K. (2018). The Multi Vehicle Stereo Event Camera Dataset: An Event Camera Dataset for 3D Perception. *IEEE Robotics and Automation Letters*, *3*, 2032-2039.

Zhu, A. Z., Yuan, L., Chaney, K., & Daniilidis, K. (2018). Ev-flownet: Self-supervised optical flow estimation for event-based cameras. *CoRR*, *abs/1802.06898*. Consulté sur http://arxiv.org/abs/1802.06898

Zhu, L., Wang, X., Chang, Y., Li, J., Huang, T., & Tian, Y. (2022). Event-based video reconstruction via potential-assisted spiking neural network. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition* (pp. 3594–3604).

Ziv, J., & Lempel, A. (1977). A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*, *23*(3), 337-343. doi: 10.1109/TIT.1977.1055714

Web pages

APROVIS3D. (2023). *Aprovis3d.* Consulté le 2021-08-18, sur http://aprovis3d.eu

BasBuller. (2019). *Pysnn: Efficient spiking neural network framework, built on top of pytorch for gpu acceleration.* Consulté sur https://github.com/BasBuller/PySNN/

Bordage, F. (2019, Sep). *Empreinte environnementale du numérique mondial.* Consulté sur https://www.greenit.fr/wp-content/uploads/2019/11/GREENIT_EENM_etude_EN_accessible.pdf

Byrne, J. H. (2020, May). *Chapter 3: Propagation of the action potentia.* The University of Texas Medical School at Houston. Consulté sur https://nba.uth.tmc.edu/neuroscience/m/s1/chapter03.html

HEAVY.AI. (2022). *What is an embedded system ? definition and faqs.* Consulté sur https://www.heavy.ai/technical-glossary/embedded-systems

i3S. (2023a). *Laboratoire d'Informatique, Signaux et Systèmes de Sophia Antipolis.* Consulté le 2021-08-18, sur https://www.i3s.unice.fr/

i3S. (2023b). *Sparks team.* Consulté le 2021-08-18, sur http://sparks.i3s.unice.fr

INT. (2023). *Institut de Neurosciences de la Timone.* Consulté le 2021-08-18, sur http://www.int.univ-amu.fr

L, B. (2021, Jan). *Neuromorphic computing : Tout savoir sur les ordinateurs à l'intelligence humaine.* Consulté sur https://www.lebigdata.fr/neuromorphic-computing-tout-savoir

Lenz, G. (2019). *Datasets — 1.2.6.* Consulté sur https://tonic.readthedocs.io/en/latest/datasets.html

Lenz, G., Chaney, K., Shrestha, S. B., Oubari, O., Picaud, S., & Zarrella, G. (2021, jul). *Tonic: event-based datasets and transformations.* Zenodo. Consulté sur https://doi.org/10.5281/zenodo.5079802 (Documentation available under https://tonic.readthedocs.io) doi: 10.5281/zenodo.5079802

Linux Fundation, T. (2017). *Datasets — torchvision main documentation.* Consulté sur https://pytorch.org/vision/stable/datasets.html

Manchester, S. (2014, June). *Simple data input and output with spinnaker - lab manual - github pages.* Consulté sur http://spinnakermanchester.github.io/spynnaker/5.0.0/SimpleIO-LabManual.pdf

Pehle, C., & Pedersen, J. E. (2021, janvier). *Norse - A deep learning library for spiking neural networks.* Zenodo. Consulté sur https://doi.org/10.5281/zenodo.4422025 (Documentation: https://norse.ai/docs/) doi: 10.5281/zenodo.4422025

Perrinet, L. (2023, Feb). *Implementing a retinotopic transform using 'grid_sample' from pytorch.* Consulté sur https://laurentperrinet.github.io/sciblog/posts/2023-02-02-implementing-a-retinotopic-transform-using-grid_sample-from-pytorch.html

Prophesee. (2019, Jul). *What is event-based vision ?* Consulté sur https://www.prophesee.ai/2019/07/28/event-based-vision-2/

qiaokaki. (2017, Jun). *Github - qiaokaki/aedattools: Tools for manipulating .aedat files (timestamped address-event data from neuromorphic hardware), in matlab and python.* Consulté sur https://github.com/qiaokaki/AedatTools

Robotics, & Perception Group, u.-r. (2020). *Event-based vision resources.* Consulté sur https://github.com/uzh-rpg/event-based_vision_resources

Rodríguez, M. S., Serrano-Gotarredona, T., & Linares-Barranco, B. (2017). SLOW-POKER-DVS Database. Sevilla, Spain. Consulté sur http://www2.imse-cnm.csic.es/caviar/SLOWPOKERDVS.html

Serrano-Gotarredona, T., & Linares-Barranco, B. (2015). POKER-DVS Database. Sevilla, Spain. Consulté sur http://www2.imse-cnm.csic.es/caviar/POKERDVS.html

SynSense. (2019a). *Synsense / rockpool · gitlab.* Consulté sur https://gitlab.com/synsense/rockpool

SynSense. (2019b). *Synsense / sinabs · gitlab.* Consulté sur https://gitlab.com/synsense/sinabs

University of Manchester, S. (2021, Sep). sPyNNaker models, limitations and extensions. In *Software for spinnaker.*

Vasković, J. (2023, Apr). *Neurotransmitters.* Kenhub. Consulté sur https://www.kenhub.com/en/library/anatomy/neurotransmitters

Yuhuang, H. (2016). *AEDAT Recordings for DVSACT16 (Version 20160610) [Data set].* doi: https://doi.org/10.5281/zenodo.4809809

My publications

Bulzomi, H., Gruel, A., Nakano, Y., Fujita, T., & Martinet, J. (2023). Object detection for embedded systems using tiny spiking neural networks: Filtering noise through visual attention. In *18th international conference on machine vision applications (mva).*

Bulzomi, H., Schweiker, M., Gruel, A., & Martinet, J. (2023, juin). End-to-end neuromorphic lip-reading. In *Cvpr 2023 workshop on event-based vision.* Vancouver, Canada. (This work

was supported by the European Union's ERA-NET CHIST-ERA 2018 research and innovation programme under grant agreement ANR-19-CHR3-0008. The authors are grateful to the OPAL infrastructure from Université Côte d'Azur for providing resources and support.)

Grimaldi, A., Gruel, A., Besnainou, C., Jérémie, J.-N., Martinet, J., & Perrinet, L. U. (2022, décembre). Precise Spiking Motifs in Neurobiological and Neuromorphic Data. *Brain Sciences*, *13*(1), 68. Consulté sur https://hal-amu.archives-ouvertes.fr/hal-03918338 doi: 10.3390/brainsci13010068

Gruel, A., di Mauro, A., Hunziker, R., Benini, L., Martinet, J., & Magno, M. (2023). Embedded neuromorphic attention model leveraging a novel low-power heterogeneous platform. In *Ieee international conference on artificial intelligence circuits and systems (aicas).*

Gruel, A., Hareb, D., Grimaldi, A., Martinet, J., Perrinet, L., Linares-Barranco, B., & Serrano-Gotarredona, T. (2022). Stakes of neuromorphic foveation: a promising future for embedded event cameras. *Resarch Square*. (Unpublished, under review at *Biological Cybernetics*)

Gruel, A., Hareb, D., Martinet, J., Linares-Barranco, B., & Serrano-Gotarredona, T. (2022, juin). Neuromorphic foveation applied to semantic segmentation. In *NeuroVision: What can computer vision learn from visual neuroscience ? A CVPR 2022 Workshop.* New Orleans, United States. Consulté sur https://hal.science/hal-03760724 (This work was supported by the European Union's ERA-NET CHIST-ERA 2018 research and innovation programme under grant agreement ANR-19-CHR3-0008.The authors are grateful to the OPAL infrastructure from Université Côte d'Azur for providing resources and support.)

Gruel, A., & Martinet, J. (2021). Bio-inspired visual attention for silicon retinas based on spiking neural networks applied to pattern classification. *CBMI*.

Gruel, A., & Martinet, J. (2022, juillet). Axonal Delay Learning: from biology to computational neuroscience. In *Journées Ouvertes en Biologie, Informatique et Mathématiques (JOBIM) 2022*. Rennes, France. Consulté sur https://hal.science/hal-03760748 (This work was supported by the European Union's ERA-NET CHIST-ERA 2018 research and innovation programme under grant agreement ANR-19-CHR3-0008.)

Gruel, A., Martinet, J., Linares-Barranco, B., & Serrano-Gotarredona, T. (2021, septembre). Stakes of foveation on event cameras. In *ORASIS 2021*. Saint Ferréol, France. Consulté sur https://hal.archives-ouvertes.fr/hal-03339647

Gruel, A., Martinet, J., Linares-Barranco, B., & Serrano-Gotarredona, T. (2023). Performance comparison of dvs data spatial downscaling methods using spiking neural networks. In *2023 ieee/cvf winter conference on applications of computer vision (wacv)* (p. 6483-6491). doi: 10.1109/WACV56688.2023.00643

Gruel, A., Martinet, J., & Magno, M. (2023a). Simultaneous neuromorphic selection of multiple salient objects for event vision. *2023 International Joint Conference on Neural Networks (IJCNN)*.

Gruel, A., Martinet, J., & Magno, M. (2023b). Simultaneous neuromorphic selection of multiple salient objects for event vision. In *Orasis 2023*.

Gruel, A., Martinet, J., Serrano-Gotarredona, T., & Linares-Barranco, B. (2022). Event data downscaling for embedded computer vision. In *Visapp*.

Gruel, A., Trillo Carreras, L., Bueno Garcia, M., Kupczyk, E., & Martinet, J. (2023, juin). Frugal event data: how small is too small ? a human performance assessment with shrinking data. In

Cvpr 2023 workshop on event-based vision. Vancouver, Canada. (This work was supported by the European Union's ERA-NET CHIST-ERA 2018 research and innovation programme under grant agreement ANR-19-CHR3-0008. The authors are grateful to the OPAL infrastructure from Université Côte d'Azur for providing resources and support.)

