# Event-Based Feature Detection, Recognition and Classification

Gregory Kevin Cohen

University of Pierre et Marie Curie

Western Sydney University

Doctorate school: SMAER

*Institut de la vision - Vision and Natural Computation*
*BENS Group – The MARCS Institute*

# Event-Based Feature Detection, Recognition and Classification

Defended by Gregory Kevin Cohen

Discipline/Speciality

**Robotic**

Supervised by:

Pr. Ryad Benosman, Pr. André van Schaik and Pr. Jonathan Tapson

Defended 5th September 2016

Defense committee:

| | |
|---|---|
| Pr. Barnabé Linares-Barranco, Instituto de Microelectrónica de Sevilla | Reviewer |
| Pr. Elisabetta Chicca, Universität Bielefeld | Reviewer |
| Pr. Bruno Gas, Institut des Systèmes Intelligents et de Robotique, UPMC | Examiner |
| Pr. Jonathan Tapson, The MARCS Institute, Western Sydney Univeristy | PhD advisor |
| Pr. Ryad Benosman, Institut de la vision, Université Pierre et Marie Curie | PhD advisor |
| Pr. André van Schaik, The MARCS Institute, Western Sydney Univeristy | PhD advisor |

# Abstract

One of the fundamental tasks underlying much of computer vision is the detection, tracking and recognition of visual features. It is an inherently difficult and challenging problem, and despite the advances in computational power, pixel resolution, and frame rates, even the state-of-the-art methods fall far short of the robustness, reliability and energy consumption of biological vision systems.

Silicon retinas, such as the Dynamic Vision Sensor (DVS) and Asynchronous Time-based Imaging Sensor (ATIS), attempt to replicate some of the benefits of biological retinas and provide a vastly different paradigm in which to sense and process the visual world. Tasks such as tracking and object recognition still require the identification and matching of local visual features, but the detection, extraction and recognition of features requires a fundamentally different approach, and the methods that are commonly applied to conventional imaging are not directly applicable.

This thesis explores methods to detect features in the spatio-temporal information from event-based vision sensors. The nature of features in such data is explored, and methods to determine and detect features are demonstrated. A framework for detecting, tracking, recognising and classifying features is developed and validated using real-world data and event-based variations of existing computer vision datasets and benchmarks.

The results presented in this thesis demonstrate the potential and efficacy of event-based systems. This work provides an in-depth analysis of different event-based methods for object recognition and classification and introduces two feature-based methods. Two learning systems, one event-based and the other iterative, were used to explore the nature and classification ability of these methods. The results demonstrate the viability of event-based classification and the importance and role of motion in event-based feature detection.

# Acknowledgements

The author would like to thank his primary supervisors, André van Schaik, Jonathan Tapson, and Ryad Benosman for their tireless support, constant stream of ideas and inspiration, and for imparting their interest and passion for their fields.

Additionally, the author would like to thank Philip de Chazal for his insights and guidance, Klaus Stiefel for countless fascinating discussions, Sio Leng and Christoph Posch for all their ideas, help, and for assisting the author with navigating the French bureaucratic system and to Francesco Galluppi and Xavier Lagorce for their insights and their friendship.

A special acknowledgement needs to be extended to Garrick Orchard, who put up with a never-ending stream of questions, provided the author with an opportunity to visit his lab in Singapore, and physically saved his laptop more than once. Also, the author would like to thank David Valeiras for his unwavering enthusiasm and for many after-hour discussions in Singapore.

Personally, the author could not have completed this work without the support and friendship of Andrew Wabnitz, David Karpul, Mark Wang, Paul Breen, Gaetano Gargiulo, Tara Hamilton, Saeed Afshar, Patrick Kasi, Chetan Singh Thakur, James Wright and Yossi Buskila.

Finally, the author would like to thank his family for their unwavering support, optimism and encouragement.

# Contents

# List of Figures

# Nomenclature

**Greek Symbols**

$\beta$ Linear Output Weights

$\theta_k$ The inverse of the correlation matrix

$w_{i,j}$ Inputer layer weights

**Acronyms**

AER Address-Event Representation

AIT Austrian Institute of Technology

ATIS Asynchronous Time-based Imaging Sensor

BRIEF Binary Robust Independent Elementary Features

DAVIS Dynamic and Active-Pixel Vision Sensor

DoG Difference of Gaussians

DVS Dynamic Vision Sensor

ELM Extreme Learning Machine

EM Exposure Measurement

FPGA Field Gate Programmable Array

GMM Gaussian Mixture Models

HOG Histogram of Oriented Gradients

LSHDI Linear Solutions to Higher Dimensional Interlayers

NEF Neural Engineering Framework

OPIUM Online Pseudoinverse Update Method

PCA Principle Component Analysis

QVGA Quarter Video Graphics Array

SIFT Scale-Invariant Feature Transform

SKIM Synaptic Kernel Inverse Method

SURF Speeded-up Robust Features

TD Temporal Difference

# Executive Summary

The importance of computer vision in modern technology is continuously increasing, both in terms of usage and the demands placed upon it. The field is also experiencing unprecedented levels of interest and growth, spurred on by large-scale investment in future technologies such as self-driving vehicles, autonomous drones and household robotics. The underlying technologies and hardware requirements for these applications have also experienced rapid growth, often eclipsing the capabilities of current state-of-the-art algorithms and processing techniques. The tasks in computer vision are inherently difficult and challenging problems, and despite the advances in computational power, pixel resolution, and frame rates, even the most sophisticated methods fall far short of the robustness, reliability and energy consumption of biological vision systems.

Silicon retinas, such as the Dynamic Vision Sensor (DVS) and the Asynchronous Time-based Imaging Sensor (ATIS), attempt to replicate some of the benefits of biological retinas and provide a vastly different paradigm in which to sense and process the visual world. Tasks such as tracking and object recognition still require the identification and matching of local visual features, but the detection, extraction and recognition of features requires a fundamentally different approach, and the methods that are commonly applied to conventional imaging are not directly applicable.

The work in this thesis explores methods to detect features in the spatio-temporal information produced by event-based vision sensors and then utilises these features to perform recognition and classification tasks. This work explores the nature of detecting features in event-based data and presents two feature detection methods inspired by concepts and methodologies employed by highly successful conventional computer vision algorithms such as the Scale Invariant Feature Transform (SIFT) and Histograms of Gradients (HOG).

The first feature detector makes use of spatio-temporal gradients to extract features from the event-stream. Applying circular statistics to the orientation data allows for efficient noise removal and feature detection using mixtures models of circular distributions.

The second feature detector operates on surfaces of time, created from the

incoming event-based visual data. Feature extraction makes use of a modified and event-based corner detection approach to identify features of interest, and the use of descriptors allows for invariance to specific affine transformations.

This thesis introduces two new neuromorphic datasets, N-MNIST and N-Caltech101, created from existing and well-established computer vision datasets and intended to address the lack of consistent and well-adopted benchmarks in the neuromorphic community. Based on the MNIST dataset and the Caltech101 datasets, these datasets maintain the structure and methodology of the originals, allowing for comparison and contrast to existing conventional computer vision systems. This thesis provides a thorough analysis of these two datasets and includes benchmarks based on statistical classifiers as a baseline for classifier performance.

The Online PseudoInverse Update Method (OPIUM) and the Synaptic Kernel Inverse Method (SKIM) form the basis of the learning systems used in this work, and the results achieved validate their use in both event-based vision systems and on large neuromorphic datasets.

Additionally, this work makes use of SKIM networks applied to the largest datasets to date, implementing the largest hidden layer sizes and simultaneously training the largest number of output neurons. The success of the classifiers built using these SKIM networks validates both the underlying SKIM algorithm and its applicability to event-based tasks such as those presented in this work. The further work on optimisation of the SKIM network for parallel GPU implementation serves as a first-step toward the faster hardware-based implementations required for real-time operation of such systems.

The work in this thesis deals with the classification of objects from the event-based output of a silicon retina and explores two approaches to performing such classification; one operating on the event-streams directly, and the other making use of extracted features from the event-based data.

Exploring the nature of performing classification on event-based data directly, this work characterises the performance of these systems and achieved accuracies up to 92.87% on the N-MNIST dataset and 11.14% on the N-Caltech101 dataset. A study into training patterns and output determination methods is also presented, along with means and methods for handling non-uniform image sizes.

This thesis also investigates the effects of both spatial and temporal downsampling on event-based vision data, and found that it produces improved classification accuracy. For a given network containing 1000 hidden layer neurons, the spatially downsampled systems achieved a best-case accuracy of 89.38% on N-MNIST as opposed to 81.03% with no downsampling at the same hidden layer size. On the N-Caltech101 dataset, the downsampled system achieved a best case accuracy of 18.25%, compared to 7.43% achieved with no downsampling.

The classification systems based on feature detection introduce an additional

layer to the network, which converts the input from a 2D space defined by the silicon retina, to a feature space defined by the feature detectors. These classification networks address a number of issues relating to network size and the handling of non-uniform image sizes, and produced the best classification results on the N-MNIST dataset, achieving $91.6\% \pm 1.69\%$ with orientation features and an ELM network, and $94.71\% \pm 0.36\%$ with the time surface features and a SKIM network.

Given the performances of the classification systems presented, this work serves to validate the two neuromorphic datasets introduced and provides a range of benchmark accuracies for them. The results also demonstrate the viability of event-based object classification and proves that it can be accomplished in an event-based manner. The work also validates the applicability of SKIM to event-based data, and specifically its applicability to classification on event-based vision data.

Finally, the improved results achieved using the feature detectors and the downsampling indicate the importance of both the temporal and spatial information in the event-streams from a silicon retina, and serve to further demonstrate the benefits to using such devices.

# Chapter 1

# Introduction

## 1.1 Motivation

Computer vision is becoming an increasingly important and active field of study and technologies arising from its applications are finding widespread adoption in both consumer and industrial products which require sophisticated visual systems in order to safely and successfully interact with the world around them.

One of the fundamental tasks underlying much of computer vision is the detection, tracking and recognition of visual features. It is an inherently difficult and challenging problem, and despite the advances in computational power, pixel resolution, and frame rates, even the state-of-the-art methods fall far short of the robustness, reliability and energy consumption of biological vision systems.

Silicon retinas, such as the Dynamic Vision Sensor (DVS) and the Asynchronous Time-based Imaging Sensor (ATIS), attempt to replicate some of the benefits of biological retinas and provide a vastly different paradigm in which to sense and process the visual world. Tasks such as tracking and object recognition still require the identification and matching of local visual features, but the detection, extraction and recognition of features requires a fundamentally different approach, and the methods that are commonly applied to conventional imaging are not directly applicable.

As a result, there exists a need for the development of new algorithms and paradigms in which to handle and process event-based image data. These algorithms and systems also require characterisation and comparison to existing approaches to visual sensing using conventional cameras in order to verify the information encoded into the spatio-temporal information produced by these event-based devices.

## 1.2 Aims

This thesis seeks to explore methods to detect features in the spatio-temporal information from event-based vision sensors, and then to utilise these features to perform recognition and classification tasks.

This work will explore the nature of detecting features in event-based data, and present methods to determine and detect features in manner that preserves the event-based nature of the data. Making use of concepts often employed in conventional computer vision, this work investigates the adaption of these methods to an event-based paradigm.

A framework for detecting, recognising and classifying features will be developed to showcase and demonstrate the efficacy and efficiency of these detectors and to validate their performance using both real-world and simulated data. Additionally, two new spiking neuromorphic datasets, based on existing computer vision datasets, will be introduced and characterised to allow for comparisons to conventional computer vision systems.

## 1.3 Main Contributions of this Work

The main contributions of this thesis are the development of novel feature detection, classification and recognition algorithms for event-based visual tasks. These algorithms and systems operate in an event-based manner, and perform learning and computation in a similar fashion where possible.

Each section in this thesis contains a detailed discussion of the specific contributions made.

## 1.4 Relevant Publications

Parts of this work have already been published, or are currently under review. A list of these works can be found below:

- **Synthesis of neural networks for spatio-temporal spike pattern recognition and processing**
  J. Tapson, G. Cohen, S. Afshar, K. Stiefel, Y. Buskila, R. Wang, T. Hamilton, and A. van Schaik
  Frontiers in Neuroscience, vol. 7, 2013.

  I assisted in the development of the SKIM algorithm, contributing to the writing, the testing and in the production of the results for the paper.

- **ELM solutions for event-based systems**
  J. Tapson, G. Cohen, and A. van Schaik
  Neurocomputing, vol. 149, pp. 435-442, Feb. 2015.

  I produced results for the paper and wrote the implementation of the algorithms used.

- **Converting Static Image Datasets to Spiking Neuromorphic Datasets Using Saccades**
  G. Orchard, A. Jayawant, G. Cohen, and N. Thakor,
  arXiv preprint arXiv:1507.07629 (2015).

  I produced the statistics and the classification results for the all the methods in the paper (with the exception of the HFirst results).

- **Learning Motion Selective Receptive Fields In Spiking Neurons**
  G. Orchard, G. Cohen, N. Thakor, and J. Tapson
  IEEE Transactions on Neural Networks and Learning Systems *(under review)*

  I implemented and produced the results for the SKIM networks used in this paper

- **A Memristor-based Hardware Implementation of Synaptic Kernels for Dendritic Computation**
  J. Burger, G. Cohen, C. Teuscher, and J. Tapson
  ELM2014 Conference, Singapore.

  I contributed the SKIM portion of this paper and integrated the memrister models into it.

## 1.5 Structure of this Thesis

This thesis is structured as follows:

- Chapter 2 presents a detailed literature review covering feature detection in conventional computer vision, the classification systems used in this thesis, and the silicon retina devices used in this work.

- Chapter 3 introduces two feature detectors for use with event-based visual data.

- Chapter 4 explores object recognition and classification using event-based visual data, and showcases two datasets created specifically for benchmarking neuromorphic spike-based systems.
- Chapter 5 extends the object recognition framework introduced in Chapter 3 through the use of the feature detectors to convert the classification task from a 2D spatial classification task to one in a feature-space.
- Chapter 6 provides the conclusions to the work performed in this thesis, and includes a discussion of potential future work.

Appendix A contains a detailed statistical analysis of the N-MNIST and N-Caltech101 datasets. Appendix B describes the optimisation techniques used for the classification systems in this thesis, which allowed for the depth and breadth of the experiments conducted. Appendix C presented additional figures and tables included as supplementary material.

# Chapter 2

# Literature Review

## 2.1 Introduction to Computer Vision

The field of computer vision is currently experiencing an unprecedented level of interest and growth, spurred on by large-scale investments in future technologies such as self-driving vehicles, autonomous drones, household robotics, and even the prolific growth of mobile phones and the cameras within them. The underlying technologies and hardware requirements for these applications have also experienced rapid growth, and have often eclipsed the capabilities of the current state-of-the-art algorithms and processing techniques. Computer vision therefore lies at the crux of a large swath of future technologies, and has the potential to offer solutions to these problems. It is a field made additionally complicated by the dramatic range of applicable contexts and platforms, ranging from low-power mobile devices, to high-precision and life-critical robotic surgery techniques.

One of the fundamental tasks underlying much of computer vision is the detection and tracking of visual features. It is inherently a difficult and challenging problem, and despite the advances in computational power, pixel resolution, and frame rates, even the state-of-the-art methods fall far short of the robustness, reliability and energy consumption of biological vision systems.

At the heart of any visual system lies a device that captures information from the 3D scene and encodes it into a representation fit for further processing, and from which further actions can be taken. In the majority of cases in computer vision, this device is a camera that creates a 2D representation of a 3D scene. It is this transformation, and the loss of information that arises from it, that causes much of the difficulty in visual feature detection and tracking. Other features that complicate the problem include noise, motion (both of objects in the scene and the camera itself), occlusions and illumination changes. The ability to exhibit the appropriate degree of invariance to these issues is a core requirement of both a feature detection system, and a feature tracking system.

The majority of the research into computer vision focuses on using a single camera. Multiple cameras can alleviate some of the issues regarding camera pose and viewpoint issues, but there are also new classes of imaging devices emerging that make use of other methods to create representations of the world around them. These include depth sensors, infrared sensors, and a class of biologically-inspired neuromorphic vision devices often referred to as Dynamic Vision Sensors (DVS).

This review chapter serves to provide the background and context for the work presented in the following chapters. It begins with a discussion of the field of neuromorphic imaging devices which generate event-based visual data and describes the nature of these devices and the means used to represent output data. Finally, the section provides a detailed description of the specific device used in this thesis.

The work in this thesis draws heavily on the existing body of knowledge encompassing the field of computer vision for conventional imaging devices. The field of feature detection on images is well-established and has produced a number of algorithms and techniques that enjoy widespread use, both in research and industry. The second part of this literature review provides a summary of the field of feature detection in computer vision and highlights the important algorithms which form the basis for much of the work in this thesis.

The learning and recognition of features is an important task, and an area in which this work makes a large contribution. The last section of this literature review introduces a class of learning algorithms which form the basis of the recognition systems presented in this work.

## 2.2 Neuromorphic Imaging Devices

The process by which information is processed in neuro-biological systems is vastly different from the mechanisms used by modern computers and devices. Whereas computers use high-speed clock rates, synchronous logic and precisely matched hardware, neuro-biological systems make use of slow signals, inhomogeneous components and make use of noise as a feature in their design. These biological systems outperform even the most sophisticated hardware at a myriad of real-world tasks, whilst consuming only a fraction of the power.

Conventional cameras output visual information in the form of frames at a constant rate. These frames include highly redundant information, as every pixel produces a value for every frame at a fixed frame rate and global exposure time for all pixels. Biological retinas, by contrast, exhibit a far more efficient encoding mechanism, and transmit only relevant information in an asynchronous manner, which leads to far greater information encoding and much lower energy require-

ments [7].

### 2.2.1 Neuromorphic Engineering

Neuromorphic engineering is the term given to the study and development of electronic systems that attempt to replicate and mimic the structure, function and organisation of these neuro-biological systems. Electronic implementations of neural circuits began with the construction of perceptrons in 1958 [8], but it was the combination of the rise of Very Large-Scale Integration (VLSI) and a focus on utilising the non-linear characteristics of transistors that lead to development of the concept of neuromorphic engineering by Carver Mead [9].

Recently, the term *neuromorphic* has come to refer to a wider range of systems that attempt to model the behaviour of neural systems in analogue, digital and mixed-signal systems [10]. The most successful of these systems have focused on emulating a single sensory transduction, for such senses as vision, sound and tactile information. These include the development of silicon cochleas, visual motion sensors and silicon retinas. It is the latter that forms the basis of the motivation for this thesis.

### 2.2.2 Address Event Representation (AER)

Of paramount importance to any information processing system is the mechanism by which information is transmitted and received. This is of particular relevance in attempting to model elements of neuro-biological systems, as they boast a degree of point-to-point connectivity that is not feasible to implement directly with current technologies. Neuromorphic systems are often spike-based, and make use of a spike-based hardware technique known as Address-Event Representation (AER).

AER has become the standard neuromorphic interfacing protocol, specifically for multi-chip neuromorphic systems [11] but also in terms of intra-chip communication in mixed-signal devices, and has proved to be a successful and powerful protocol for simulating large point-to-point connectivity, in which sparse events need to be communicated from multiple sources over a narrow channel [12]. A handshaking protocol allows multiple devices to share a common communication bus, and static timing between events is statistically preserved, due to the asynchronous nature of the event generation and the random order in which events occur [13].

Figure 2.1 shows a simple AER implementation, in which an arbiter on the transmitting device assigns a unique address to each of its neurons, and when a neuron spikes, an event is generated that contains its unique address and any additional information required. The AER protocol then transmits this spike

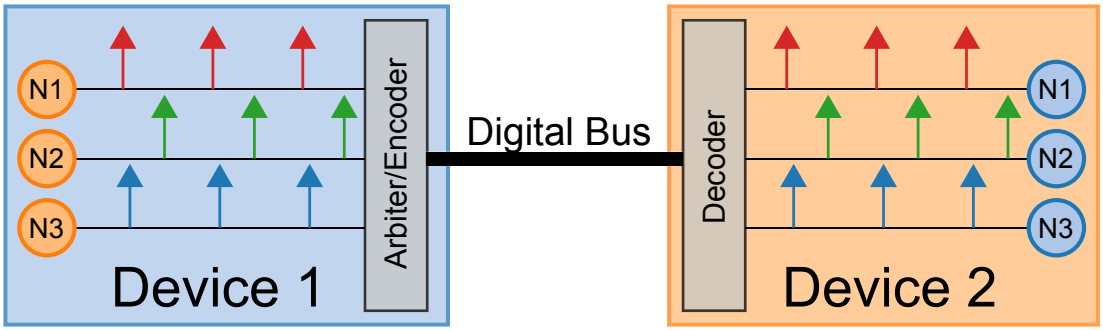



Figure 2.1: **Inter-device communication using Address Event Representation (AER)**. The neurons in Device 1 are assigned a unique address by the arbiter, and when they spike, an event is sent serially down the AER data bus that connects the two devices together. A decoder on the second device reconstructs the spike and makes it available to the target neurons. This figure has been adapted from [1].

serially over a communication bus to a receiver, which decodes the event and selects the appropriate destination. Multiplexing and arbitration only occur when the spike rate of the chip exceeds the transmission capacity of the communication link.

AER has been widely adopted for use in silicon retinas [1, 14, 15, 16, 17], neural processors [18, 19, 20, 21] and in silicon cochleas [22, 23, 24, 25, 26, 27].

### 2.2.3 Silicon Retinas

Although the first electronic models of the retina were developed in the 1970's [28], it was the integrated silicon retina by Mahowald and Mead [29] that represented the first viable imaging device. This was followed by a more biologically realistic implementation from Boahen's group in 2004 [30, 31], which performed better but suffered from large variability in pixel response. An implementation by Ruedi et al. solved a number of issues involving transistor mismatch and poor dynamic range found in previous retinas [32].

There have been a number of other silicon retinas developed. These include a number of retinas that compute and output spatial contrast [14, 33, 34], temporal intensity retinas [35], temporal difference retinas [36, 37], centre of mass detectors for multiple objects [38], a prototype microbolometer [39] and the motion detection system for the Logitech trackball products [40].

The two most relevant devices to this thesis are two of the most recent Temporal Difference (TD) devices, namely the Dynamic Vision Sensor (DVS) [41] and the Asynchronous Time-based Imaging Sensor (ATIS) [2] .

AER is a natural fit for the data arising from silicon retinas, as the address encode the physical $(x, y)$ location of the pixel, and optionally can include some information regarding the illumination falling onto it. Inter-spike intervals and spatial relationships between spike times encode the information about the scene.

In a Temporal Difference (TD) device, each pixel only outputs data in response to a change of illumination [42]. Given a static scene, these pixels will not produce any output and therefore the data-rate from the device is dependent on the activity in the scene. Each pixel emits an AER event containing the physical location of the pixel in the array, and generally a single bit of information to indicate whether the illumination on the pixel increased or decreased.

The following notation is commonly used to represent an event emitted by a TD pixel:

$$\mathbf{e} = [x.y, t, p]^T \tag{2.1}$$

In which the event $\mathbf{e}$ indicates that the pixel located at $\mathbf{u} = [x, y]^T$ on the camera sensor generated a change event in response to an illumination change at time $t$, with the direction of the change in illumination encoded as $p \in [-1, 1]$, in which $p = 1$ is conventionally referred to as an ON event, representing an increase in illumination, and $p = -1$ correspondingly representing an OFF event in which a decrease in illumination occurred.

The output of a TD sensor represents a spatio-temporal image of the changes in illumination in a scene. The temporal resolution is limited by the rate at which events can be read from the physical hardware (usually on the order of microseconds). Unlike conventional cameras, there is no concept of frames, as the data arrives entirely asynchronously.

A second class of devices, known as Exposure Measurement (EM) devices, measure the absolute pixel illumination intensity. Applying the same asynchronous approach to these devices allows each pixel to integrate independently, which greatly improves the dynamic range of the sensor [43], as the minimum integration time is set by the transmission rate of the arbiter, and the maximum integration time is subject only to effects of noise and dark current.

The exposure measurement pixels also operate in a spike-based manner by encoding the relative illumination intensity in the inter-spike interval of two sequential events from the same pixel. The exposure measurement circuit operates using two thresholds applied to the change of illumination on the given pixel. When triggered, the EM circuit begins integrating its photocurrent, emitting a spike when reaching the lower threshold, and a second spike when encountering the upper threshold. This results in an inter-spike time between the two events that is proportional to the illumination falling on the pixel itself.

The major work and contributions of this thesis are in the development of

a feature detection, tracking and recognition framework for use with spatio-temporal data derived from a TD sensor.

### 2.2.4 Asynchronous Time-based Imaging Device (ATIS)

The data and the datasets used in this work make use of a specific DVS sensor known as the Asynchronous Time-based Imaging Device (ATIS), which contains both a TD and EM circuit for every pixel. Other imaging devices, such as the Dynamic and Active-Pixel Vision (DAVIS) camera [44], offer similar capabilities, but were not used in this work.

The ATIS is a CMOS dynamic vision and image sensor developed by the Austrian Institute of Technology (AIT) and draws inspiration from the data-driven nature of biological vision systems [45]. Unlike conventional camera methodologies, which rely on artificially created timing signals that are independent of the source of the visual information [41], biological retinas do not produce frames or pixel values but rather encode visual information in the form of sparse and asynchronous spiking output.

The ATIS sensor offers a QVGA resolution with a $304 \times 240$ array of autonomously operating pixels, which combine an asynchronous level-crossing detector (TD) and an exposure measurement circuit (EM). These pixels do not output voltages or currents, but rather encode their output in the form of asynchronous spikes in the address-event representation (AER) [46].

Figure 2.2 shows the structure and function of an individual pixel and nature of their asynchronous event outputs. The change detection circuit outputs events in response to a change in illumination of a certain magnitude, and is also capable of triggering the exposure measurement circuit, which then generates two events with the absolute instantaneous pixel illumination encoded as the inter-spike timing between them.

This ability to couple the TD and EM circuits results in EM readings generated only in response to changes in the scene, providing a form of hardware level compression which allows for highly efficient video encoding. This is most significant in slowly-changing scenes, where the majority of the illumination on each pixel remains constant.

In addition, the combination of change events from the TD and the two spikes emanating from the exposure measurement circuitry can be combined to produce a quicker gray-scale image approximation by using the inter-spike times between the TD event and the first EM measurement [47]. This can then be later updated using the final EM spike to produce a more accurate result.

As each pixel operates autonomously, there is no need for a global exposure rate and therefore each pixel is able to optimise its integration time independently. This results in a sensor with high dynamic range and improved signal-to-noise

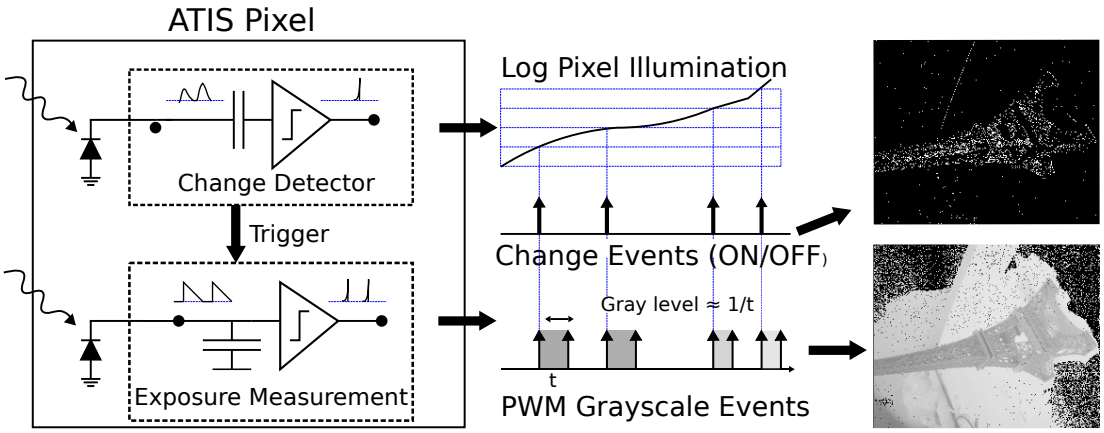



Figure 2.2: **Functional diagram of an ATIS pixel** [2]. Each pixel in the ATIS camera contains both a change detector circuit (CD) and an exposure measurement circuit (EM). The CD circuitry generates events in response to a change in illumination of a certain magnitude (specified by an externally configurable threshold). The CD circuitry can also be configured to trigger the EM circuitry, which encodes the absolute light intensity on the pixel in the inter-spike interval between two spikes. Each pixel operates asynchronously, eliminating the need for a global shutter and is capable of individually encoding change events and events containing illumination information. These events are generated and transmitted asynchronously from each pixel in the sensor.

ratio. The asynchronous nature of the change detection also yields almost ideal temporal redundancy suppression and results in sparse encoding of the visual information in the scene.

The output of these cameras is a spatio-temporal pattern with the location of the pixel generating the event contributing the spatial information ($\mathbf{u}$ from Equation (2.1)), and the time ($t$) at which the pixel generated the event providing the temporal information. The temporal resolution is limited only by the speed at which the sensor can produce events (typically on the order of microseconds [2]).

It is important to note that the ATIS is characterised as supporting a maximum event rate of 30 million events per second [48], yielding a theoretical time resolution of 33 ns. This value represents the smallest possible time measurable between two events and is generally not sustainable. In practise, it is not always possible to read events at the speed at which they occur, and the action of the arbiter may result in events being read in a different order to the order in which they occurred. Temporal event accuracy is generally on the order of microseconds [49], and for that reason, most ATIS hardware limits the temporal resolution to microseconds.

Figure 2.3: **Example output from the ATIS sensor showing the TD output (left) and the EM output (right) for a typical outdoor scene [3].** The scene represents a person walking away from the camera, with the background remaining static. The TD image on the left was constructed by buffering all events over a fixed time period and displaying a black or white dot for pixels which have recently experienced an increase or decrease in intensity respectively. The EM image on the right shows the absolute pixel intensity, which receives pixel value updates whenever the TD circuit detects a change.

An example of the output of the ATIS camera is shown in Figure 2.3. It shows both the output of the TD circuitry (left), and the EM circuitry (right). For the purposes of representation, the TD events generated have been collected over a period of time to form the image shown for the TD output, whereas the EM output consists of absolute intensity levels which have been iteratively updated as the TD events trigger the EM module to obtain measurement of illumination.

It is important to note that the notion of frames is absent from the acquisition process. Data is generated in response to activity in the scene. Frames can be reconstructed, when needed, by buffering the events generated over a given period of time. As a result, the potential frame-rates that can be generated are limited only by the temporal resolution of the ATIS chip itself.

The ATIS camera used in this work consists of an ATIS chip connected to an Opal Kelly XEM-6010 FPGA board. The FPGA interacts with an AER arbiter on the ATIS chip to extract visual events as they occur, and the FPGA applies a time-stamp to them. This preserves the timing between events before they are buffered and transmitted to a PC. Varying levels of processing can be performed on the FPGA itself, such as preliminary noise filtering.

## 2.3 Feature Detection

Feature detection is the process of identifying and describing sections of an image representation for the purposes of identification, tracking or classification. When dealing with conventional cameras, these image representations are frames of illumination intensity, either given as monochrome or colour. Feature detection, in the context of event-based cameras, creates a new representation of the scene, one in which the encoding of information includes a temporal component not present in conventional image representations.

For the purposes of this section, and the rest of this work, feature detection refers to the process of identifying features. Once a feature is detected, an appropriate means of capturing the properties of the feature is required, and this feature representation is referred to as the feature descriptor. Often, the calculation of the descriptor yields a level or degree of invariance toward one or more affine or photometric factors, which can complicate feature detection.

The tasks of feature detection and feature tracking are inherently linked. A good description of a feature is required in order to track or classify it, and often, what constitutes a good feature is one that tracks or classifies well. The quality of a feature is therefore a direct factor in the quality of the tracking and classification of a given system.

The most desirable property of a feature is its ability to be uniquely distinguished in feature space [50], with the choice of feature, and the means of describing it, being of paramount importance. There exist a large number of feature detectors and descriptors that are often specialised to tackle specific tasks or situations as different image features capture different properties of the image. The correct choice of feature is highly dependent on the scene, the context and the system in which it is used.

The development of feature detectors and feature descriptors is an active field of research, and there exist a number of powerful and efficient methods to detect and describe visual features in conventional images. This section explores the most important ideas and methods used in conventional feature detection, and which form the basis of much of the work on event-based vision detection presented in this thesis.

Features that are based on spatial gradients have seemingly provided some of the most robust and widely-used feature descriptors to date. Gradient features can be divided into two major categories; those that extract shapes and contours from gradient information, and those that exploit a statistical representation of the underlying gradients. Both types of methods are described below.

Human detection in images has been one of the driving forces behind the use of gradient features in computer vision. This is due primarily to the widespread need for a reliable means of human detection in autonomous vehicles, surveillance

systems and robotic systems. There are also well established and labelled datasets available, allowing performance metrics to be easily calculated.

The feature detection methods described in this section represent the most successful and widely-used methods in computer vision with conventional cameras. As a result, these methods operate on static frames, and not event-based data. Much of the work presented in this thesis represents the implementations of similar feature detection methods to the event-based data streams from the silicon retina introduced in Section 2.2.3.

### 2.3.1 Shape-based and Contour-based Features

Template, shape and contour matching techniques attempt to identify, detect and classify objects and features through unique edges and silhouettes. These techniques operate either at a global level, which might attempt to identify a human silhouette through matching a full body template, or at a local level, in which a human form may be detected by identifying a number of sub-parts to make up the whole. Global methods have the advantage of easily being able to detect multiple objects in one scene, but lack the robustness to occlusions and cluttered scenes that part-based methods can offer.

These gradient-based techniques have been applied in numerous different and varied scenarios, often combining multiple techniques or methods. For example, Garvrila tackled the problem of identifying pedestrians from a moving vehicle through the use of hierarchical template matching using a Chamfer detector to find proper contours [51]. Lin et al. also used a hierarchical approach to human segmentation, but combined a global template and local parts approach, and additionally formulated the task as a Bayesian Maximum Posteriori (MAP) optimisation problem [52].

Another interesting example is the work of Ferrari et al., who tackled the problem of object detection in cluttered environments by extracting edges and fitting contours, and using a map of their interconnects to perform detection [53]. Elements of this technique led to work of Anvaripour and Ebrahimnezhad, who constructed exact object boundaries for object detection, using boundary fragment extraction and using Gaussian Mixture Models (GMM) [54]. Wu and Nevatia extended the concept further by introduced edgelet features, which are short segments of a line or curve and used these in a joint-likelihood model, also formulated as a Maximum Posteriori probability (MAP) problem [55].

These approaches which make use of shapes or contours are particularly interesting in the context of event-based vision. Silicon retinas, such as the ATIS, perform an operation similar in nature to edge extraction for scenes with motion, ego-motion or photometric changes, but performs the computation at the hardware level and asynchronously. This creates the possibility of applying and

extending these techniques into an event-based paradigm.