Gruel, A., Vitale, A., Martinet, J., & Magno, M. (2022). Neuromorphic event-based spatio-temporal attention using adaptive mechanisms. In *Ieee international conference on artificial intelligence circuits and systems (aicas).*

List of figures

1.1 Evolution of ImageNet image classification performance over time. 2

2.1 Behaviour of a spiking neuron. 21
2.2 Biological neuron. 22
2.3 Comparison of the neurocomputational properties of spiking models. 23
2.4 Comparison between three SNN coding schemes. 27

3.1 Human retina, reduced to essential layers for neuromorphic visual sensors. 41
3.2 Block diagram of an event camera pixel. 41
3.3 Illustration of event-based encoding of visual signals. 41
3.4 CNN architecture of Ev-SegNet. 45

4.1 Network topology of Dikov's cooperative stereo-matching network. 57
4.2 Topology of Dikov's micro-ensembles preventing homolateral self-matching. . . 58

5.1 Visual cortex. 65
5.2 Biological foveation mechanism. 70
5.3 Visual representation of retinotopic foveation. 71
5.4 Interaural time difference. 73
5.5 Reichardt detector. 73
5.6 Myelination and delay learning. 74
5.7 Samples from the input dataset employed by Nadafian *et al.* 76
5.8 Spatio-temporal features learnt by the model of Nadafian *et al.* 76

6.1 General concept of spatial downscaling. 82
6.2 Spatial funnelling and event count methods. 83
6.3 Linear log-luminance reconstruction. 85
6.4 *Constant* strategy applied to linear log-luminance reconstruction. 86
6.5 *Interpolation* strategy applied to linear log-luminance reconstruction. 86
6.6 Linear log-luminance reconstruction. 87
6.7 *Constant* strategy applied to cubic log-luminance reconstruction. 88
6.8 *Interpolation* strategy applied to cubic log-luminance reconstruction. 88
6.9 Architecture of separate and mutual SNN downscaling methods. 92
6.10 Schematic illustration for SNN pooling. 92
6.11 Comparison of activity measures according to a varying synaptic weight ω for separate and mutual SNN downscaling methods applied to DVS128 Gesture. . . 93
6.12 Temporal funnelling downscaling methods. 95
6.13 Temporal structural downscaling methods. 96
6.14 Comparison of activity measures between different methods and reduction factors for DVS128 Gesture and N-MNIST datasets. 99
6.15 Run time for offline spatial and temporal downscaling methods. 100

6.16 Run time for online spatial downscaling methods. 101
6.17 Run time for spatial downscaling by SNN pooling. 102
6.18 Evolution of the accuracy performance of the SLAYER benchmark classifier applied to DVS128 Gesture and N-MNIST reduced spatially. 104
6.19 Evolution of the accuracy performance of the PLIF-based classifier applied to DVS128 Gesture and N-MMNIST reduced spatially. 104
6.20 PLIF-based classification performance according to polarity. 106
6.21 Poster presented to the participants of our human assessment study. 107
6.22 Interface presented to the participants during the human assessment experiment. . 109
6.23 Participants performing the human assessment experiment. 109
6.24 Human accuracy, rate of "unknown" responses and human response time to classify event data downsized spatially. 110
6.25 Human accuracy, rate of "unknown" responses and human response time to classify event data downsized temporally. 110
6.26 Human accuracy according to the number of events after spatial downscaling. . . 111
6.27 Human accuracy according to the number of events after temporal downscaling. . 111
6.28 Ratio of the human accuracy to the classifier accuracy, for each downscale factor, for spatial and temporal downscaling methods. 112
6.29 Overall comparison between spatial and temporal event downscaling methods. . . 113

7.1 Visualisation of the difference between a RoI and an OoI. 119
7.2 SNN model used to detect saliency by event density. 122
7.3 Bounded exponential WTA used as a radial basis function in the attention model. 123
7.4 Visual illustration of the behaviour of the weight adaptation rule. 123
7.5 Architecture of the spatiotemporal attention model filtering one OoI at a time. . . 125
7.6 Visual illustration of the behaviour of the threshold adaptation rule implemented in the saliency detection of one output. 126
7.7 Architecture of the saliency detection model adapted to SpiNNaker. 127
7.8 Architecture of the multi spatiotemporal attention. 129
7.9 Visual illustration of the behaviour of the threshold adaptation rule implemented in the saliency detection of multiple outputs. 130
7.10 Samples from the Prophesee Pedestrian dataset. 132
7.11 Evolution of quantitative and qualitative properties of the DVS 128 Gesture dataset after attentional filtering. 134
7.12 Simulation time for an event input of $1s$, detection latency and PLIF-based classification accuracy of the OoIs detected in DVS128 Gesture. 136
7.13 Mean number of events in and percentage of sensor activated by the three-gesture dataset. 136
7.14 Simulation on PyNN of the attention model applied to two temporally shidted samples. 137
7.15 Size invariance of the attention model with selection of one OoI, run on Loihi. . . 138
7.16 Spatial invariance of the attention model with selection of one OoI, run on Loihi. 138
7.17 Performance of the attentional filtering. 139
7.18 Mean precision-recall comparison in different noise ratio intervals. 141
7.19 Performance of simultaneous multi-OoIs detection according to the parameters ω_{WTA}, $\Delta\theta$ and Δt. 143

7.20 Multi-OoIs detection performance, the latency of the OoIs detection and classification accuracy of the selected OoIs of the multi-OoIs selection model. 143
7.21 Latency of the OoI selection according to the shift. 144

8.1 Overview of the neuromorphic foveation concept. 149
8.2 Binary foveation of events using the corresponding high and low resolution and based on a known RoI. 151
8.3 Classification performance on DVS 128 Gesture according to the number of events in the dataset in high resolution, low resolution, with only RoIs and after foveation. 152
8.4 Semantic segmentation performance according to the number of events in the dataset in high resolution, low resolution, with only RoIs and after foveation. 153
8.5 Semantic segmentation performance evolution according to the percentage of total events in the dataset after processing for the event data in high resolution, low resolution, only from RoIs and after foveation. 154
8.6 Qualitative study of the saliency detection, based on the semantic segmentation performance. 157
8.7 Quantitative study of the saliency detection. 157

9.1 Example of events accumulated into a frame, picturing the Mediterranean coastline. 165
9.2 Visual explanation of delay learning. 167
9.3 Movements of the events recorded by a UAV over the Mediterranean coastline. . 168
9.4 Movements of the events recorded by a UAV over the Mediterranean coastline. . 168
9.5 Features learned experimentally on noisy and non-noisy input data. 171
9.6 Evolution over time of the features learned experimentally on noiseless input data. 171
9.7 Features learned experimentally on noiseless input data, with a varying number convolution layers. 172
9.8 Issue in the identification of patterns based on delay learning. 172
9.9 Metrics to assess the quality of the features learned experimentally on noiseless input data, with a varying number convolution layers. 173
9.10 Tentative model of delay learning. 174
9.11 SNN architecture for vertical and horizontal line detection, with no knowledge of the line position. 175
9.12 SNN architecture for vertical and horizontal line detection, with knowledge of the line position. 175
9.13 SNN overall architecture and pattern of connections for one detector for the detection of diagonal lines. 176
9.14 Examples of line reconstruction from various detector's neurons activations. . . . 177
9.15 Communication between the SpiNNaker board and an external device as handled by the external_devices module. 180
9.16 Accumulated events from a DVS 128 Gesture sample. 182
9.17 Architecture of two lip reading SNN models. 184
9.18 Architecture of the overall spiking low-rate branch MSTP topology. 185

A.1 Samples from control and random symmetric datasets with $n = 2$ and $shift = 0$, constituted from the DVS 128 Gesture dataset. 231

A.2 Samples from control and random symmetric datasets with $n = 2$ and $shift = 0$, after spatial reduction using the event count method then attentional selection of one OoI. 231
A.3 Samples from control and random symmetric datasets with $n = 2$ and $shift = 0$, after spatial reduction then attentional selection of one OoI. 232
C.4 Visual representation of a DVS 128 Gesture sample after various processes. . . . 239
F.5 Example of line detection on static lines, without noise. 246
F.6 Example of line detection on moving lines, with noise. 247
G.7 Real time reconstruction of the disparity map of the "two fans" dataset from the cooperative stereo-matching results obtained on SpiNN-3. 252

List of tables

1.1 Overview of publications published during this PhD thesis. 14

2.1 Overview of existing SNN simulators on CPU. 32

3.1 Overview of existing event-based datasets. 47

6.1 Hyperparameters used to initialise separate and mutual SNN downscaling. 93
6.2 Features of the original DVS128 Gesture dataset and of the six spatial downscaling methods, for a downscaling factor of 4. 97
6.3 Accuracy of the PLIF-based classifier applied to DVS128 Gesture reduced using the temporal structural downscaling methods. 105
6.4 Accuracy of the PLIF-based classifier applied to DVS128 Gesture reduced using the temporal funnelling downscaling methods. 105
6.5 Demographic repartition of the human assessment experiment participants. . . . 108

7.1 Neuronal and synaptic parameters used for the model implementation on PyNN and Kraken. 133
7.2 SNE latency and energy consumption of the saliency detection functional modules. 135
7.3 Kraken platform energy consumption and latency. 135
7.4 Recall, accuracy, and F1 score of compared methods over the whole dataset. . . . 140
7.5 Size and mean spike counts comparison for an 640×480 input downscaled to 80×60. 141
7.6 Energy used by the SpiNNaker implementation 141
7.7 Hyperparameters used to implement the multi-OoI selection model. 142
7.8 Comparison between our contribution and the state-of-the-art. 146

8.1 Hyperparameters used to implement the saliency detection model. 151

9.1 Number of each diagonal pattern per cluster and corresponding Gini index, with a network trained on noiseless data and tested on noisy and non-noisy input data. . 170
9.2 Number of each diagonal pattern per cluster and corresponding Gini index, with a network trained on noisy data and tested on noisy and non-noisy input data. . . . 170
9.3 Neurons parameters of the lip-reading SNN model. 185
9.4 SNN1 accuracy on a subset of 10 classes from DVS-Lip, using different surrogate functions. 187
9.5 SNN results on the entire DVS-Lip dataset. 187
9.6 Spiking MSTP results on the entire DVS-Lip dataset. 187
9.7 Size and accuracy comparison between our models and the state-of-the-art. . . . 187

1 Accumulated frames from the spatially downscaled DVS 128 Gesture dataset. . . 233
2 Accumulated frames from the spatially downscaled N-MNIST dataset. 235

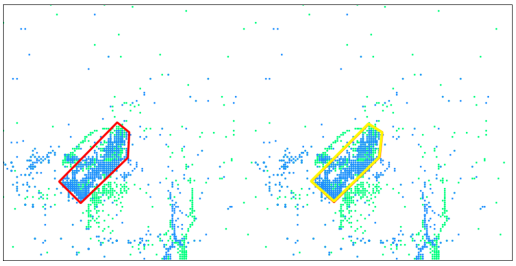
3 DDD17 events and corresponding segments identified by Ev-SegNet after various process. 237
4 Structure of the AEDAT data as saved by IMSE's embedded DVS sensor 241
5 Events repartition over time in the NTUA data. 242
6 Number of patches according to the number of events in the NTUA data. 243
7 Events at specific coordinates over time in the NTUA data. 244
8 Events repartition at specific coordinates over time in the NTUA data. 245
9 Comparison of the performance of Dikov's cooperative stereo-network the on original and downscaled "two fans" dataset. 248
10 Comparison of the performance of Dikov's cooperative stereo-network on yhr original and downscaled "pendulum" dataset. 249
11 Comparison of the performance of Dikov's cooperative stereo-network to the ground truth on DSEC. 250

List of algorithms

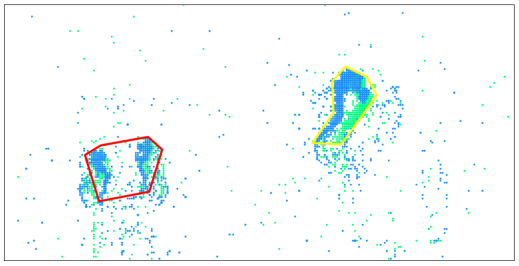
.1 Script to transform AEDAT2 event data into CSV data. 240
.2 Script to transform CSV event data into NPZ data. 240
.3 SpiNNaker's neural population "SpikeInjector". 251
.4 Function to send input spikes into a sPyNNaker neural network. 251
.5 Function to receive output spikes from a sPyNNaker neural network. 251

Annexes

A Visualisation of RoIs detected by our spatiotemporal visual attention mechanism

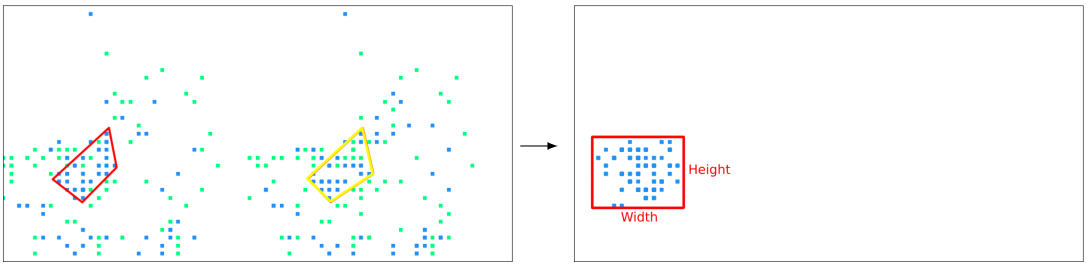


A) Control dataset — (left and right) gesture "right arm clockwise" performed by user 5.