### 2.3.2 Scale-Invariant Feature Transform (SIFT)

The Scale-Invariant Feature Transform (SIFT) is perhaps one of the most well-known and widely-used feature detection methods. SIFT is a cascaded filter chain, which includes a scale-invariant detector and a rotationally invariant descriptor [56], and has proved to be effective and efficient across a wide variety of applications.

The SIFT detector belongs to a class of feature detectors that look for regions within an image that exhibit the properties of a good features - namely one that is either useful for recognition, classification or tracking purposes. The Moravec corner detector represents one of the first attempts to detect such points through identifying points of local maximum or minimum intensity changes. The Harris Corner Detector [57] solved some of the issues surrounding sensitivity to noise and edges that afflicted the Moravec detector.

The Harris Corner Detector proved to be a better means of identifying points of interest within an image, but lacked the ability to handle changes in scale or viewpoint. SIFT tackles this problem directly through the use of scale-space theory, and more specifically linear scale-space representation [58], allowing it to detect features across a wide range of scales. The SIFT detector approximates the scale-space through a technique known as Difference-of-Gaussians (DoG), and uses a $2 \times 2$ Hessian matrix of image gradients about each point in order to remove edge responses.

The SIFT detector identifies a set of keypoints for a given image, and then calculates a descriptor for each one. The descriptor needs to provide a consistent and reliable means of recognising the same feature in a different image or location, and the SIFT descriptor provides a rotationally-invariant and partially illumination-invariant descriptor calculated through the use of local orientation histograms, which are re-aligned to the dominant orientation.

The success of SIFT has led to numerous improvements and modifications to the algorithm. Sukthankar made use of Principle Component Analysis (PCA) in order to reduce the size of the SIFT descriptors [59], Abdel-Hakim and Farag created a colour invariant version called CSIFT [60], and Scovanner et. al extended SIFT to handle 3D features [61].

SIFT has also given rise to a whole class of similar feature detection methods based on similar principles. These include the Speeded Up Robust Features (SURF), which makes use of a fast Hessian detector based on a discretised Gaussian filter [62], an efficient binary-based descriptor called Binary Robust Independent Elementary Features (BRIEF) [63], a rotationally invariant version of BRIEF called ORB [64] and a descriptor based on Weber's Law called WLD [65].

The methods of detecting points of interest first described by Moravec, and then later refined by the Harris Corner Detector and SIFT, are used in the development of similar techniques for detecting features on event-based time surfaces in Section 3.3.

### 2.3.3 Histograms of Oriented Gradients (HOG)

Another important feature detection method also makes use of histograms of orientations, and applies it a recognition task. Dalal and Trigs introduced the Histograms of Oriented Gradients (HOG) method, which makes use of dense and overlapping grids of local and normalised histograms of orientations as an input to a pedestrian recognition system [66].

These histograms of oriented gradients are the same as those used in SIFT, and the HOG approach showcases the power and simplicity of just using those histograms as opposed to the entire cascading filtering chain. An important point raised by Dalal and Triggs is that the distribution of edge directions in a region containing an object provides a good characterisation of its appearance [67].

Dalal found that the simplest methods of calculating the orientation gradients produced better results. The best performing method made use of the simple intermediate difference gradient formula as follows:

$$\frac{\partial F}{\partial x} = \frac{F(x+1, y) - F(x-1, y)}{2} \tag{2.2}$$

$$\frac{\partial F}{\partial y} = \frac{F(x, y+1) - F(x, y-1)}{2} \tag{2.3}$$

Using methods of calculating the spatial gradient that made use of larger areas tended to decrease performance, as did performing any smoothing prior to computing gradients. The HOG descriptor normalises the local histograms, as it proved to be essential to achieving good performance in their recognition tasks.

The paper introducing the HOG descriptor also explored the nature of the spatial binning for the histograms. SIFT made use of the full range of orientations $[0, 360°)$, but the HOG method chose to discard the sign of the gradient and use a range of $[0, 180°)$, as the use of the full range caused a decrease in performance. The authors do note that the use of the signed orientations did serve to improve accuracy when dealing with non-human objects.

These results are particularly important as they formed the basis for the design and configuration of the feature detectors based on temporal orientations in Section 3.4.

## 2.4 Neuromorphic Approaches

The recent advances made in artificial visual sensing have occurred primarily in three areas. The first relates to the development of silicon retinae hardware, and more specifically the development of applications and algorithms designed specifically to work with them.

The second area in which there has been significant research in recent years is the creation of large-scale neuromorphic hardware platforms. These systems represent viable hardware on which event-based and spike-based algorithms can be implemented.

The third area of research involves the development of spike-based learning methods and systems. These including algorithms and techniques for learning spatio-temporal patterns, and software frameworks for implementing spike-based computation.

This section presents a short summary of the recent advancements in these three areas.

### 2.4.1 Event-Based Visual Algorithms

Event-based cameras, such as the ATIS and DVS devices, have been the subject of active research for a number of years, and a number of algorithms and applications have emerged that make use of the event-based paradigm offered by these devices. This section describes some of the recent vision-specific algorithms and applications relevant to the work presented in this thesis.

Working with the data directly, silicon retinae have been used in a number of hardware systems making use of the unique nature of the event-based output. These include visuo-motor control in a humanoid robot [68], a pole-balancing robot using an embedded DVS camera [69], a robot goalie system [70], and as a device for fall detection in the elderly [71].

The event-based nature of these cameras also allows for an interesting approach to stereo vision processing, as temporal coincidence can be used to match events from different cameras. An event-based approach to epipolar geometry was proposed by Bensosman et al. [72], and developed into full 3D reconstruction techniques for the DVS camera [73, 74] and for the ATIS camera [75].

Techniques for calculating local optical flow in an event-based manner have also been proposed [76, 77] and for motion estimation [78]. The devices have also found applications in micro-particle tracking [79] and in providing stable haptic feedback in micro-robotics [80]

Two algorithms of particular relevance to this work are the part-driven shape tracking by Valeiras et al. [81] and the multi-kernel tracking system by Lagorce et al. [82]. Both algorithms are event-based systems capable of operating in real-

time on the output from an ATIS device, and both embody a similar approach to event-based visual processing as adopted in this work. Haeng et al. [83] implemented a similar real-time gesture interface using a pair of DVS cameras.

Another work of particular importance is that of Kim et al. [84] in which a DVS camera was used to provide simultaneous localisation and mosaicking. The system produced excellent results, but the camera motion was limited to pure rotation. This work represents an important step toward the implementation of an event-based simultaneous localisation and mapping algorithm.

Perhaps the most closely related work is that of Lagorce et al. [85], who introduced a framework for detecting spatio-temporal features in event-based data using recurrent neural networks and a winner-take-all architecture. Trained and tested on similar digits to the work presented in this thesis and operating in an unsupervised manner, the approach presented in this work represent a parallel and promising approach to the event-based visual processing tasks.

### 2.4.2 Neuromorphic Hardware Systems

There have been a number of significant advances in the design and development of large-scale biology-inspired spiking neural hardware platforms. These hardware devices compute in a power efficient manner inspired by neurons in the brain and can be used to process the captured visual signals from neuromorphic devices such as silicon retinae.

SpiNNaker [86] is one such system, and comprises specially designed processing and communication hardware developed to simulate large populations of spiking neurons in a massively parallel manner, and at relatively low power compared to existing systems for simulation. A number of systems have been developed for the SpiNNaker hardware, including a framework for plasticity [87] and an implementation of Deep Belief Networks [88].

Another such system is Neurogrid [89], which augments the purely digital approach of SpiNNaker with the use of mixed signal hardware for neuron simulation. Implementing systems up to a million neurons and eight billion synapses, the system proved to be power efficient, consuming just 941 pJ per synaptic activation.

The TrueNorth processor from IBM [90] is a commercial device and is capable of implementing one million neurons and 256 million synapses in real time. It is also reported to consume less than 100 mW under normal operating conditions. The TrueNorth system comprises custom-built hardware and an extensive software development framework with which to develop applications for the system.

Another class of hardware implementations make use of the concept of polychronous networks, first introduced by Izhikevich [91]. Wang et al. [92] produced a large-scale hardware implementations of such networks on an FPGA. Such net-

works are also capable of learning specific spike-timing and are also applicable for use in an event-based vision system.

The work done by Basu et al. on hardware implementations of the Extreme Learning Machine (ELM) using silicon spiking neurons [93] is of particular relevance to this work, as many of the algorithms make use of classifiers similar in structure, and the work represents a promising hardware platform on which to implement the algorithms and feature detectors presented in this work.

### 2.4.3 Spiking Neural Networks

In addition to the hardware implementations, the development of algorithms and neural networks that operate on spikes have also received much attention. One such network is the Tempotron [94], which is capable of exploiting information in the spatio-temporal arrangement of spike patterns rather than the mean firing rate. The DELTRON, which is based on a similar learning rule to the Tempotron, represents a viable hardware implementation using FPGAs. In both these networks, the weights for the network are incrementally learnt.

The Synaptic Kernel Inverse Method (SKIM) represents an alternative method in which the network weights are synthesised rather than iteratively calculated. This makes it very useful in characterising and contrasting the other components and extractors used in a visual recognition system. As the SKIM algorithm is used throughout this work, it is covered in more detail in Section 2.5.3.

Additional methods exist for the recognition of the inter-spike timing in spatio-temporal patterns. Masquelier and Thorpe demonstrated the sensitivity of Spike Timing Dependent Plasticity (STDP) to recognising spatio-temporal patterns [95]. Other methods include the Remotely Supervised Method (ReSuMe) [96] and SPAN [97].

The development of multi-layer deep learning approaches also present exciting opportunities for neuromorphic applications. O'Connor et al. provided a Deep Belief Network capable of performing sensor fusion on both audio and visual information. [98]. Carrasco et al. provided a means of processing event-based visual information using a cascade of convolution processors [5], further validating the efficacy of the deep learning approach. Neftci et al. proposed an event-based implementation of a Restricted Boltzmann Machines suitable for such deep belief systems [99], with the work of Pedroni et al. further expanding on a neuromorphic implementation.

In parallel to the growth of hardware implementations, a similar increase in the number of available and viable software frameworks has also emerged. These include the Neural Engineering Framework (NEF) [100], Brian [101], PyNCS [102] and PyNN [103].

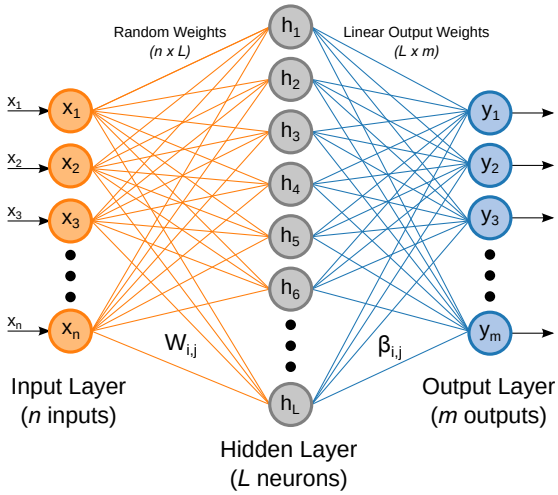



Figure 2.4: **Structure of a typical 3-Layer LSHDI Network**. The inputs are projected through a set of random weights to a larger hidden layer of neurons, each implementing a non-linear activation function. The outputs of these neurons are linearly weighted and summed for each output neuron.

## 2.5 Linear Solutions to Higher Dimensional Interlayers (LSHDI)

The term Linear Solutions to Higher Dimensional Interlayers (LSHDI) [4] refers to a class of neural networks that resemble a three-layer perceptron, except that they possess a large number of hidden layer nodes. The input weights, which are often random, project the input into a higher dimensional space with the intention of spreading out the inputs and allowing for the fitting of a hyperplane for either regression or classification tasks, resulting in the need to compute only linear output weights for each output neuron.

Examples of these techniques include the Neural Engineering Framework (NEF) [100] and the Extreme Learning Machine [104] . Using the analytical approach has the advantages of requiring no parameters and of producing an optimal least squares solution in a single step.

### 2.5.1 Structure of an LSHDI Network

The structure of a typical LSHDI network closely resembles a standard three-layer perceptron, as shown in Figure 2.4. These networks consist of an input layer, a large hidden layer and then an output layer, all of which can vary in size according to the nature and requirements of the task.

LSHDI networks typically possess a larger hidden layer than most standard

perceptron implementations, and it is often set as a multiple of the number of input layer neurons used, with this multiple referred to as the fan-out factor. As mentioned before, a large hidden layer size is often used and a fan-out of 10 is common in ELM systems [105].

Another common property of an LSHDI network is that fixed weights project the input to the hidden layer $\mathbf{w}_{i,j}$, and are not altered or changed during the training. This greatly simplifies the calculation of the analytical solution, as only the output weights $\beta$ require calculation. It is common to use random values for the input layer weights, as is the case for ELM implementations [104]. The random hidden layer weights serve to project the input into the higher dimensional space of the hidden layer, in a process similar to the *kernel trick* used in the majority of kernel methods. Each hidden layer neuron implements a non-linear activation function, such as a sigmoid.

For a given ELM network presented with a set of inputs $x_l \in \mathbb{R}^{n\times 1}$, the output $y_m$ representing the output value at output neuron $m$ is determined as follows:

$$y_m = \sum_{j=1}^{M} \beta_{mj}\ g\left(\sum_{i=1}^{L} w_{i,j}x_l\right) \tag{2.4}$$

In which $w_{i,j}$ represents the random input weight from the input layer to the hidden layer and $g$ represents a non-linear function implemented at each hidden layer neuron. The outputs from each hidden layer neuron project to the output neurons through the output layer weights $\beta$. The values for $\beta$ can be determined using a number of different methods, including back-propagation, but LSHDI networks take an analytical approach to determining these weights through the Moore-Penrose pseudoinverse.

Given N training sample $(\mathbf{x_i}, \mathbf{y_i})$ such that the inputs $\mathbf{x}_i[x_{i1}, x_{i2}, ..., x_{in}]^T \in \mathbb{R}^n$, and the outputs $\mathbf{y}_i = [y_{i1}, y_{i2}, ..., y_{im}]^T \in \mathbb{R}^m$, the equations for $\mathbf{N}$ using Equation (2.4) can be represented using the following compact notation:

$$\mathbf{H}\beta = \mathbf{Y} \tag{2.5}$$

In which $\beta$ contains the output weights, $T$ is the input pattern set and $\mathbf{H}$ represents the outputs from the hidden layer as follows:

$$\mathbf{H}(w_1, w_2, ..., w_n, x_1, x_2, ...x_n) = \begin{bmatrix} g(w_1 \cdot x_1) & g(w_1 \cdot x_2) & \cdots & g(w_1 \cdot x_n) \\ g(w_2 \cdot x_1) & g(w_2 \cdot x_2) & \cdots & g(w_2 \cdot x_n) \\ \vdots & \vdots & \ddots & \vdots \\ g(w_l \cdot x_1) & g(w_l \cdot x_1) & \cdots & g(w_l \cdot x_n) \end{bmatrix} \tag{2.6}$$

The definition of the $\mathbf{H}$ matrix given by Huang [106] differs as it includes a bias term $\mathbf{b} = [b_1, b_2, ..., b_N]^T$ which is added after the input weight summation

for each hidden layer neuron. Huang et al. [107] demonstrated that if the random weights (and the bias term if used) remain constant, then training the network is equivalent to finding the least-squares solution $\hat{\beta}$ to the linear system presented in Equation (2.5), and can be calculated as follows using the Moore-Penrose Generalised Inverse:

$$\hat{\beta} = \mathbf{H}^{\dagger}\mathbf{Y} \tag{2.7}$$

Where $\mathbf{H}^{\dagger}$ is the Moore-Penrose generalised inverse of $\mathbf{H}$. Calculating this inverse becomes a limiting factor in the application of LSHDI techniques, as it is a large and memory intensive operation. The scale of the inverse also grows with the number of training samples, thereby placing a limit on that as well.

### 2.5.2 The Online Pseudoinverse Update Method

The Online Pseudoinverse Update Method (OPIUM) [105] is an incremental method for solving the pseudoinverse for the type of LSHDI systems detailed in the previous section. Whereas the addition of a new training item requires the complete retraining of a conventional ELM system, the OPIUM method allows for online updates. In addition, the ability to iteratively update the network allows for the handling of non-stationary data.

The OPIUM method tackles the problem of calculating the Moore-Penrose pseudoinverse $\mathbf{H}^{\dagger}$ by making use of Greville's algorithm for iteratively calculating the psuedoinverse [108].

The OPIUM method provides a means to update an LSHDI network for data from a process sampled $k$ times, with a new set of input and output data, labelled $x_k$ and $y_k$ respectively. From $x_k$, the hidden layer activations $h_k$ can be calculated. Thus, two vectors can be formed as follows:

$$H_k = [H_{k-1}h_k] \tag{2.8}$$
$$Y_k = [Y_{k-1}y_k] \tag{2.9}$$

The OPIUM method requires the calculation of the inverse of the correlation matrix of the hidden layer activations, denoted as $\theta_k \in \mathbb{R}^{L\times L}$, and which, under certain conditions discussed in [105], can be updated using the following rule:

$$\theta_k = \theta_{k-1} - \theta_{k-1}h_k b_k^T \tag{2.10}$$

Where $b_k$ is calculated as follows:

$$b_k = \frac{\theta_{k-1}h_k}{1 + h_k^T\theta_{k-1}h_k} \tag{2.11}$$

The linear output weights, denoted by $\beta$ also make use of $b_k$ for its update equation, which is shown below:

$$\beta_k = \beta_{k-1} + (y_k - \beta_{k-1} h_k) b_k^T \tag{2.12}$$

The two update equations form the basis of the mechanism for iterative learning, and require calculating for each new and incoming input/output pair. Each step requires only the new data, the previous output weights $\beta_{k-1}$, and the inverse correlation matrix $\theta_{k-1}$.

As the OPIUM method is inherently an iterative method, it is important that the initial conditions for $\beta$ and $\theta_k$ be set appropriately. The initial values suggested in [105] suggest that $\beta_0 = 0^{n \times L}$ and $\theta_0 = \frac{1}{\epsilon^2} I^{L \times L}$.

The nature of these initial conditions and the effects in the under-determined region are explored in Section 6.5.

### 2.5.3 The Synaptic Kernel Inverse Method

The Synaptic Kernel Inverse Method (SKIM), proposed and outlined in [4], is a neural synthesis technique which produces networks of neurons and synapses that are capable of implementing arbitrary functions on spike-based inputs. The network generally contains a single input neuron for each input, and a single neuron for each desired output. The conventional fan-out to a higher dimensional space, present in most Linear Solutions to Higher Dimensional Interlayer (LSHDI) network systems [4] and usually implemented through a hidden layer of neurons, is replaced with multiple synaptic connections, which are shared between output neurons.

SKIM differs from other LSHDI systems, such as the Extreme Learning Machine (ELM), in that it is specifically designed to learn spike-dependent signals. It therefore bears a closer resemblance to synthesis methods such as the Neural Engineering Framework (NEF), which is also capable of spike-based input-output relationships [100]. SKIM differs from the Neural Engineering Framework in that it does not rely on rate-encoded signals, and rather relies on both the spatial and temporal information in the incoming spike-trains.

SKIM is based on a biologically plausible network structure modelled on the synaptic connections between neurons. An overview of the SKIM network is shown in Figure 2.5. In the SKIM, input neurons are considered analogous to pre-synaptic neurons and input event streams are projected to a layer of synapses through a set of random weights. Each weight is representative of a specific dendritic branch leading towards a synapse. These synapses implement non-linear responses to the received current from the pre-synpatic dendritic branches through the use of non-linear kernels, such as exponentials or decaying-alpha

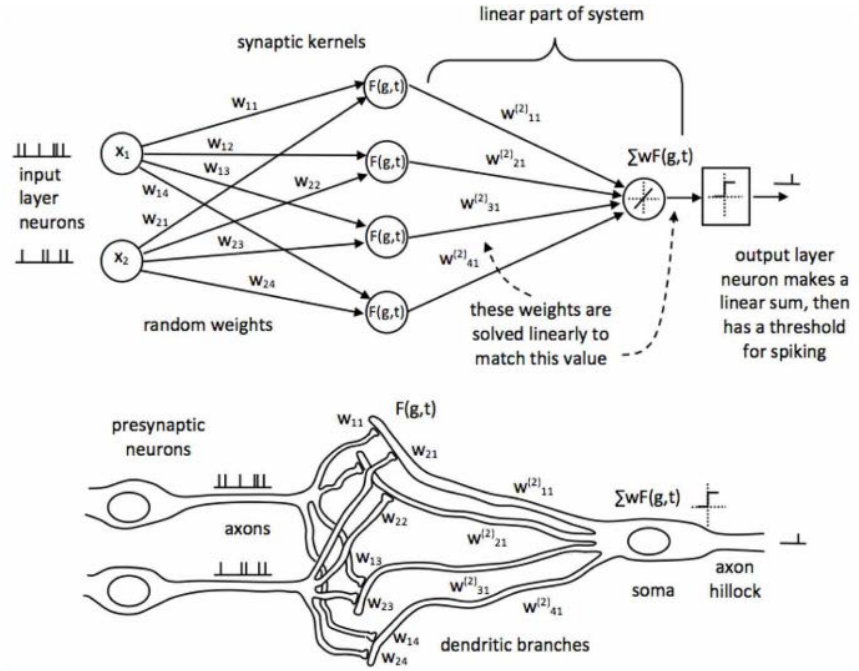



Figure 2.5: **Topology of the Synaptic Kernel Inverse Method (SKIM)** [4] . Each input layer neuron (left) will be an input from a separate pixel. At initialisation, the static random weights and synaptic kernels are randomly assigned and they remain fixed throughout the learning process. During learning, the output weights in the linear part of the system are solved to minimise the error between the system output and the desired output.

functions. It is these kernel functions that provide the SKIM with the ability to respond to temporal information in the input as they convert discrete incoming spikes into a continuous value.

The outputs of these synapses proceed down the post-synaptic dendritic branches, which connect to the soma of the output neurons. The dendritic branches sum the currents from the synapses at the soma of the output neuron, causing it to fire if the soma potential exceeds a specified threshold. It is the properties of these post-synaptic dendritic branches which are analytically calculated in the SKIM method as they are analogous to the linear output weights in other similar systems.

# Chapter 3

# Event-Based Feature Detection

## 3.1 Introduction

The ability to reliably detect and extract distinct and robust local features in a visual scene is a fundamental building block in a wide range of computer vision problems. Much of the work in the field has focused on the detection and matching of features in 2D images, and has been used for such tasks as object recognition [56], image stitching [109] and scene classification [110]. These techniques have also been applied to find stable features in video streams [111], and have been used in off-line tasks such as feature detection and video analysis [112], and for more demanding on-line tasks such as camera localisation and visual mapping [113].

Event-based cameras provide a vastly different paradigm in which to sense and process the visual world. Tasks such as tracking and object recognition still require the identification and matching of local visual features, but the detection and extraction of features requires a fundamentally different approach, and the methods that are commonly applied to conventional imaging are not directly applicable to event-based data.

This section introduces a number of methods for detecting and extracting features from the spatio-temporal data generated from an event-based camera.

## 3.2 Contributions

The field of event-based vision is relatively new when compared to its conventional counterpart, and therefore there is little existing work on the subject of event-based feature detection. This section introduces a number of new techniques and makes the following contributions, as it:

- Explores the use of time surfaces and descriptors based on time surfaces for use in feature tracking and recognition problems.
- Introduces a new type of feature based on orientations of spatio-temporal gradients, and outlines a method to select good features based on a mixture model of von Mises distributions.
- Applies the principles of circular statistics and uniformity to create a new class of noise filter for event-based data.

## 3.3 Feature Detection using Surfaces of Time

This section describes a method of detecting features on surfaces derived from the spatio-temporal information produced from the event-based cameras introduced in Section 2.2.4. This section discusses different methods of creating these surfaces from the event-streams, the means to detect features of interest and the methods to calculate unique and robust descriptors from these features.

Figure 3.1 shows an overview of the process used to generate a spatio-temporal time surface and feature for a given event. An event source, such as an ATIS camera, generates a stream of events which include both a temporal and spatial component. These events update a spatio-temporal surface, as shown in Figure 3.1 (c) and discussed in depth in Section 3.3.1.

From this time surface, an area around the given event is extracted and considered as a potential feature through a method detailed in Section 3.3.2, and from which an appropriate descriptor is calculated. The process of calculating descriptors and the transformations to which it can be made invariant are described in Section 3.3.3.

### 3.3.1 Surfaces of Time

In the context of computer vision, a local feature is generally defined as an image patch that exhibits one or more properties that differentiates itself from its immediate neighbourhood [114]. Implicit in this definition is the assumption of a conventional imaging system where the entire frame of pixel information is already available.

The data obtained from an event-based camera includes a temporal component, and the definition of a local feature in the context of event-based feature detection may better be defined as an identifiable spatio-temporal pattern representing an aspect of the visual scene, distinct in the local spatial and temporal context, and robust to geometric and temporal transformations.

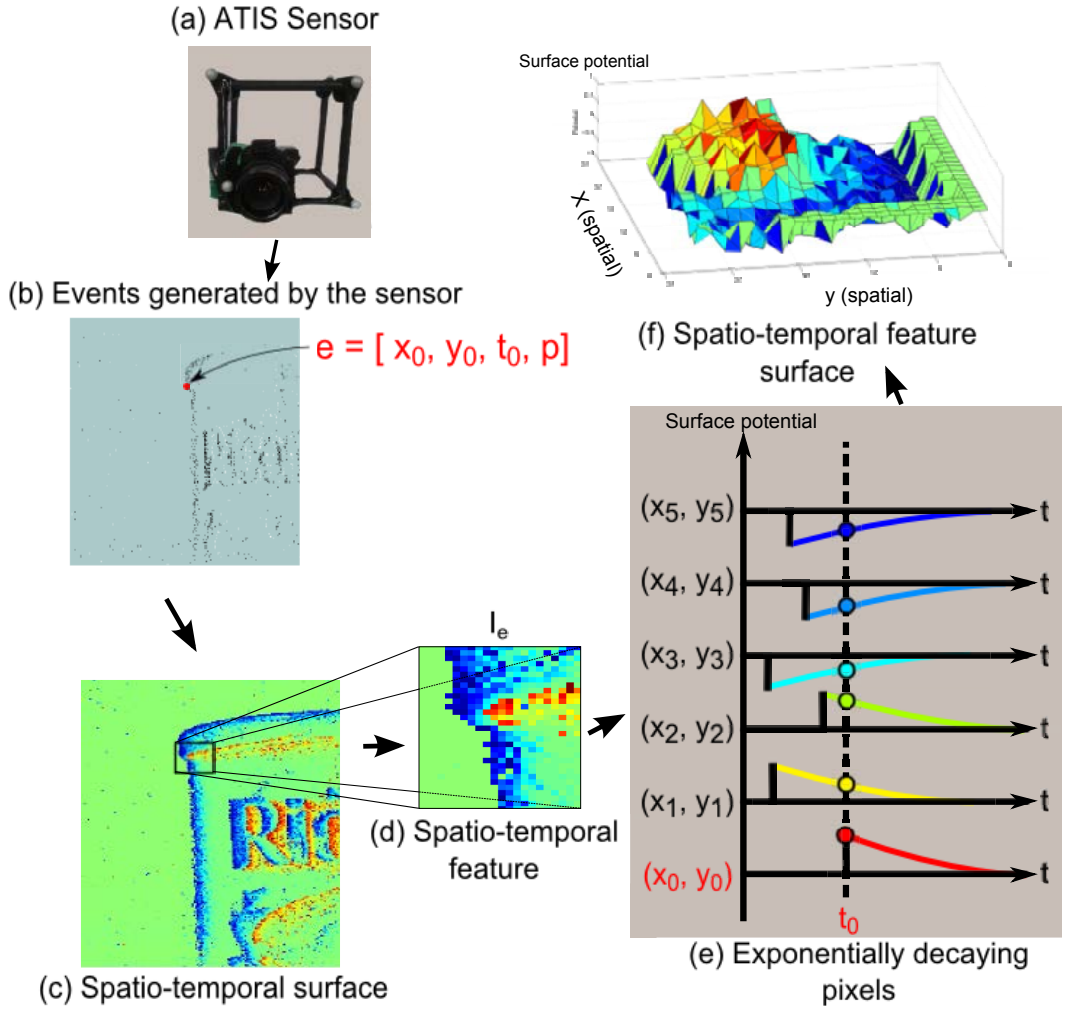



Figure 3.1: **Construction of a spatio-temporal time surface and features from the event-based data**. The time surface encodes the temporal information from the sensor into the magnitude of the surface as each event arrives. The ATIS sensor shown in (a) and attached to a rig with fiducial markers for 3D tracking, generates a stream of events (b) as the camera is moved across a static scene. When an event occurs, such as event $e$ from the pixel $(x_0, y_0)$ at time $t_0$, the value of the surface at that point is changed to reflect the polarity $p \rightarrow \{-1, 1\}$. This value then decays exponentially as $t$ increases, yielding the spatio-temporal surface shown in (c). The intensity of the colour on the surface encodes the time elapsed since the arrival of the last event from that pixel. A spatio-temporal feature consists of a $w \times w$ window centred on the event $e$, as shown in (d). An example of the decaying exponentials in the feature that generate the time surface at $t = t_0$ is shown in (e). The spatio-temporal feature can also be represented as a surface, as shown in (f).

Any method of detecting features or patterns in spatio-temporal data requires some form of memory through which previous events are considered when updating or calculating features. This is of particular importance in event-based processing, where updates occur on the arrival of new data as opposed to occurring at regular intervals. The irregular nature of the data rate makes implementations that directly buffer event information difficult, and therefore motivates the need for functions that are capable of mapping the event-based data onto a time-dependent surface.

Defining a feature in the context of a spatio-temporal pattern requires some additional definitions relating to the processing of event-based visual data. A visual event from the ATIS camera can be mathematically defined as a vector with four components:

$$e(\mathbf{u}, t) = [\mathbf{u}, t, p]^T \tag{3.1}$$

Where $\mathbf{u} = (x, y)^T$ is the 2D spatial coordinate of the event on the silicon retina, $t$ is the absolute time at which the event occurred and $p \in \{-1, 1\}$ encodes the polarity of the event. Motion or change of illumination in the scene will cause the pixels in the ATIS camera to generate a stream of vectors, each corresponding to a relative luminance change in the scene.

We can then define the function $\Sigma_e$ to map a time $t$ to each 2D spatial coordinate $\mathbf{u}$:

$$\begin{aligned} \Sigma_e : &\quad \mathbb{R}^2 \to \mathbb{R} \\ \mathbf{u} : &\quad t = \Sigma_e(\mathbf{u}) \end{aligned} \tag{3.2}$$

And a similar function $P_e$ to map the polarity to each spatial coordinate:

$$\begin{aligned} P_e : &\quad \mathbb{R} \to \{-1, 1\} \\ \mathbf{u} : &\quad p = P_e(\mathbf{u}) \end{aligned} \tag{3.3}$$

As time is inherently a monotonically increasing function, the function $\Sigma_e$ defined in (3.2) describes a monotonically increasing surface. Making use of this surface does not yield much benefit, as it is just another representation of the temporal component from each event. Applying a function to this surface generates other surfaces from which more descriptive features that better represent the underlying visual scene can be extracted. Surfaces derived from $\Sigma_e$ will be referred to as $\Omega_e$, and three such surfaces are discussed below.

One candidate surface, the linear-decaying time surface $\Lambda_e(\mathbf{u}, t)$, is defined

below:

$$\Lambda_e(\mathbf{u},t) = \begin{cases} P_e(\mathbf{u}).\left(1+\frac{\Sigma_e(\mathbf{u})-t}{\tau}\right), & \Sigma_e(\mathbf{u})-t >= \tau \\ 0, & \Sigma_e(\mathbf{u})-t < \tau \end{cases} \tag{3.4}$$

The above equation utilises the two functions defined in equations (3.2) and (3.3) to create a linear decay for each event based on a time-constant defined by the variable $\tau$ in the equation. The $\tau$ parameter defines the duration over which an event will have an effect on the surface and therefore is responsible for implementing the required memory aspect needed for spatio-temporal pattern identification. The actual value for $\tau$ is highly dependent on the application and nature of the scene and will be specified whenever such a surface is used.

A second candidate surface extends the concept of the linear decaying surface by implementing an exponential decay rather than a linear decay. The exponentially decaying time surface $\Gamma_e(\mathbf{u},t)$ is defined as follows:

$$\Gamma_e(\mathbf{u},t) = \begin{cases} P_e(\mathbf{u}).\mathrm{e}^{\left(\frac{\Sigma_e(\mathbf{u})-t}{\tau}\right)}, & \Sigma_e(\mathbf{u}) <= t \\ 0, & \Sigma_e(\mathbf{u}) > t \end{cases} \tag{3.5}$$

As with the previous surface, the $\tau$ constant in the equation determines the duration over which events have impact on the scene. Figure 3.1 illustrates the exponential time surface and the manner in which the surface is calculated from incoming events. It also shows how a snapshot of the decaying exponential activations from previous events contributes to the surface at time $t$. The length of the decay for each event is set by the *tau* parameter.

A third surface further extends the exponentially decaying time surface by incorporating the residual value of the previous decaying exponential into the update for each event. This accumulating temporal surface $\Phi_e(\mathbf{u},t)$ is defined as follows:

$$\Phi_e(\mathbf{u},t) = \begin{cases} (\Phi_e(\mathbf{u},\tau_e) + P_e(\mathbf{u})).\mathrm{e}^{\left(\frac{\Sigma_e(\mathbf{u})-t}{\tau}\right)}, & \Sigma_e(\mathbf{u}) <= t \\ 0, & \Sigma_e(\mathbf{u}) > t \end{cases} \tag{3.6}$$

Where $\tau_e$ is the time of the arrival of the last event from any pixel. Whereas the ranges of both the linear surface $\Lambda_e$ and the exponential surface $\Gamma_e(\mathbf{u},t)$ are limited to $[-1,1]$ as $\Sigma_e <= t$, the range of the accumulating surface $\Phi_e$ is unbounded.

Figure 3.2 illustrates the nature of an update to a single pixel for each type of surface. If a pixel receives another event whilst the corresponding value for that pixel on either the linear or exponential surfaces is still decaying, the value on the time surface will revert to either $+1$ or $-1$ depending on the polarity of the new event, discarding the previous value. The inclusion of the residual from the

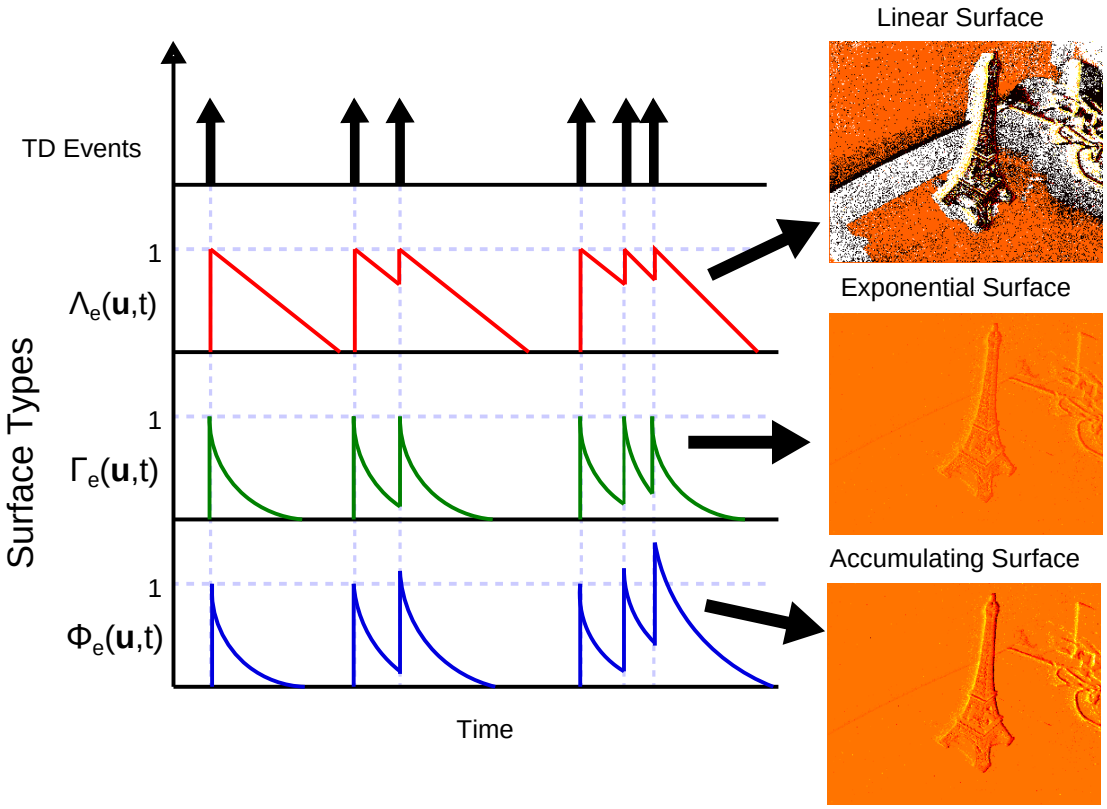



Figure 3.2: **Linear, Exponential and Accumulating Surfaces of Time**. Illustration of the updates for a single pixel on the linear surface $\Lambda_e$ (top), the exponentially-decaying surface $\Gamma_e$ (middle) and the accumulating exponential time surface $\Phi_e$ (bottom).

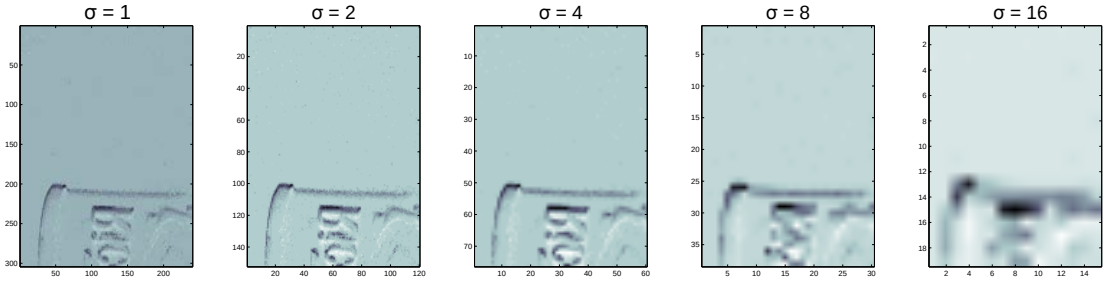



Figure 3.3: **Exponential Time surfaces generated at five different spatial resolutions**. Five different surfaces of varying scales are shown. These were generated by calculating the accumulating time surface described in equation (3.6) and making use of the scale mapping in equation (3.7) with five different $\sigma$ values as shown in the above figure. The five surfaces are all updated in an event-based fashion for each incoming event.

previous event for that pixel on the accumulating surface increases the range of the surface beyond the $[-1, +1]$ and allows for staggered and partial updates of the surface.

The accumulating nature of this surface opens up the possibility of calculating surfaces with lower spatial resolutions on the arrival of each event in an efficient manner. Given a scale factor of $\sigma$, event $\mathbf{e} = (x, y, t, p)$ can be mapped to the lower spatial surface as follows:

$$\mathbf{u}' = [\frac{x}{\sigma}, \frac{y}{\sigma}] \tag{3.7}$$

$$p' = \frac{||p||}{\sigma^2} \tag{3.8}$$

The contribution of a given event to a lower spatial resolution is weighted by the scale factor. A pixel in a lower spatial resolution therefore corresponds to the averaged mean of the pixels mapped to it on the higher spatial surfaces. The accumulating surface is required here as the value on the lower resolution surface would otherwise only represent the contribution from the last event to arrive, and not the history of all events in the past.

Each spatial surface is updated for each incoming event, and each surface can be updated in parallel. Figure 3.3 shows five different spatial resolutions for a given stream of events. These surfaces can be used to provide a level of spatial invariance in a similar fashion to the pyramid of difference-of-Gaussians used in the SIFT approach in Section 2.3.2.

### 3.3.2 Feature Detection on Time Surfaces

The techniques used by conventional feature detection algorithms do not translate well to the realm of event-based data. Usually, these detectors operate on an entire image at once, applying a filter or multiple filtering steps to entire frames of pixels and then searching the resulting space for candidate points. When dealing with event-based data, such an approach is not possible as algorithms need to operate iteratively on received data as it arrives in order to benefit from its event-based nature.

An event-based approach also introduces the challenge of detecting desirable features in spatio-temporal data, and the challenge of not only detecting where interest points occur, but also when they occur. The use of a temporal surface reduces the problem down to a two dimensional representation and it would be possible to generate frames at regular intervals from these surfaces and then to apply conventional feature detection techniques, but this would discard the high temporal resolution and data-redundancy offered by the sensor.

Therefore, feature detectors that operate in an event-based manner only have access to past events and need to operate in an efficient manner. It is not feasible to search or calculate across the entire image resolution for each incoming event, and therefore it is desirable to limit the information to spatially local areas around the event. Therefore, we can define a spatio-temporal feature on a time surface $\Omega_e$ of size $w \times w$ pixels centred on position $\mathbf{u} = (x, y)^T$ and occurring at time $t$ as follows:[1]

$$I_e(x, y, t) = \{\Omega_e(x + i, y + j, t) \mid |i, j| < w\} \tag{3.9}$$

Borrowing from conventional feature detection in computer vision, these feature detectors make use of the principles behind the corner and edge detectors from the Harris Corner detector [57]. Harris and Stephens based their work on the Moravec corner detector [115], which operated by examining a local window about each pixel and determining the average changes in intensity that would result from small shifts in the window in both the $x$ and $y$ directions. Given a region of approximately uniform intensity, a small shift will produce a small change in the average intensity, an edge will produce a small change in the direction of the edge, but a large change in the direction parallel to the edge, and finally, a corner or region of interest, will produce a large change in all directions.

Mathematically, this method makes use of the weighted sum of squared difference (SSD) between an image patch $I(u, v)$ and the same sized patch shifted

[1] This definition makes the assumption that the imaging device makes use of pixels aligned to a rectangular grid, and all the work in this thesis makes use of devices that contain such arrangements of pixels. Other pixel configurations, such as a hexagonal arrangement of pixels, would require a different feature definition.

by a small factor $(x, y)$. The weighted SSD between $I(u, v)$ and $I(u+x, v+y)$ is given as follows:

$$S(x, y) = \sum_u \sum_v w(u, v)(I(u + x, y + v) - I(u, v))^2 \tag{3.10}$$

In which the weighting function $w(u, v)$ alters the sensitivity to noise, and a Gaussian weighting function is commonly used. The term $I(u + x, v + y)$ can be approximated using a first-order Taylor expansion as follows:

$$I(u + x, v + y) \approx I(u, v) + I_x(u, v)x + I_y(u, v)y \tag{3.11}$$

In which $I_x$ and $I_y$ are the partial derivatives of $I$. Using this definition, Equation (3.10) can be re-written as:

$$S(x, y) \approx \sum_u \sum_v w(u, v)(I_x(u, v)x + I_y(u, v)y)^2 \tag{3.12}$$

Or, written in matrix form:

$$S(x, y) \approx \begin{bmatrix} x & y \end{bmatrix} A \begin{bmatrix} x \\ y \end{bmatrix} \tag{3.13}$$

In which, $A$ is a structural tensor as follows:

$$A = \sum_u \sum_v w(u, v) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} = \begin{bmatrix} \langle I_x^2 \rangle & \langle I_x I_y \rangle \\ \langle I_x I_y \rangle & \langle I_y^2 \rangle \end{bmatrix} \tag{3.14}$$

The angle brackets denote the result of applying the weighting function to the region and calculating the average value. This $A$ matrix is a 2D Hessian matrix which is often referred to as a Harris matrix [56]. Examining the magnitudes of the eigenvalues of this $A$ matrix, yields insight into the nature of the point in terms of quality as a feature. A desirable feature would exhibit two large eigenvalues, indicating sensitivity to shifts in both directions.

However, the calculation of eigenvalues is a computationally expensive operation which involves a square root operation, and Harris and Stephens proposed the following corner metric based on the trace and determinant of $A$:

$$R = \lambda_1 \lambda_2 - \kappa(\lambda_1 + \lambda_2) = \left|A\right| - \kappa Tr(A)^2 \tag{3.15}$$

In which $\lambda_1$ and $\lambda_2$ are the two eigenvalues of $A$ and $\kappa$ is a tunable parameter that adjusts the sensitivity of the corner detector (with a value ranging from 0.04 to 0.2 often used). This method is applied to every pixel in the image in the fashion of a filter, and a point is considered to be an interest point if the value is greater than the surrounding pixels (usually the neighbouring 8 pixels).

Shi and Tomasi proposed another means of ranking corners by calculating the eigenvalues of $A$ and comparing them to a specified threshold [116]. They found that if the smaller eigenvalue exceeded a specified noise threshold, then $A$ is generally well-conditioned and the region can be considered a good feature. Both the Harris Corner Detector and the Shi and Tomasi detector require a threshold or a constant which needs to be empirically determined, but there exist other measures which avoid the need for a tunable parameter. The first such method is the harmonic mean of the eigenvalues as follows:

$$R = \frac{\left|A\right|}{Tr(A) + \epsilon} = \frac{I_x^2 I_y^2 - I_{xy}^2}{I_x^2 + I_y^2 + \epsilon} \tag{3.16}$$

Where $\epsilon$ is a small positive constant to prevent division by zero problems.

It is important to note that these corner detectors operate on traditional images. Adapting these methods to an event-based approach requires that they operate on the time surfaces instead. The process of converting the spatio-temporal pattern to a time surface yields a 2D spatial surface containing real-valued integers representing the inter-spike times around the current event. Whereas these methods detect corners in conventional images, these same techniques identify similar structures on the time-surface. These structures are spatio-temporal corners, resulting from a specific combination of contours in the visual scene.

Figure 3.1 shows how a spatio-temporal feature $I_e$ is extracted from the time surface about an incoming event. It is on this surface that the corner detection methods are applied. Unlike the usual case in conventional computer vision, the algorithm is evaluated iteratively for each incoming event and not applied globally across the whole image surface (as the time surface is continuously evolving with each incoming event). The corner metric is calculated for the point on the time surface corresponding to the current event (i.e. it makes use of a window centred on $\mathbf{u} = (x, y)^T$ from the current event $\mathbf{e}$). The same corner metric is also calculated for all eight neighbouring points on the time surface, and a feature is only considered if it represents either a local maximum or local minimum in its local spatial neighbourhood. This removes the need for an explicit threshold.

### 3.3.3 Time Surface Descriptors

Identifying a candidate feature of interest on the time surface represents only half of the problem involved in robustly detecting a feature. The second half of the problem involves finding a representative manner in which to describe the feature to allow for future matching and re-detection. This involves the calculation of a descriptor from the given feature.

An important aspect of a descriptor is that it provides a quantitative means of comparing two features, often yielding a metric indicating their relative simi-

larity. Carefully choosing an encoding method from feature to descriptor, and an appropriate means of quantitatively comparing two features, can yield a feature detection system that displays invariance to a variety of different transformations (such as affine transformation, photometric transformation, and in the case of event-based visual data, changes in optical flow velocities).

The simplest descriptor for a given feature $I \in \mathbb{R}^{m \times n}$ on a time surface $\Omega_e$ is to simply transform the 2D region into a one dimensional vector as follows:

$$d = vec(I) = [I_{1,1}...I_{m,1}, I_{1,2}...I_{m,2}, I_{1,n}...I_{m,n}]^T \tag{3.17}$$

Making use of such a descriptor is paramount to simple template matching, but provides no invariance to any transformations. One particular issue with this descriptor is its sensitivity to the magnitude of the values within $I$, which makes direct comparison of vectors sensitive to noise and outliers.

Achieving a level of invariance to the magnitude of $I$ is possible by normalising the descriptor as follows:

$$d = \frac{vec(I)}{||vec(I)||} \tag{3.18}$$

In terms of event-based data, the magnitude of the values contained in $I$ are an encoding of the last spike times for events in the spatial neighbourhood of the last event. Therefore, the relative difference between pixels in $I$ is a measure of velocity (either of objects within the scene, or of the camera itself). It is even possible to calculate local optical flow about an event given just the linear time surface $\Lambda_e$ [77].

It is also possible to sort the values in the descriptor, which yields a bag-of-words approach to feature detection in which the presence of certain magnitudes within the feature itself becomes the feature.

Borrowing from the SIFT approach, a histogram of orientations for the given image patch can be calculated, and then re-oriented about the most significant orientation to yield rotational invariance.

Figure 3.4 shows three different descriptors calculated for a given region on the time surface. The figure shows how the descriptors change under rotation, mirroring and skewing operations and when noise is applied to the image patch. The figure shows that the orientation descriptor displays the best invariance to the rotation, skewing and noise transformations.

Unfortunately, this method is by far the slowest as it requires far more calculations than the other two descriptors and needs to be calculated for each incoming event. This highlights one of the important issues regarding feature detection, which is the trade-off between computational cost and required invariance of the

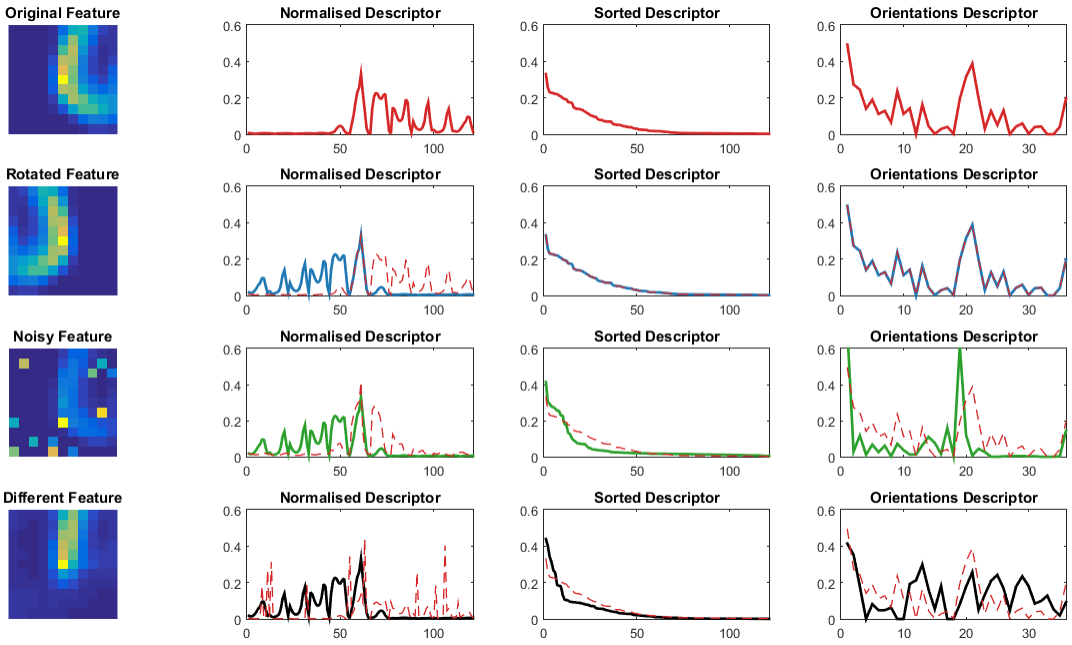



Figure 3.4: **The effect of transformations and noise on the normalised, sorted and orientation descriptors applied to event-based data**. Three different descriptors are shown for four different features. The first row shows the original feature and their resulting descriptors for each method. The second row shows the results for a rotated version of the initial feature, with descriptor from the original feature shown as a dotted line. The third line shows the initial feature corrupted by noise and the fourth shows a completely different feature.

descriptor. Unlike feature detection in conventional computer vision, the algorithm does not have access to the full frame of data and therefore cannot optimise the search for points of interest in the same manner.

Instead, each event must be considered as a potential feature as it arrives. Furthermore, these calculations must occur when events arrive, and this often places a hard limit on the computational time available in order to guarantee some degree of runtime performance.

One facet that is not often discussed is that there is not always a need for invariance to all types of transforms. For example, rotational invariance may not always be required, as the orientation of a feature may be important to the classification or tracking task. Additionally, the high temporal resolution of the camera translates to a fast update rate and therefore there are often only small changes in features from detection to detection.

## 3.4 Feature Detection using Orientated Histograms

This section introduces a method of detecting features based on the use of oriented histograms of time. As with the other methods presented in the previous section, the structure and implementation of this method is designed to be event-based. Leveraging the success of histograms of orientated gradients (HOG) in conventional computer vision [66], the same principles are applied to the temporal information provided by the ATIS camera.

Unlike the traditional applications of histograms of orientations which calculate orientations from image derivatives, these orientations are calculated from the spatio-temporal gradients derived from the ATIS camera. For each incoming event, the last spike time is recorded and the spatial neighbourhood around the current event is used to calculate the gradients used to form the orientations.

Figure 3.5 shows how the last spike times for the current event (shown as a green circle) are used to construct an image of time around the current pixel. A threshold is used to prevent events that occurred far in the past from exerting a strong dominance over the orientations by excluding them from use in further calculations. A threshold value of 50ms was found to be a good parameter for the given ATIS hardware used. Additionally, during initialisation, all pixels have their last spike time set to an arbitrarily high number to indicate that they have never fired.

The temporal derivatives, $dx(x, y)$ and $dy(x, y)$, were calculated using a $3 \times 3$ Sobel operator, operating over a feature size of $8 \times 8$ pixels around each event. The temporal gradient magnitudes $m(x, y)$ and the orientations $\theta(x, y)$ were then calculated from these temporal derivatives using the following equations:

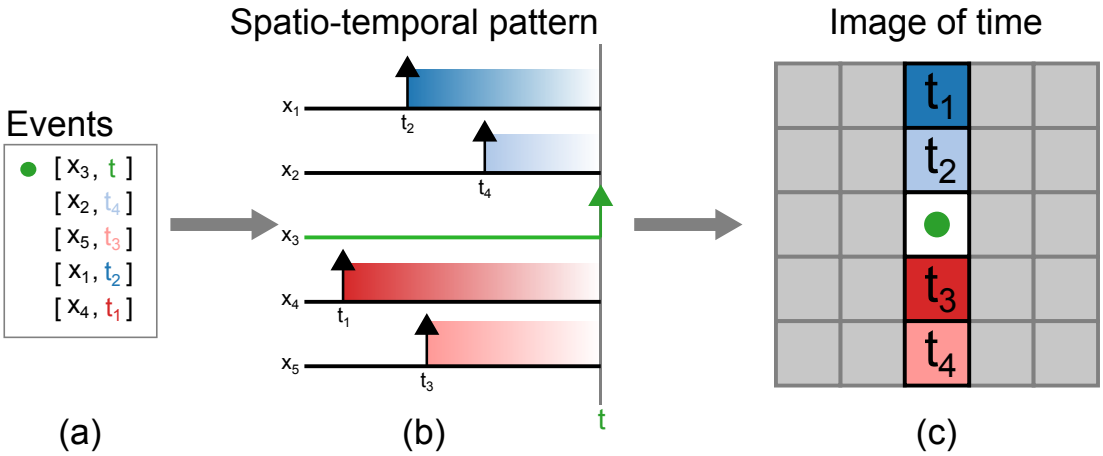



Figure 3.5: **Diagram showing how the spatio-temporal image of time is created for the 1D case**. The ordered list of spike events is retrieved from the camera (a), which in the case of the ATIS device, contains an additional y-dimension. The relative time difference in the spike times from the most recent event to the surrounding events defines the image of time for the arriving event (b). These relative spike timings are then stored (c) and are used to calculate the temporal orientations about the event. Only the vertical dimension is shown in this diagram for the sake of clarity, but when performed on data from the ATIS camera, horizontal and vertical orientations are considered, yielding a full matrix of values, rather than a single vector. This is shown in (c) as the greyed boxes.

$$m(x,y) = \sqrt{dx(x,y)^2 + dy(x,y)^2} \tag{3.19}$$

$$\theta(x,y) = \arctan \frac{dy(x,y)}{dx(x,y)} \tag{3.20}$$

Every valid pixel in the neighbourhood produces an orientation and a magnitude, and as no two events can occur at exactly the same time-step[1], a small difference is always present between two events. This always generates an orientation albeit with a small magnitude for pixels that have fired in close proximity. Although these magnitudes of orientation are not used in generating the histogram representation of each feature, they are still calculated and used to reject weak orientations.

When a neighbouring pixel has not generated an event in a reasonable amount of time (defined by the threshold mentioned above), the process to calculate the orientations ignores the value, resulting in the total number of orientations produced by a feature having an upper bound equal to the size of the neighbourhood used.

It should also be noted that the chosen method of calculating the orientations used the arctan function and all the resulting orientations were then shifted into the range $[0, 2\pi)$. The result of this process is a set of orientation angles present in the spatio-temporal neighbourhood around the incoming event.

As orientations are intrinsically angles, and as there is a need to make inferences on the distributions that give rise to them, a special sub-branch of statistics, known as circular statistics, is required in order to properly draw statistical inferences from the data. Circular statistics are required for two purposes in this method. The first task is as a test of uniformity in order to reject features arising from noise, and the second is in the process of detecting and characterising good features.

It is important to make the distinction between the notion of temporal gradients and spatio-temporal gradients. As time is monotonically increasing, the temporal gradient for the latest event will always be negative, as every event prior to it inherently occurred at a previous point in time. For the orientations discussed in this section, the temporal difference dictates the magnitude of the orientations, but the spatial alignment of pixels determines the orientation. Therefore, the polarity of all magnitudes is constant. The polarity depends on the reference chosen for the temporal values. Referencing all temporal values to

[1]The ATIS device used in this work applies a time-stamp to each incoming event using an internal clock which also controls the acquisition system. As a result, two events that occur close together will be read out sequentially, and will receive different time-stamps. Thus, when using the full resolution of the timing signal, no two events can occupy the same time-stamp.

the latest event yields a negative polarity, whilst referencing it to the start of the timing for the sequence produces a positive polarity. In either event, the polarity is constant due to the monotonic nature of time.

### 3.4.1 Introduction to Circular Statistics

The orientations extracted to create the histograms used in this section belong to a special class of data known as circular or directional data. Circular data occurs frequently, and extends beyond angles into many other types of periodic data throughout engineering, and across other such diverse fields as neuroscience [117], oceanography [118], psychology [119] and even sleep research [120].

All circular data has an underlying periodicity, which can map to angles in the range of $[0°, 360°)$. In the case of angles, which are perhaps the simplest example of circular data, there is no distinction between 0° and 360° and all other angles that are greater than 360° or smaller than 0, are simply mapped back into that range through the following equation:

$$\theta = \theta_k \mod 360° \in [0°, 360°) \tag{3.21}$$

The same occurs when examining other types of circular data, such as days of the year, in which Day 1 occurs after Day 365[1]. In a general sense, there is no concept of high or low values in circular data. There is also no inherent concept of zero, as the data can be mapped to a unit circle, and the choice of any particular point of reference is entirely arbitrary.

This complicates the process of using conventional statistical methods to describe the data or to draw inferences from it. Generally, linear statistics are not applicable and the notion of the arithmetic mean provides a straightforward example of this point. Given a set of angles, such as $\theta = [15°, 45°, 315°]$, calculating the arithmetic mean would yield an angle of $\bar{\theta} = 125°$. Figure 3.6a shows these three angles and the resulting arithmetic mean, and demonstrates how it fails to capture the true mean direction represented by the angles.

In reality, the arithmetic mean does provide an estimate of the 'centre' of the distribution of the data as it does in linear statistics, but it is heavily biased by the choice of a zero reference and the conventions of direction and rotation in the data [122]. Although there may be times when the arithmetic mean of angles is appropriate, there is a need for a more intuitive means of describing the mean of a circular data.

A common approach to this problem is to project the angles onto a unit circle [123] by treating the angles as the orientation component of a set of polar

[1] Assuming that it is not a leap year. In practise, leap years are often handled by inserting an artificial day at the end of February. An example and proof of robustness is found in [121].

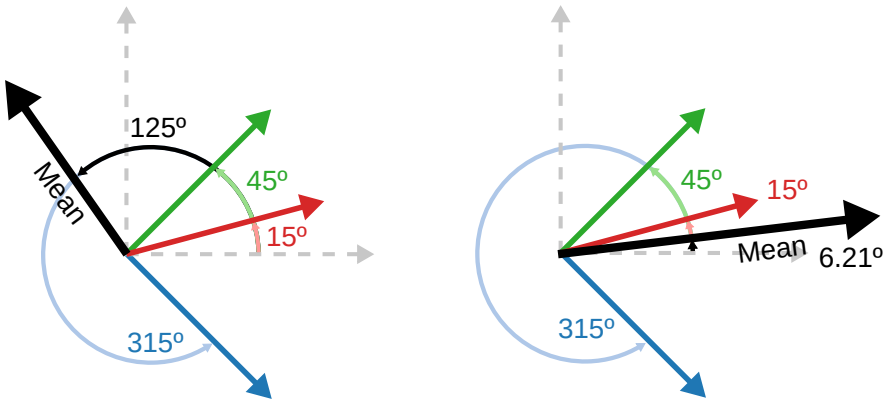



Figure 3.6: **Diagram showing the differences between the arithmetic and circular means for three angles** 15°**,** 45° **and** 315°**.** The diagram presented highlights the difficulty in applying conventional statistical measures to circular data. Although the calculation of the arithmetic mean shown in (a) is valid, it is heavily biased by the choice of a zero reference and conventions and rotation in the data. The resulting value does not capture the true mean direction represented by the angles. A common solution to this project is to project the angles onto a unit circle and take the mean angle of the resulting vectors as shown in (b). This value better represents the concept of a mean direction for circular data.

coordinates with unit magnitude as follows:

$$r_i = \begin{pmatrix} cos\theta_i \\ sin\theta_i \end{pmatrix} \tag{3.22}$$

For the sake of clarity, the angles extracted from histograms will be referred to as orientations instead of angles. Given a set of orientations $\theta = \theta_1, \theta_2 ... \theta_n$, the above equation transforms these points into unit vectors $\mathbf{r} = r_1, r_2 ... r_n$. Taking the average of this vector yields the mean resultant vector $\bar{r}$:

$$\bar{r} = \frac{1}{N} \sum_{i=1}^{n} r_i \tag{3.23}$$

The angle of this vector represents the mean direction for the data. This value better represents the concept of the mean direction for angular data, and provides a good analogue for the arithmetic mean. For this reason, it is often referred to as the circular mean. The following equation provides a method to calculate the circular mean direction directly from the sample orientations:

$$\bar{\theta} = \arctan(\frac{\sum_{j=1}^{n} cos\theta_j}{n}, \frac{\sum_{j=1}^{n} sin\theta_j}{n}) \tag{3.24}$$

When applied to the example given above, the circular mean defined in Equation (3.24) yields $\bar{\theta} \approx 6.21°$. As Figure 3.6 (b) shows, this value better represents the concept of a mean direction for angular data and it represents the mean direction about which the given orientations are clustered.

The magnitude of the resulting vector $\bar{r}$ also plays an important role in circular statistics, as it provides a measure of the concentration of the angles about the mean direction. The following formula calculates the resultant vector length $\|\mathbf{R}\|$ directly from the sample orientations:

$$\|\mathbf{R}\| = \frac{\sqrt{(\sum_{j=1}^{n} \cos\theta_j)^2 + (\sum_{j=1}^{n} \sin\theta_j)^2}}{n} \tag{3.25}$$

The circular variance, which describes the concentration about the mean direction, and is defined as follows:

$$V = 1 - \|\mathbf{R}\| \tag{3.26}$$

As $\|\mathbf{R}\|$ is the magnitude of the mean direction, the variance is bound to the interval $[0, 1]$, and a value close to 0 indicates that the orientations all point in same direction, and correspondingly, a value close to 1 indicates that the points are spread evenly about the unit circle. It is important to note that this does not

necessarily imply that the points are uniformly spread, but rather that they are not clustered about a single direction.

Armed with descriptive statistics similar to the mean and variance for describing a sample of circular data, it becomes possible to explore the analogous distributions used to describe circular data. The most common circular distribution is the von Mises distribution, which closely resembles the Normal distribution. Both are unimodal and symmetrical distributions, and the von Mises distribution possesses a similar shape to the Normal distribution over the range of $[-180°, 180°]$.

Whereas a normal distribution $x$ can be fully described by $x \sim N(\mu, \sigma)$ where $\mu$ is the mean and $\sigma$ is the standard deviation, a von Mises distribution $y$ can be fully described by $y \sim VM(\bar{\theta}, \kappa)$, where $\bar{\theta}$ is the mean direction (from Equation (3.24)) and $\kappa$ is the concentration parameter.

The probability density function for a von Mises distribution is given by [124]:

$$p(\theta, \bar{\theta}, \kappa) = \frac{1}{2\pi I_0(\kappa)} e^{\kappa \cos(\theta - \bar{\theta})} \tag{3.27}$$

Where $I_0$ is the modified Bessel function of order zero. The subject of using a Maximum-Likelihood approach to finding the parameters of a von Mises distribution presents a number of complexities, especially involving the estimation of the concentration parameter $\kappa$ [125]. Although there exist a number of different approximations for computing $\kappa$, these are often computationally costly. As the feature detection step runs frequently, a less accurate approximation was chosen as its simplicity makes it fast to compute. More accurate approximations were introduced by Banerjee et al. [126] and further refined by Sra [125]. For the purposes of this work, the $\kappa$ value can be determined from $\|\mathbf{R}\|$ using the following approximation [127]:

$$\kappa^* = \begin{cases} 2\|\mathbf{R}\| + \|\mathbf{R}\|^3 + \frac{5\|\mathbf{R}\|^5}{6} & \|\mathbf{R}\| < 0.53 \\ -0.4 + 1.39\|\mathbf{R}\| + \frac{0.43}{1-\|\mathbf{R}\|} & 0.53 \leq \|\mathbf{R}\| < 0.85 \\ \frac{1}{3\|\mathbf{R}\| - 4\|\mathbf{R}\|^2 + \|\mathbf{R}\|^3} & \|\mathbf{R}\| \geq 0.85 \end{cases} \tag{3.28}$$

This estimate $\kappa^*$ requires a correction to remove an inherent bias for samples containing 15 or fewer orientations as follows [124]:

$$\hat{\kappa} = \begin{cases} max(\kappa^* - \frac{2}{n\kappa^*}, 0) & \kappa^* < 2 \\ \frac{(n-1)^3 \kappa^*}{n(n^2+1)} & \kappa^* \geq 2 \end{cases} \tag{3.29}$$

Therefore, for a given set of $n$ orientations, $\kappa$ can be approximated as follows:

$$\kappa \approx \begin{cases} \hat{\kappa} & n \leq 15 \\ \kappa^* & n > 15 \end{cases} \tag{3.30}$$

The von Mises distribution becomes a uniform circular distribution when $\kappa = 0$. It is also worth noting that there is also a wrapped Normal distribution, which is also suitable for circular data, although this method is not commonly used as it makes drawing inferences more complicated [120].

The von Mises distribution and its derivatives are used throughout this work as the uniform distribution forms the basis of the circular noise filter presented in Section 3.4.3, and Section 3.4.4 uses mixture models of von Mises distributions to select good spatio-temporal features.

### 3.4.2 Mixture Models of Circular Distributions

Given a set of orientations, it is useful to be able to identify and separate the data into a set of underlying distributions. In linear statistics, this can be performed using mixture of models that rely on the Expectation-Maximisation (EM) algorithm. Typically used with Gaussian distributions, these techniques are capable of iteratively deriving the properties of the underlying distributions for a set of data. Usually, however, the nature and number of underlying distributions is required beforehand.

In the context of this work, there arises a need to attempt to find and identify underlying distributions making up the set of orientations about each incoming event. Applying a technique similar to Gaussian Mixture Models (GMM) allows the identification and rejection of unimodal distributions, and furthermore allows for features exhibiting bimodal distributions to be identified and characterised.

Unfortunately, Gaussian Mixture Models cannot be directly applied to sets of circular data. Figure 3.7 shows the results of applying a GMM to a bi-modal distribution of angles. The two distributions shown were generated by creating two normal distributions of angles over the range $[-720°, 720°)$ with random means and standard deviations. The angles were then mapped back to the range $[0, 360°)$ using the periodic nature of circular data shown in Equation (3.21). The figure shows how the GMM is unable to handle the distribution that wraps around the mean. The method failed to converge, and was stopped after 1000 iterations.