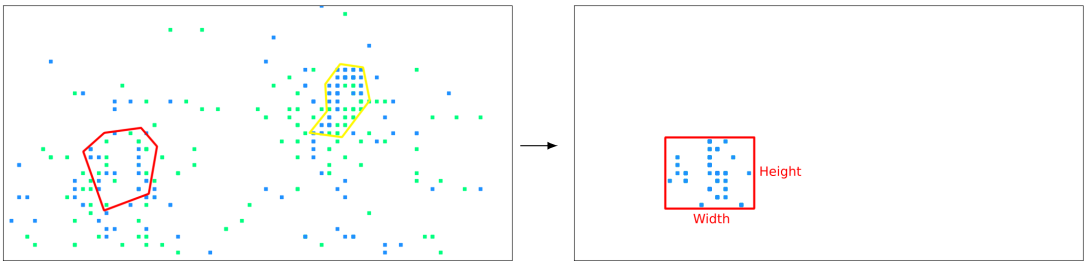


B) Random symmetric dataset — (left) gesture "hand clap" performed by user 16 and (right) gesture "right arm counter clockwise" performed by user 5.

Figure A.1 – Samples from control and random symmetric datasets with $n = 2$ and $shift = 0$, constituted from the DVS 128 Gesture dataset.

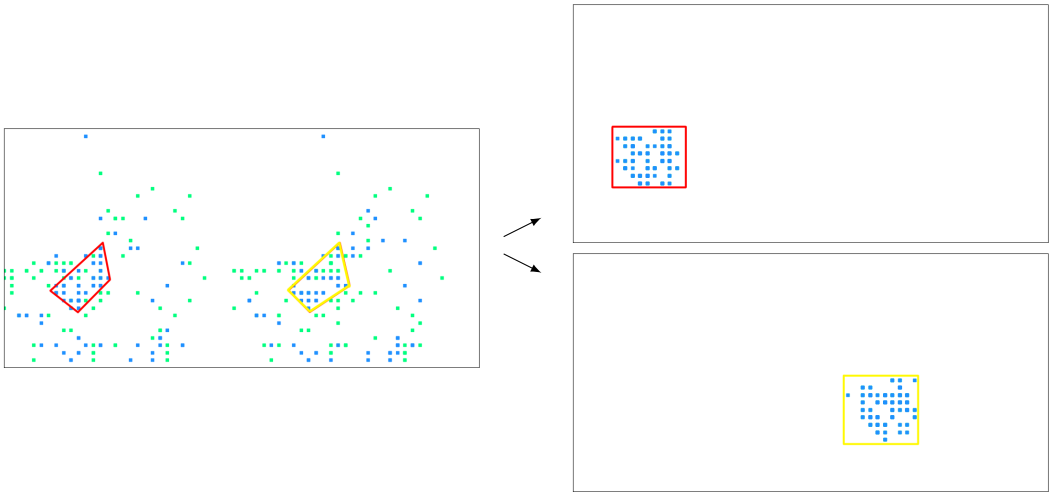


(a) Input reduced data to attentional selection of a "control" sample.

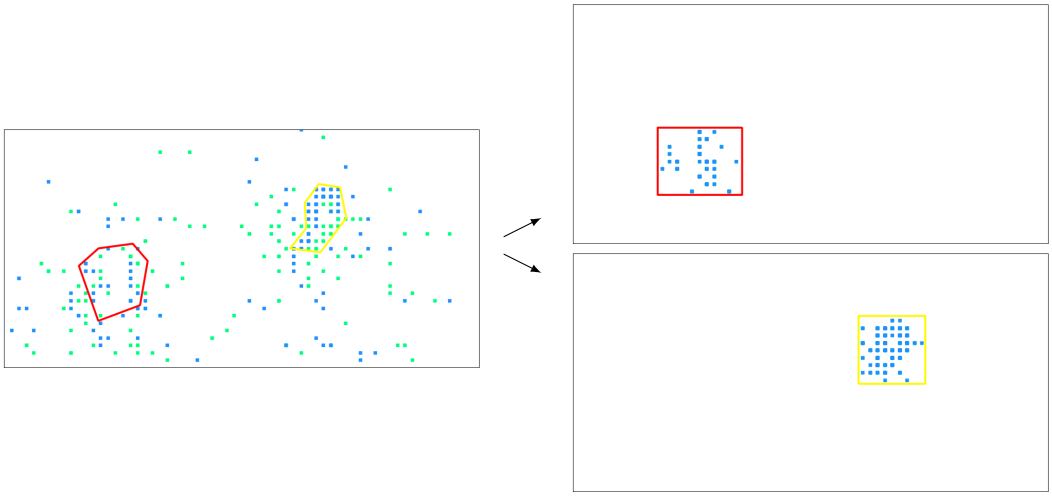


(b) Input reduced data to attentional selection of a "random symmetric" sample.

Figure A.2 – Samples from control and random symmetric datasets with $n = 2$ and $shift = 0$, after spatial reduction using the event count method introduced in (Gruel et al., 2022) then attentional selection of one OoI.



(a) Input reduced data to attentional selection of multi-OoIs in a "control" sample.



(b) Input reduced data to attentional selection of multi-OoIs in a "random symmetric" sample.

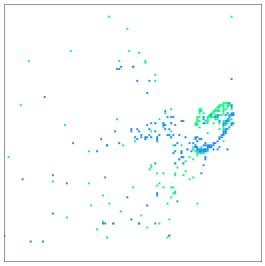
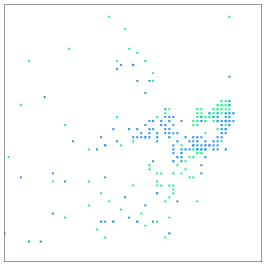
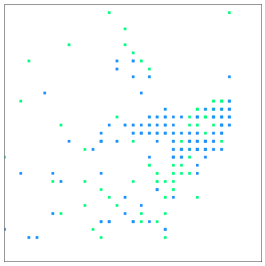
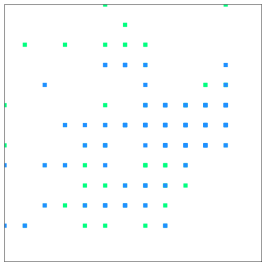
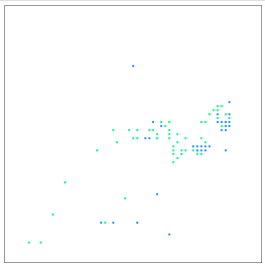
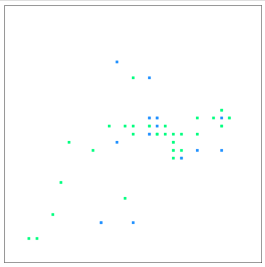
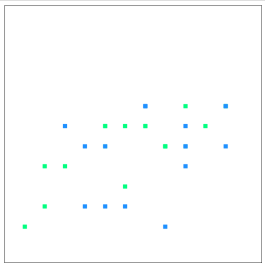
Figure A.3 – Samples from control and random symmetric datasets with $n = 2$ and $shift = 0$, after spatial reduction then attentional selection of one OoI.

B Visualisation of different spatial downscaling methods.

Tables 1 and 2 present one frame extracted respectively from the DVS 128 Gesture (Amir et al., 2017) and N-MNIST (Orchard et al., 2015) datasets.

Three examples are presented for each method, one corresponding to a spatial downscaling by a factor of 2 (first column), another by a factor of 4 (second column) and the last by a factor of 10 (third column). The corresponding number of events produced by each method from this sample is indicated below each frame.

Table 1: Frames produced from the sample `user14_led_0.npz`, corresponding to the gesture "left hand clockwise" (class 6) from the DVS128 Gesture dataset, processed with the spatial downscaling methods implemented during this PhD. Each frame corresponds to the accumulation of the events occurring during the sample's first 5 ms. Green and blue pixels correspond respectively to positive and negative events.

Original			
Number of events:		670, 118	
Methods	**Downscaled by 2**	**Downscaled by 4**	**Downscaled by 10**
Simple event funnelling			
Number of events:	649, 507	649, 414	648, 989
Log-luminance with event count			
Number of events:	105, 164	25, 400	3, 958

Continued on next page

Table 1: Frames produced from the sample `user14_led_0.npz`, corresponding to the gesture "left hand clockwise" (class 6) from the DVS128 Gesture dataset, processed with the spatial downscaling methods implemented during this PhD. Each frame corresponds to the accumulation of the events occurring during the sample's first 5 ms. Green and blue pixels correspond respectively to positive and negative events. (Continued)