To alleviate this problem, this work makes use of a mixture model adapted specifically for circular data, and structured around the von Mises distribution. Although not as commonly used as Gaussian Mixture Models, there have been a number of applications of similar principles to circular data. Mixture models

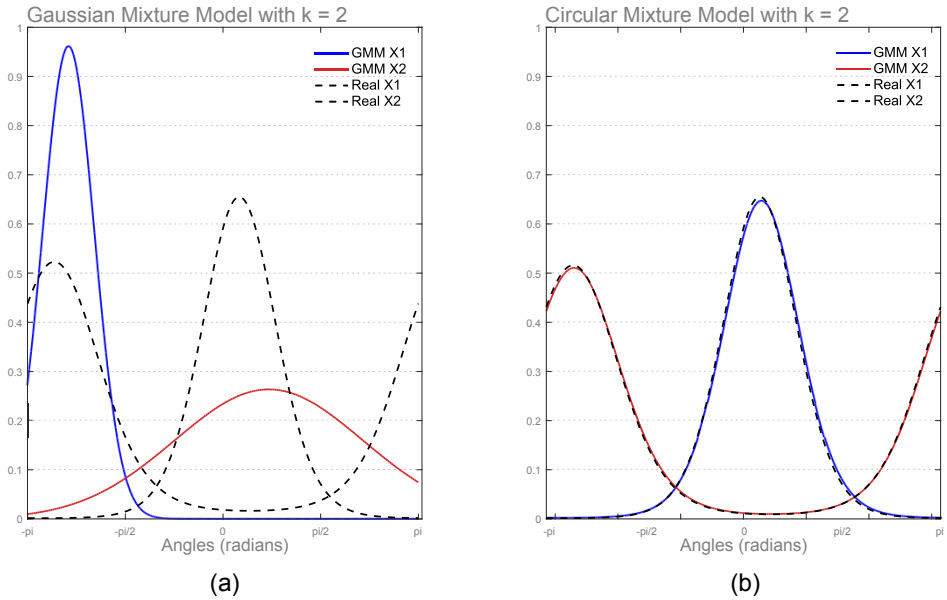



Figure 3.7: **Comparison of the fits provided by a Gaussian Mixture Model and a Circular Mixture Model for a bimodal set of orientations ($\mathbf{k = 2}$).** (a) The result of applying a standard Gaussian Mixture Model to circular data showing how the GMM is unable to handle the circular distribution. The two underlying distributions used to create the data are shown using the black dotted lines, and the distributions identified through the GMM algorithm are shown in red and blue. The GMM failed to converge and was terminated after 1000 iterations. (b) The results of the Circular Mixture Model applied to the same data. The algorithm converged after 61 iterations and successfully identified the underlying distributions.

using wrapped normal distributions have been implemented by Agiomyrgiannakis and Stylianou [128, 129] for use in analysing phase data for speech recognition, whereas Mooney et al. [120] and Banerjee et al. [130] explored the use of mixture models based on von Mises distributions. The method implemented in this work is based on the methods outlined in the latter publications.

The circular mixture models make use of the same expectation maximisation steps as performed in finding standard Gaussian Mixture Models, except that the underlying distribution fitted is a von Mises distribution, and therefore the defining characteristics of each distribution change from mean and variance to mean direction and concentration. Figure 3.7 shows the result of the applying the circular mixture model to the same distributions as used for the Gaussian Mixture Model. Unlike the GMM, the circular mixture model was able to fit both distributions successfully after 61 iterations.

### 3.4.3 Noise Filtering through Circular Statistics

The statistical properties of circular data allow for an interesting class of filters that remove noise through examining the nature of the temporal orientations surrounding the event. These noise filters operate by selectively filtering events based on the concentration and uniformity of the temporal orientations.

Consider the example provided in Figure 3.8, in which a single noise spike occurs in a region which has received no other events recently, but has had events occur within the 50 ms window set by the threshold as shown in (a). Calculating the orientations surrounding this single spike results in a set of strong temporal orientations in all the cardinal and inter-cardinal directions around the noise event (as none of the surrounding pixels have received a spike) as illustrated in (b). The orientations for the patch around the event form the histogram shown in (c). The number of bins used in the histogram controls the inaccuracies resulting from timing inaccuracies and jitter, but the orientations form distinctive peaks at the cardinal and inter-cardinal angles.

Projecting these angles onto a unit circle, as shown in (d), it becomes clear that there is no dominant mean direction for this set of angles, resulting in a mean direction vector with a length close to zero. As shown in Equation (3.26), this results in a high variance, indicative of a poor fitting mean direction.

Setting a minimum threshold on the magnitude of the mean direction vector, and filtering out events which produce small resultant vectors produces a quick method for eliminating stochastic noise events. Additionally, when using large features, this method also discards any events that have multiple non-overlapping noise spikes, as these also produce orientations on the cardinal and inter-cardinal directions and increase only the number of orientations on each direction, instead of the spread.

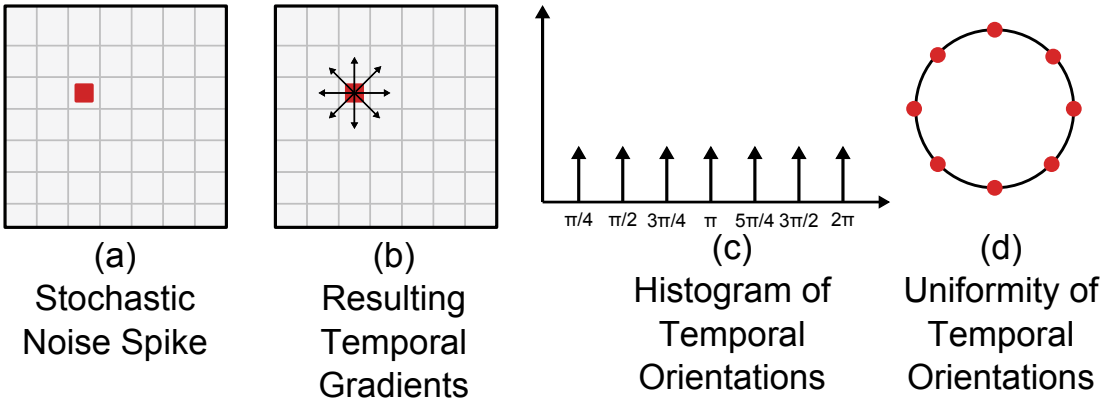



Figure 3.8: **Diagram showing how noise filtering is performed using circular statistics**. When a noise spike occurs in region in which no events have occurred recently (a), it generates strong temporal orientations in all directions around the event (b). When a histogram is constructed from these events, peaks form at the cardinal and inter-cardinal directions (c). These orientations are uniformly distributed around a circle (d), and therefore can be rejected using a test for uniformity.

Unfortunately, setting this threshold too high can cause the filter to discard useful events as well as useful features may also exhibit a non-modal spread and therefore generate a mean direction vector with a small magnitude. Making use of a test for uniformity serves to alleviate this problem.

Uniformity in the case of circular statistics refers to the situation in which probability of any angle is equally likely [131]. The probability density function for a uniform distribution is simply:

$$p(a) = \frac{1}{360} \tag{3.31}$$

There exists a range of test for uniformity, and each has its own specific characteristics and applications. The Rayleigh test [127] tests the magnitude of the mean direction vector against a suitable value for a uniform distribution, and is best suited for unimodal data. Rao's spacing test [132] examines the spacing between successive samples, with the assumption that the spacing should be $\frac{360}{N}$, where $N$ is the number of samples.

The Modified Kuiper's test and the Watson test [133] are two other methods that test for uniformity against any alternative. Finally, the Omnibus test [134] is a test that works well for multi-modal distributions, and can operate on as little as 8 samples. These tests assume as a null hypothesis that the distribution is uniformly distributed, and generate probabilities for rejecting it. These tests can filter events by setting a significance level at which to reject sets of orientations

as being uniformly distributed and therefore likely spurious noise. Implementing the Rayleigh test has the advantage of directly checking the $\|\mathbf{R}\|$ length, checking it against a statistically derived value as opposed to a static threshold. However, it performs best when the underlying data is unimodal.

The Omnibus testing method also produces good results at rejecting noise spikes, and places less emphasis on the true nature of the distribution. It is important to note that all of these tests require at least 8 angles in order to produce meaningful results. Therefore, any event that generates fewer than 8 orientations is automatically discarded.

These methods provide an advantage of the conventional filtering techniques for event-based data, in that the statistical approach does not require a scene-specific threshold. As the magnitude of the orientations is not considered, the nature of these tests is theoretically capable of displaying invariance to event-rate.

### 3.4.4 Feature Selection using Mixture Models

As every event generates a set of orientations, it is beneficial to select only events that represent features encoding relevant and repeatable information. This feature selection step, and the prior noise filtering step, are important to the system as they remove spurious sets of orientations. This is of particular importance in later sections where a temporal downsampling step calculates the average orientation response for each 1 ms time-step, and is sensitive to the number of spurious events. Features, such as those generated by edges, perform poorly in recognition as they will match at multiple points along the length of the edge. Detecting and removing these features can be accomplished using the Circular Mixture Models (CMM) presented above.

A feature containing an edge will present a unimodal distribution of orientations, as most of the orientations will be perpendicular to the edge itself. Therefore, attempting to fit a bi-modal distribution to the data will fail, or will produce two distributions with mean directions that are close together.

A CMM based on the von Mises Mixture Model technique described in Section 3.4.2, runs on each set of orientations and attempts to fit a bi-modal distribution to the data. It produces two sets of parameters (a mean direction $\theta$, and a concentration parameter $\kappa$) for each distribution, and a mixing factor indicating the proportion of the orientations fitted to each distribution.

Features are discarded if the mixing factor is skewed toward one distribution (70% or more of the points belonging to one of the distributions) as this is a strong indication of a unimodal fit. The distance between the two means is also checked, and features are discarded if the mean directions are not separated by at least 30°.

## 3.5 Discussion

The feature detection methods presented in this chapter demonstrate two viable means of detecting features in the event-based vision data from a silicon retina such as the ATIS camera. These methods operate in an event-based manner, and highlight a number of the issues surrounding the development of such detectors. These two methods extend the core principles of widely-used computer vision techniques such as HOG and SIFT to an event-based paradigm in which the identification of features occurs in an online manner as data arrives.

The issue of speed and complexity of the algorithms is an important point in the development of such event-based systems. One of the advantages of event-based cameras is the high temporal resolution, and the ability to receive events as they occur, instead of at a fixed frame rate. For real-time operation, this requires either fast algorithms or ones that allow for a large degree of parallelisation in order to cope with the rate at which the sensors produce events.

As there is little in the way of existing work in this field, this thesis focuses more on the concepts of feature detection than on efficient and parallel implementations. The algorithms presented often favour speed over accuracy (as in the use of a quick but inaccurate estimation of $\kappa$ in Section 3.4.2 and in the choice and implementation of descriptors in Section 3.3.3). The implementation of these methods can benefit from pipelining techniques, hardware implementations and algorithmic optimisation, but are presented here in their simplest form as they constitute the basis of the work presented in the succeeding chapters.

# Chapter 4

# Event-Based Object Classification

## 4.1 Introduction

Object classification is one of the primary tasks required by most computer vision systems and is the process of robustly identifying or classifying objects based on visual characteristics or metrics. In order to be robust, an object classification algorithm should be able to reliably classify or recognise under a wide range of affine transformations, photometric changes and cluttered scenes.

This chapter explores a means of performing robust object classification on event-based data streams and thoroughly examines and characterises the performance. This chapter also introduces two new datasets, created from established computer vision datasets using the ATIS cameras described in Section 2.2.4. Both the datasets themselves, and the methods introduced are compared to their counterparts in conventional computer vision in order to provide a deeper understanding and means to compare and interpret the results.

This work primarily makes use of a single classifier based on the Synaptic Kernel Adaptation Method (SKIM). Although it is possible to use other spike-based learning methods in place of it, the SKIM network provides a number of important features which make it particularly well suited to the applications used in this section.

The SKIM method relies upon a pseudo-inverse update step which also underpins the OPIUM method for iterative ELM solutions. This allows for the implementation of parallel and conceptually similar ELM-based classification systems that operate on the equivalent conventional computer vision datasets, providing additional means of comparing and contrasting the performance obtained in the event-based paradigm.

The SKIM network also does not require training iterations, and produces an analytical and deterministic solution given the same network configuration. The SKIM network itself is also capable of acting as a feature detector when applied to the event-based input and spatial and temporally downsampled representations.

This section focuses primarily on the development and exploration of event-based visual processing techniques and is not intended to fully explore the realm of spike-based computation or machine learning. A single class of classification system was chosen and characterised, and the use of a single and consistent classification system allows for the drawing of comparisons between the system and techniques introduced.

Although it is possible that other classification and learning algorithms may achieve better performance in the classification tasks, this work focuses on the methodology, algorithms and processing of event-based visual data rather than on achieving the highest possible accuracy on the available datasets.

The results and discussion presented in this work are intended to provide validation of the event-based processing paradigm, and serve as a guideline to the development of future hardware and algorithmic systems.

A common problem with learning and recognition systems is the amount of time taken to train such networks. Often this problem results from having to simulate the parallel nature of the system, which reduces the problem to a sequential state and drastically increases the computational time required. In order to address this issue, these algorithms were optimised and adapted for use on parallel architectures such as computing clusters and GPUs.

The successes of these adaptations have dramatically reduced the time taken to train each network, and have enabled a far greater range of experiments to be completed, allowing for a richer and more complete understanding of the field.

## 4.2 Contributions

This section makes the following contributions to the existing body of knowledge:

- It introduces and explains two new datasets for use in event-based computer vision problems. These datasets are the largest and most consistent spiking neuromorphic datasets to date, and allow for comparisons between conventional computer vision and the event-based techniques presented in this work.

- It derives and demonstrates a mechanism for performing classification on event-based data for binary and multi-class problems. Using the two new datasets and a consistent and well-characterised classifier, this section val-

idates the information encoded in spatio-temporal patterns and serves to validate the event-based processing paradigm for visual data.

- It demonstrates and explores methods to accelerate the algorithms underlying the learning component and evaluates their practical and theoretical performances. These include a discussion of different training patterns, spatio-temporal pattern conversion processes and iterative approaches needed for handling large datasets.

- It provides an in-depth investigation into the effects of both temporal and spatial downsampling on the classification accuracies for both event-based and conventional recognition problems. It also explores the nature of downsampling as a form of feature detection.

This section also builds upon and extends the ideas and contributions presented in previous chapters.

## 4.3 Spiking Neuromorphic Datasets

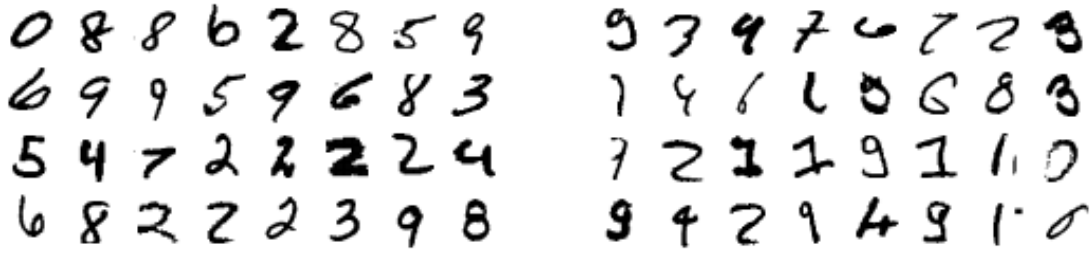
As computer vision and machine learning are active and well-established fields of research, each possesses a number of important datasets, benchmarks and figures of merit, which serve as a readily accessible and comparative means of presenting and comparing the performance of algorithms and systems. Benchmarks and challenges are also important in spurring interest and development, especially in the field of computer vision.

High quality and widely-used datasets provide a qualitative means of comparing and evaluating different algorithms by providing a known and established problem to provide metrics that are fair, and that exist within an well understood context.

Importantly, and of particular relevance with regard to event-based vision, the existence of datasets also provides access to data for researchers who lack access to proprietary, custom or expensive physical equipment. The existence of high-quality and reliable datasets thereby serve to extend the range and accessibility of this research field.

The Neuromorphic Engineering community readily acknowledges the shortage of good datasets, and can attribute this to the rarity and limited availability of neuromorphic sensors, the lack of a defined and consistent data format, varying outputs from device to device, and difficulty in providing comparable metrics.

In considering a dataset, it is important that it be large and present a problem of sufficient difficulty. The number of training and testing samples is a metric of relative importance, but the difficulty requirement can be qualified as ensuring



(a) (b)

Figure 4.1: **Examples of the digits in the MNIST Dataset.** (a) An example of the digits within the MNIST dataset showing 40 randomly selected digits from the training set. (b) The 40 most commonly miscategorised errors from the testing set.

that the state-of-the-art algorithms are not readily capable of achieving close to 100% accuracy. A good dataset should provide room for improvement over the current state-of-the-art techniques in order to foster growth and interest in the area.

This section introduces two new spike-based neuromorphic datasets, created from existing computer vision datasets that enjoy widespread use and adoption.

### 4.3.1 MNIST and Caltech101 Datasets

Instead of creating new datasets from scratch, the spike-based datasets were created from existing computer vision datasets that are among the standard set of benchmarks used by computer vision systems. The datasets chosen were the MNIST dataset [135] and the Caltech101 dataset [136].

Although current computer vision algorithms are capable of achieving good performance against these benchmarks, they were instrumental in the drive to develop the field, and the hope is that having similar datasets for event-based paradigms may have a similar effect on the Neuromorphic computation community.

In the realm of machine learning, the MNIST dataset has emerged as the de-facto means of testing and qualifying the performance of a classification system. This is due, in part, to the small dataset size (in terms of physical storage), the intuitive nature of the data, and the ease with which it can be accessed and used.

Unfortunately, the MNIST dataset is not a particularly difficult classification task, which has resulted in accuracies rates approaching 100%, leading to a competitive field in which improvements of a fraction of a percent are sought.

The MNIST dataset is actually a subset of the NIST Special Database 19

Figure 4.2: **Examples of some object images from the Caltech101 dataset.** The Caltech101 dataset contains 101 different object classes, with each class having a variable number of images. In addition, the images are of varying sizes. The five images presented here show an example of the variety in pose, aspect ratio and framing found in the Caltech101 dataset.

[137], which contains both digits and letters represented as $128 \times 128$ binary images. To generate the MNIST dataset, the NIST digits were downsampled to $28 \times 28$ pixels, with an anti-aliasing step in the downsampling algorithm which produces grey-scale levels from the original binary information [135]. Examples of the digits in the MNIST dataset are show in Figure 4.1.

The Caltech 101 dataset provides a far more challenging dataset than MNIST. It contains images of varying sizes and belonging to 101 object classes, with an additional background class. Whereas the MNIST dataset contains centred, scaled and isolated digits for classification, the Caltech101 dataset contains a far richer set of objects within environments and requires algorithms that are translation and scale invariant. Figure 4.2 shows examples of the images in the Caltech101 dataset.

### 4.3.2 Existing Neuromorphic Datasets

There have been a number of neuromorphic datasets produced over the past five years, and these can be segmented into two categories; new datasets and converted datasets. New datasets have been created from scratch, explicitly for testing a neuromorphic recognition system, whereas converted datasets utilise existing conventional datasets and reproduce them to form a neuromorphic representation or output. The new datasets presented in later sections are both of the converted dataset type.

Linares-Barranco [5] created a novel dataset created by flicking through a deck of cards and recording the pips on each card with a DVS sensor. The nature of the data acquisition is particularly well suited to event-based vision sensors, as it showcases the speed of the devices, and would not easily be matched using conventional camera equipment. The dataset comprised ten presentations of each category of pip, resulting in a total of 40 samples.

Orchard et al. [6] produced an original dataset consisting of digits and upper-

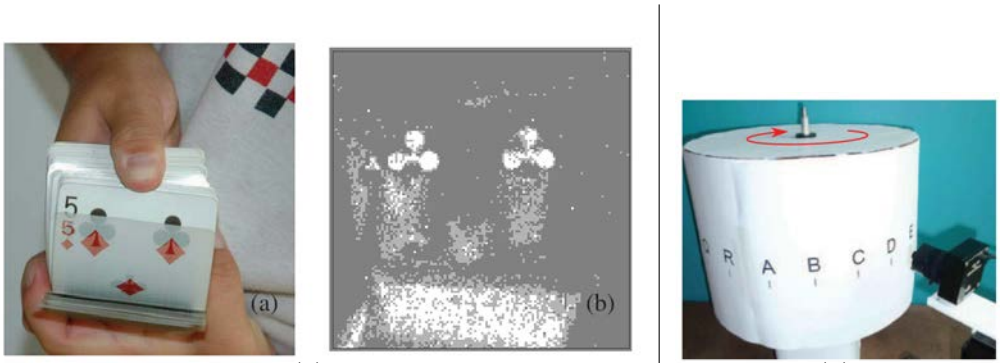



Figure 4.3: **Examples of the configurations used to generate Neuromorphic Datasets**. (a) Flicking through a deck of cards to generate the pips dataset [5]. (b) An example of the data captured from the DVS sensor. (c) The rotating barrel with letters and digits used to generate the 36 character recognition dataset used by Orchard et al. [6]

case letter and digits (36 in total). To create the data, digits and characters were printed onto the surface of a cylinder, which was then rotated at 40 rpm and recorded using a DVS camera. Two recordings were produced for each character.

These datasets tend to contain a small number of training and testing sequences, primarily due to the time taken to acquire data, the limited availability of appropriate hardware, and the prototype-nature of some of the equipment.

Another approach to creating datasets is through simulation of event-based hardware. Other AER simulation tools have been developed, and some include the ability to encode 2D image information (or video frames) into an event-based format. Dominguez-Morales et al. [138] thoroughly explored the conversion of video frames to spikes on a variety of CPU architectures. The underlying assumption being that the intensity of the pixel is the information to be encoded into the synthetic spike output for that pixel. Other efforts, by López-Torres et al. [139], extended this work to GPUs demonstrating the computational benefits, but did not attempt any classification tasks.

A notable and relevant example is the work of Carrasco et al. [140], as they created an AER simulation tool that included the ability to convert 2D images into events and were able to achieve a recognition accuracy of 91% against the MNIST dataset.

Finally, the MNIST-DVS dataset [141] is perhaps the most directly comparable dataset as it consists of MNIST digits recorded from a monitor with a DVS sensor. The images were scaled to three different levels and the images were moved across the monitor. This resulted in an effect visible in the FFT spec-

trum resulting from the monitor refresh rate, and serves to highlights the issues surrounding the use of sensors with high temporal resolution. As the researchers explored different scales, only a subset (10,000) of the available MNIST samples were converted, complicating the process of direct comparisons to computer vision algorithms.

### 4.3.3 Conversion Methodology

As mentioned above, a conversion approach was taken to create the datasets used in this work. Both the MNIST and the Caltech101 datasets were converted using the same process, and under the same conditions. It was decided that capturing the data using an ATIS camera would be the best method, as the data would then include noise and artifacts present in a real-world application. No noise filtering was performed at any stage during the production of the dataset.

The choice of conversion methodology for the datasets is an important consideration as it can impose structure onto the underlying data. The conversion of existing datasets requires a strategy to project the static images such that the ATIS camera can detect them. This can be achieved through motion of the camera, by moving the images on a screen or by changing the illumination through flashing the images on a display.

The conversion approach adopted for the creation of these datasets moved the camera in a fixed saccade motion, bearing resemblance to the retinal movements observed in primate and human experiments [142]. This method was chosen over moving the digit on a screen, primarily as an alternative to the existing MNIST-DVS dataset which was generated in that manner [143]. Flashing the image on the screen is another viable option but was not used due to concerns over maintaining a consistent illumination environment over the course of the data acquisition.

As these datasets represent the first use of this conversion method, a fixed saccade pattern was chosen to mitigate any issues relating to the characteristics of the motors used in the pan/tilt equipment. The fixed saccades do impose a structure on the output data and this is explored in this work. The use of random saccades, more closely resembling the random micro-saccades of the eyes, is the next logical step for future work on these datasets.

In the case of the datasets presented here, recordings were produced by displaying each image on an LCD monitor, and in order to avoid spurious noise events, the image remained stationary for the entire duration of the recording. Instead, the camera moved to create events. As the image on the LCD screen is inherently planar, all points can be considered to have the same depth relative to the camera. Translating the camera produces an effect that bears little resemblance to a similar translation across an actual 3D scene. To mitigate this issue,

(a) (b)

Figure 4.4: **The configuration used to capture the N-MNIST and N-Caltech101 datasets.** (a) The ATIS camera mounted on a custom-built Pan/Tilt mount. (b) A top-down view of the configuration showing the relative placement of the camera and the monitor.

pure rotation about the origin of the camera was chosen, as it is not dependent on scene depth.

After the images were projected onto the LCD screen, a short delay was inserted in order to prevent any issues relating to transitions between images, flickering or refresh rates. The camera, an ATIS device as detailed in Section 2.2.4, was mounted on the pan-tilt platform shown in Figure 4.4, enabling it to move precisely through two degrees of freedom. The motors and the imaging chip itself were controlled through a single FPGA, enabling accurate capturing and time-stamping of both the visual data and the motor commands issued to the pan-tilt mount.

Figure 4.5 shows the motions of the camera. Inspired by the notion of saccades, the camera trajectory traces an isosceles triangle, which starts at the top-left of the digit. In the first saccade motion, the camera pans so that its field of view moves downward and to the right until it reaches a point just beneath the projected digit and in line with the vertical centre of the digit. The camera then proceeds from this centreline to the top right point of the digit in an upward and right motion, forming the second saccade. The third saccade motion translates solely horizontally until the initial point is reached.

The commands were sent to the motors to initiate each saccade at 100 ms intervals. The timing of each saccade and the angles through which the camera moves are given in Table 4.1. In addition, markers were inserted into the event-stream to signify the start of each of the three movements, allowing accurate extraction of any of the motions.

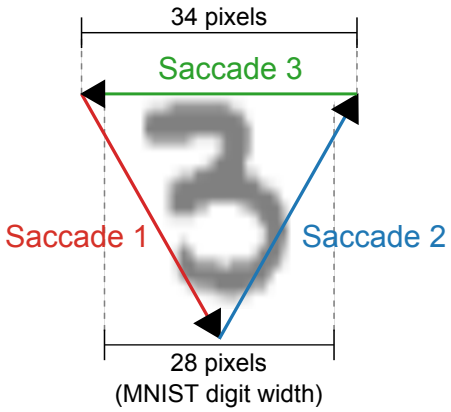



Figure 4.5: **Diagram showing the three saccade-inspired motions across each digit used in creating the spiking neuromorphic datasets.** Each MNIST digit was sized so that it occupies a $28 \times 28$ pixel region on the ATIS sensor, and three saccade motions starting in the top-left and moving in triangular motion across each digit. Due to the motion, the resulting digits sequences span a $34 \times 34$ pixel region to ensure that the full digit remains in view during the entire saccade motion.

When converting the MNIST digits, the distance between the camera and the LCD screen was calibrated such that each digit occupied a $28 \times 28$ pixel region on the ATIS sensor. In order to accomplish this, the projected image had to be scaled up to $56 \times 56$ pixels on the LCD. The digits were scaled up using a linear bitmap scaling.

Each of the training and testing digits were converted to an event-stream using the above procedure. Although each digit was calibrated such that it occupies $28 \times 28$ pixels, the movement of the camera necessitates that the actual size of each recording be larger so that the entire digit remains in focus for the entire duration of the motion. Therefore, each digit sequence was centred and cropped [1] to spatial resolution of $34 \times 34$ pixels around its centre. The events were timestamped using the full resolution of the ATIS camera (1 MHz clock) and no on-board noise filtering was applied at any point during the acquisition process.

Both the full training set (60,000 samples) and the full testing set (10,000 samples) were converted to an event-based format using the same image index as used in the MNIST dataset. This not only preserves the testing and training split, but allows direct comparison of elements to the original MNIST dataset. As this is a neuromorphic version of the MNIST dataset, it will be referred to as the N-MNIST dataset.

[1] These effects are achieved on the event data by truncating the range of the spatial coordinates (centering) and then removing any constant offset in the x and y range (cropping)

Table 4.1: **Saccade motor timings and speeds for the conversion process.** The positions are given in degrees due to the rotational motion of the camera. Similarly, the speeds quoted are specified in units of degrees per second.

| | | Start | | End | | Speed | |
|---|---|---|---|---|---|---|---|
| | Start Time | x | y | x | y | x | y |
| Image Change | 0 ms | -0.5 | 0.5 | -0.5 | 0.5 | 0 | 0 |
| Wait | 0 ms | -0.5 | 0.5 | -0.5 | 0.5 | 0 | 0 |
| Saccade 1 | 100 ms | -0.5 | 0.5 | 0 | -0.5 | 10 | 20 |
| Saccade 2 | 200 ms | 0 | -0.5 | 0.5 | 0.5 | 10 | 20 |
| Saccade 3 | 300 ms | 0.5 | 0.5 | -0.5 | 0.5 | 20 | 0 |

The images in the Caltech101 dataset were processed in a similar manner, but differed as the image sizes are much larger, are not consistent (and often have different aspect ratios). To compensate for this, each image was resized to fit within the width requirements of the ATIS sensor whilst still maintaining the original aspect ratio. In a similar vein to MNIST, this new event-based dataset is referred to as the N-Caltech101 dataset.

A more complete discussion of the dataset, the configuration used and the software procedure used in generating these datasets can be found in [144] and the associated appendices.

### 4.3.4 Conclusions

The two neuromorphic datasets presented in this section are used throughout this work to investigate and characterise the event-based feature detectors and visual processing techniques introduced in this work. As the two datasets are derived from existing datasets that enjoy widespread use in both the fields of machine learning and computer vision, it is possible to draw parallels and contrasts between the new techniques and similar methods or tasks in those established fields.

The N-MNIST dataset represents the easier of the two classification tasks and the large number of training and testing patterns allows for in-depth characterisation of the techniques presented. The data and nature of the classification task are also intuitive and the original MNIST dataset benefits from extensive research and study.

The N-Caltech101 dataset poses a more challenging classification task, and is used to further explore the manner in which the event-based techniques handle

complications such as non-uniform image sizes, objects with backgrounds, fewer training samples and a higher number of object classes. As it also has a counterpart in conventional computer vision, it allows for similar comparison to existing techniques such as HMAX [136] and the neuromorphic derivative HFirst [6].

As the focus of this work is the development of feature detection and classification techniques for event-based visual data, the use of reliable and consistent datasets is of paramount importance to understanding and comparing the techniques presented. These datasets serve to fill that requirement and when coupled with a well-characterised classifier, provide the essential basis on which event-based techniques can be analysed.

## 4.4 Object Classification using the N-MNIST Dataset

This section examines a number of different approaches to performing recognition and classification tasks on the N-MNIST dataset. The approaches primarily make use of the SKIM algorithm introduced in Section 2.5.3 as the mechanism of learning and recall, as it is specifically designed to learn spatio-temporal patterns.

This section serves to characterise both the nature and performance of the SKIM algorithm and includes an investigation into the nature of the distribution of errors when using SKIM as a classifier. The SKIM classifier serves as the means of evaluating the performance of the feature detectors and techniques presented in this thesis, but could be substituted with any other classification system capable of learning spatio-temporal patterns.

In terms of the MNIST datasets and all its derivatives, recognition and classification represent the same task. The lack of clutter or background in the images, and the lack of non-character sequences negate the need for a separate class recognition task (as required by the nature of the N-Caltech101 dataset presented in Section 4.5.4). Instead, all experiments treat the problem as a 10-class recognition task.

The diagram in Figure 4.6 shows the structure of the classification system used in this section. Each training and testing sequence consists of a stream of events in an AER format and a label indicating the encoded digit. The first stage of processing converts this data into a spatio-temporal pattern for use with the SKIM algorithm. The methodology section below discusses in detail the conversion of the event stream into a spatio-temporal pattern. When training, the digit label also requires encoding into a spatio-temporal pattern and the nature of this encoding and the different training patterns used are presented and discussed in Section 4.4.5.

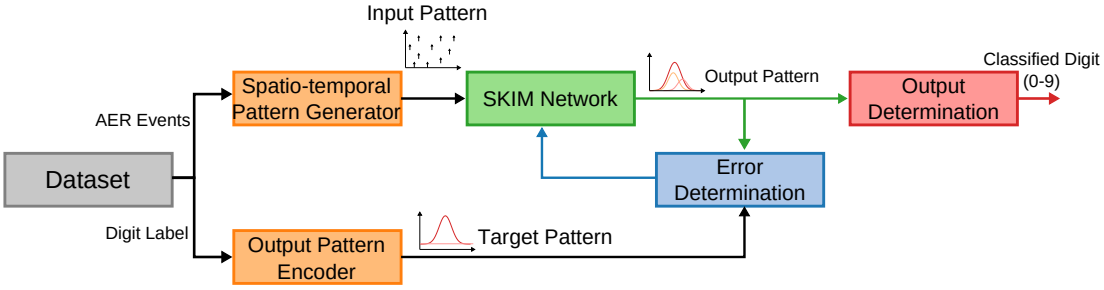



Figure 4.6: **Overview of the classification system used for the N-MNIST classification task.** The classification system presented retrieving training and testing samples from the dataset and encodes them into a spatio-temporal pattern for use in the SKIM network. The process for encoding the AER data into a spatiotemporal pattern is described in Section 4.4.1. The label for the dataset is also encoded using the training methods described in Section 4.4.5. The SKIM network produces a spatio-temporal pattern as an output, which is used either for output determination during recall operations or is compared to the training signal and used to update the SKIM network during training.

The SKIM network itself receives the spatio-temporal input patterns, and a supervisory signal when training. An error determination step calculates the error from the supplied training pattern and the output from the SKIM network. The error signal is then used to perform the update to the SKIM network. The same output from the SKIM network is also used to determine the final classification for the sequence, and the nature of the determining the winning class is discussed in Section 4.4.4.

### 4.4.1 Classification Methodology

Applying the SKIM algorithm to a classification task with the scale of the N-MNIST dataset requires a number of modifications to the underlying SKIM algorithm and a complete software framework. The framework needs to manage the state, implement the neural layers, collate the outputs and manage the training and testing regime.

Prior works have explored the theoretical performance of SKIM networks using pre-determined patterns with varying levels of noise and jitter. The authors applied the technique to the Mus Silica dataset in the original SKIM paper [145] and then later applied the SKIM to a real-world separation problem [146]. Others have used the algorithm to determine angle and direction in biological motion estimation [144], and in gait detection [147].

The application of SKIM to the N-MNIST dataset requires networks with an

order of magnitude more input neurons and hidden layer neurons than previous applications. In addition, the majority of the applications to date have formulated their outputs as binary classification tasks, whereas the N-MNIST is inherently a 10-class problem.

The training methods used for the prior SKIM implementations have, to date, created a single pattern consisting of all the training data, accompanied by a supervisory learning pattern of equal duration. The input pattern is a spatio-temporal pattern representing spikes arriving at the input nodes of the SKIM network, and the supervisory pattern contains the desired output spikes from the output neurons.

When dealing with the N-MNIST dataset, the direct application of the above method is not feasible. Instead, the system trains each pattern in a stand-alone fashion, preserving the random weights, configuration of the hidden layer nodes, inverse correlation matrix and linear output weights between training sequences. This allows the network to train on any number of training samples as it is no longer bound by the available memory.

Although the SKIM network handles event-based data, software implementations usually make use of discretised time steps to simplify processing as the outputs of the hidden layer nodes must be evaluated continuously[1] in order to calculate the soma potentials on each output neuron, and evaluate whether or not they output a spike.

If implemented in hardware, it would be possible to design a truly event-based implementation of SKIM that operates on continuous time. A system of this nature is of particular interest when combined with the ATIS hardware, which natively outputs event-based data. In the case of the N-MNIST dataset, the ATIS camera applies timestamps with a resolution of 1 $\mu s$ during the encoding of the pixel data into the AER stream (see Section 2.2.2). Tightly coupling a hardware implementation of SKIM with the hardware of the ATIS sensor could remove the need to internally timestamp incoming events, and allow the system to operate on the spikes as they arrive.