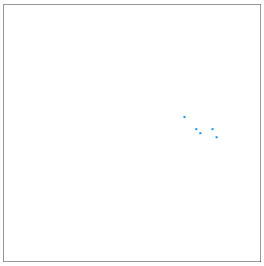
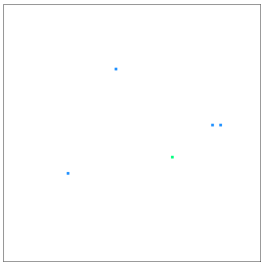
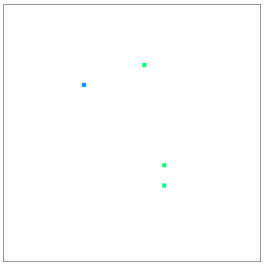
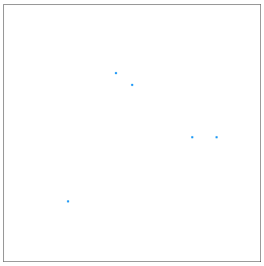
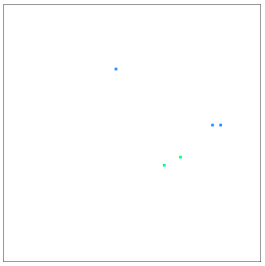
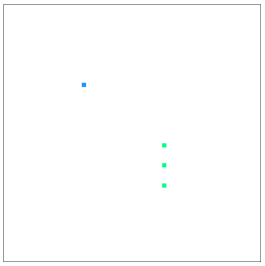
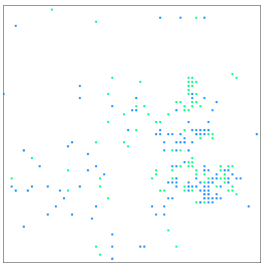
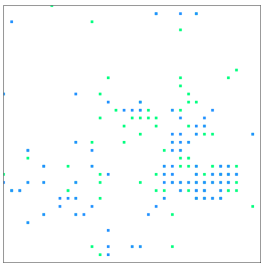
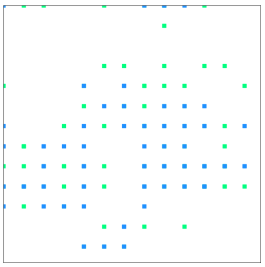
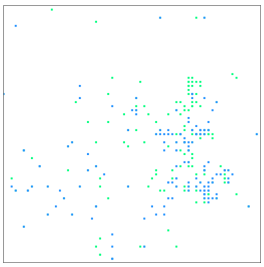
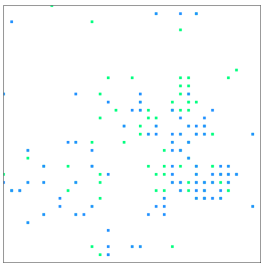
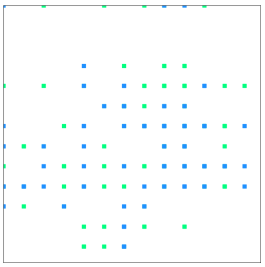
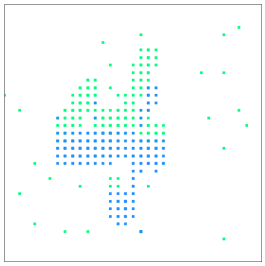
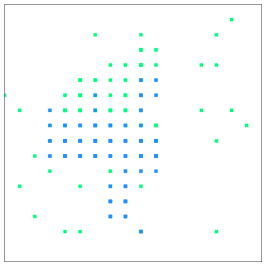
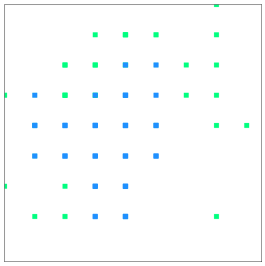
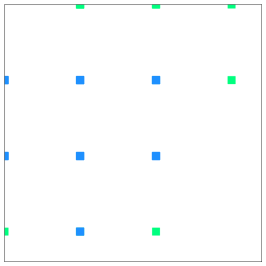
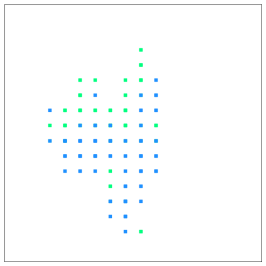
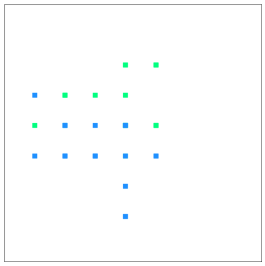
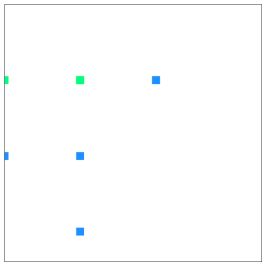
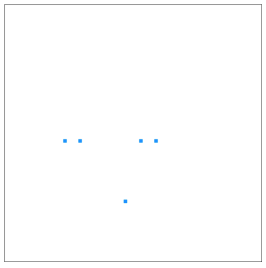
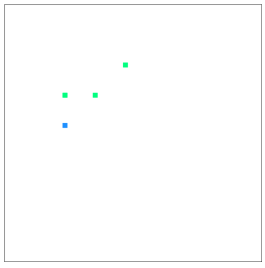
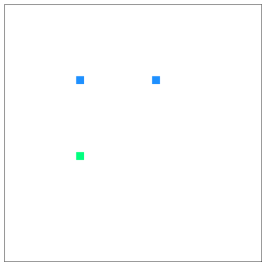
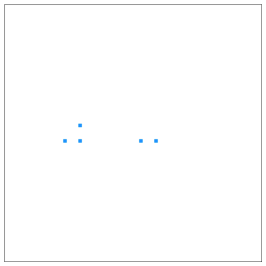
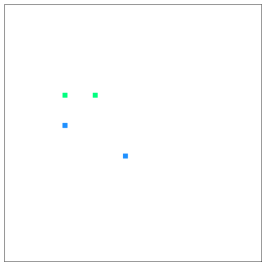
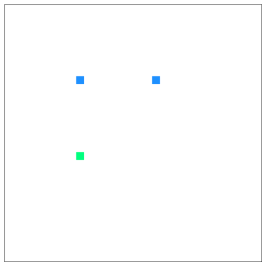
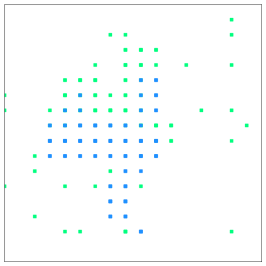
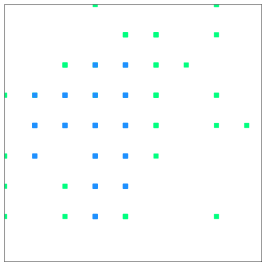
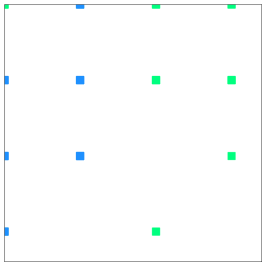
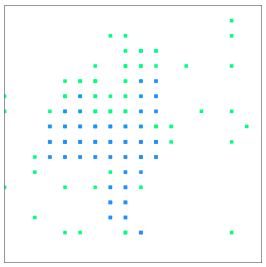
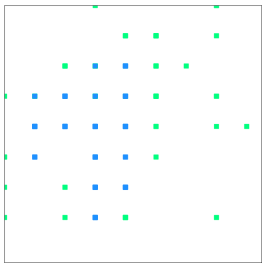
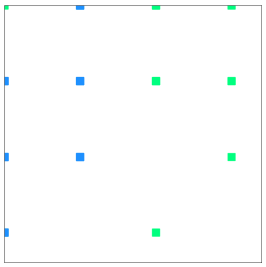
Log-luminance with linear estimation			
Number of events:	$7,016$	$1,156$	143
Log-luminance with cubic estimation			
Number of events:	$7,513$	$1,267$	129
Separate SNN pooling			
Number of events:	$121,508$	$88,629$	$48,155$
Mutual SNN pooling			
Number of events:	$109,395$	$80,551$	$43,597$

Table 2: Frames produced from the sample `user14_led_0.npz`, corresponding to the number "4" from the N-MNIST dataset, processed with the spatial downscaling methods implemented during this PhD. Each frame corresponds to the accumulation of the events occurring during the sample's first 5 ms. Green and blue pixels correspond respectively to positive and negative events.

Original			
Number of events:		$3,848$	
Methods	**Downscaled by 2**	**Downscaled by 4**	**Downscaled by 10**
Simple event funnelling			
Number of events:	$3,845$	$3,844$	$3,838$
Log-luminance with event count			
Number of events: 716	165	19	
Log-luminance with linear estimation			
Number of events:	566	76	3

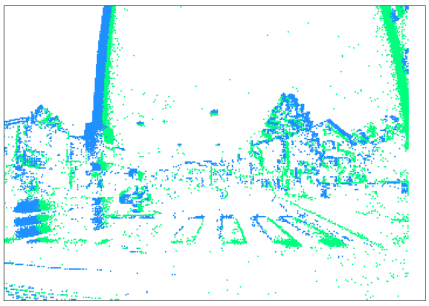
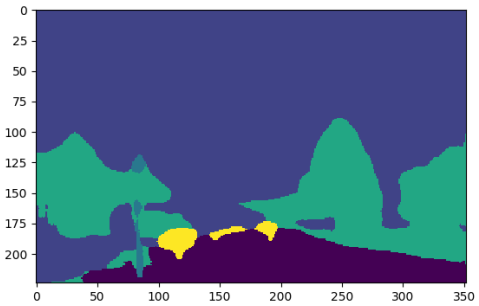
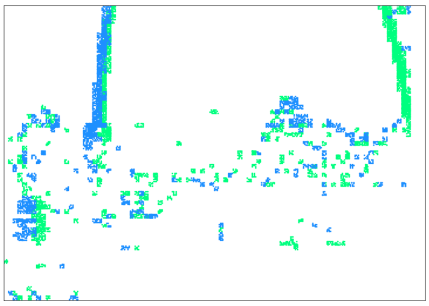
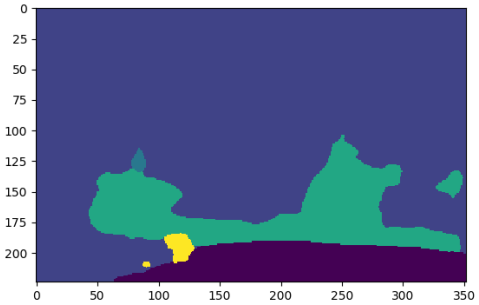
Continued on next page

Table 2: Frames produced from the sample `user14_led_0.npz`, corresponding to the number "4" from the N-MNIST dataset, processed with the spatial downscaling methods implemented during this PhD. Each frame corresponds to the accumulation of the events occurring during the sample's first 5 ms. Green and blue pixels correspond respectively to positive and negative events. (Continued)

Log-luminance with cubic estimation			
Number of events:	597	84	3
Separate SNN pooling			
Number of events:	5,373	3,507	1,766
Mutual SNN pooling			
Number of events:	4,800	3,157	1,617

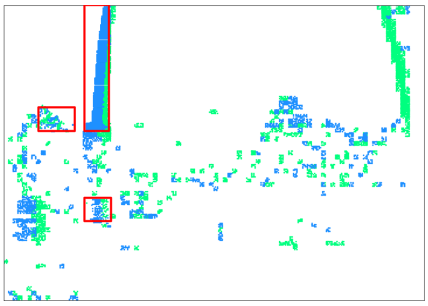
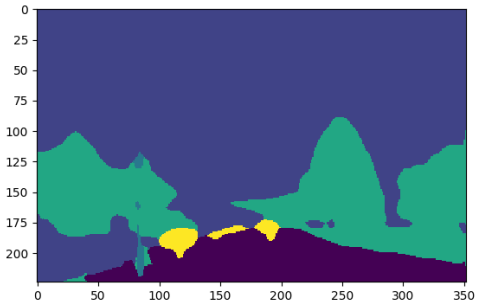
C Visualisation of the stakes of neuromorphic foveation

Table 3: Visual representation of the events and the different segments identified by Ev-SegNet in the sample `rec1487417411_export_1467` from the DDD17 dataset, after various processes. The subplot on the left corresponds to the original data and in the middle–left to the same event data spatially reduced by 4 using the *event count* method. The middle-right subplot corresponds to the events contained in the RoIs detected on the data reduced using the *event count* method, and the right one to the sample foveated according to those RoIs (identified by the red bounding boxes). On the left, each frame corresponds to the accumulation of the events over 50ms, the sample's time-window. Green and blue pixels correspond respectively to positive and negative events.