Regarding the internal timestamping, it is possible to reduce the temporal resolution of each event from the order of microseconds to milliseconds without a significant loss of events. This is due to the slow movement of the camera relative to the rate at which events are time-stamped. In addition, the camera biases were not configured for high-speed acquisition but rather to reduce noise and maximise the balance between ON and OFF events. This is an important step as it allows the SKIM algorithm to simulate millisecond time-steps, instead

[1] It is possible to alter the design of the SKIM network such that the firing of output neurons can be analytically determined. This can be accomplished by using only decaying kernel functions in the hidden layer and forcing the linear output weights to be positive.

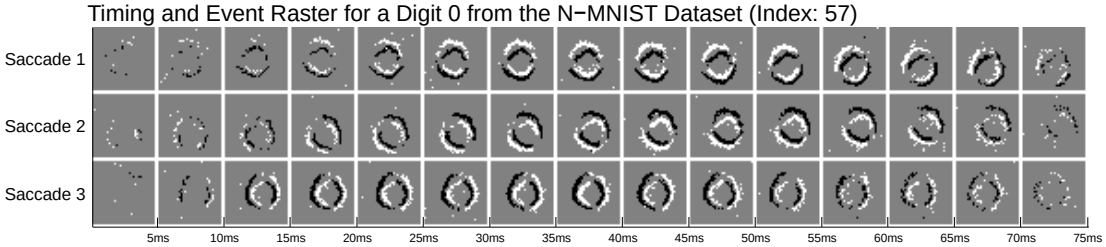



Figure 4.7: **Timing and Event Raster for a Digit 0 from the NMNIST Dataset.** Each sequence in the training and testing set contains three equal duration saccades. The plots shown are frame rendered by accumulating events over a period of 5ms.

of microsecond ones, which dramatically increases the speed of computation.

Each training and testing sequence in the dataset consists of a stream of AER events and a label indicating the digit class to which it belongs. Figure 4.7 shows an example sequence from the N-MNIST training set, showing the three distinct saccades. Events were accumulated over 5 ms periods to generate the frames shown in the figure. These plots represent a raster plot of the raw events generated in each time-step, without any filtering or overlap between frames.

As mentioned in Section 2.2.3, the AER events generated from the ATIS camera have the following form:

$$\mathbf{e} = [x, y, t, p]^T \tag{4.1}$$

In which $\mathbf{u} = (x, y)$ denotes the spatial location of the pixel generating the event, $t$ contains the value of the FPGA timer at the moment at which the FPGA receives the event from the sensor and $p \in [-1, 1]$ denotes the polarity, indicating whether it was an ON event or an OFF event. The SKIM network cannot operate on the event stream directly, and a conversion to a fully specified spatio-temporal pattern in which rows ascribe input channels and columns denote time-steps is necessary. We can denote such a spatio-temporal pattern as $I$ such that $I(c, \delta t)$ denotes the dendritic current on channel $c$ at the time step denoted by $\delta t$.

The spatial information contained in $\mathbf{u} = (x, y)^T$ is inherently lost when applied to the SKIM network as the all-to-all connectivity and their random weights discard the spatial location of channels. Therefore, any transformation that consistently maps $\mathbb{R}^2 \to \mathbb{R}$ is a suitable candidate for the conversion of spatial locations to input channels for SKIM. This only hold true if there is no interaction between input channels. Operations such as spatial downsampling can invalidate this condition depending on the implementation. When downsampling, the order of channels becomes significant as it dictates to which pixel (and then subsequent channel) the information from a region of pixels aggregates. All the experiments

performed on the N-MNIST dataset made use of a simple matrix flattening operation shown in Equation (4.2).

$$c = (34 \times \lfloor \frac{y}{\beta} \rfloor) + \lfloor \frac{x}{\beta} \rfloor \tag{4.2}$$

In the above equation, the constant value $\beta$ represents the spatial downsampling factor applied to the pattern and $\lfloor n \rfloor$ operation represents the floor function applied to $n$. The downsampling factor operates on both the $x$ and the $y$ coordinates, effectively reducing the number of channels by a factor of $\beta^2$. The value of 34 derives from the pixel dimensions of the N-MNIST digits (see the discussion on conversion methodology in Section 4.3.3). Downsampling the temporal information is far simpler as $t$ is a monotonously increasing single-valued variable, requiring only a division and flooring operation to quantise it into the appropriate time step as shown in Equation (4.3).

$$\delta t = \lfloor \frac{t}{\alpha} \rfloor \tag{4.3}$$

Therefore, for each incoming event $e$, the update to the spatio-temporal input pattern for SKIM is as follows:

$$I(c, \delta t) = I(c, \delta t) + p \tag{4.4}$$

The above equation demonstrates that the effects of multiple events accumulate when mapped to the same channel and time-step, and that their polarity dictates the nature of their contribution. It is also important to remember that the value of $t$ is always monotonically increasing, allowing the iterative construction of the spatio-temporal pattern, and allowing processing to begin for a time step once the time value for the next event exceeds it.

The output of a SKIM network is a continuous value for each output class representing the soma potential at the output neuron. In the original SKIM implementation by Tapson et al. [145], the application of a set threshold to this value converted the continuous value into a binary spike, allowing the creation of an output spatio-temporal pattern. The nature of this output spatio-temporal pattern retains the same temporal resolution, namely the same number of equally sized time-steps, with the rows changed to represent the output of each output neuron.

As the training update in a SKIM network requires a measure of error between the network output and the desired output, it follows that the format for the learning sequence must adhere to the same format as the network output. We can therefore define the output pattern $O$ such that $O(n, \delta t)$ represents the real-valued potential at output neuron $n$ at time $\delta t$. Therefore, for every $\delta t$, the input pattern $I$ contains the instantaneous input values for all $c$ channels, and the

training pattern $O$ contains all the desired output values for each output neuron $n$.

The analysis of the N-MNIST dataset presented in Section 6.5 shows that a pattern length of 316 ms is sufficient to encode every pattern in both the training and testing set. Appending an additional 45 ms onto the pattern allows the last events to have an effect on the learning system, resulting in a total pattern length of 360 ms. It is within this additional 45 ms that both the training and recall occur.

### 4.4.2 Digit Classification using SKIM

This section introduces and explores the initial applications of SKIM to the N-MNIST dataset, and in a similar manner to the statistical classifiers from the previous section, attempts to set benchmark accuracies for the dataset. As a result, the underlying methodology requires the performing of the minimum number of alterations to the underlying data. In the proceeding experiments, each training sequence includes all three saccades and the data was not subjected to any noise filtering. The training used the full spatial resolution, and 1 $ms$ time-steps.

Figure 4.8 shows a plot of the accuracy as function of the number of training presentations for networks of varying numbers of hidden layer nodes. Each line represents the results of testing the given configuration against the test dataset at regular intervals during the training process.

The testing phase occurred separately from the learning process, and the results were never included in the training or in any update mechanism. The plot displays only the first 10,000 training samples as the network performance stabilises and remains constant after that point. Intermediate tests used 1000 testing samples randomly drawn from the total testing sample set, and the final tests at 10,000 utilised the entire testing set. Both training and testing orders were always random.

It is interesting to note that a network trained with only 10 hidden layer neurons achieves a performance of approximately 26.36% at 10,000 training samples, which is almost exactly the performance of 26.52% yielded by the simple event-based classifier presented in Section 6.5. This suggests that the network requires only 10 hidden layer neurons to learn and respond to the event lengths.

The figure also demonstrates the quick convergence of the network, with the accuracy stabilising after fewer than 2000 presentations in almost every case, and often much earlier. There were no problems resulting from over-fitting, and the accuracy remained constant through the full 60,0000 training presentations. This is significant, given the iterative nature of the training process and proves the viability of using the system in an online manner.

Figure 4.8 demonstrates a selection of the full range of hidden layer sizes tested

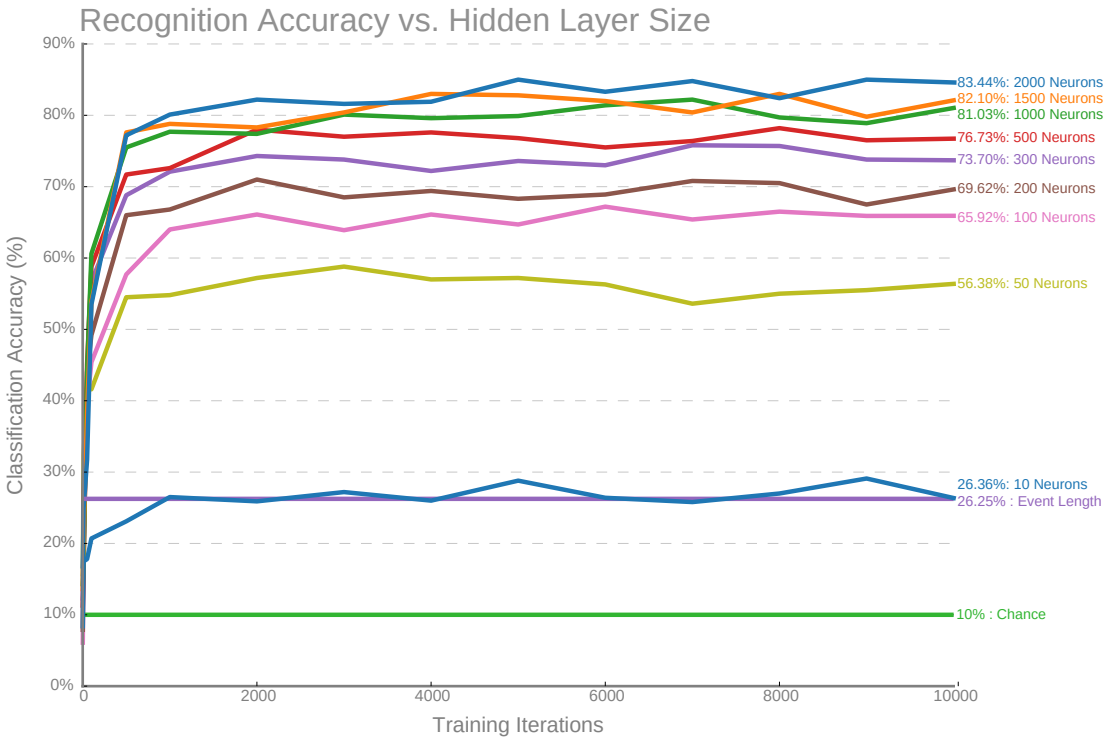



Figure 4.8: **Training accuracy over the first 10,000 training samples for N-MNIST using SKIM for increasing numbers of hidden layer neurons**. Each configuration of hidden layer neurons was trained sequentially through a randomly shuffled training sequence, and tested against the testing dataset at increments of 1000 training samples. Each test was performed independently of the training, and no learning took place during the testing phase. Also shown on the graph are the accuracies due to chance and the accuracy based solely on mean number of events. The final results shown on the right represent the full training accuracy tested against the full 10,000 training samples whilst intermediate points on the curve were calculated over a randomly drawn subset of 2000 testing samples.

for this network configuration, and Table 4.2 presents the full classification results for larger networks still. The amount of computational power and time required to complete these tests were responsible for the granularity of 100 neurons in the range of configurations. The scale of testing performed was made possible through the use of a computing cluster and the GPU optimisation techniques outlined in Section 6.5.

Table 4.2: **Results for classification of N-MNIST with larger hidden layer sizes.** This table shows results for SKIM networks with large numbers of hidden layer neurons and trained on the full 60,000 training samples. Due to the time taken to train these large networks, only a single trial of each was performed.

| Hidden Layer Size | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 1000 | 2000 | 3000 | 4000 | 5000 | 6000 | 7000 | 8000 | 9000 | 10000 |
| 81.03% | 83.44% | 85.6% | 85.1% | 86.6% | 86.3% | 88.6% | 90.22% | 91.56% | 92.87% |

Characterising the relationship between the number of hidden layer neurons and the overall classification accuracy provides a useful metric for determining the size of the network required to achieve a specific accuracy and yields insight into how the learning mechanism scales with respect to network size. Figure 4.9 shows the results of such an analysis for the accuracy achieved after training on 10,000 training samples and tested against the full 10,000 testing set. The graph shows results for the N-MNIST dataset, and the same analysis for an iteratively trained ELM network making use of the OPIUM update method and trained using the same 10,000 subset as the N-MNIST dataset. The MNIST results provide an interesting comparison as both networks make use of the same underlying algorithms and operate on the same dataset albeit in different forms.

Figure 4.9 also shows that the classification accuracy of the N-MNIST dataset is always lower than the accuracy for the MNIST dataset. This occurs as the MNIST dataset is the source for the N-MNIST dataset and the conversion process only serves to add noise and artifacts to the data. Therefore, the N-MNIST results should at best match the MNIST results. The conversion process is also lossy in that it discards grey-scale information, and this serves to lower the accuracy.

The graphs show the actual classification accuracies as points, and an exponential fit to the data with an accompanying 95% confidence interval. Immediately apparent from the figure is the similarity between the fits of the MNIST and N-MNIST datasets. Both have the same shape, and appear to exhibit the same relationship between hidden layer size and accuracy. The MNIST dataset produces a more accurate result, but seems to have similar convergence profile to the N-MNIST dataset.

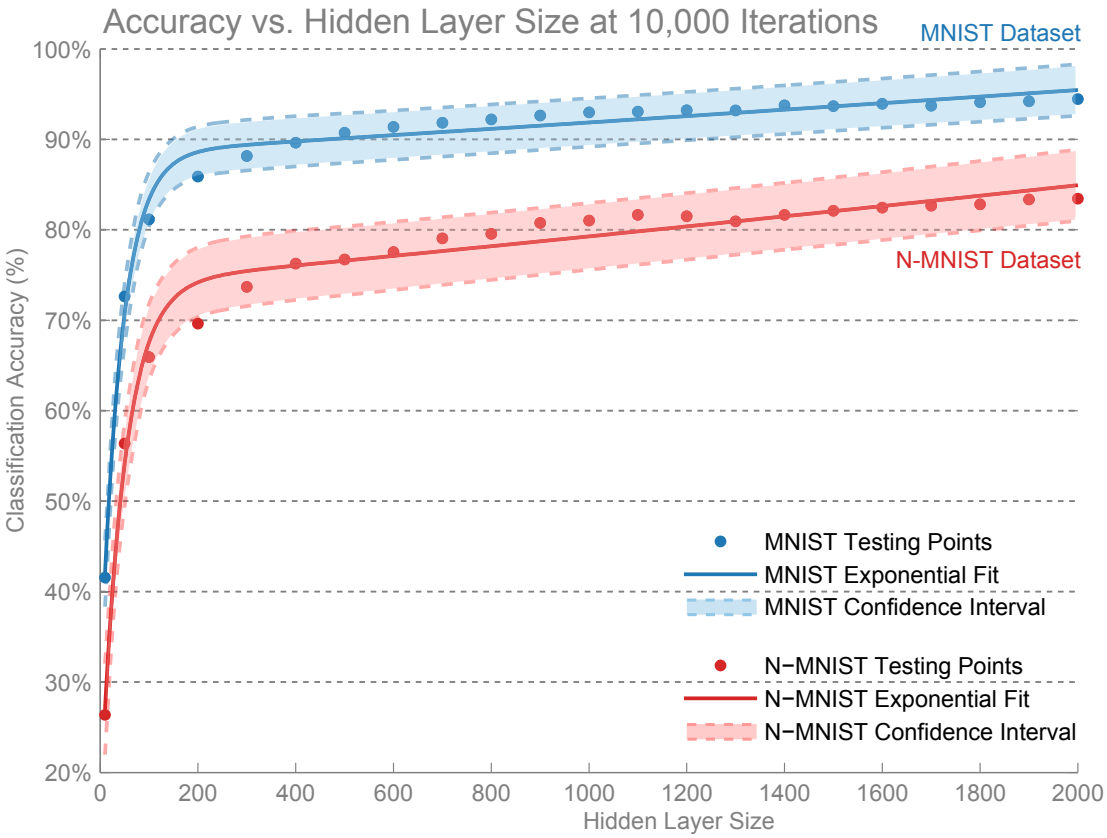



Figure 4.9: **Accuracy of MNIST and N-MNIST as a function of Hidden Layer Size**. Both MNIST and the N-MNIST datasets were trained and tested using an increasing number of hidden layer neurons. An ELM classifier was used to learn the MNIST dataset, whilst SKIM was used to learn the N-MNIST saccades. The same training and testing order was preserved, and the results plotted were the results of testing against the entire testing dataset. Both results show the actual test results, a fitted exponential approximation and the 95% confidence interval.

Table 4.3: **Accuracy vs. Hidden Layer Size fitting parameters.** This table provides parameters for two different fitted curves to the results for the SKIM network as a function of hidden layer size presented in Figure 4.9.

| | Fit Parameters | | | | Fit Accuracy | |
|---|---|---|---|---|---|---|
| Method | $a$ | $b$ | $c$ | $d$ | p-value | RMSE |
| Logarithmic: | $6.35 \pm 0.58$ | $305.2 \pm 264.7$ | $-2988 \pm 3434$ | n/a | 0.99 | 1.29 |
| Exponential: | $73.99 \pm 1.81$ | $6.8 \times 10^{-5} \pm 1.9 \times 10^{-5}$ | $-58.2 \pm 5.68$ | $-0.02 \pm 0.004$ | 0.9849 | 1.706 |

The figure shows the exponential fit, and requires two exponential terms and has the following form:

$$y = ae^{(bx)} + ce^{(dx)} \tag{4.5}$$

Table 4.3 provides the full fit coefficients for the N-MNIST dataset and includes the values required for the 95% confidence interval. Information on the quality of the fits is also provided. Unfortunately, unlike the range of these functions, the true accuracy of any classification system is inherently asymptotic to a theoretical maximum accuracy of 100%, and in reality, a value well below that.

These fits are therefore only accurate up to a certain point. The exponential fit, for example, exceeds 100% after 22,362 hidden layer neurons, forming the uppermost bound on the accuracy of the prediction. The result is still relevant, as within the range of validity, using the lower bound of the confidence interval provides a good estimate of the classification accuracy to expect, and proved useful in checking and validating the results.

### 4.4.3 Error Analysis of the SKIM network Result

As the networks used in this section make use of either random weights or random training orders, it is often important to conduct multiple trials of each experiment to fully characterise the networks, systems and performance. The results provided in this section are often given as either a mean accuracy, or a mean accuracy and standard deviation, as is common practise within the machine learning community.

However, results presented in such a manner only fully characterise the error distribution when the errors are normally distributed, and this is often an implied assumption when stating results in such a fashion. This section seeks to explore and characterise the nature of the errors arising from a typical SKIM experiment, and to validate this assumption for a typical SKIM network. All the statistics reported use either the standard t-test or the paired t-test as appropriate.

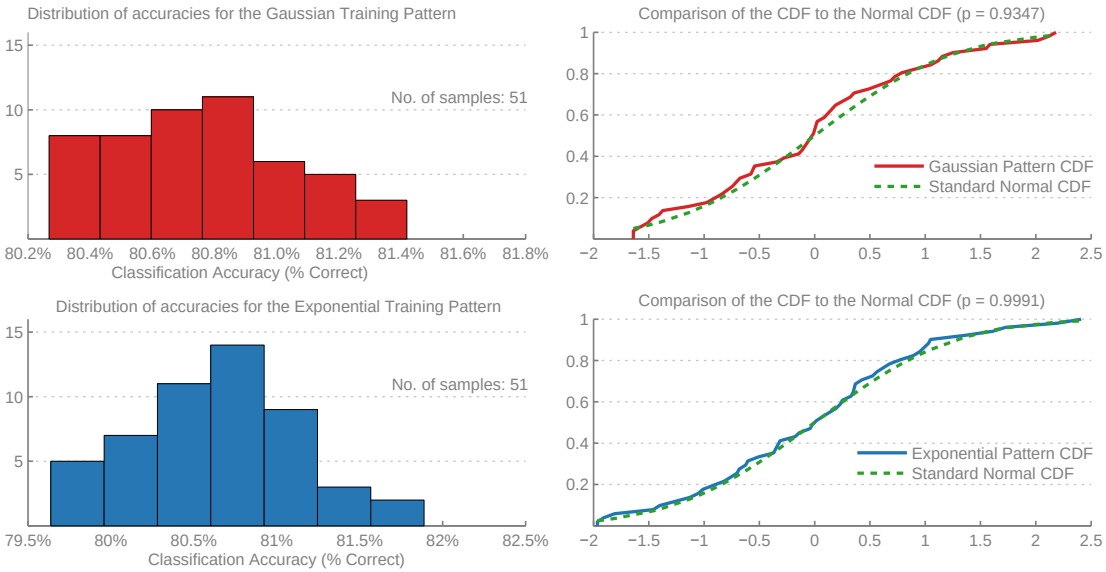



Figure 4.10: **Histograms of the distribution of errors and their CDFs for the Gaussian and Exponential training patterns.** The results were calculated over 51 independent trials of each network. A comparison to the CDF of a Standard Normal distribution is included, and the p-value for a one-sample Kolmogorov-Smirnov test provided, demonstrating that both distributions are normally distributed.

The network chosen to characterise was a standard SKIM implementation with 1000 hidden layer neurons. A number of sections in this thesis, such as the downsampling study presented in Section 4.6, make use of this network configuration when exploring other aspects of the feature recognition problem. This network represents a good balance between accuracy and training time, making it well suited to experiments that require multiple trials.

Testing included the two variations of this network, making use of the Gaussian and Exponential training patterns introduced and discussed in Section 4.4.5 (which includes a discussion and comparison of the training patterns). The characterisation involved 51 complete tests on each network, with only the random weight matrix and training order varying from trial to trial.

The networks with the Gaussian and Exponential patterns received the same random weights for each trial, and the Area method of the output determination methods run on the same network output (see Section 4.4.4 for a discussion on the output determination methods).

Figure 4.10 shows the distribution of accuracies for the Exponential and Gaussian patterns for the 51 trials. A one-sample Kolmogorov-Smirnov test was used to test the normality of the distributions [148], and the null hypothesis was re-

tained for both the Gaussian pattern ($p = 0.9347$) and the Exponential pattern ($p = 0.9991$).

Furthermore, applying the Lilliefor's composite goodness-of-fit test of composite normality [149] (which itself is a specialised version of the Kolmogorov-Smirnov test) also retained the null hypothesis that the data is normally distributed ($P > 0.5$) for both patterns.

These results show that the output error are normally distributed, and therefore are sufficiently represented by the mean and standard deviation of the accuracies or error rates.

### 4.4.4 Output Determination in Multi-Class SKIM Problems

All prior work with the SKIM algorithm was always limited to training a single output neuron and employed a threshold to generate output spikes from the soma potential of output neurons. If trained with the same random weights and hidden layer configuration, it is possible to independently train and then combine multiple output neurons (and their associated thresholds) to implement a multi-class classifier. As the outputs are already spikes, it is possible to use existing spike-based techniques such as first-to-spike and a winner-take-all approaches to selecting an output class. Unfortunately, due to the need for fine-tuning and determining thresholds this approach does not scale well when dealing with datasets such as N-MNIST, and there exists a need for a more robust and automated means of output determination.

In practise, multi-class problems constructed from independently trained output neurons suffer from the need for individual and specific thresholds for each output class, or the use of a global threshold which is often sub-optimal, as the ranges and characteristics of the output neurons may differ. This introduces additional parameters into the classification methodology, which is difficult to empirically determine given the size of the datasets and the time required to train on them.

In response to this issue, the approaches detailed in this section all serve to remove the need for fixed thresholds, and replace them with a comparison between output neurons directly. For this approach to work, the outputs must therefore be relative in magnitude, which requires the simultaneous training of all the output classes. Although the OPIUM method underpinning SKIM does include a normalisation step, the range of the linear weights can vary from class to class when training individually, and prevents the direct comparison of output class values. When training all outputs simultaneously with SKIM (underpinned with OPIUM), the normalisation applies to all output weights, keeping them

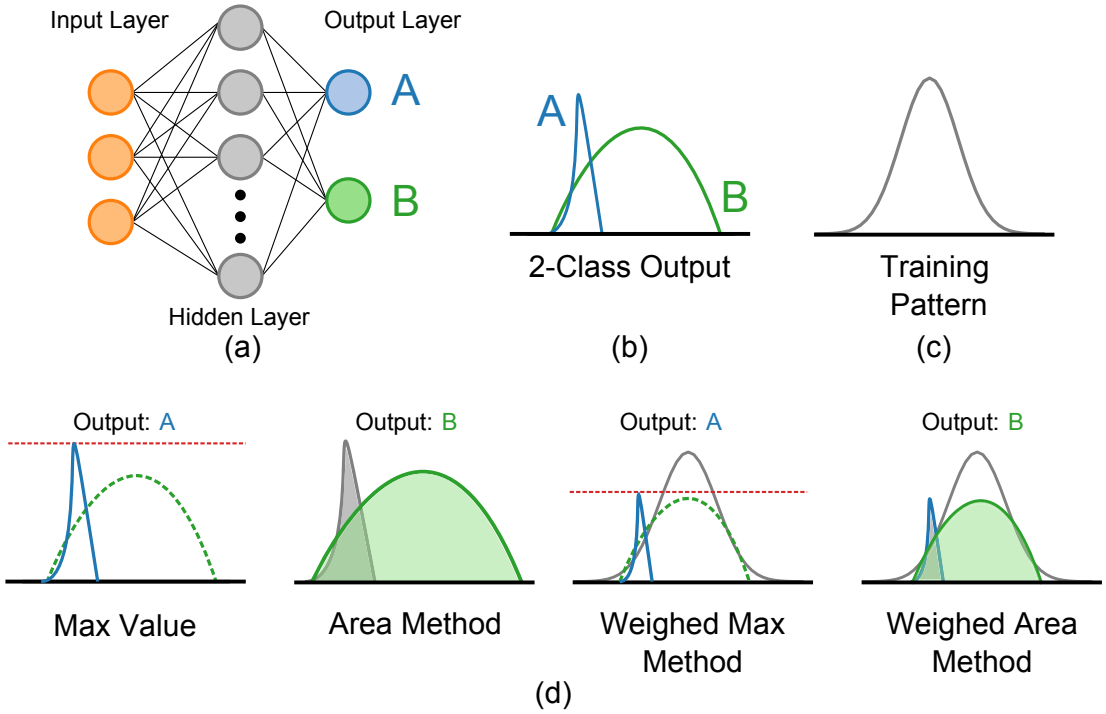



Figure 4.11: **Diagram showing the four output determination methods evaluated for use in multi-class SKIM problems.** (a) An example SKIM network with two simultaneously trained output neurons coding for two classes; A and B. (b) Example outputs from the two output neurons during the period in which the training spike occurs. In this example, the training utilised is the Gaussian training spike shown in (c). Diagrams showing the manner of calculating the four output selection methods are presented in (d), and include a label showing the theoretical winning class in each case. Note that for the Weighted Sum and Weighted Area methods, the training spike is also shown.

relative in magnitude to one another.

This section proposes and investigates four approaches to determining the output in a multi-class problem using SKIM, primarily applied to the N-MNIST dataset and also applicable to the multi-class classification problems in the N-Caltech101 dataset (see the 5-way classification problem and full classification problem in Section 4.5.5). Each method utilises the real-valued output from each of the output neurons during the section of the pattern in which the supervisory learning signal is expected to be present. There is no thresholding, and the methods operate on the real-valued outputs from the output neurons.

Figure 4.11 demonstrates the four methods used. The first approach is the Max Method, and simply takes the output class that achieves the maximum value

during the output period. This maximum does not necessary have to correspond with the intended location of the maximum in the training signal, but simply represents the maximum of any output class during the output phase. The second approach calculates the area under each curve, and selects the output class with the highest overall area. This is analogous to integrating the incoming values, and picking the highest value. It is important to note that the output can be either positive or negative, and any areas arising from portions of the curves below zero are negative in value and require subtracting from the total positive area.

The third and fourth methods exploits knowledge about the training pattern, and attempts to weight the outputs accordingly before applying the same techniques used in the first two methods. This has no effect when using a square training pulse, but has a significant effect when using a Gaussian or exponential training sequence (as the flat pattern results in a uniform pattern as shown in Figure 4.13). The third method weights the outputs proportionally to the training pattern, and then finds the maximum value. This method, dubbed the Weighted Max method, places emphasis on the portions of the output range where a high value is expected. The fourth method, referred to as the Weighted Area method, weights the output values and then calculates the areas under the curve and selects the highest output.

Figure 4.12 shows a comparison of the four output methods over a 61 trials of a SKIM network consisting of 1000 hidden layer neurons and trained using a Gaussian training pattern with a $\mu$ of 10 and a $\sigma$ of 5. The network structure and training order remained constant between each trial, with only the random input layer weights differing from trail to trial. Each output determination method ran on the same output for each trial, and calculated classification accuracy in terms of percentage of digits correctly identified.

It is immediately and apparently clear from the figure that the Area method produces the best result overall ($p < 0.00$). The performance of the other two methods did not show any statistically dominance at the 5% significance level.

The superior performance of the Area Method over the Weighted Area method is an interesting result, and shows that the learning mechanism makes use of the whole training pattern, and not simply the maximum value. As this method consistently produces the best results, all experiments henceforth report this result unless otherwise specified.

### 4.4.5 Analysis of Training Patterns for SKIM

In theory, a SKIM network should require only an output spike as a supervisory training pattern. In reality, a single spike (i.e. a pulse with a duration of a single time step) does not produce an error signal with enough magnitude or duration to allow rapid learning. It is possible to train with a short duration pulse, but it

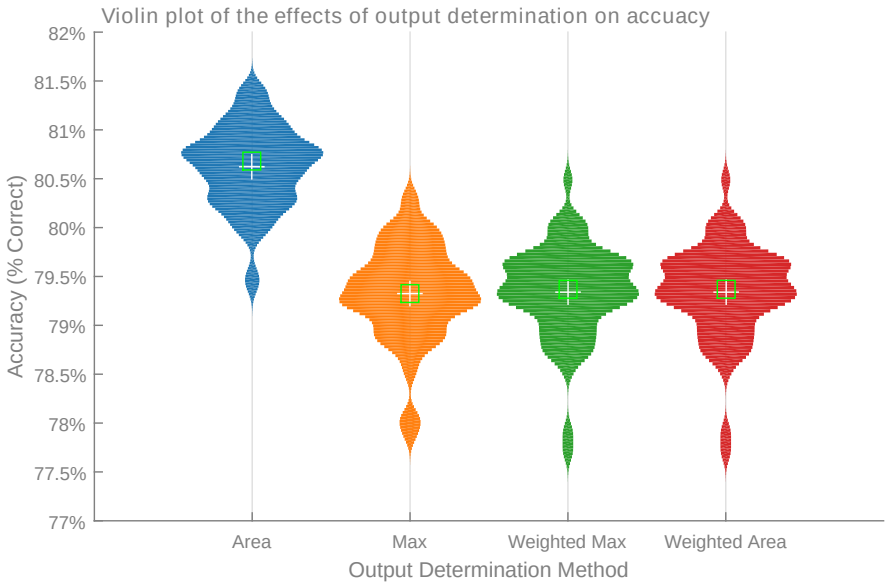



Figure 4.12: **Comparison of the effects of the four different output determination methods on classification accuracy with a Gaussian training pattern.** The figure shows the distribution of classification accuracies over 61 trials of a SKIM network with 1000 hidden layer neurons. Each network made use of the same random training order and hidden layer configuration, with only the random weights varying from trial to trial. The network made use of a Gaussian training pattern.

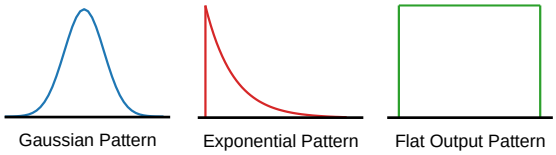



Figure 4.13: **Diagram showing different training patterns used to train the SKIM network.** The three training patterns described and tested in Section 4.4.5 are shown in this figure. The Flat Output pattern represents the logical extension of a single output spike and includes two sharp discontinuities on each end. The Gaussian pattern represents the opposite approach and contains a smooth curve with no discontinuities and peaks in the centre of the pattern. The Exponential pattern represents a combination of the two approaches, and contains an initial discontinuity followed by a smooth and gradual decrease.

requires multiple presentations of each digit and does not reliably converge. In place of a single spike, using a training pattern of a longer duration produces a better result without the need for multiple presentations of the training sequence. Having a pattern that spans multiple time-steps also allows the use of different training patterns, which can have a significant impact on both the training itself and the most appropriate method of determining the output class.

Figure 4.13 shows three different training patterns that produce good results with the SKIM network. The flat output pattern is the logical extension of the single output spike, but has two sharp discontinuities on each end. The Gaussian pattern represents the opposite approach, and exhibits a smooth (although discretised) curve which peaks during the middle of the output pattern. The exponential pattern represents the combination of the two approaches, and maintains the initial discontinuity but gradually decreases so as to exhibit a smooth return to zero.

To evaluate the performance of these training patterns, full tests against the N-MNIST datasets were performed. Each network contained 1000 hidden layer neurons, and made use of the full training set. Sixty trials of each experiment were performed, and the average error rate and standard deviation reported. These two properties are sufficient to characterise the classification accuracy as this exact configuration exhibits normally distributed output weights, as shown in Section 4.4.3.

The same random weights and hidden layer alpha functions were maintained across all trials, with only the training pattern varied across the tests. The classifiers all achieved accuracies in the range of $77\% \pm 2.2\%$, which is consistent with the results expected for a network with 1000 hidden layer neurons (see Section 4.4.2). The experiments and tests (along with all others in this section) make use of an output pattern of 10 time-steps in length.

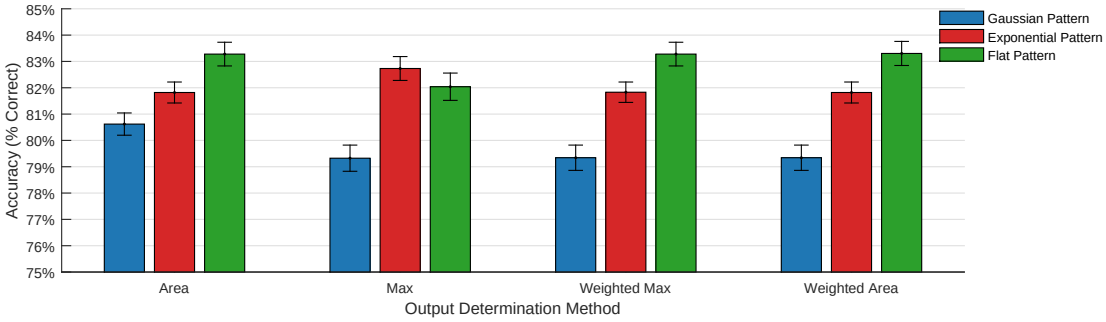



Figure 4.14: **Classification error when using different training patterns and output determination methods.** Sixty trials for each pattern were conducted and the mean training accuracy shown in terms of percentage correctly identified. It is immediately clear from the figure that the Gaussian training pattern produces the lowest classification accuracy across all output determination methods. The Flat pattern produces the best results for all methods with the exception of the Max determination method. In that case, the Exponential pattern produced the best results.