	Accumulated events in the sample	Segments and performance values identified by Ev-SegNet
Original data	 $N_{events} = 20,199$	 $Accuracy = 86.06\%$ $MIoU = 0.5$
Reduced data	 $N_{events} = 895$	 $Accuracy = 78.92\%$ $MIoU = 0.3508$

Continued on next page

Table 3: Visual representation of the events and the different segments identified by Ev-SegNet in the sample `rec1487417411_export_1467` from the DDD17 dataset, after various processes. The subplot on the left corresponds to the original data and in the middle–left to the same event data spatially reduced by 4 using the *event count* method. The middle-right subplot corresponds to the events contained in the RoIs detected on the data reduced using the *event count* method, and the right one to the sample foveated according to those RoIs (identified by the red bounding boxes). On the left, each frame corresponds to the accumulation of the events over 50ms, the sample's time-window. Green and blue pixels correspond respectively to positive and negative events. (Continued)

Only RoIs		
	$N_{events} = 4,283$	$Accuracy = 74.96\%$ $MIoU = 0.1249$
Foveated data		
	$N_{events} = 4,716$	$Accuracy = 84.75\%$ $MIoU = 0.4648$

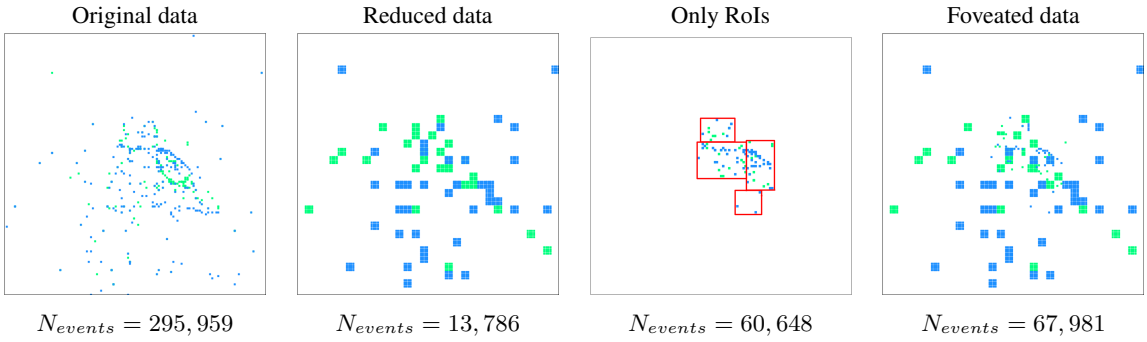


Figure C.4 – Visual representation of the events in the sample `user25_led`, corresponding to the gesture "left hand clockwise" (class 6) from the DVS128 Gesture dataset (Amir et al., 2017), after various process. The subplot on the left corresponds to the original data and in the middle-left to the same event data spatially reduced by 4 using the *event count* method. The middle-right subplot corresponds to the events contained in the RoIs detected on the dataset reduced using the *event count* method, and the right one to those RoIs (identified by the red bounding boxes). Each frame corresponds to the accumulation of the events occurring during the sample's first 10 ms. Green and blue pixels correspond respectively to positive and negative events.

D Pipelines for event data handling

D.1 From AEDAT2 to CSV files

```
clearvars
close all;
dbstop if error
aedat = struct;

%-------------------------------- USER INPUT --------------------------------%
aedat.source = 'Dvs128';
aedat.importParams.filePath = '/path_to_existing_aedat2_file/';
csv_path = '/path_to_new_csv_file/';
%----------------------------------------------------------------------------%

aedat = ImportAedat(aedat);

events = zeros(aedat.data.polarity.numEvents, 4);
events(:, 1) = aedat.data.polarity.x;
events(:, 2) = aedat.data.polarity.y;
events(:, 3) = double((aedat.data.polarity.timeStamp) - aedat.data.polarity.
    timeStamp(1))/1000000;
events(:, 4) = aedat.data.polarity.polarity;

csvwrite(csv_path,events);
```

Algorithm .1 – Matlab script transforming the AEDAT2 event data provided by NTUA into CSV files.

D.2 From CSV to NPZ files

```
from numpy import genfromtxt, save

csv_path = 'path_to_existing_csv_file';
events = genfromtxt(csv_path, delimiter=',')
save(csv_path.replace('.csv', '.npy'), events)
```

Algorithm .2 – Python script transforming the event data contained in the CSV files produced by Algo. .1 into NPZ event data.

E Analysis of DVS and RGB coastline dataset

data	polarity	numEvents x y timeStamp polarity
infos	firstTimeStamp lastTimeStamp numEventsInFile dateTime beginningOfDataPointer source deviceAddressSpace fileFormat	
source		
importParams	filePath	

TABLE 4 – Structure of the AEDAT data as saved by IMSE's embedded DVS sensor. Each attribute is the mother of the following attributes on the same line.

TABLE 5 – Events repartition over time, where the samples are temporally split into successive patches (each $1s$ long).

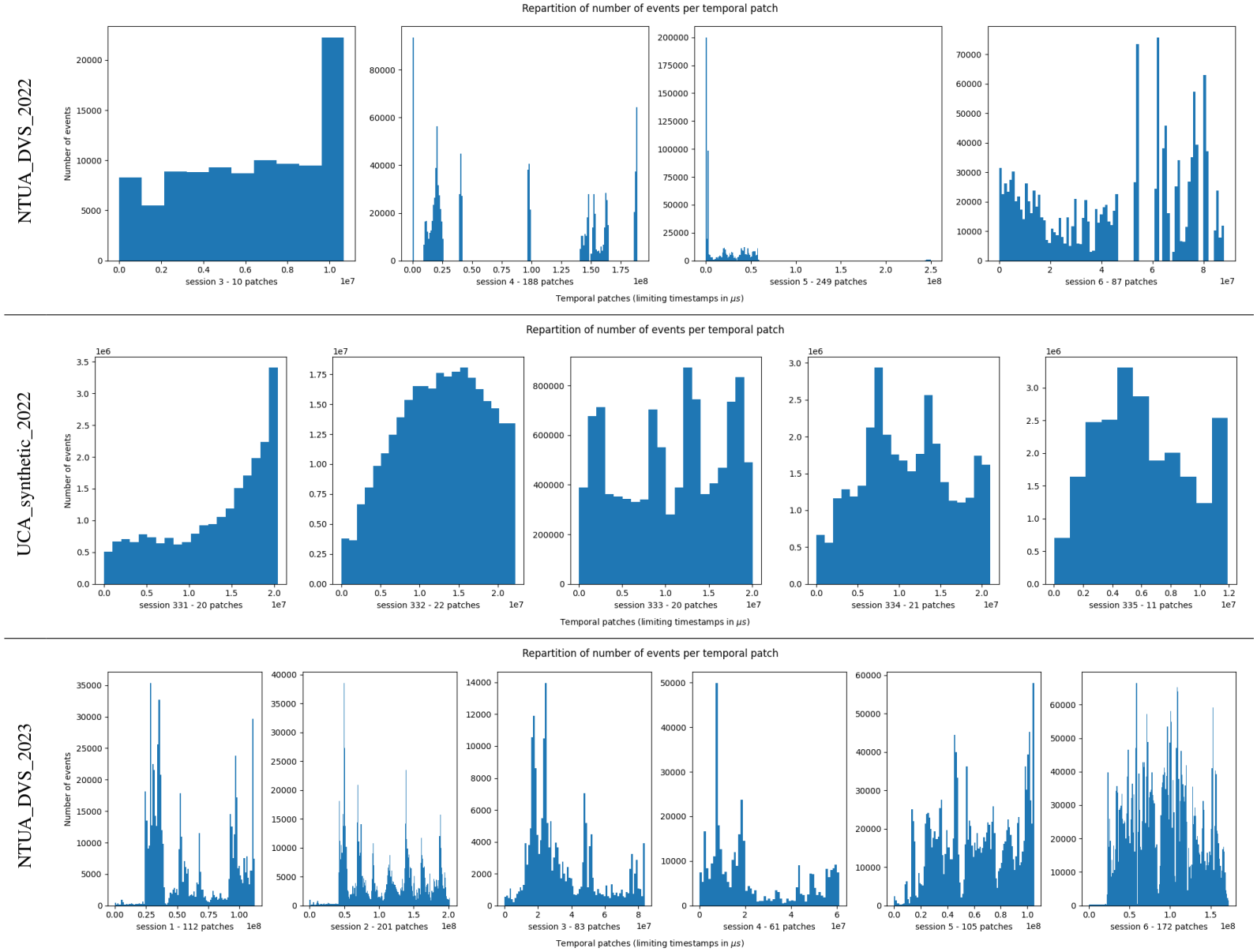


TABLE 6 – Number of patches according to the number of events in said patches, where the samples are temporally split into successive patches (each 1s long). It is particularly interesting to note the high number of empty patches in the "NTUA_DVS_2022" dataset and their absence in the "NTUA_DVS_2023" and "UCA_synthetic_2022" datasets.