Figure 4.14 shows the results of the training the three training patterns for the four output determination methods presented in Section 4.4.4. The graph shows the mean error rate across all sixty trials for each training pattern. These results indicate that the Flat training pattern produces the best results in every case except for the Max determination method, and that the Gaussian method produces the worst result in every case.

Figure 4.15 presents a further study into the effect of training pattern and output determination methods. The figure shows the distribution of accuracies for the exponential and the Gaussian training patterns over sixty independent trials with a network configuration containing 1000 hidden layer neurons. The random weights and training order varied between trials, but not between training methods for each trial. Two output determination methods, the Area method and the Max method were assessed, and the histograms for each overlaid for both training patterns.

The histograms show an overall boost in accuracy when using the exponential method, for both output determination methods. Table 4.4 shows the mean accuracy and standard deviation resulting from the trial over sixty independent tests for all three training patterns.

A difference was not observed in the performance of the area method under different training patterns ($p = 0.091$), but the performance of the max method greatly improved ($p < 0.00$). The standard deviation in the results did not vary between the two output determination methods, and remained consistent with

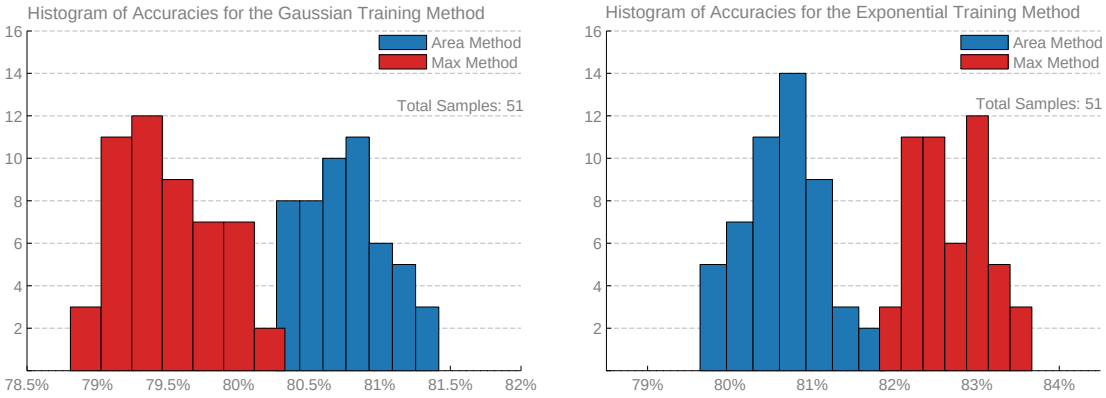



Figure 4.15: **Histograms of accuracies showing the comparison between the Gaussian and Exponential training patterns under the Max and Area determination methods.** The figure shows the histograms of accuracies obtained when using the Exponential and Gaussian patterns under the Max and Area output determination methods. It serves to demonstrate the important link between training method and output determination method. It is clear that the Exponential training method is a better choice when using the Max output determination method. This is of particular importance as the Max output determination method represents the easiest method to implement in hardware.

Table 4.4: **Mean accuracies and standard deviation for the comparison between the Exponential, Flat and Gaussian training patterns.** This table presents a comparison of the different training patterns under both the Area and Max output determination methods.

| | Gaussian Pattern | | Exponential Pattern | | Flat Pattern | |
|---|---|---|---|---|---|---|
| | Area | Max | Area | Max | Area | Max |
| Mean | 80.62% | 79.32% | 81.82% | 82.73% | 83.28% | 82.04% |
| STD | 0.42% | 0.49% | 0.45% | 0.40% | 0.45% | 0.51% |
| Max | 81.42% | 80.33% | 82.62% | 83.67% | 84.17% | 83.16% |
| Min | 79.41% | 77.88% | 81.12% | 81.82% | 82.36% | 81.06% |

| Training Pattern | Recommended Output Method |
|---|---|
| Gaussian Pattern | Area Method |
| Exponential Pattern | Max Method |
| Flat Pattern | Area or Max Method |

Table 4.5: **Recommended output determination methods for different training patterns**

the results found in Section 4.4.3.

Further investigation into these results shows that the initial discontinuity present in the flat and exponential patterns is the primary source of the performance improvement. The discontinuity produces a large and sudden spike in the error signal for the update stage. The Gaussian method produces a smooth error signal without any discontinuities, which has the effect of smoothing away the maximum peak. For this reason, the Max method is least effective when used with the Gaussian pattern.

Figure 4.15 also demonstrates an important link between training pattern and output determination method, and suggests that the choice of training pattern determines the optimal output determination method. The results show that the Area method is the best choice when using a Gaussian training pattern, and the Max method produces the best results when using an exponential training pattern. This makes sense when considering that the area under the Gaussian training pattern is larger, and thereby increases the total area under the output curve during recall. In a similar fashion, the sharp discontinuity, and resulting spike in the error signal, creates a more pronounced maximum value at the onset of the output pattern.

The flat output pattern benefits from the same effects from the sharp initial discontinuity, and also from the sharp negative error in the training signal resulting from the second discontinuity.

Table 4.5 summarises the recommended output determination method given the different training patterns. For reference, all experiments make use of the Gaussian pattern and the Area method unless otherwise specified.

One significant finding arising from the comparison of training patterns and output determination methods is that the Max method produces the best results when trained with an exponential pattern. This is important as the Max method is perhaps the simplest way of determining outputs in a system as it requires only a comparison operation.

### 4.4.6 Conclusions

As the N-MNIST dataset is a new dataset, the results presented in this thesis form the initial classification benchmarks. In an effort to provide a context in which to interpret these results, the statistical classifiers detailed in Section 6.5 attempted to set a theoretical lower bound on performance. These measures are important to understanding the nature of the classification problem, but additional value is gained from applying existing and state-of-the-art spike-based classification techniques to the classification problem. The comparison results presented in this section represent an initial application of existing techniques but have not benefited from any problem-specific optimisations or tuning beyond those required to run the systems on the N-MNIST dataset, and it is expected that better accuracies can be achieved.

As a point of comparison, the HFirst algorithm [6] was applied to the N-MNIST dataset. HFirst is an implementation of the HMAX algorithm [136] that makes use of spiking neurons and operates in an event-based manner. Table 4.6 contains the parameters used to configure the HFirst model.

Table 4.6: **HFirst parameters for the N-MNIST Dataset. These parameters were used to generate the results of the HFirst model applied to the N-Caltech101 and N-MNIST datasets. These same parameters were used for both the hard and soft HFirst classifiers.**

| Layer | $V_{thresh}$ | $I_l/C_m$ | $t_{refr}$ | Kernel Size | Layer Size |
|---|---|---|---|---|---|
| S1 | 150 | 25 | 5 | $7 \times 7 \times 1$ | $34 \times 34 \times 12$ |
| C1 | 1 | 0 | 5 | $4 \times 4 \times 1$ | $9 \times 9 \times 12$ |
| S2 | 150 | 1 | 5 | $9 \times 9 \times 12$ | $1 \times 1 \times 10$ |
| C2 | 1 | 0 | 5 | $1 \times 1 \times 1$ | $1 \times 1 \times 10$ |
| Unit | **mV** | **mV/ms** | **ms** | **synapses** | **neurons** |

Ten S2 layer neurons were trained (one representing each output class) and the synaptic weights for each were determined by collating and summing the output spikes from the C1 layer. This process was performed independently for each digit, and across the whole training set. Two classification methods were presented in [6]; a hard and a soft classifier. The hard classifier picks the class that generates the most output spikes during the sequence, whilst the soft classifier makes use of a calculated percentage probability that represents the percentage of total output spikes attributed to that class. If an output neuron produces no output spikes, it is assigned an accuracy of 0%.

The hard classifier produced an accuracy of 71.15%, whilst the soft classifier

produced a lower accuracy of 58.40%. These results are noticeably lower than the accuracies achieved on the 36-character recognition task presented in [6]. It should be noted that the digits used in those experiments exhibited far less variation between digits (see the description of the dataset in Section 4.3, and the apparatus used to create them in Figure 4.3), which could account for the lower accuracy. In addition, the optimisations of the HFirst model focused on detecting small objects, and the method not received the same level of specific optimisation and tuning yet.

This section provides the first steps toward an implementation of an event-based classification system. In these examples, the SKIM classifier is used both as a means of learning the spatio-temporal patterns and as a rudimentary feature detector. Due to the size and scale of the classification task posed by the N-MNIST dataset, adaptations to the SKIM method were required and were extensively characterised.

This is important as it allows for these methods to be applied to the feature detectors introduced in this work, and also serves to guide future hardware implementations. The finding that the Max output determination method works best with the Exponential training pattern is of significance, especially if implementing SKIM on existing hardware platforms such as NEF [100] and SpiNNaker [86].

The focus of this work lies primarily on the feature detectors and processing techniques required for event-based visual systems. Although a number of existing spike-based learning techniques are suitable for the classification task, a single network was chosen for all classification tasks to allow for comparison between tests and results.

## 4.5 Object Classification on the N-Caltech101 Dataset

The testing results on the N-MNIST dataset show that the event-based processing system is well suited to performing a classification task when provided with a large dataset containing a roughly equal distribution of object classes. The SKIM algorithm successfully generalised and produced good classification results when trained on less than 5% of the total available training samples. Although this is an impressive result, it still requires between 1500 and 2000 training presentations (approximately 150 to 200 samples from each category) before the classification accuracy begins to stabilise.

The N-Caltech101 problem poses an interesting challenge for the event-based processing system. The structure of the dataset provides more categories with fewer samples, has varying image sizes, and contains images of real-world objects

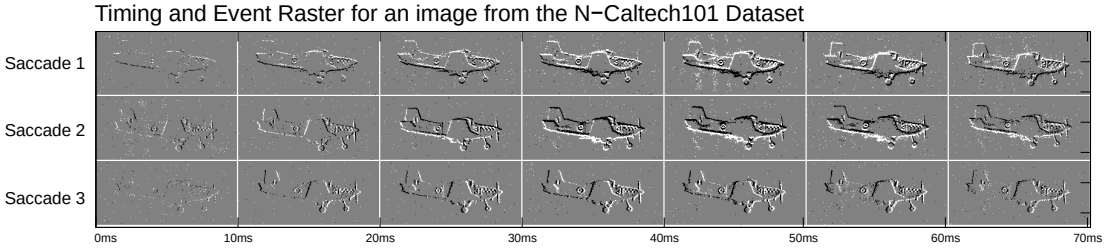



Figure 4.16: **Example of a sequence from the Airplanes object category in the N-Caltech101 dataset**. The above images represent frames created by accumulating the events captured over 10 ms time periods. Black pixels represent ON events generated by the ATIS sensor and white pixels indicate OFF events. Grey regions indicate that no events were received.

that do not exhibit the same consistency between samples of the same class that is present with the handwritten digits in the N-MNIST dataset.

The visual classification system, comprising the filter chain and the SKIM classifier, is not well suited to handling these problems. It does no pre-processing of the inputs and performs no explicit feature detection or extraction. This section explores the use of a visual processing system similar to the one presented in the previous section, and specifically adapted for tackling this dataset and the issues surrounding the implementation of a larger and more complex classification task.

### 4.5.1 Classification Methodology

All the work performed on the N-Caltech101 dataset attempted to use as much of the same processes and methodology as used in the classification of the N-MNIST dataset detailed in Section 4.4.1. In fact, these experiments utilise the same processing framework, with only minor modifications made to support the differing nature of the input data and the larger number of output classes. The same iterative SKIM method formed the underlying learning component, and employed the same methods to convert the event streams into spatio-temporal patterns. The majority of the alterations concern the handling and extraction of sequences from the dataset, the generation of training and testing splits, and the type of classification task required. This section proceeds to explore these changes and considerations.

Figure 4.16 shows an example of a sequence from the Airplanes category of the N-Caltech101 dataset. It shows the three saccades as rows, with frames generated over discrete time-steps of 10 ms. The structure of the data, and the encoding as AER events is identical to the sequences in the N-MNIST dataset, with the only differences being the range of the $x$ and $y$ addresses.

Processing the N-Caltech101 dataset poses a different problem to that of the N-MNIST dataset, and requires special consideration. Whereas the splits between testing and training data are explicitly defined for the M-NIST dataset, such an explicit division does not exist for the N-Caltech101 dataset. Complicating the task further is the varying number of training images in each category, and the presence of a background class which contains no defined objects. There exist a number of different training and testing regimes used by other researchers, and using these methods allows comparison to their results.

The N-Caltech101 dataset lends itself to two separate tasks; an object recognition task and an object classification task. In the object recognition task, the goal is to discriminate between elements of the class and the background class. It is inherently a binary problem in which each output class requires independent testing. The object classification task is the more challenging task, in which the output needs to be classified as one of the output classes. This section contains a thorough discussion of the implementation of these two problems.

Common to both problems is an issue arising from training with the full dataset, as it introduces a bias toward object classes with a higher number of available sequences. Due to the order of magnitude difference between the lowest and the highest number of objects per category, this can have a significant impact on the results. When training a multi-class problem, restricting the number of training items to that of the smallest category is the most commonly adopted solution. This dramatically reduces the number of trainable items for most categories when performing a full 101 class classification for the object recognition problem.

This then raises the issue of how to select the subsets used for training. A subset is randomly drawn from each category, and trained in a random order. Running multiple trials of the same experiment with different random subsets reduces the effect of any bias from any particular subset. Ten trials of each experiment were performed for all the experiments listed in this section, unless otherwise stated, in line with other experiment methodologies that make use of the Caltech101 dataset. The computational cost of each test became the primary limiting factor in determining the number of trials to run.

It is also possible to make use of a leave-one-out cross-validation scheme for the testing of the algorithm, which would maximise the number of available training samples through testing every combination of subsets. The computational requirements arising from the number of training and testing procedures make this approach difficult to implement given the size of the SKIM network required.

Whereas the training is inherently limited to a subset of the dataset, the testing in the object classification task can make use of the full testing set. This is preferable, as having more available samples for testing yields a better indication of the performance of the system. The results of the testing, however, require

special handling due to the differing number of samples in each category.

It is also important to note that the background class should not be included in the full classification problem. It contains no valid objects, and the training mechanism in SKIM can produce random and erratic results if trained with it. Including the background class can have a dramatic effect on the overall accuracy of the system.

For the object recognition task, which is binary and therefore only considers the background class and each object class independently, the number of training samples can be raised to include an equal number of samples from both the background category and the object category. This increases the number of training sequences for certain object classes, but the differing number of samples per category again requires special consideration.

Due to the varying number of images per category, a smaller subset of categories is often used instead [6]. The subset consists of the five categories containing the largest number of images, and are the Airplanes, Motorbikes, Faces, Cars and the Background category. All the experiments presented in this section make use of either this five-way dataset or the full 101-way dataset.

### 4.5.2 Handling Non-Uniform Inputs

The varying image sizes pose a challenge to the implementation of an event-based visual processing system that uses a SKIM network to classify the N-Caltech101 dataset. The challenge is primarily a practical one, relating to the mapping of pixels to SKIM input channels. This problem does not exist when dealing with the N-MNIST dataset as all the sequences have the same dimensions. The conversion process outlined in Section 4.3.3, details the steps taken to preserve this uniformity in the N-MNIST dataset. The choice to maintain aspect ratio in the images of the Caltech101 dataset makes it impossible to follow the same process. SKIM requires a fixed number of input channels, and the choice of mapping strategy can have a dramatic effect on the classification system. This same limitation is also present when using other classification systems, as the majority of methods requires an input layer of fixed and known size.

The channel limitation in SKIM is a result of the random weights that connect to the input layer to the hidden layer. This mapping effectively describes the receptive field of each hidden layer neuron. In the original SKIM implementation, this receptive field is a randomly weighted sample of the entire image space, and needs to remain exactly the same during all phases of training and testing. With differing image sizes, it becomes difficult to ensure that the mapping of input channels to input weights remains consistent and logical. Making use of a feature extraction layer between the input and the SKIM network can alleviate this problem altogether by producing a new event-stream in feature space rather

than 2D space.

There are two direct strategies for overcoming the difference in image sizes. The first method is to simply crop each image to the dimensions of the smallest image in the dataset. This is by far the simplest strategy to implement, and it has the added advantage of negating any classifying power that the image size may convey (see Figure 4). Unfortunately, the image sizes vary from $(41 \times 51)$ pixels to $(223 \times 173)$ pixels, and cropping would thereby discard a significant portion of most images. Regardless of the chosen strategy, it would be desirable to keep as much of the information from each image as possible.

The second strategy takes the polar opposite approach and ensures that all training sequences have the same number of channels as the largest image in the sequence. This results in the majority of images having a large number of empty channels, but has the advantage of not discarding any information. The network does encode the size of the image, both directly through the sharp contrast edges around the border of the original image, and indirectly through the ratio of event spikes to total number of pixels. Under this methodology, each sequence (corresponding to an image in Caltech101) requires centring within a larger frame of $(223 \times 173)$ pixels, requiring SKIM to have $38,579$ input channels.

Adding additional empty channels to SKIM contributes nothing to the classifier; neither does it have any negative impact on classification results. Empty channels produce no activation at any of the hidden layer neurons, and are therefore effectively invisible.

One of the interesting characteristics of the SKIM network is that the computational load does not increase exponentially with respect to the number of input channels, as it does with the number of hidden layer neurons. Adding hidden layer neurons requires an extra row and column in the inverse correlation matrix $\theta_k$, increasing the number of operations needed to perform the dot product and update by an exponential factor. Increasing the number of input channels, however, only results in a larger random weights matrix $w_{i,j}$, also requiring an additional row and column, but used only in a simple multiplication operation. However, the reverse effect applies to the testing, where the penalty is higher for additional input channels, and the number of hidden layer neurons is less significant as the inverse correlation matrix is neither used, nor updated.

### 4.5.3 Revising the Binary Classification Problem with SKIM

The original implementation of SKIM primarily tackled binary problems - detecting whether or not a specific spatio-temporal pattern is present in the input pattern. The implementations used to perform recognition tasks on the N-MNIST dataset employs SKIM in a slightly different manner, using it as a form of feature extraction by training it on complex input patterns with a supervisory learn-

ing signal. The N-MNIST task is a multi-class problem, rather than a binary one, making the determination of the final output far easier as it becomes a relative comparison between output neurons as explored in Section 4.4.4, rather than a binary decision on a single output. The object recognition task with the N-Caltech101 dataset is inherently a binary problem.

The output of the original SKIM network is a spatio-temporal pattern consisting of output spikes indicating the presence of the trained pattern. To convert the real-valued and continuous outputs of the hidden layer neurons (multiplied by the linear output weights) into a spike, the output neuron sums all the outputs from the hidden layer neurons and then emits a spike if the value crosses an empirically determined threshold. This threshold is usually kept constant throughout the entire training and testing regiment, and allows the adjustment of the sensitivity of the network.

When training large systems, such as the datasets used in this section, the process of empirically determining the threshold value becomes problematic, both computationally and when comparing different runs and trials. This free parameter can be removed by recasting the binary problem as a two-class problem, in which one output neuron codes for the presence of a signal, and the other codes for the lack of one. The two outputs are simply compared, with an output spike occurring if the response of the neuron coding for the presence of the pattern is greater than that of the one coding for the lack of it. In terms of implementation, both output neurons need only to produce an output after the presentation of the pattern.

Two strategies are proposed for coding for the lack of an output, and depend on the nature of the problem. Figure 4.17 shows a graphical representation of these two methods. The first strategy (shown as Method 1 in Figure 4.17) applies in situations where there is structure to both sequences in the binary problem. Comparing two classes from N-Caltech101 dataset is an example of such a situation, as both object classes contain sequences with common features or structures for the SKIM algorithm to learn. Under such a strategy, the output neuron coding for the lack of a signal does not continuously fire, but is only active during the same phase as the one coding for the presence of one.

The second strategy (shown as Method 2 in Figure 4.17) applies more generally, and is applicable to situations where it is not guaranteed that both classes contain data compatible with the learning mechanism employed by SKIM. The binary object detection problem represents a good example of such a situation, as there may not be any similarity between the images in the background class. Under such a system, training an output to represent the lack of an object would cause the algorithm to seek features where none may exist, and therefore to become unstable. Training the output with a continuous signal of a lower magnitude that the positive threshold produces a superior result. This signal becomes

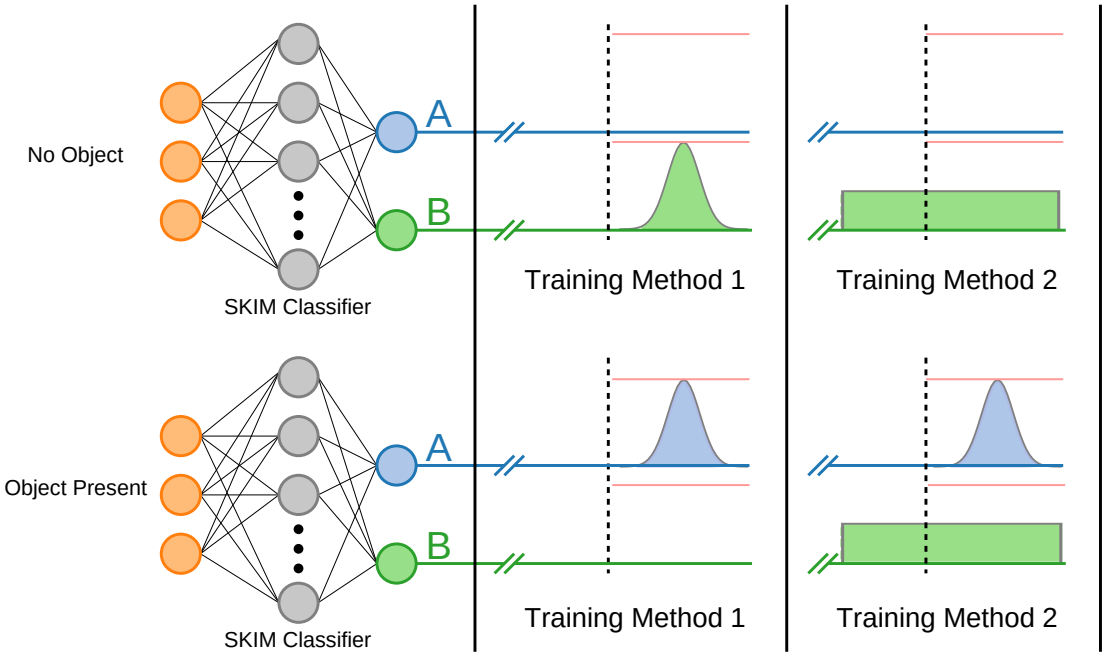



Figure 4.17: **Illustration of the two methods for performing automated binary classification using SKIM without the need for explicit thresholding**. The two methods show how the binary problem can be recast as a two-class problem. Two methods are presented, which cater for different types of problems. Method 1 deals with cases where there are features in both conditions of the binary problem, and Method 2 handles the case where the one condition does not necessarily contain any separable information.

a variable threshold calculated across every time-step of every training pattern.

The choice of training pattern for such a network is an important concern, as the output neuron coding for the lack of the object is sensitive to the sharp discontinuities found in the exponential and flat patterns (see Figure 4.13). For this reason, it is best to use the Gaussian training pattern for output of the output class coding for the lack of the object.

The secondary signal in the above configuration acts as a dynamic threshold, calculated from the same signal as the one used for classification. It is a relative and pattern-specific measure of activity, but one that improves with each presentation from the dataset.

As the binary task against the N-Caltech101 dataset involves separation from the background class, only the more general method (Method 2) is applicable. For this reason, it is the chosen method for all experiments performed in this section.

### 4.5.4 Object Recognition with SKIM

Implementing the binary classification problem with the same five classes used for the kNN approach in Section 6.5 using SKIM did not produce good results. The tests made use of the same methodology and dataset splits as used with the kNN classifiers, with equal number of samples drawn from each category and the background class. The subsets chosen from the category and the base class are always randomly selected from the full range of sequences available, and the order in which the training and testing proceeds is also random from trial to trial and from class to class.

Table 4.7 presents the results of the classification using varying numbers of hidden layer nodes for the four classes. Note that the comparison is always made to the background class and each result is the average computed across ten trials (in accordance with the methodology used in [150]). It is clear from the results that the classification accuracy is not far from chance (which would be 50%), thereby displaying little predictive power. Increasing the number of hidden layer neurons to 3000 did serve to increase the performance to levels slightly above chance, although the increase to 2000 hidden layer neurons paradoxically produced an overall weaker result across most of the classes.

These results are surprising, given the ability of the kNN classifier to discriminate with a far higher accuracy on the same problem and under the same conditions. The output of the kNN classifier also appeared more stable, and fluctuated less between trials. This may be due to the aggregating nature of the statistics used in the kNN classifier, and the difficulty in capturing the same statistical information given the nature of the SKIM algorithm and the large number of input channels used. The slight increase that arises from increasing the num-

Table 4.7: **Binary classification results of the N-Caltech101 dataset with SKIM.** The results for the binary classification task were calculated over 15 trials of each experiment and involved the comparison of the four most populated categories with the background class. As this is a binary classification task, the accuracy due to chance is 50%.

| Class | 1000 Neurons | 2000 Neurons | 5000 Neurons |
|---|---|---|---|
| Airplanes | $46.2\% \pm 5.1\%$ | $45.1\% \pm 3.0\%$ | $58.3\% \pm 5.2\%$ |
| Cars | $43.9\% \pm 4.7\%$ | $49.1\% \pm 5.6\%$ | $62.3\% \pm 6.6\%$ |
| Faces | $58.4\% \pm 3.4\%$ | $51.7\% \pm 1.0\%$ | $54.7\% \pm 4.4\%$ |
| Motorbikes | $57.7\% \pm 5.8\%$ | $41.9\% \pm 2.8\%$ | $52.4\% \pm 5.4\%$ |

ber of hidden layer neurons suggests that there may not be enough hidden layer neurons to adequately separate the classes in a higher dimensional space.

Using SKIM in this manner forces the algorithm to act like a feature extractor, and given the variability in input composition and size, the number of samples available for training is perhaps the most likely explanation for the poor performance. The largest of the five categories used in the above results contained only 428 sequences for training. Examining the convergence graphs for the N-MNIST problem shown in Figure 4.8 shows that the networks appear to converge within the first 2000 training samples for all hidden layer sizes. Unfortunately, the maximum available samples in any category falls well below this value, due to the inherent design of the Caltech101 dataset.

Given that the N-MNIST dataset provides a far easier challenge due to the consistent image size and the lack of background or clutter in the images, it is likely that a SKIM network trained on more complicated data, such as the sequences in the N-Caltech101 dataset, will require a higher number of training samples before the accuracy will converge. It is also likely that the equations for the number of hidden layer nodes will also change, primarily due to the increased number of input channels.

### 4.5.5 5-Way Object Classification with SKIM

The object classification task on the N-Caltech101 dataset is a difficult problem. Implementing such a task with a SKIM network presents additional challenges due to the immense size of the problem, the data and the inputs. The size of the problem also demands significant computational power, and a full classification result can take in excess of 24 hours to complete, even with all the optimisations outlined in Section 6.5. One of the major contributing factors is the number of

input channels required to cater for patterns in the N-Caltech101 dataset. As discussed in Section 4.5.2, the irregular sized images in the original Caltech101 dataset require 38,579 input channels in order to fully cater for all image sizes. Compared to the 1,156 required to support the $34 \times 34$ patterns in the N-MNIST dataset, this is an enormous increase which presents a difficult problem for optimisation as these input channels need interact with a similarly large matrix of random weights. The essence of this operation is a large multiplication requiring either sequential execution on the CPU, or to incur the penalties involved with transferring the entire input pattern vector to the GPU for parallel execution.

There is also a ten-fold increase in output classes (101 for N-Caltech101 as opposed to 10 for N-MNIST), which increases the complexity and computation cost of the update process required during training. The impact of increasing the number of output classes is less significant than it may appear, as it does not affect the size of the inverse correlation matrix $\theta_k$ (defined solely by the number of hidden layer neurons), and the update involving the most computationally intensive operation in the algorithm (consuming approximately 78% of processing time during training). Therefore, the number of output classes is not the factor that contributes to the slower execution time and even the smaller 5-way classification problem suffers from the same training speeds per training sequence, albeit with far fewer training elements available for training.

The 5-way classification problem provided the initial means for evaluating the performance of SKIM against the N-Caltech101 dataset. The experiment followed the structure of the initial 5-way classification tasks performed using the kNN classifiers in Section 6.5. The testing regiment used the same classes as the previous experiment, but trained in a different random order as there is no explicit training order in the kNN classifiers. The overall accuracies shown represent the averaged accuracy over ten trials of each experiment, but the confusion matrices displayed show only the results of the last trial.

Figure 4.18 shows the confusion matrices and accuracies for the 5-way classification problem performed with 1000 and 5000 hidden layer neurons. It is important to note the inclusion of the background class in these results, despite the fact that training an output neuron for the background class with SKIM can become unstable (see the discussion of training for non-features in Section 4.5.3). The inclusion of the background class is solely to provide consistency with the kNN classifier methodology (see Appendix A) and therefore allow for the comparison of the results.

The networks achieved a performance of 48.26% for the 1000 hidden layer neuron configuration, and 59.66% for the 5000 hidden layer neuron configuration across the ten trials performed. The confusion matrices show that the increase in hidden layer neurons resulted in a more even spread of errors across the classes. The confusion matrix from the 1000 hidden layer neuron configuration shows a

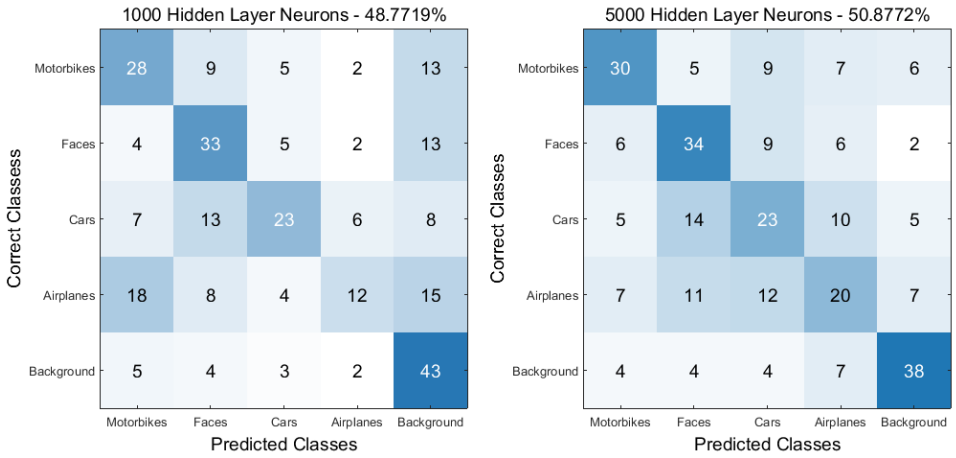



Figure 4.18: **Results and confusion matrices for the 5-way classification performed using SKIM networks of 1000 and 5000 hidden layer neurons**. The confusion matrices for the 5-way N-Caltech101 classification task demonstrate the effect of hidden layer size on classification accuracy. It is important to note that the inclusion of the background class can cause instability in the SKIM network due to the lack of consistent features, and was only included to allow for comparison with the kNN classifiers explored in the Appendices.

bias toward the Airplanes and Background classes.

Increasing the number of hidden layer neurons improved the results slightly, but also changed the distribution of errors. The network no longer favoured two classes, but rather spread the results more evenly, achieving a more balanced accuracy among the classes. The large increase in network size only produced a small gain in accuracy, although given the nature of the problem and the lack of suitability of the SKIM algorithm, it is still an interesting result.

In comparison the results achieved with the statistical classifiers presented in Section 6.5, the SKIM results beat all but the classifier based on the Standard Deviation of the $y$ addresses, which achieved an accuracy of 64.82% on the same task. For purposes of comparison, the classifiers based on the $x$ and $y$ sizes (the performance of which exceeded that of all the others) were not included as the pre-processing performed on each image seeks to negate that classification advantage due to image size. As spatial information is discarded in the conversion from event streams to input channels, it is possible that the information encoded in the distribution of pixels in the Y-axis are no longer available to the SKIM classifier.

To further investigate the performance, the background class was removed from the classification problem. Figure 4.19 shows the confusion matrices and

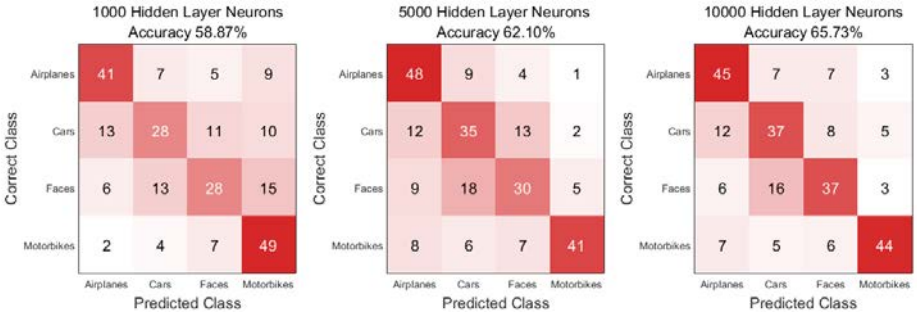



Figure 4.19: **Classification results and confusion matrices for the 4-way classification problem with SKIM.** The confusion matrices presented here show the effect of the hidden layer size on the 4-way classification task. In this case, the background class was removed, resulting in an increase in classification accuracy. Although the number of classes decreased, the classification accuracy of the 1000 hidden layer neuron configuration in the 4-way task exceeded the 50.87% accuracy achieved using 5000 hidden layer neurons in the 5-way task.

accuracies for three SKIM networks, with 1000, 5000 and 10,000 hidden layer neurons respectively. Results for a range of different hidden layer sizes are also given in Table 4.8. The accuracy improved as the number of hidden layer neurons increased, achieving an accuracy of 65.73% when trained with 10,000 hidden layer neurons. Computational and memory limits prevented further experimentation.

Table 4.8: **Results of the 4-way classification of the N-Caltech101 Dataset with SKIM.** The results of the 4-way classification task (Airplanes, Cars, Faces and Motorcycles) are presented along with the standard deviation across 15 trials of each classification task.

| | Hidden Layer Size | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 1000 | 2000 | 3000 | 4000 | 5000 | 6000 | 7000 | 10000 |
| Mean | 58.87% | 62.50% | 58.07% | 57.66% | 65.32% | 62.10% | 57.66% | 65.73% |
| STD | 3.22% | 4.78% | 4.12% | 2.03% | 4.44% | 5.01% | 3.32% | 1.89% |

The 4-way classification task produced better results than the 5-way example, at both hidden layer sizes. Figure 4.20 shows the accuracies for each class in both the 4-class and 5-class problems for a network consisting of 5000 hidden layer neurons. The 4-class classifier is superior in three out of the four classes, and by a wide margin in the case of the Airplane and Motorbike classes.