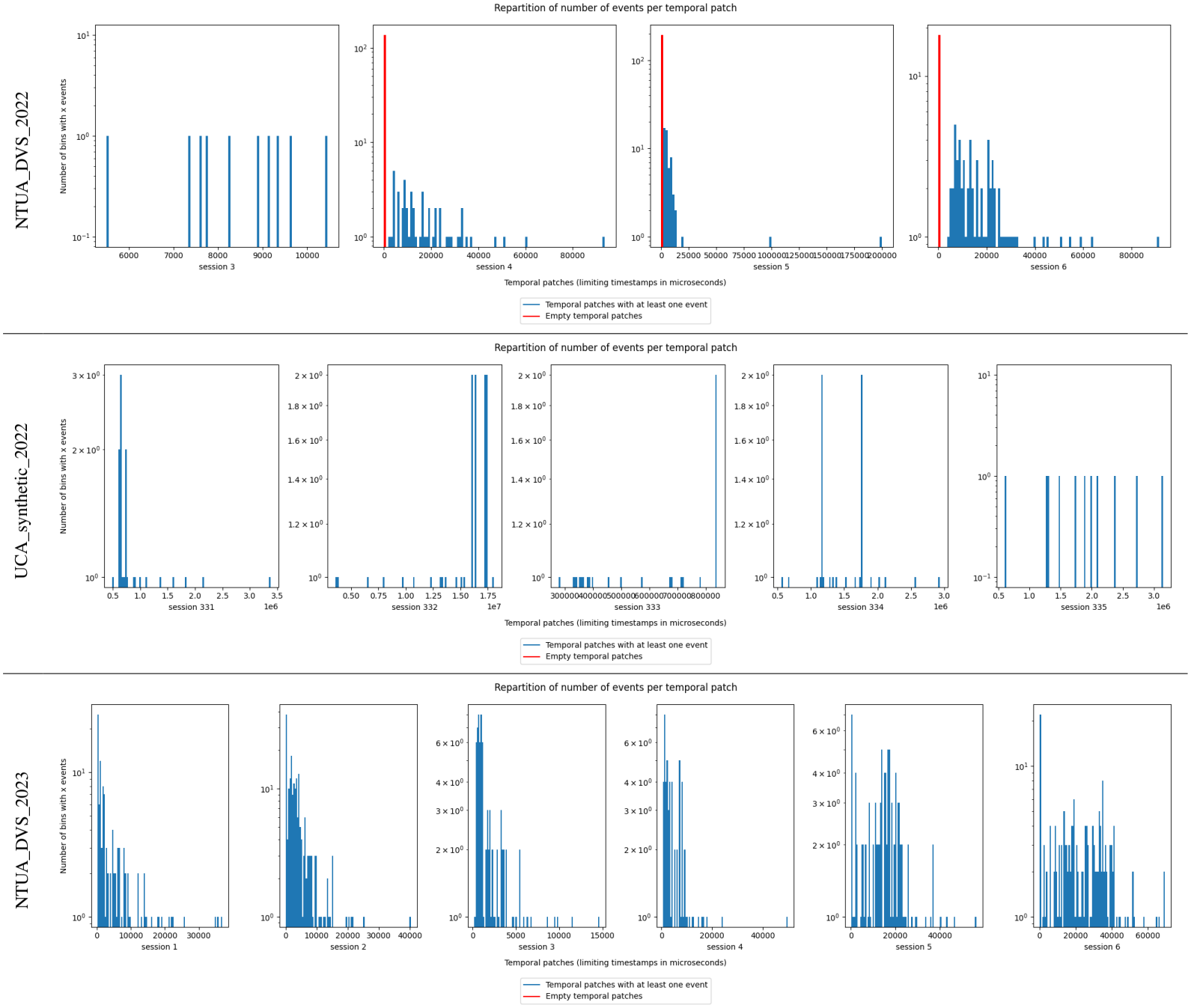


TABLE 7 – Events of coordinates $(x, 0)$ (top) and $(0, y)$ (right) over time.

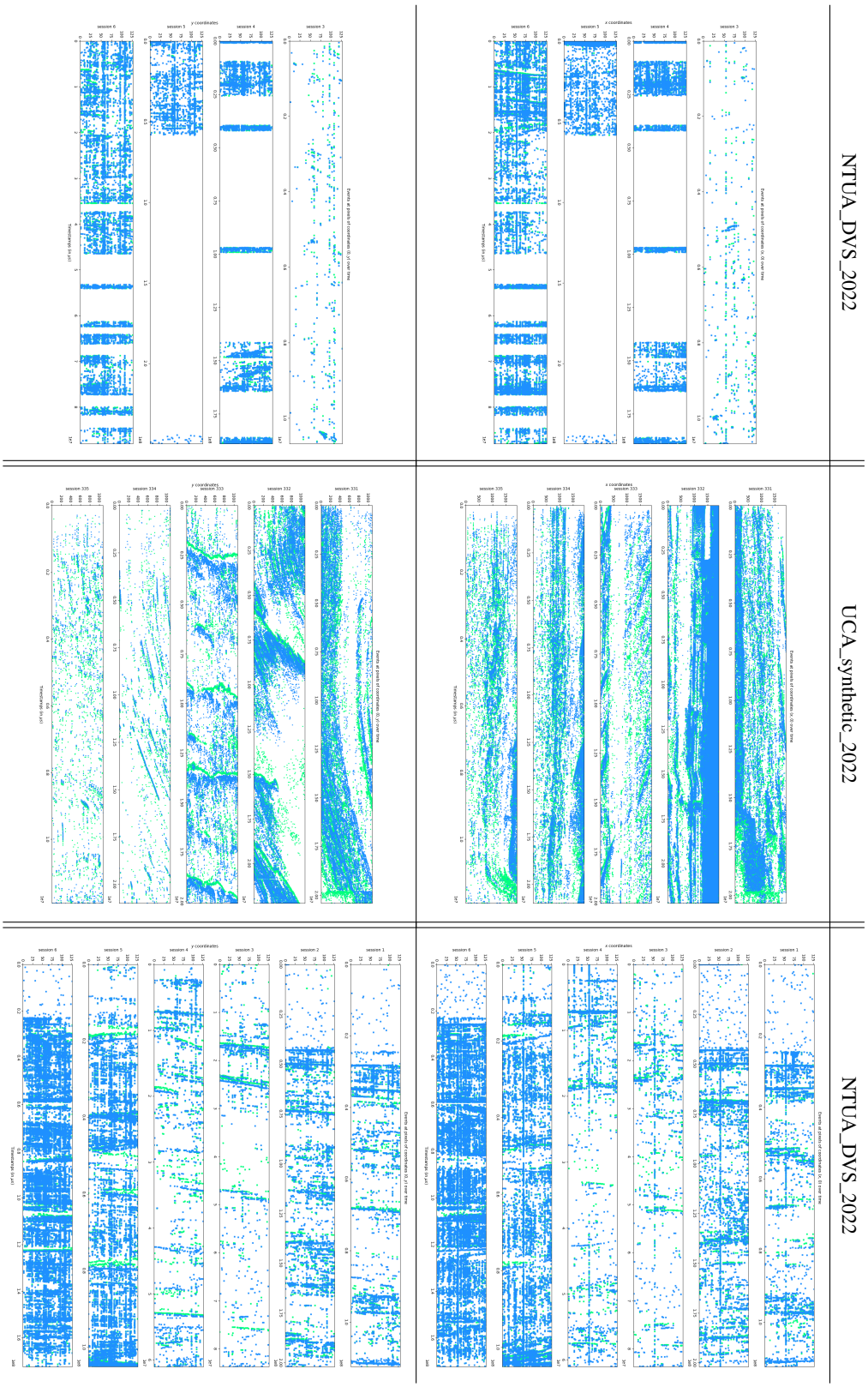
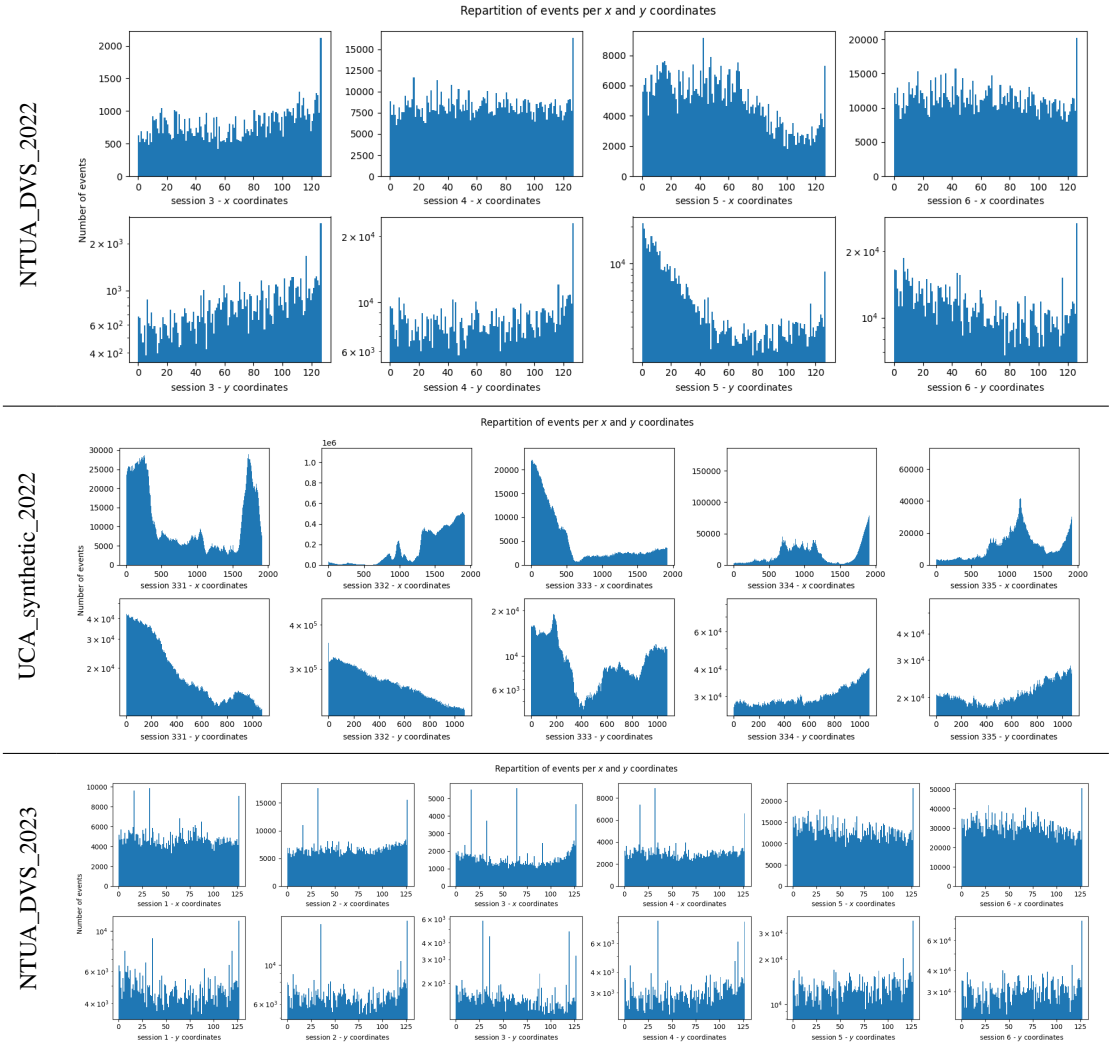


TABLE 8 – Repartition of events per x (top) and y (bottom) coordinates.



F Line detection

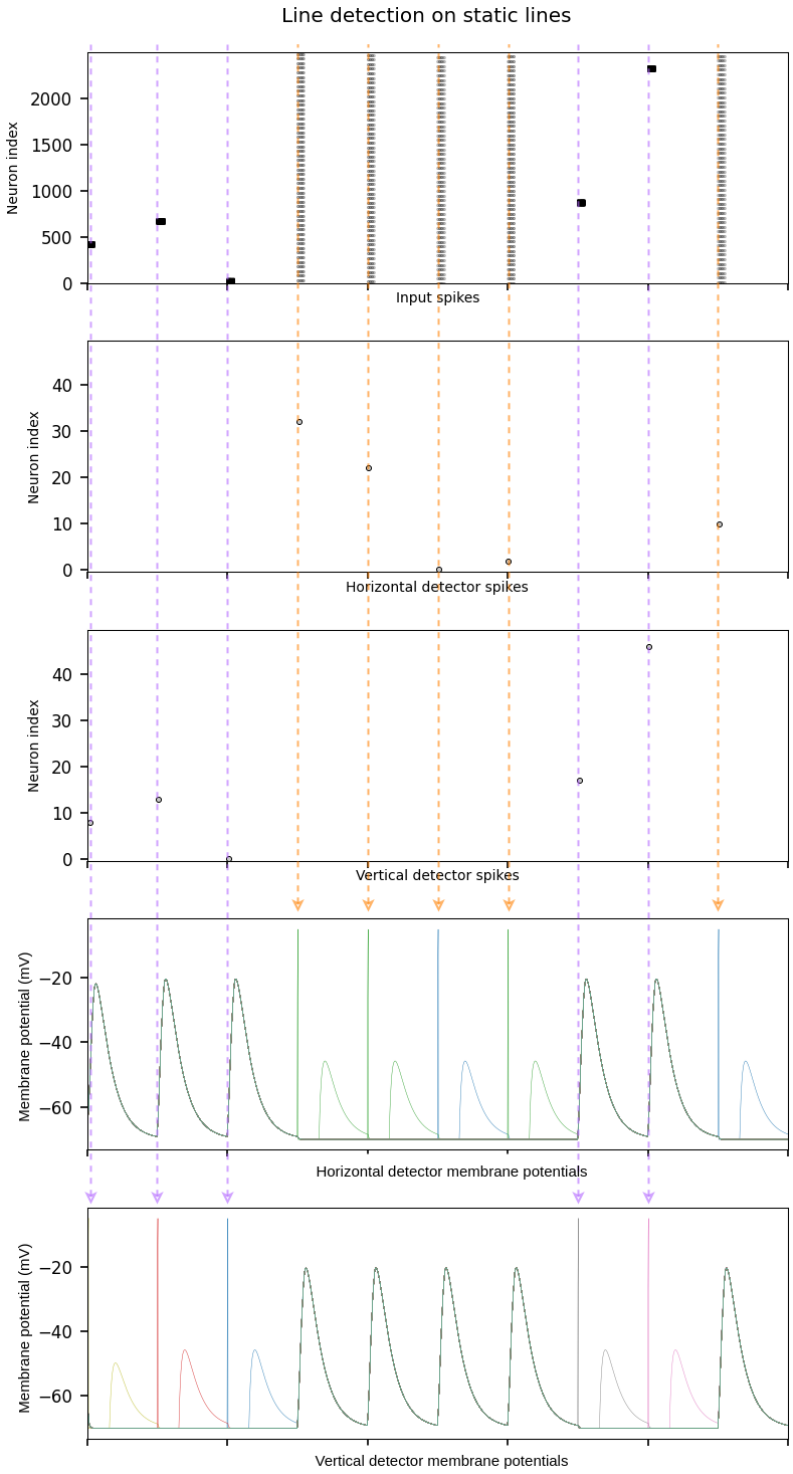


Figure F.5 – Example of line detection on static lines produced by the line detection SNN. The neuronal activity is presented as a raster plot according to their index (y-axis) and time (x-axis). Static lines are generated during $10\mu s$ in input every $100\mu s$, randomly horizontal or vertical (first plot). The corresponding layer, either the horizontal (second plot) or the vertical line detector (third plot), activates when it identifies the presence and type of line in the output. This activation is correlated with the corresponding membrane threshold (fourth and fifth plots). The orange and purple dotted lines indicate respectively the apparition of the horizontal and vertical static lines.

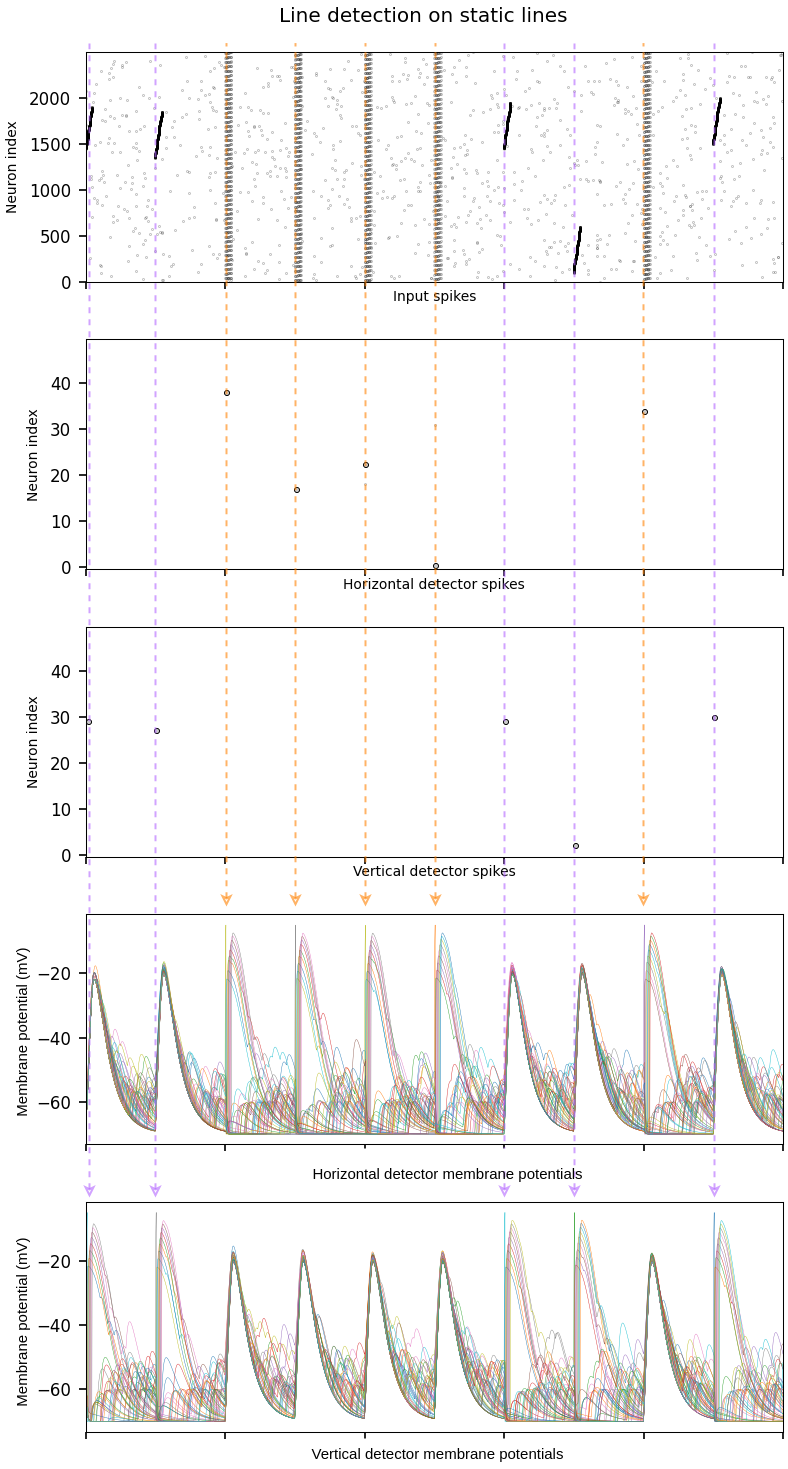
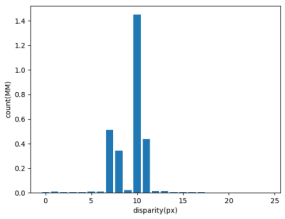
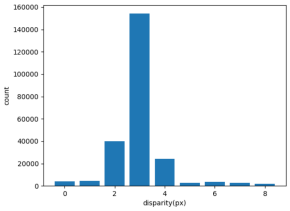
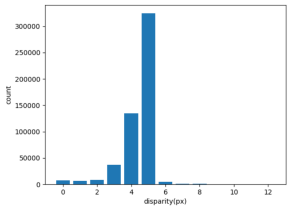
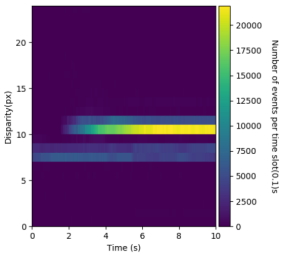
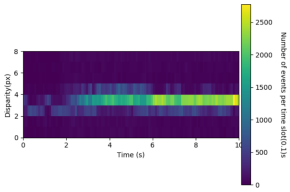
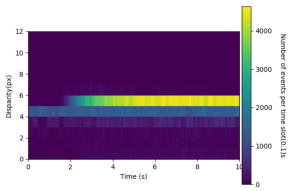
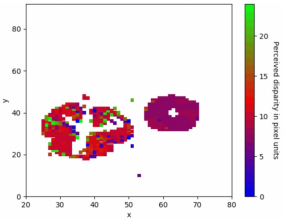
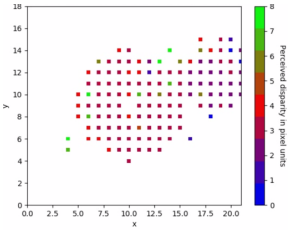
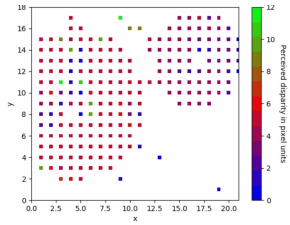


Figure F.6 – Example of line detection on moving lines with background noise produced by the line detection SNN. The neuronal activity is presented as a raster plot according to their index (y-axis) and time (x-axis). Moving lines are generated during $10\mu s$ in input every $100\mu s$, randomly horizontal or vertical (first plot). The corresponding layer, either the horizontal (second plot) or the vertical line detector (third plot), activates when it identifies the presence and type of line in the output. This activation is correlated with the corresponding membrane threshold (fourth and fifth plots). The orange and purple dotted lines indicate respectively the apparition of the horizontal and vertical moving lines.

G 3D reconstruction

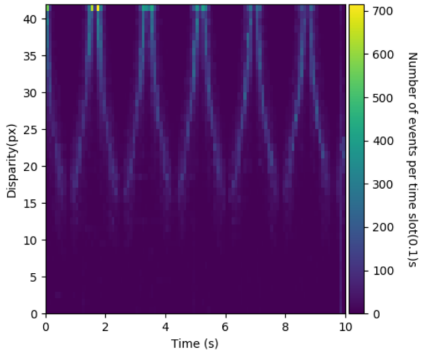
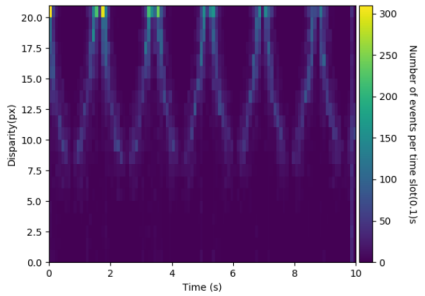
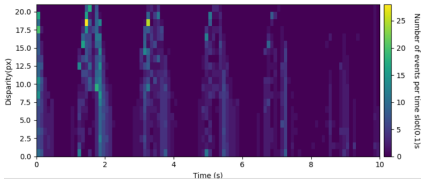
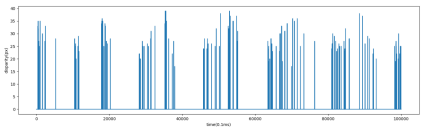
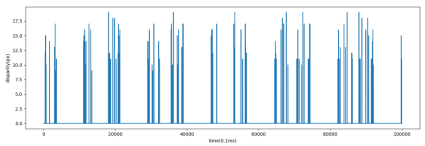
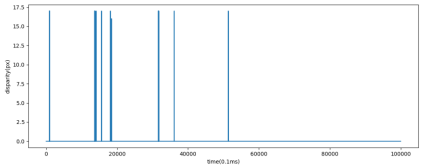
G.1 "Two fans" dataset

TABLE 9 – Comparison of the performance of the cooperative stereo-network on original (first column) and downscaled (second and third columns) data from the "two fans" dataset. The input data is spatially downscaled using the event count method by a factor 3 (second column) or downscaled similarly by a factor 2 then cropped (third column), before being fed to the cooperative stereo-matching network implemented on SpiNN-3.