Although the 4-way classification problem is essentially a subset of the 5-way classification task, the addition of the output neuron coding for the background

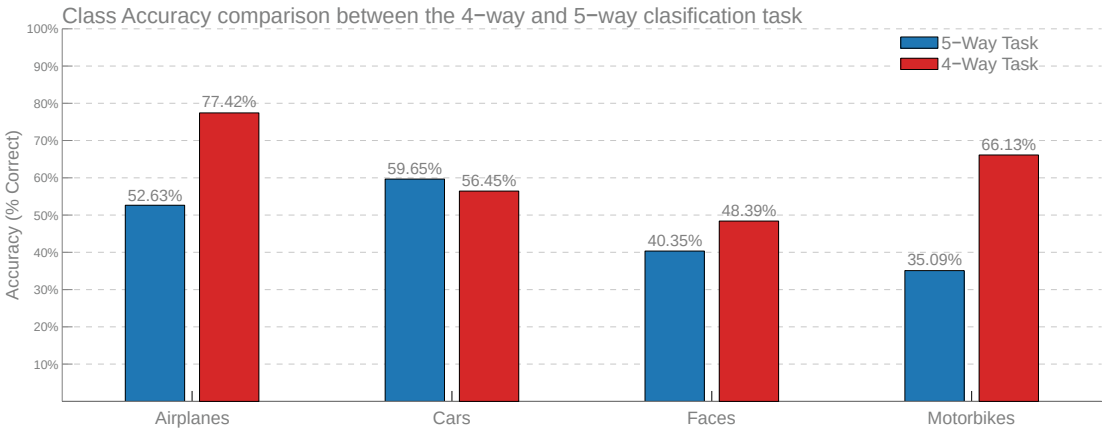



Figure 4.20: **Comparison per class of the results from the 4-way and the 5-way classification problems.** This comparison demonstrates how the inclusion of an additional class can impact the accuracy of a classification network. Although the 4-way classification task is a subset of the 5-way classification task, it is clear from the above figure that the 4-class classification system produces superior classification results in three out of the four classes. The same number of training samples was used per category to generate these results (57 samples).

class impacts the accuracy in two ways. Firstly, the addition of another class affects the output determination aspect of the problem, allowing for potential mis-classifications. This also affects the accuracy due to chance, which increases from 20% in the 5-way problem, to 25% in the 4-way version.

A second manner by which the inclusion of an addition class can affect accuracy is through alterations to the training regiments, as it needs to include the additional image sequences for the additional class. As all the output neurons require training simultaneously, when one class is receiving a learning signal, the other classes receive a suppressing signal to prevent their firing. These two factors are responsible for the difference in performance shown in Figure 4.20.

### 4.5.6 101-Way Object Classification with SKIM

Applying the same classification methodology to the full classification problem yielded the results shown in Table 4.9. Although these results represent low accuracies, they remain well above chance (which is less than 1% accuracy) and show an increase with the number of hidden layer neurons. These results show promise given that the original HMAX algorithm achieved an accuracy of 44.6%± 1.14% [136] and a similar performance of 47.2% ± 1.0% was achieved with the FPGA version of HMAX presented in [150]. Given that the design of the HMAX algorithm focused on object recognition, the performance of the SKIM algorithm

Table 4.9: **Results of the full 101-way classification of the N-Caltech101 Dataset with SKIM.** Note that due to the size of these datasets and the computation involved, only a single run of these experiments were performed.

| Hidden Layer Size | | | | | | |
|---|---|---|---|---|---|---|
| 100 | 1000 | 2000 | 3000 | 4000 | 5000 | 10000 |
| 3.47% | 7.43% | 8.60% | 8.98% | 9.65% | 9.65% | 11.14% |

on this task - with no explicit feature detection - is remarkable.

Figure 4.21 shows the confusion matrix for the full 101-class recognition task. The network utilised 10,000 hidden layer neurons, and performed no spatial or temporal downsampling. The N-Caltech101 dataset represents the first encoding of the Caltech101 images into an event-based framework, and this work forms the basis of a benchmark with which to compare future results.

The results shown in Table 4.9 show that the accuracy increases with the size of the hidden layer. Unfortunately, networks with more than 10,000 hidden layer neurons become too memory-intensive for simulation using current hardware and limits further exploration in that direction. Instead, there are other areas in which to improve the results. Spatial and temporal downsampling may yield better accuracies, and placing a dedicated feature detection layer before the input layer will allow the SKIM network to operate in a much smaller feature space, than the spatial dimensions in which the above tests operate.

### 4.5.7 Conclusions

This section demonstrates the application of the event-based visual processing system to the N-Caltech101 dataset. Building on the system presented in Section 4.4 for use on the N-MNIST dataset, this section presents the adaptations and modifications necessary to apply the filtering and classification system to the N-Caltech101 dataset.

Although the HFirst model was successfully applied to the N-MNIST dataset, the large image sizes prevented the use of the same implementation on the N-Caltech101 database. The work presented in this section forms the current state-of-the-art benchmarking result for the N-Caltech101 dataset, as presented in [144].

When compared to the results of the full 101-way classification performed using the statistics-based kNN classifiers presented in Section 6.5, these results demonstrate that the network is capable of learning and performing classification on more than just the statistical properties of the sequences. Even the results with only 100 neurons achieved an accuracy of 3.47%, which exceeds the performance

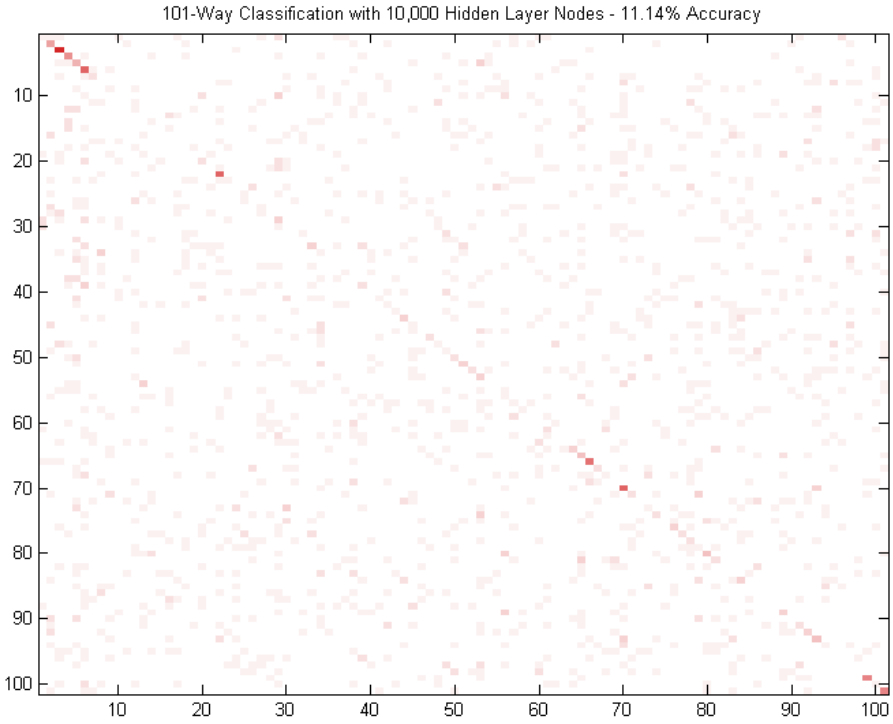



Figure 4.21: **Confusion matrix for the 101-way object classification task using SKIM with 10000 hidden layer neurons.** The network achieved an accuracy of 11.14% accuracy, and trained with 15 randomly selected samples from each category.

of all but one of the statistical classifiers. The only statistical classifier to produce a higher accuracy is the $y$ size classifier, which achieved 4.32%, but makes use of spatial information that is lost during the conversion to spatio-temporal patterns.

The use of a similar SKIM network to the ones used on the N-MNIST dataset allow for an examination of how the visual processing system handles the more challenging classification task posed by the N-Caltech101 dataset. This section presents a technique for handling non-uniform image sizes which is applicable to any spatio-temporal classification system requiring a fixed or known number of input channels.

It is clear from the results that the proposed visual processing system is capable of achieving accuracies above both chance and the statistical classifiers for the N-Caltech101 dataset. The N-Caltech101 dataset was created specifically to test object recognition using only a few samples for each category, and that the systems presented in this section are not optimised for such configurations.

## 4.6 Spatial and Temporal Downsampling in Event-Based Visual Tasks

This section explores the effects of performing downsampling operations on the input patterns prior to learning them with the event-based visual processing system presented in the previous sections. It discusses the means and effects of performing downsampling on the input data for the N-MNIST dataset, and performs similar operations on the MNIST dataset as an analogue. In addition, this section introduces a new dataset based on a rudimentary method of converting the MNIST dataset to a spike-based representation, and uses this to further explore the nature of downsampling with the SKIM algorithm.

This section also investigates the nature of downsampling as a method of feature detection for event-based data. The action of downsampling, both spatially and temporally, serves to aggregate information before it is presented to the classification system. As with the previous sections, the SKIM classifier is used in order to allow comparisons with other results obtained in this thesis.

### 4.6.1 The SpikingMNIST Dataset

The N-MNIST dataset represents a conversion of the original MNIST dataset into spikes through the use of a physical camera. This process introduces noise through stochastic noise events, latency in the AER arbiter and through mechanical motion. The creation of the SpikingMNIST dataset intends to provide an alternative means of converting the MNIST digits into a spatio-temporal for-

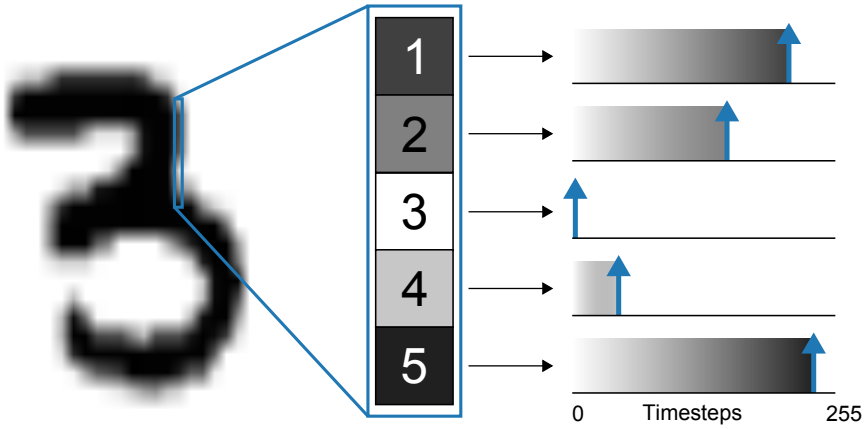



Figure 4.22: **Illustration of the encoding of intensity to delays used in creating the SpikingMNIST dataset.** Each pixel in the image is assigned to a separate input channel and the intensity encoded using the mechanism illustrated above. For each pixel, a spike is placed in its respective channel at a time proportional to the grey-scale intensity of the pixel. Thus, the darker the pixel, the later in the channel the spike will occur. As the input images in the MNIST dataset use a single unsigned byte to encode intensity, the pattern length is limited to 255 time-steps.

mat through a software process, thereby allowing for total control over the entire conversion process.

A variety of techniques exist for the encoding of real-valued inputs into spike trains or spatio-temporal patterns. The purpose of the SpikingMNIST dataset is to make use of the most direct means of encoding the MNIST digits into spikes, thereby introducing as little external noise into the system as possible. The conversion makes use of a simple, deterministic and straightforward encoding scheme over more realistic techniques such as probabilistic encoding and rate-based encoding.

The SpikingMNIST dataset converts the MNIST digits to spikes by assigning each pixel to an input channel and placing a spike in that channel at a time proportional to the grey-scale intensity of the pixel in the image. Figure 4.22 provides an example of this intensity-to-delay conversion process. This results in a sparse input pattern, as the original MNIST digits contain a number of empty pixels, which in turn results in spikes occurring in the first time-step of the pattern. It is important to include the spikes corresponding to an empty pixel, as the lack of a grey-scale value is information relevant to the classification of the digit. Leaving the spike out altogether results in an input channel with no spikes at all, which has no effect on the SKIM network. These empty input channels can be combined into a single channel, as having multiple identical

channels contributes no new information for the system and has no effect when using randomised input weights with all-to-all connectivity.

The original MNIST digits are $(28 \times 28)$ pixels in size, with each pixel containing a grey-scale value in the range $[0, 255]$. Encoding the intensity as time directly results in a pattern of length 255 time steps (excluding the output pattern), and 784 input channels [1]. The units of the time-steps remain entirely arbitrary, but choosing millisecond resolution makes them consistent with the N-MNIST dataset sequences.

In terms of implementation, the SpikingMNIST dataset converts the MNIST data into a stream of AER events that are directly compatible with the processing system used for the N-MNIST dataset. To ensure that the resulting dataset is directly compatible with the N-MNIST processing chain, the SpikingMNIST sequences make use of a spatial resolution of $34 \times 34$ pixels instead of the $28 \times 28$ from MNIST. The digits are centred in this larger spatial window. The conversion process encodes the images as spatio-temporal sequences but does not simulate the motion of the saccades used to create the N-MNIST dataset. The encoding as AER events allows for the same filtering chain to be used and the same methods for conversion into a spatio-temporal pattern that is compatible with the SKIM network.

### 4.6.2 Downsampling Methodologies

This section investigates two forms of downsampling, temporal downsampling and spatial downsampling, and considers the effects of downsampling on three different datasets which all represent the MNIST digits in varying forms. Although the application of these methods differ slightly in implementation, the overall structure of the three experiments remains the same.

Each experiment sweeps across a range of spatial and temporal downsampling parameters for each dataset. The relevant section for each dataset discusses the nuances and limitations placed on the downsampling operations. One common facet of these imitations is that these experiments only use integer steps when sweeping parameters and additionally attempt to remain symmetrical wherever possible.

The datasets all contain the full 60,000 training samples and each individual experiment trained on the full complement of samples. In line with the previous examples, each experiment used a random training order and random hidden layer weights. Each dataset also contains the full testing set, and each experiment tested all 10,000 sequences in a random order to generate the accuracies presented

[1] The $34 \times 34$ pixel sizes in the N-MNIST dataset arise from the need to leave space to accommodate for the motion of the $28 \times 28$ pixel digits. See Section 4.3.3 for more information on the conversion process for N-MNIST.

in the following sections. The testing set formed no part of the training and served only as a means to evaluate and report performance.

The number of hidden layer neurons required a constant value for all tests, as computation costs made it impractical to consider as a parameter over which to sweep. Every network in this section made use of 1000 hidden layer neurons, regardless of whether the underlying learning mechanism was SKIM or ELM. This value represents a balance between computational time and accuracy, and benefited from a thorough exploration in Section 4.4.3, which showed that the statistical distribution of the errors were normal and benefited from a standard deviation of $\sigma < 0.5\%$ with respect to classification accuracy.

Due to the number of tests required to fully explore the effects of downsampling, it was not possible to run multiple trials of each experiment, and only a single trial of each configuration was performed. As the nature of the distributions of errors has been explored, and as all tests are entirely independent of one another, the overall trend relating to downsampling can be observed whereas the importance of any single result is less significant.

It is likely that networks implementing larger hidden layer sizes will achieve better accuracies, in a similar fashion to N-MNIST.

### 4.6.3 Downsampling on the N-MNIST Dataset

The advantages of downsampling on the N-MNIST dataset are numerous. Firstly, it paradoxically produces an increase in accuracy for the networks of the same hidden layer size. It also requires either fewer time-steps or input channels, both which reduce computation time and similarly would reduce resource requirements if implemented in hardware. The act of downsampling inherently reduces the size of the input data, requiring less data to transmit. However, the event-based nature of the system retains all the benefits of ATIS camera, especially the scene-dependent data rate.

For the N-MNIST dataset, the downsampling occurs during the conversion of event-based data (the AER stream) to the spatio-temporal data provided to the input layer of the SKIM network, and is therefore an AER filter. The downsampling operates as an intermediate step, allowing it to generalise and operate on any 2D event-based information. As Figure 4.23 shows, the SpikingMNIST method takes advantage of this property by generating new event-based data from the MNIST digits and then passing it through the same intermediate layer as used for the N-MNIST data.

The downsampling operates on the events received in each training and testing sequence and adjusts these values accordingly, before converting the event-based data into a spatio-temporal pattern. This operation is just an AER transform operating on events, and allows for a theoretical hardware or straightforward

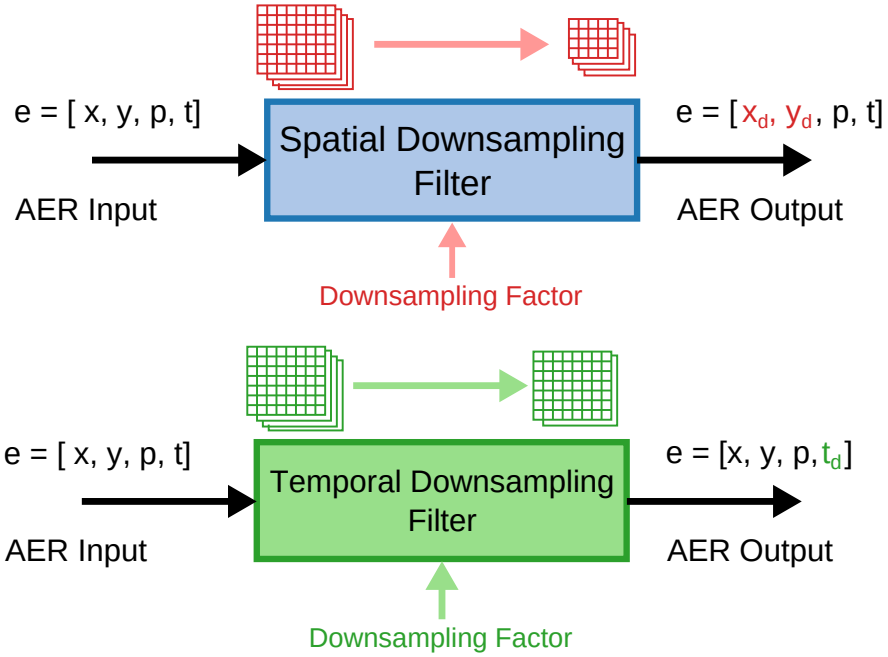



Figure 4.23: **Diagram of the structure and function of the AER-based downsampling filters** Both downsampling filters represent AER transformations and operate on AER data and produce output in the AER format. The spatial downsampling filter reduces the spatial resolution of the input data by a fixed downsampling factor, resulting in an output stream that maintains the temporal resolution of the input. The temporal downsampling filter reduces the temporal resolution of the input stream, leaving the spatial resolution unchanged. These filters can be cascaded to provide both spatial and temporal downsampling.

FPGA implementation. The spatial downsampling affects the mapping from $x$ and $y$ addresses to input channels, and the temporal downsampling alters the resolution and number of time-steps in the pattern.

The potential to implement in hardware, arising from using an AER filter to perform the downsampling, allows for on-board processing within the ATIS camera or acquisition device. This would have the added advantage of reducing the data rate from the camera directly.

The conversion process from events to spatio-temporal patterns discussed in Section 4.4.1 includes a discussion of the process for performing spatial and temporal downsampling, with the equation for spatial downsampling given in Equation 4.2 and the temporal downsampling provided in Equation 4.3.

The information contained within each event is integer in nature. The time-steps possess microsecond resolution encoded with a 32-bit integer value, the $x$ and $y$ pixel addresses are integers within the range of the camera frame and the

Table 4.10: **Downsampling factors and the resulting pattern sizes.** The table shows the resulting sequence sizes based on the spatial downsampling factor applied. Note that the sizes of the MNIST dataset are smaller as each original image sequence is $28 \times 28$ as opposed to the $34 \times 34$ resulting from the conversion process described in Section 4.3.3.

| | Spatial Downsampling Factor | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Method | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| N-MNIST | $34 \times 34$ | $17 \times 17$ | $12 \times 12$ | $9 \times 9$ | $7 \times 7$ | $6 \times 6$ | $5 \times 5$ | $4 \times 4$ |
| Spiking MNIST | $34 \times 34$ | $17 \times 17$ | $12 \times 12$ | $9 \times 9$ | $7 \times 7$ | $6 \times 6$ | $5 \times 5$ | $4 \times 4$ |
| MNIST | $28 \times 28$ | $14 \times 14$ | $10 \times 10$ | $7 \times 7$ | $6 \times 6$ | $5 \times 5$ | $4 \times 4$ | $3 \times 3$ |

polarity field of the event contains only a Boolean value indicating the event type[1]. As the output of this downsampling AER filter must conform to the same conventions as the input (allowing integration at any point in a filtering chain), the output events must contain only integer values.

The range of permissible downsampling factors is therefore limited to integer values. Furthermore, as discussed in the methodology in Section 4.6.2, the two spatial dimensions are downsampled by the same factor as to maintain the original aspect ratio. Temporal downsampling is also limited to integer downsampling values, and a flooring operation handles any fractional time-steps. Table 4.10 shows the spatial downsampling factors and the resulting pattern sizes. For the purposes of these experiments, these map directly to input channels. From the table, it can be seen that the smallest networks resulted in only 16 channels (9 for MNIST), and further downsampling was not possible.

The choice of a 1 ms resolution for the network allows for the interpretation of the temporal downsampling factor as the time resolution for the system. Therefore, a temporal downsampling factor of 5 results in a network that makes use of 5 ms time-steps instead of 1 ms. It is important to note that events mapped to the same time-step accumulate, as discussed in Section 4.4.1.

The parameters used for the downsampling also affect some parameters within the SKIM network. The bulk of these concern the duration of the pattern, and are simply scaled down by the same temporal factor. The maximum delays of the alpha functions used for the hidden layer networks require scaling by the same factor to ensure that the full range of the alpha functions fall within the duration of the pattern. The duration of the output pattern, and the shape of the training

[1]The polarity field possesses the ability to confer other information, such as the orientation of a filter, and therefore is usually stored as a full byte. Regardless of the mechanism conveyed through the polarity field, it still remains integer in principle

pattern remains fixed for all experiments.

Table 4.11 presents the results of a full downsampling sweep for the N-MNIST dataset. The training used a Gaussian training pattern and the Area output determination method (See Section 4.4.5 and Section 4.4.4 respectively). The figure demonstrates that downsampling serves only to improve the results, with the lowest accuracy of 80.04% achieved with no spatial or temporal downsampling. In fact, this result is similar to that achieved under maximum downsampling, achieved with a pattern of $4 \times 4$ in size and under 40 ms in length.

Table 4.11: **Results from spatial and temporal downsampling of the N-MNIST dataset.**

| | 1 ms | 2 ms | 3 ms | 4 ms | 5 ms | 6 ms | 7 ms | 8 ms |
|---|---|---|---|---|---|---|---|---|
| $34 \times 34$ | 80.04% | 84.04% | 84.23% | 85.18% | 85.06% | 85.04% | 85.33% | 85.05% |
| $17 \times 17$ | 82.94% | 85.77% | 86.18% | 85.88% | 86.70% | 87.11% | 87.52% | 87.08% |
| $12 \times 12$ | 85.43% | 87.00% | 87.43% | 87.89% | 88.56% | 88.14% | 88.26% | 88.14% |
| $9 \times 9$ | 87.33% | 88.09% | 88.68% | 89.09% | 89.11% | 89.08% | 89.38% | 89.00% |
| $7 \times 7$ | 88.64% | 89.47% | 88.59% | 88.35% | 87.93% | 87.50% | 87.28% | 86.70% |
| $6 \times 6$ | 87.93% | 89.03% | 88.14% | 87.63% | 86.23% | 86.28% | 84.60% | 84.76% |
| $5 \times 5$ | 87.45% | 87.00% | 87.03% | 85.36% | 84.82% | 83.95% | 83.05% | 82.62% |
| $4 \times 4$ | 85.71% | 86.78% | 84.99% | 84.21% | 82.51% | 81.95% | 81.08% | 80.73% |

The mid-range values for the spatial and temporal downsampling factors produced the best overall results, both individually and when used together. A pattern of $7 \times 7$ with a temporal resolution of 2 ms achieved the best overall accuracy of 89.47%. The distribution of accuracies shows that downsampling provides an advantage up until the point where the loss of information overcomes the encoding efficiency of the network, and the accuracies decrease back to the original accuracy under no downsampling.

These results are interesting, as the chosen spatial and temporal resolutions are based on logical extensions from the parameters of the camera. The number of input channels is derived directly from the number of pixels, and the temporal resolution is set by the on-chip acquisition counter (1 MHz). These results show that this encoding is not optimal for the N-MNIST dataset, as there is no apparent information loss under downsampling. The full resolution patterns must contain at least the same amount of information as the downsampled variants, indicating that there exists an optimal level of spatio-temporal density for the SKIM network.

### 4.6.4 Downsampling on the SpikingMNIST Dataset

The SpikingMNIST dataset is intended to examine the performance of the AER downsampling filter on an artificial spike pattern that is free of noise and is therefore a clean encoding of the information into a format similar to that of the N-MNIST dataset. The conversion from MNIST data into AER events is detailed in Section 4.6.1.

The method of temporal downsampling for the SpikingMNIST datset is identical to that used for the N-MNIST dataset, and operates using the same linear scaling as discussed in Section 4.4.1. Spatial downsampling requires a different approach, and is not performed on the spatio-temporal pattern but rather on the original input MNIST digit. In the N-MNIST dataset, reducing the range of the $x$ and $y$ pixel addresses in the event stream performs the spatial scaling, but has no analogue in terms of the static MNIST images. Instead, resizing the images using a bi-cubic filter performs the conversion to a lower resolution, which is then encoded in the same manner as before. This results in an event stream with a smaller $x$ and $y$ range, and one that is comparable in nature to that of the spatially downsampled N-MNIST sequences.

Table 4.12: **Results from the spatial and temporal downsampling of the SpikingMNIST dataset.**

| | 1 ms | 2 ms | 3 ms | 4 ms | 5 ms | 6 ms | 7 ms | 8 ms |
|---|---|---|---|---|---|---|---|---|
| $34 \times 34$ | 46.25% | 59.27% | 61.80% | 64.62% | 67.34% | 71.43% | 74.01% | 74.81% |
| $17 \times 17$ | 47.00% | 79.88% | 83.24% | 83.11% | 82.92% | 83.92% | 83.07% | 82.75% |
| $12 \times 12$ | 44.03% | 81.84% | 84.40% | 86.17% | 86.62% | 86.48% | 86.12% | 86.04% |
| $9 \times 9$ | 35.59% | 80.37% | 83.46% | 85.56% | 86.79% | 86.89% | 86.97% | 87.21% |
| $7 \times 7$ | 35.15% | 78.11% | 82.82% | 84.24% | 85.34% | 85.99% | 85.71% | 85.05% |
| $6 \times 6$ | 28.02% | 77.62% | 81.78% | 82.30% | 82.59% | 82.40% | 81.39% | 80.49% |
| $5 \times 5$ | 29.00% | 73.65% | 77.74% | 78.12% | 78.40% | 76.21% | 74.98% | 68.84% |
| $4 \times 4$ | 26.87% | 73.64% | 76.69% | 77.72% | 77.04% | 75.99% | 75.18% | 69.26% |

Table 4.12 presents the results from a sweep of the spatial and temporal downsampling factors for the SpikingMNIST dataset. The table highlights the way in which the nature of the image resizing is of paramount importance to the effectiveness of the SpikingMNIST dataset. The results for no spatial and temporal downsampling are poor, with an accuracy below 50%. Performing spatial downsampling only results in poorer performance still, dropping to a low of 26.87%. However, applying only temporal downsampling improves the results. This makes sense, as applying a downsampling factor of 255 would result in a single time-step containing a flattened and scaled version of the original MNIST digit, and the

SKIM algorithm would operate in the exact same manner as the OPIUM method used in the ELM approach in Section 4.6.5.

It is therefore the spatial downsampling that provides the interesting results, as it constitutes a similar pattern to the digit sequences in N-MNIST dataset. The spatial downsampling also demonstrates a similar pattern to the results obtained on the N-MNIST dataset. As with the N-MNIST, the accuracy increases with the degree of spatial downsampling, up until a point at which the information loss becomes significant and the results begin to decrease. In this case, however, the accuracy drops more steeply than in the case of N-MNIST under higher levels of spatial downsampling, but this may be a result of the SpikingMNIST dataset containing far fewer spikes than the N-MNIST dataset.

These experiments also yield insight into the nature of the SKIM algorithm for such problems. As downsampling is a lossy process, the results containing the full resolution pattern should be able to achieve the same accuracy bound as any of the downsampled variants. In the case of SKIM, there appears to be a spatio-temporal density at which the algorithm performs best. It is possible that this density is determined by the effective fan-out defined by the relation of the number of input channels to the size of the hidden layer network. In that case, the spatial downsampling serves to effectively increase the fan-out, allowing for better separation in the higher dimensional space created by the hidden layer neurons. This effect provides the improvement in accuracy, until the amount of information discarded in the downsampling process become significant and serves to degrade the performance.

### 4.6.5 Downsampling on the MNIST Dataset

As both the N-MNIST and the SpikingMNIST datasets are derived from the MNIST dataset, it is illustrative to perform analogous downsampling on the MNIST dataset as it provides a theoretical accuracy maximum for the previous systems. The MNIST dataset represents the original data, and any operations performed on it can serve only to either discard information or to introduce noise. As both the SpikingMNIST and N-MNIST methods make use of a SKIM implementation based on OPIUM, these experiments made use of an ELM classifier built around the OPIUM method.

The downsampling process for the MNIST dataset is fundamentally different from the other two datasets, as the data is not an event stream but rather a static image composed of grey-scale values. Spatial downsampling can therefore be performed directly on the pixels, and affects the grey-scale values. This is in contrast to the spatial downsampling in the N-MNIST dataset, where the spatial downsampling affects the range of $x$ and $y$ values, but is the same process used to downsample the digits prior to conversion in the SpikingMNIST dataset. As

with the SpikingMNIST dataset, a bicubic algorithm performs the image resizing. As the MNIST digits are only $28 \times 28$ pixels in size, the spatial resolution for the MNIST dataset differs from the $34 \times 34$ pixels present in the N-MNIST and SpikingMNIST datasets.

The MNIST dataset is static, and therefore has no temporal aspect to downsample. Instead, the range of intensity is scaled in an analogous fashion. This is fundamentally the same operation as temporal downsampling on the SpikingMNIST dataset as the time values in that dataset encode the intensity. Therefore, increasing the temporal downsampling factor effectively decreases the dynamic range of the image. Although not technically temporal downsampling, this analogous process of altering the intensity values of the MNIST dataset will be referred to as temporal downsampling from this point onward.

Table 4.13: **Results from the spatial and resolution downsampling of the MNIST dataset.**

| | 256 | 128 | 85 | 64 | 51 | 42 | 36 | 32 |
|---|---|---|---|---|---|---|---|---|
| $28 \times 28$ | 94.38% | 94.32% | 94.39% | 94.32% | 94.40% | 94.36% | 94.22% | 92.23% |
| $14 \times 14$ | 93.65% | 94.18% | 94.26% | 94.45% | 94.28% | 94.38% | 92.68% | 92.57% |
| $10 \times 10$ | 93.44% | 94.28% | 94.63% | 94.77% | 94.61% | 94.50% | 94.42% | 94.27% |
| $7 \times 7$ | 77.61% | 78.74% | 79.98% | 79.70% | 80.31% | 80.50% | 80.03% | 79.54% |
| $6 \times 6$ | 88.78% | 89.83% | 90.34% | 90.35% | 90.44% | 90.46% | 88.21% | 90.26% |
| $5 \times 5$ | 80.82% | 81.83% | 82.79% | 83.43% | 83.90% | 84.02% | 83.08% | 82.74% |
| $4 \times 4$ | 50.41% | 52.63% | 53.71% | 53.71% | 53.52% | 53.68% | 53.17% | 53.00% |
| $3 \times 3$ | 74.98% | 76.63% | 77.18% | 77.51% | 77.48% | 77.04% | 76.73% | 76.31% |

Table 4.13 shows the results of then downsampling sweep performed on the MNIST dataset. An OPIUM network consisting of 1000 hidden layer neurons was used to perform the classification. The configuration of the network attempted to remain as close to that used in SKIM as possible, and makes use of the same sigmoid non-linearity and the same range for the random input weights. The network was trained on all 60,000 training samples in a random order, and tested on the full testing set.

The results show that spatial downsampling on the MNIST dataset produces a far more pronounced effect than temporal downsampling. In fact, the temporal downsampling rarely caused a drop in performance, and often improved the classification result under varying levels of spatial downsampling. This may be a result of the quantisation resulting from the lower input range.

Immediately visible in the results are the poor performances at image sizes of $7 \times 7$ and especially at $4 \times 4$, where the accuracy is only marginally above 50%. The results for the steps above and below each are in the order of 20%

better, and even the $3 \times 3$ network outperforms the $4 \times 4$ network by almost 25%. These poor results are caused by the choice of downsampling algorithm, as 28 is divisible by 4 and 7. This results in a direct mapping for the resizing, and requires no inter-pixel interpolation. Therefore, pixel information is simply lost instead of being transferred to neighbouring pixels as is the case for the other parameters. The effect should also be present at the $14 \times 14$ spatial resolution, but it is likely that the image contains enough redundancy to negate this effect.

### 4.6.6 Downsampling on the N-Caltech101 Dataset

Performing downsampling on the N-Caltech101 dataset requires a small deviation in the methodology used for the N-MNIST dataset. The lack of defined splits between training and testing datasets and the irregular number of images per category necessitate the use of only a subset of the total training samples available.

Applying a similar downsampling approach to the N-Caltech101 dataset also yields a performance increase, and additionally allows for a smaller set of input channels to the SKIM network when downsampling spatially. Temporal downsampling reduces the number of discrete time-steps in the input pattern, greatly reducing the number of updates required by the SKIM algorithm.

These reductions improve the speed at which the network learns and also serves to reduce the memory requirements. The testing time also benefits from the downsampling reductions, but receives only a modest gain in speed by comparison. The discussion in Section 6.5 describes the reasons for the asymmetrical speed increases between training and testing.

The practical effects of the speed increase are more pronounced on the N-Caltech101 dataset than similar experiments on the N-MNIST dataset due to the larger scale of the problem posed by the N-Caltech101 dataset.

Table 4.14 presents the results of the downsampling as applied to the full 101-way classification task on N-Caltech101 dataset. Due to the number of experiments performed, only one trial of each configuration was tested. Table 4.14 presents only a subset of the parameters tested for this dataset, and the full results for the 101-way classification task can be found in Table 5 in the Appendices.

The same set of downsampling experiments were performed on the 5-way classification task, and displayed a similar pattern of results. The effect of spatial downsampling also served to only worsen the classification performance, and the best classification performance (81.05%) was achieved using no spatial downsampling and the maximum level of temporal downsampling.