Performance	Original data on EBRAINS	Downscaled data (divided by 3) on SpiNN-3	Downscaled data (divided by 2 and cropped) on SpiNN-3
Parameters	$x_{max} = y_{max} = 80$ $d_{max} = 25$	$x_{max} = y_{max} = 22$ $d_{max} = 21$	$x_{max} = y_{max} = 22$ $d_{max} = 21$
Accumulated network output			
Network output over time			
Disparity map	 $t = 2.5s$	 $t = 4s$	 $t = 4s$
Performance accuracy	Distant fan: 95.8% of events at $d = 7 \pm 1$ Proximate fan: 96.5% of events at $d = 10 \pm 1$	Distant fan: 87% of events at $d = 2 \pm 1$ Proximate fan: 90.1% of events at $d = 3 \pm 1$	Distant fan: 92% of events at $d = 3 \pm 1$ Proximate fan: 90% of events at $d = 5$

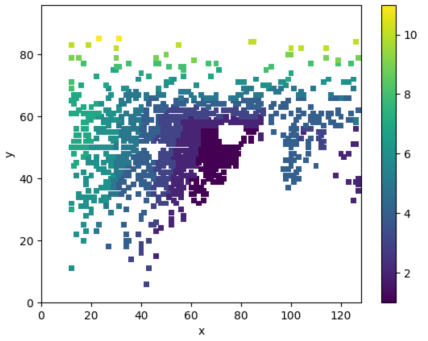
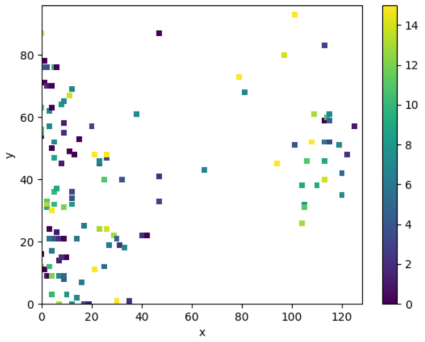
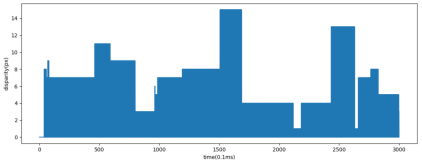
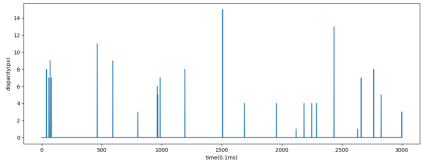
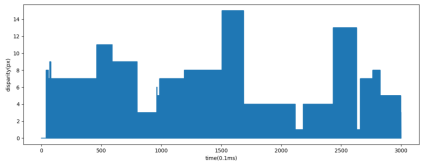
G.2 "Pendulum" dataset

TABLE 10 – Comparison of the performance of the cooperative stereo-network on original (first column) and downscaled (second and third columns) data from the "pendulum" dataset. The input data is spatially downscaled using the event count method by a factor 2 then fed to the network implemented on EBRAINS (second column) or downscaled similarly by a factor 2 then cropped before being fed to the network implemented on SpiNN-3 (third column).

	Performance on original data on EBRAINS	Performance on downscaled data (divided by 2) on EBRAINS	Performance on downscaled data (divided by 2 and cropped) on SpiNN-3
Parameters	$x_{max} = y_{max} = 130$ $d_{max} = 50$	$x_{max} = y_{max} = 65$ $d_{max} = 21$	$x_{max} = y_{max} = 22$ $d_{max} = 21$
Accumulated network output			
Network output of one ball pixel over time			

G.3 DSEC

TABLE 11 – Comparison of the ground truth (first column) and the performance of the cooperative stereo-network pre (second column) and post (third column) adoption of the maintenance of the previously calculated disparity. The input data is spatially downscaled using the event count method by a factor of 5 and then fed to our cooperative stereo-matching network implemented on EBRAINS (second and third column).

		Performance on downscaled data (divided by 5) on EBRAINS pre disparity maintenance	Performance on downscaled data (divided by 5) on EBRAINS post disparity maintenance
Disparity map at $t = 200ms$			
Network output over time	X		

G.4 Communication between the SpiNNaker board and an external device

```
my_population = pyNN.spiNNaker.Population(
    'size␣of␣population',
    pyNN.spiNNaker.external_devices.SpikeInjector(),
    label='population␣label')
```

Algorithm .3 – SpiNNaker's neural population "SpikeInjector" and its attributes.

```
def send_spike(label, sender):
    sender.send_spike(label, 0, send_full_keys=True)
```

Algorithm .4 – Function from the sPyNNaker library to send input spikes in real time (during a simulation run) into a neural network.

```
def receive_spikes(label, time, neuron ids):
    for neuron_id in neuron_ids:
        print(time, label, neuron_id)

live_spikes_connection =
    pyNN.spiNNaker.external_devices.SpynnakerLiveSpikesConnection(
        receive_labels='label',
        local_port='your␣port␣number',
        send_labels=['label']
    )
live_spikes_connection.add_receive_callback('label', receive_spikes)
```

Algorithm .5 – Function from the sPyNNaker library to receive in real time (during a simulation run) spikes output by a neural network.

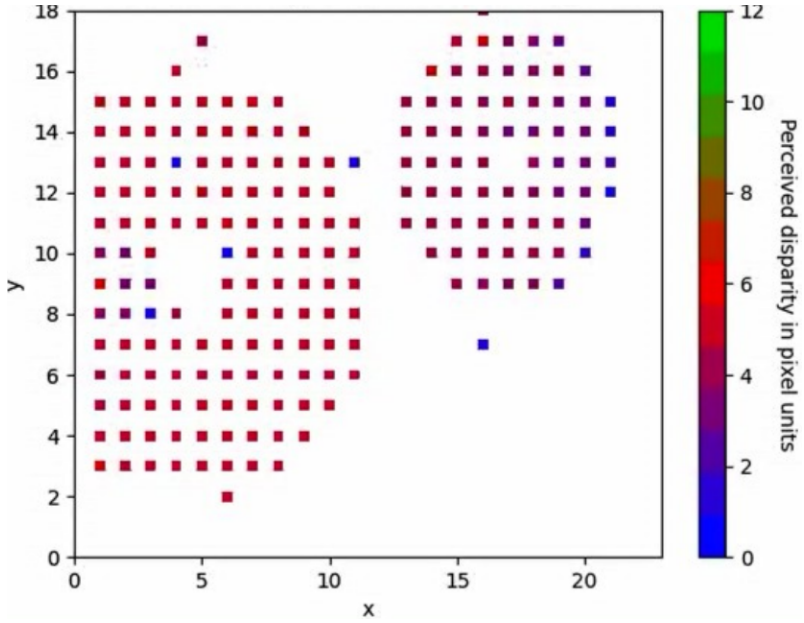


Figure G.7 – Real time reconstruction of the disparity map of the "two fans" dataset from the cooperative stereo-matching results obtained on SpiNN-3.

H i3S dataset

H.1 Composition of the i3S dataset

- addition
- appétit
- attirer
- baguette
- ballon
- beau
- bonne
- carnaval
- cheval
- chiffon
- coeur
- concombre
- décider
- dessin
- doudou
- école
- fillette
- hache
- huître
- joli
- jouet
- joyeux
- lèvres
- lundi
- maison
- maman
- manoir
- moto
- musique
- neige
- panier
- papa
- parking
- pomme
- pyjama
- requin
- robe
- ruisseau
- sapin
- serpent
- tentation
- terminer
- voiture
- xylophone
- zèbre

H.2 Tree structure of the i3S dataset

```
words/
    mot_1/
        user1_XXX/
```

```
            recording_1.bias
            recording_1.npy
            recording_1.raw
            recording_2.bias
            recording_2.npy
            recording_2.raw
            ...
        user2_XXX/
            ...
mot_2/
    ...
```

Spiking neural networks for embedded event-based vision

Amélie GRUEL

Abstract

In recent years, embedded computer vision has become omnipresent. It encompasses tasks such as detection, recognition and tracking of visual elements, with applications in robotics (autonomous driving), industries (assessment of product quality, automation of repetitive tasks), security, customer experience, social networks, etc. This widespread impetus only reinforces the need to overcome the challenges posed by this area of research, which requires gargantuan energy, high memory, and support for a wide range of algorithms.
We believe a promising solution to these challenges can be found in the combined use of spiking neural networks (SNNs) and event-based cameras. SNNs consist of a bio-inspired artificial neural network aiming to mimic the dynamics of biological neurons by processing the information as spike trains. Event cameras are a novel type of bio-inspired visual sensor which generates asynchronous data according to pixel intensity changes. It is ideal for real-time applications, but the great amount of timed information is challenging to process using standard computer vision models. Moreover, event data make a natural match for SNNs in terms of biological inspiration, energy savings, latency and memory use for dynamic visual data processing.

However, the novelty of SNNs and event cameras leaves room for many improvements in terms of optimal preprocessing of data as well as how this data is processed, making the most of the particularities of these scientific concepts. In this thesis, we identified several questions related to this broad field of research, which we have condensed into two main topics.
The first issue concerns the optimisation of the embedded preprocessing of event data acquired by an onboard camera to facilitate subsequent analysis. We propose three solutions: event data could either 1) be spatially or temporally reduced, either online or offline; 2) keep only salient elements and discard the rest; 3) be subject to foveation, as a bio-plausible compromise between the two previous solutions. We compared qualitatively and quantitatively the event data obtained after applying each preprocessing method to assess which method yields the best trade-off between the amount of data (i.e. events) kept *versus* the relevance of information maintained.
The second challenge is the exploitation of the SNN relevance to processing the specific chronology of event data in an embedded context.
Overall, we hope to have made a valuable contribution to the exploitation of the unique advantages of combining SNNs and event cameras for embedded computer vision, especially concerning event data preprocessing.

Keywords: Spiking neural networks, Event-based cameras, Neuromorphic, Computer vision, Computational neuroscience, Pattern recognition.