The results differ from those achieved on the N-MNIST dataset in that spatial downsampling served only to produce worse results. Temporal downsampling, on the other hand, produced a dramatic increase in accuracy, with the best result (15.72%) achieved using the lowest spatial downsampling factor and the highest

Table 4.14: **Selected spatial and temporal downsampling results for the 101-way classification problem on the N-Caltech101 dataset using SKIM.** The table shows the classification accuracy achieved with a network containing 1000 hidden layer neurons. The full set of results can be found in Table 5 in the Appendixes

| | Temporal Downsamplng (millisecond resolution) | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 1ms | 2ms | 3ms | 4ms | 5ms | 10ms | 15ms | 20ms |
| $223 \times 173$ | 6.37% | 7.55% | 9.10% | 9.78% | 9.59% | 14.73% | 16.03% | 15.72% |
| $111 \times 86$ | 6.06% | 7.18% | 8.54% | 8.66% | 9.47% | 12.44% | 13.80% | 14.97% |
| $74 \times 57$ | 7.05% | 6.68% | 8.35% | 9.16% | 9.53% | 12.75% | 14.85% | 15.41% |
| $55 \times 43$ | 5.94% | 7.74% | 7.49% | 8.23% | 8.85% | 11.94% | 12.75% | 15.66% |
| $44 \times 34$ | 5.75% | 6.19% | 7.49% | 8.04% | 7.98% | 11.45% | 11.57% | 13.18% |
| $37 \times 28$ | 5.57% | 6.62% | 6.13% | 8.17% | 6.68% | 13.12% | 12.31% | 13.80% |
| $31 \times 24$ | 5.14% | 5.26% | 6.99% | 6.93% | 6.99% | 10.89% | 12.69% | 12.19% |
| $27 \times 21$ | 4.52% | 5.20% | 6.31% | 6.74% | 7.49% | 9.59% | 11.63% | 12.19% |
| $24 \times 19$ | 6.93% | 5.07% | 6.06% | 7.12% | 7.05% | 10.33% | 10.77% | 11.88% |
| $22 \times 17$ | 5.69% | 5.32% | 6.81% | 6.81% | 7.18% | 9.65% | 12.31% | 12.25% |
| $20 \times 15$ | 5.69% | 5.01% | 5.57% | 7.49% | 7.98% | 9.53% | 11.32% | 13.86% |
| $18 \times 14$ | 4.46% | 4.70% | 6.93% | 5.69% | 7.67% | 9.72% | 10.77% | 12.69% |

temporal one.

The degrading effect of spatial downsampling may be a result of the non-uniform image sizes in the N-Caltech101 dataset. Whereas each image in the N-MNIST dataset receives the same benefit from the spatial downsampling, it is likely that only the differing image sizes results in an uneven spatial compression, which makes separation by SKIM difficult.

As the accuracy due to temporal downsampling served only to increase as the size of the time-steps were increased, a second set of experiments served to explore the effects of further downsampling temporally.

Figure 4.24 shows the results of extending the temporal downsampling from 1 ms time-steps to 250 ms time-steps. The results show that there is a clear range over which the method produces optimal results, with a peak accuracy of 18.25% achieved using at a time-step size of 25 ms. After that point, the results gradually decrease until the size of the time-steps are approximately 100 ms after which the accuracy settles on approximately 5%.

It is clear that temporal downsamping improves the results on the N-Caltech101 dataset, whereas spatial downsampling serves to only degrade the results. The varying image sizes in the N-Caltech101, coupled with the far larger spatial resolution of each image may serve to explain the performance of spatial downsampling. It is possible that the larger image sizes may require a greater level of downsam-

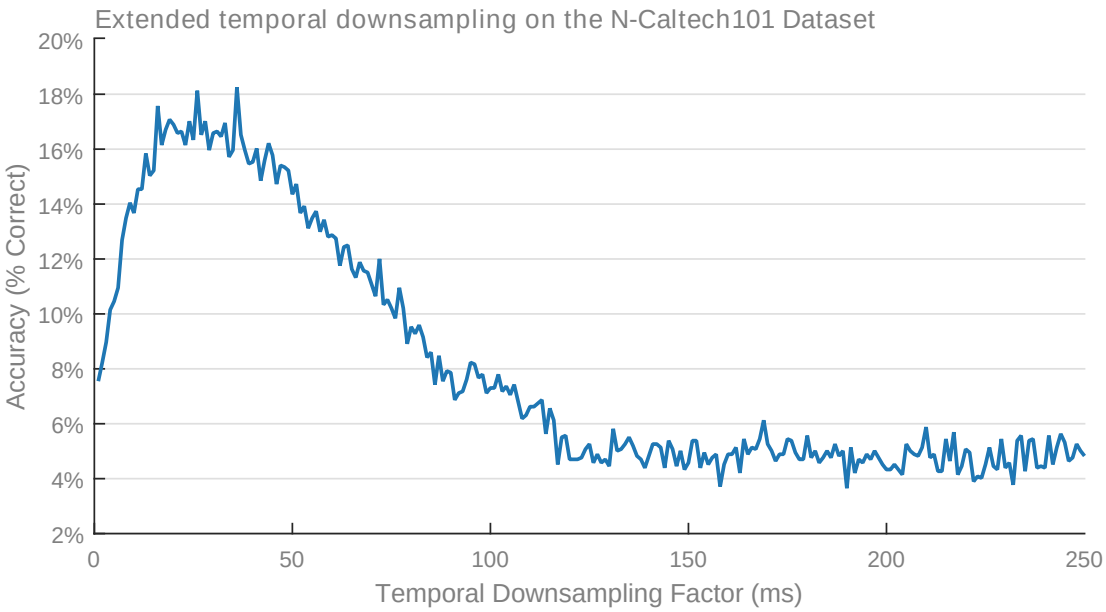



Figure 4.24: **Classification results from extended temporal downsampling on the N-Caltech101 dataset**. The results presented in Table 4.14 present temporal downsampling results up to 20 ms time-steps. This figure shows the classification accuracy as the time-steps are increased up to 250 ms. The accuracy peaks with a time-step resolution between 25 ms and 30 ms, achieving accuracies up to 18.25%. Due to the length of time taken to perform each test, only a single trial of each experiment could be performed.

pling to achieve the same effect as found on the N-MNIST dataset, but that would cause the smaller images in the dataset to reduce to impractical sizes.

### 4.6.7 Discussion

The importance of spatial downsampling in event-based visual processing is subtle but important. The downsampling serves as a form of feature extraction, encoding additional information into each spike in the spatio-temporal pattern. Given the nature of the SKIM and OPIUM classifiers in which the inputs to both are independent of one another, there is no importance or significance placed on the order in which the network receives inputs.

The all-to-all connectivity between the input layer and the hidden layer masks the originating channels, as the hidden layers only see an aggregated sum of all input layer activity (albeit through different sets of random weights). Therefore, the spatial structure of any input is inherently lost when the data arrives at the hidden layer neurons.

Spatial downsampling serves to mitigate this effect somewhat by mapping a region of the input to a single channel, effectively creating a small receptive field for each input to the SKIM or ELM network and preserving and encoding some of the spatial information within that region. The network still discards the spatial relationships between the inputs, but each input now encodes a little spatial information, and the larger the spatial downsampling, the more spatial information is encoded.

This effect is plainly visible in the results from this section. The effect of spatial downsampling improved the results on the N-MNIST dataset as visible in Table 4.11, with the best accuracy achieved using a downsampling factor of 4. The accuracy then decreases as the downsampling increases, as the amount of information lost due to the downsampling operation begins to outweigh the benefits from the encoding of spatial information.

The same effects are visible with the SpikingMNIST dataset, in which the same improvements appear when downsampling. The downsampling mechanism used in the SpikingMNIST dataset differs in that it is applied to the image before the conversion to spikes, and makes use of a bicubic filter which results in pixels that represent some aggregate statistic over their neighbouring regions. In fact, the network achieved a low performance when no spatial or temporal downsampling was applied.

The MNIST dataset provides additional insight into the downsampling effects. The MNIST dataset is not spike-based, and therefore makes use of OPIUM to implement an ELM classifier which also discards any relationship between input channels. When examining the results presented in Table 4.13, it becomes immediately clear that the networks making use of downsampling factors that

are divisors of the image size produce poor performance. These are the $7 \times 7$ and the $4 \times 4$ resulting image sizes, which represent a downsampling factor of 4 and 7. When downsampling by these factors, the filters do not need to perform any inter-pixel interpolation, resulting in a less effective encoding of the spatial information.

The temporal downsampling also has an interesting effect on the classification performance for both the N-MNIST and the N-Caltech101 datasets. Temporal downsampling alone produces a boost in performance in both datasets, with a far more pronounced effect on the N-Caltech101 dataset. It appears to become less effective as the level of spatial downsampling is increased as well, which may have to do with the loss of information eclipsing the performance gains.

The underlying question is why temporal downsampling should yield such a positive increase in performance, and the answer may lie in the conversion process used for the datasets themselves. As detailed in Section 4.3.3, the movement of the camera led to the generation of spikes from ATIS camera, as a stationary image would produce no output. Through the motion, the temporal information actually encodes spatial information from static objects in the relative timings of adjacent events, and the temporal downsampling allows the classifiers to access this information.

Downsampling is therefore an important pre-processing technique in event-based visual processing. It serves to help in mitigating the effects of input independence required by many classifiers (especially for LSHDI networks such as SKIM and ELM). Spatial pooling can be thought of as a generalisation of the concept of spatial downsampling to include overlapping regions, but still retains the same underlying principle of mapping an input channel to a spatial structure within the source data.

# Chapter 5

# Object Classification in Feature Space

## 5.1 Introduction

This section explores the use of feature detectors as a processing step in event-based recognition tasks by converting the classification problem from 2D space to a feature space defined by the feature detectors outlined in Section 3.1.

Implementing these feature detectors adds another processing layer to the visual processing systems implemented in the previous sections, moving the feature detection from an implicit step in the classification or downsampling operations to an explicit step based on the feature detectors presented earlier in this work. Through the use of the same datasets and the same classification system, the performance and results of this section are directly comparable those achieved in Section 4.

The use of feature detectors allows for a number of interesting benefits to a classification system. It allows the visual processing system to remain unaffected by images of varying sizes and can also allow for varying levels of invariance to rotations, translations and any other affine or photometric changes. It also serves to reduce the size of the input channels, which has a marked effect on the computational speed of the classification process.

This section explores the implementations of the feature detectors based on the surface of time presented in Section 3.3.2, and the orientation-based feature detectors discussed in Section 3.4, and presents insights into the nature of classifying data in feature space.

## 5.2 Contributions

The work in this section makes the following contributions:

- It presents and demonstrates a complete and modular classification system, capable of extracting features from AER data and performing classification in an event-based manner.
- It demonstrates the effectiveness of using orientations as features directly, and provides insight into why the ELM produces a better result than the SKIM network.
- It introduces an adaptive-based thresholding technique for clustering of features extracted from time surfaces, and demonstrates the effectiveness of such an approach.
- It validates the benefits of explicit feature detection in event-based systems.

## 5.3 Classification using Orientations as Features

The orientation feature detector, introduced in Section 3.4, calculates the spatio-temporal orientations about each incoming event and provides a means of assessing the viability of using the orientations as a feature. Applying this principle to incoming events, opens up the possibility of classifying on the orientations features rather than the spatial pixel locations as presented in Section 4.4 and Section 4.5. Therefore, these classification algorithms operate in a feature-space defined by the feature detectors, rather than the spatio-temporal space used in previous sections.

Instead of a using a separate input channel for each pixel in the image sequence, the classifiers use a fixed number of input channels, with each channel corresponding to orientation range. This approach has the important advantage of being better suited to handling non-uniform image sizes, as the number of inputs to the SKIM network remains fixed regardless of the size of the input image sequence.

### 5.3.1 Classification Methodology

Classifying based on orientations requires a more complicated system than those used for the direct application of SKIM to the spiking datasets. The feature-space classifiers include an explicit feature detection step and additional noise filtering. The particular configuration presented in this section differs slightly from the other methods presented in this chapter as it forgoes the usual spike-based input

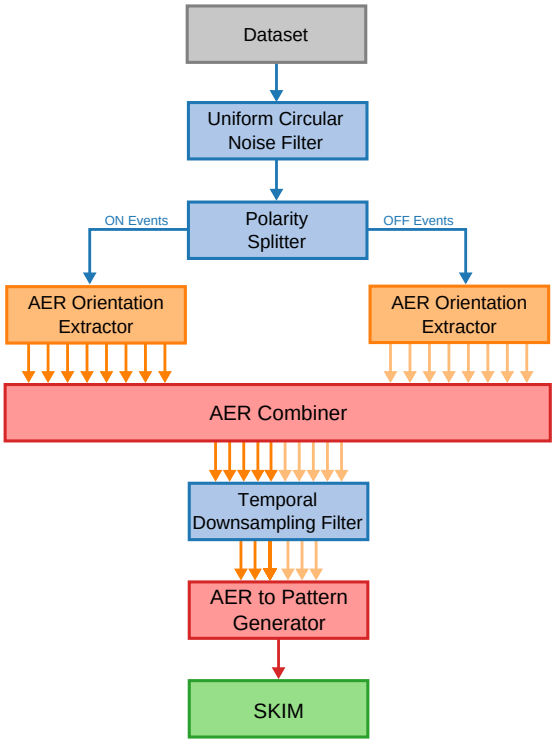



Figure 5.1: **Diagram of the filtering chain used for classifying directly on the orientation outputs using SKIM.** The diagram shows the steps performed for each event received from the camera, unless discarded by the noise filtering step after which the processing for that event will cease. It is important to note the polarity splitter, which allows the ON and OFF events to be handled separately. Orientations are extracted from the data for each polarity and then combined into a single AER stream. Temporal downsampling is applied to the AER data prior to conversion to a spatio-temporal pattern for the SKIM network.

to SKIM and provides a continuous input instead. The system and configuration still operates in an event-based manner, but replaces all binary spikes with integer valued signals.

Therefore, the input more closely resembles a set of time-varying signals than a true spike-based spatio-temporal pattern. The experiments in this section serve to demonstrate the classifying power of orientation features and provide an example of a viable event-based classification method.

Figure 5.1 shows the filtering chain used in the generation of orientation patterns for the classification system. The dataset provides a stream of events in AER format for each sequence in the dataset, which then pass through the circular noise filter described in Section 3.4.3. This filter removes events arising

from spurious noise and events which generate poor orientation features. At this point, a polarity filter divides the input into ON and OFF events for separate processing, which results in orientations specific only to those classes of events. Combined with the saccade motion, this serves to generate direction-specific features during the three saccades, which would otherwise combine if the polarity was not used in this fashion.

For each incoming event, an orientation extractor calculates the temporal orientations about the event independently for each polarity. The orientation extractor uses the method described in Section 3.4 and operates as an AER filter, thereby also outputting AER events.

The results of the orientation extractors are two histograms of orientations, one for ON events and another for OFF events, and each containing 40 bins (with each bin representing a range of 9°). These two histograms are combined and encoded into the payload of a new AER event, which receives the timestamp of the initiating event. This event contains a vector of 80 orientations. Each orientation corresponds to a feature and an input channel to SKIM. The order of these orientations in the vector does not impact the performance of the network as the SKIM algorithm discards the ordering of the input channels.

As the filtering chain translates each incoming event from a coordinate space into an orientation space, the amount of information per event increases but retains the original 1 $\mu s$ time resolution. As the SKIM algorithm operates at discrete time-steps of 1 $ms$[1], it requires a temporal downsampling step for the incoming orientation events in order to operate.

A modified temporal downsampling filter (similar in structure to the one used in the downsampling experiments in Section 4.6.2) performs the temporal scaling by aggregating feature events over 1 ms intervals, and produces an event containing the average of all the orientations received during the 1 ms period.

As the pattern presented to SKIM is no longer a spike-based spatio-temporal pattern, it raises the question of whether the SKIM method is the most appropriate means of classifying this data. The output pattern from the filtering chain presented in Figure 5.1 consists of a dense pattern of integer values, rather than a sparse pattern of spikes. It is possible to interpret this output as a static image for each sequence, one where each pixel codes for a specific average intensity of an orientation range at a specific time during the pattern. Mapping this static image onto an input layer of a standard ELM network creates a network structure similar to the one used to classify the MNIST dataset in Section 4.6.5. This approach is explored in Section 5.3.3.

[1]SKIM can operate at any timescale, but is limited to millisecond resolution due to computational limits. Additionally, using a higher time resolution results in a far more sparse input pattern, and the results of the temporal downsampling presented in Section 4.6 suggest that this can have a negative impact on classification accuracy.

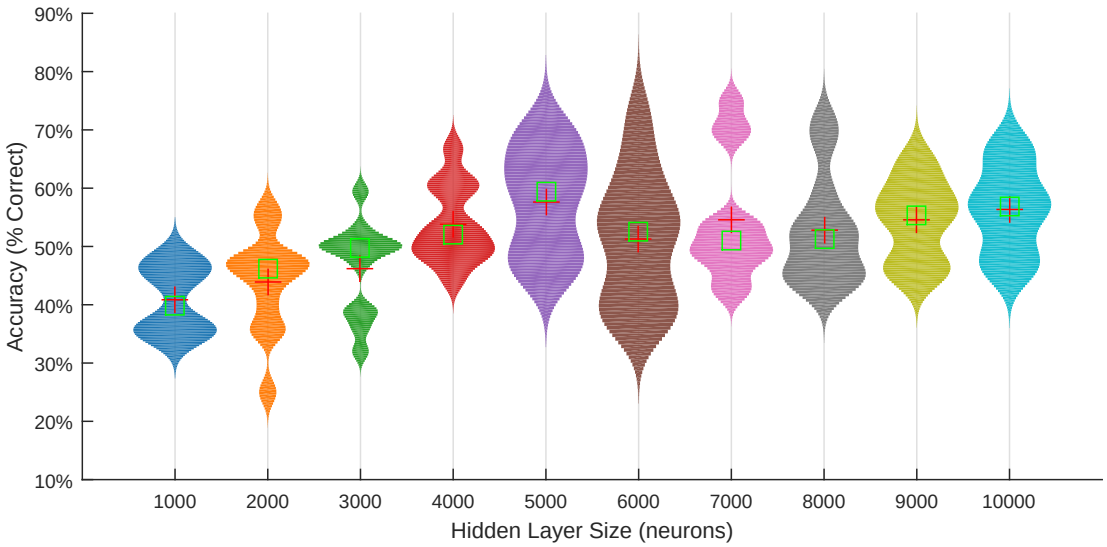



Figure 5.2: **Violin plot of the 4-way classification task on the N-Caltech101 dataset using orientation features and SKIM for varying hidden layer sizes**. Twelve trials of each hidden layer size were performed, and display the large variability in the results. As this is a 4-way classification task, the accuracy due to chance is 25%.

Although this approach is perfectly viable, there are two advantages to using a SKIM network over a direct ELM approach. The first concerns the size of the hidden layer network for the ELM. Unlike the MNIST dataset, the orientation patterns require a much larger input layer (25,280 vs. 784 for MNIST) and therefore cannot sustain the large fan-outs typically required by ELM networks. This is primarily due to memory constrains.

The second reason for the application of SKIM is the use of Alpha functions in the hidden layer neurons, which allow interactions across the columns of the spatio-temporal pattern. Such behaviour is not possible using a typical ELM network as it treats each input independently.

### 5.3.2 Classification using the SKIM Network

Due to the nature of the classes in the N-Caltech101 dataset, the training and testing splits cannot make use of all the available objects in each class. Section 4.4.1 presents a discussion of the reasons for this limitation and the widely used means for dividing the dataset. The results focused on the classification of the five most popular categories (Airplanes, Faces, Cars, Motorbikes and the Background class), as these classes contain the largest numbers of items per class.

Testing utilised networks of varying hidden layer sizes, but retained the same

Table 5.1: **Summary of the accuracies for the 4-way classification task on the N-Caltech101 dataset across 10 different hidden layer sizes.**

| | Hidden Layer Sizes | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 1000 | 2000 | 3000 | 4000 | 5000 | 6000 | 7000 | 8000 | 9000 | 10000 |
| Mean | 40.9% | 43.9% | 46.3% | 53.8% | 57.7% | 51.2% | 55.3% | 52.8% | 54.5% | 56.3% |
| Max | 47.5% | 56.6% | 59.4% | 66.8% | 69.7% | 72.5% | 75.0% | 71.7% | 65.2% | 67.2% |
| Min | 34.8% | 25.0% | 32.0% | 44.3% | 45.9% | 36.9% | 41.8% | 42.2% | 45.1% | 45.9% |
| STD | 5% | 8% | 8% | 6% | 9% | 11% | 11% | 9% | 7% | 7% |

configuration of random weights. Alpha functions and delays varied as the hidden layer size changed, but the maximum durations remained constant across all tests. The operation of the filter chain is deterministic, and therefore was only executed once with the intermediate results stored and presented to each network.

Figure 5.2 shows a summary of the results from the classification performed on the 4-class classification task. The data shows the results from 12 independent trials of each network configuration, with only the training order varying from trial to trial. Although all the results are well above chance (25% for the 4-way task), it is immediately apparent that the standard deviation is far greater than found in other networks (see Section 4.4.3 for a detailed discussion of the nature of the errors from a typical SKIM network).

The results, expanded in Table 5.1, show that the networks produced a strong performance (achieving a maximum of 75% accuracy with 7000 hidden layer neurons), and that there exists a positive relationship between the number of hidden layer neurons and the mean accuracy. However, the same configuration of 7000 hidden layer neurons also produced a worst-case accuracy of just 41.8%, causing that particular configuration to exhibit the highest overall standard deviation of any configuration.

The results indicate that this particular configuration is particularly sensitive to the subset of training and testing samples used, as is visible in the full set of results for all trials presented in Figure 5.2. Each trial made use of the same sequence of training objects, which produced vastly different results from configuration to configuration, although there is visible upward trend as the hidden layer size increases.

It is possible that the increased sensitivity is due to the density of the input patterns, which are no longer sparse spike-based spatio-temporal patterns. It is likely that the varying image sizes play a role in this configuration, as smaller images will produce fewer events than larger images, especially around the borders of the image as it is inset in the larger frame (See Section 4.5.2 for a complete dis-

cussion on the manner in which non-uniform images in the N-Caltech101 dataset).

### 5.3.3 Classification using an ELM Network

The design and methodology used to test the ELM classifier attempted to follow those used in existing work performed on the Caltech101 dataset. In particular, this work follows the same training and testing regiment used by Orchard et al. [150], making use of the same five category and full classification tasks.

The first classification task involves the five largest categories in the N-Caltech101 dataset. These are the airplanes, cars, faces, motorbikes and background classes. The large number of samples in each category allows for the largest possible training sets from the dataset, allowing the use of 250 samples from each category, with the remaining images reserved for testing. Each trial randomly drew a different subset from each category, and trained the network in a random order. In total, 30 trials of each experiment were performed.

Figure 5.3 shows a violin plot of the classification results for network configurations containing an increasing numbers of hidden layer neurons. The graph shows the classification accuracy in terms of percentage of sequences correctly identified. As there are five object categories and an equal number of each sample per category, the expected classification accuracy arising due to chance is 20%. For each hidden layer size, the violin plot shows a symmetrical plot of the distribution of accuracies across the 30 trials for that specific configuration.

The results show that even a small network containing only 1000 hidden layer neurons achieves a mean accuracy of $69.34\% \pm 2.84\%$, which is surprising given that the network exhibits a fan-in factor of approximately 25.5. Table 5.2 shows that as the number of hidden layer neurons increases, the mean accuracy also increases whilst the standard deviations about the means decreases, yielding a final accuracy of $91.6\% \pm 1.69\%$. At this network size, the fan-in factor reduces to approximately 2.5. Memory constraints prevented the exploration of hidden layer sizes beyond 10,000 hidden layer neurons.

The distribution of accuracies for each hidden layer size were checked for normality using the Lilliefors test for normality [149], with only the network containing 10,000 hidden layer nodes rejecting the null hypothesis that the data are normally distributed at the 5% significance level.

Expanding the classification task from the five categories to the full 101 object categories in the N-Caltech101 dataset has the effect of increasing the overall number of training items, but reducing the number of samples drawn from each category. This is due to certain categories possessing a small number of images (as low as 30), which sets the maximum available for training and testing. As a result, only fifteen samples of each category were available for training.

Figure 5.4 shows a violin plot of the classification accuracies achieved for the

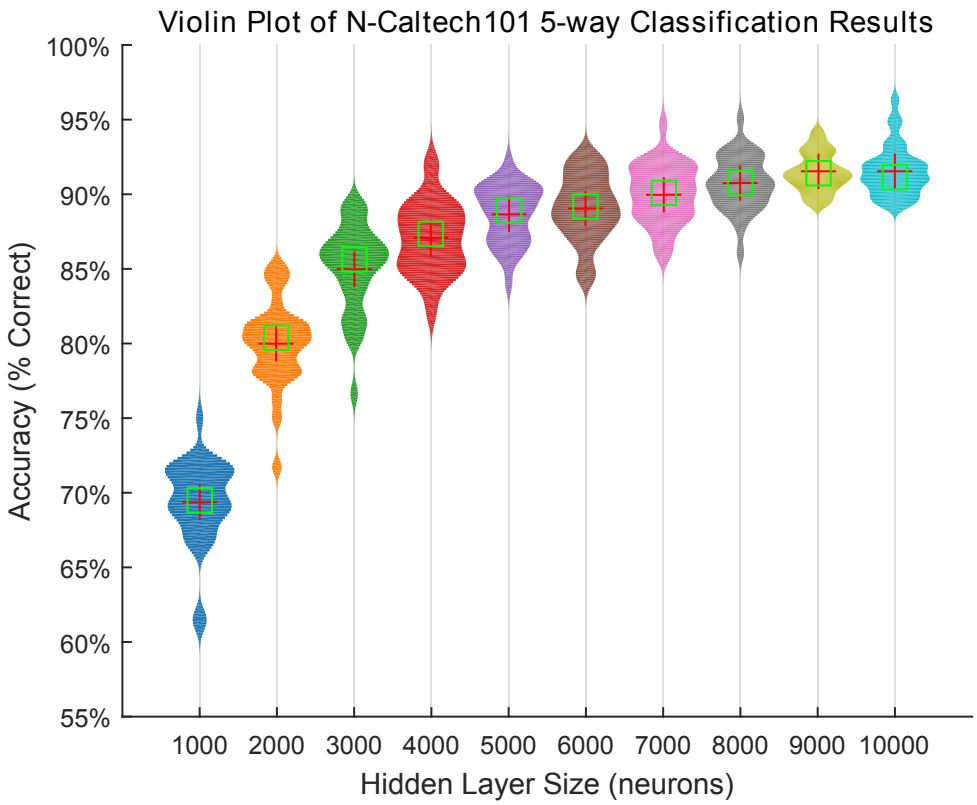



Figure 5.3: **Violin plot of the classification accuracies for networks of varying hidden layer sizes trained and tested against five categories of the N-Caltech101 dataset (airplanes, cars, faces, motorbikes and the background class)**. The violin plot shows the distribution of accuracies for each hidden layer size (containing 30 trials of each). Crosses indicate the mean accuracy and the green square indicates the median value.

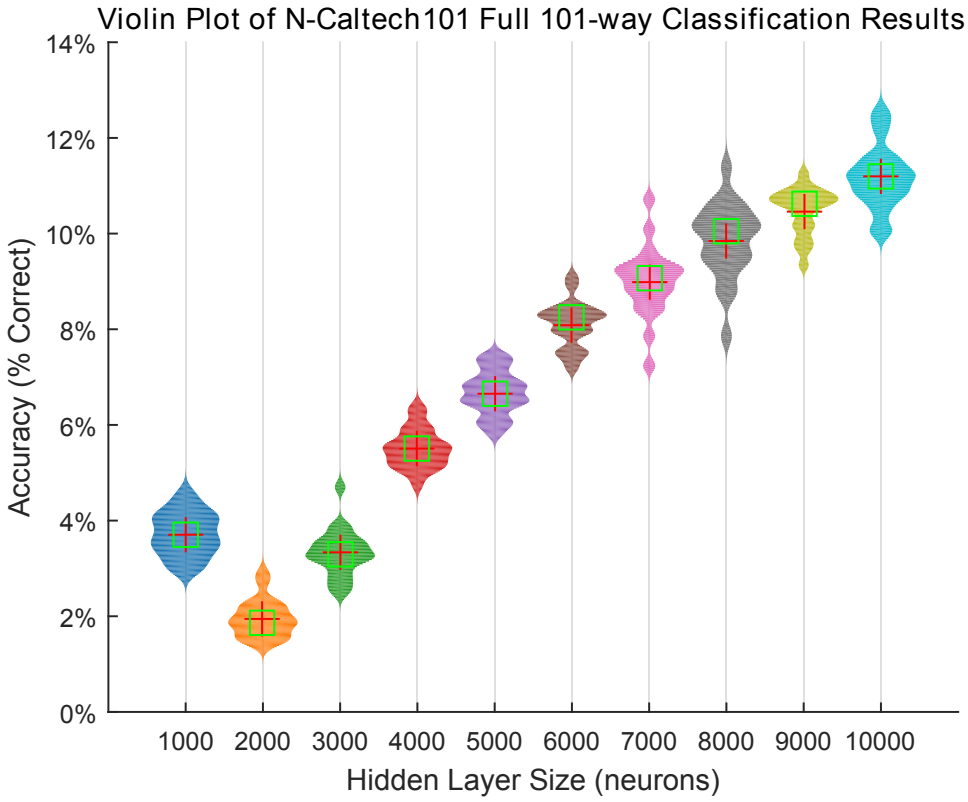



Figure 5.4: **Violin plot of the classification accuracies for networks of varying hidden layer sizes trained and tested against the full 101 categories in the N-Caltech101 dataset**. The violin plot shows the distribution of accuracies for each hidden layer size (containing 30 trials of each). Crosses indicate the mean accuracy and the green square indicates the median value.

full classification task. Due to the increased number of categories, the classification accuracies are much lower than those for the 5-way classification task. However, the expected accuracy due to chance sits at less than 1%, and therefore the classification accuracies are still significant. All distributions were tested for normality using the Lilliefors test at the 5% significance level, with the null hypothesis being upheld for each configuration.

Paradoxically, the network with 1000 hidden layer nodes achieved a mean accuracy of $3.72\% \pm 0.45\%$, outperforming the networks with 2000 and 3000 hidden layer neurons. As Table 5.2 shows, with the exception of the first three configurations, the accuracy increased as the size of the hidden layer increased, yielding an accuracy of $11.19\% \pm 0.65\%$ for the network with 10,000 hidden layer neurons.

These results compare favourably to those achieved by Orchard et. al. [144] for the full classification problem on the N-Caltech101 dataset, in which they achieved best-case accuracy of 8.30% using the SKIM algorithm. It is important to note that the SKIM network in that example makes use of only 2000 hidden layer neurons and also relies on a method similar to OPIUM for learning. Although these two networks use similar terminology, the nature of the hidden layer in SKIM make use of kernel functions, and the method itself was designed specifically for spatio-temporal patterns. It is therefore not possible to draw direct comparisons between the numbers of hidden layer neurons used.

The performance on the N-Caltech101 dataset appears to be heavily impacted by the low number of training samples available in each object category. As a result, the algorithm performs far better on the 5-way task, in which there are far more available samples, and also may explain the inconsistencies shown in the accuracies of the first three networks in the full classification problem. In order to explore this hypothesis, the same classifiers can be applied to the N-MNIST dataset.

Unlike the N-Caltech101 datasets, the N-MNIST dataset has well-defined and established splits between training and testing data. There are also ample training and testing samples available across all ten output classes. As a result, it is a suitable means of exploring the relationship between the number of training samples and the final classification accuracy.

Applying the same classification methodology to the N-MNIST dataset produced the results shown in Table 5.2 and a violin plot of the results is shown in Figure 5.5. The accuracies presented in the table represent the average classification accuracy across 30 trials against each configuration, and made use of the full training and testing set. The N-MNIST task is a 10-class classification problem, resulting in a 10% classification accuracy due to chance. Even with just 1000 hidden layer neurons, the network achieved a classification performance of $81.75\% \pm 0.38\%$. All the distributions of accuracies underwent the Lilliefors test,

Table 5.2: **Summary of the classification accuracies, in terms of percentage of test digits correctly identified, obtained for the N-Caltech101 and N-MNIST datasets**. The table shows the mean classification accuracy and the standard deviation over 30 trials of each configuration.

| | 5-Way N-Caltech101 | | 101-Way N-Caltech101 | | N-MNIST | |
|---|---|---|---|---|---|---|
| Hidden Layer Size | Mean | STD | Mean | STD | Mean | STD |
| 1000 Neurons | 69.34% | 2.84% | 3.72% | 0.45% | 81.75% | 0.38% |
| 2000 Neurons | 80.00% | 2.83% | 1.96% | 0.39% | 86.80% | 0.29% |
| 3000 Neurons | 85.03% | 2.91% | 3.33% | 0.49% | 88.79% | 0.26% |
| 4000 Neurons | 87.05% | 2.38% | 5.51% | 0.40% | 89.97% | 0.19% |
| 5000 Neurons | 88.69% | 1.94% | 6.67% | 0.45% | 90.69% | 0.25% |
| 6000 Neurons | 89.07% | 2.22% | 8.10% | 0.48% | 91.25% | 0.21% |
| 7000 Neurons | 90.03% | 1.99% | 9.00% | 0.73% | 91.54% | 0.12% |
| 8000 Neurons | 90.75% | 1.83% | 9.85% | 0.78% | 91.85% | 0.16% |
| 9000 Neurons | 91.49% | 1.29% | 10.48% | 0.49% | 92.10% | 0.18% |
| 10000 Neurons | 91.60%* | 1.69% | 11.19% | 0.65% | 92.21% | 0.18% |

*Failed the Lilliforth test for normality at the 5% significance level.

with the null hypothesis for normality upheld in every case.

The results represent the best classification to date on the N-MNIST dataset. In the original paper, the benchmark values achieved on the N-MNIST dataset was 83.44% using a SKIM network containing 2000 hidden layer nodes. The HFirst method [6] was also run on the dataset and achieved an accuracy of 71.15% using the hard classifier. Typical ELM networks used to classify the original MNIST dataset routinely achieve accuracies greater than 97.5% [105] with smaller hidden layer networks (usually 7840 hidden layer neurons representing a fan-out factor of 10), and these results achieved with the N-MNIST dataset show that much of the separable information is retained across the conversion to an event-based representation, and then into orientation features.

To further explore the performance as a function of the number of samples trained, a network consisting of 1000 hidden layer neurons was trained on an increasing number of subsets of the full N-MNIST dataset, and then tested on the full testing set. The size of the training subsets varied from a single training sequence up to 3000 training samples in steps of 50. Figure 5.6 shows a plot of the classification accuracy against the size of the training sample. Ten trials of each experiment were performed.

As expected, training with a single input sequence produced a classification accuracy of $9.97\% \pm 0.21\%$, which is at the level of chance. The classification accuracy then climbs up until a peak accuracy of $51.99\% \pm 1.35\%$ at 300 training

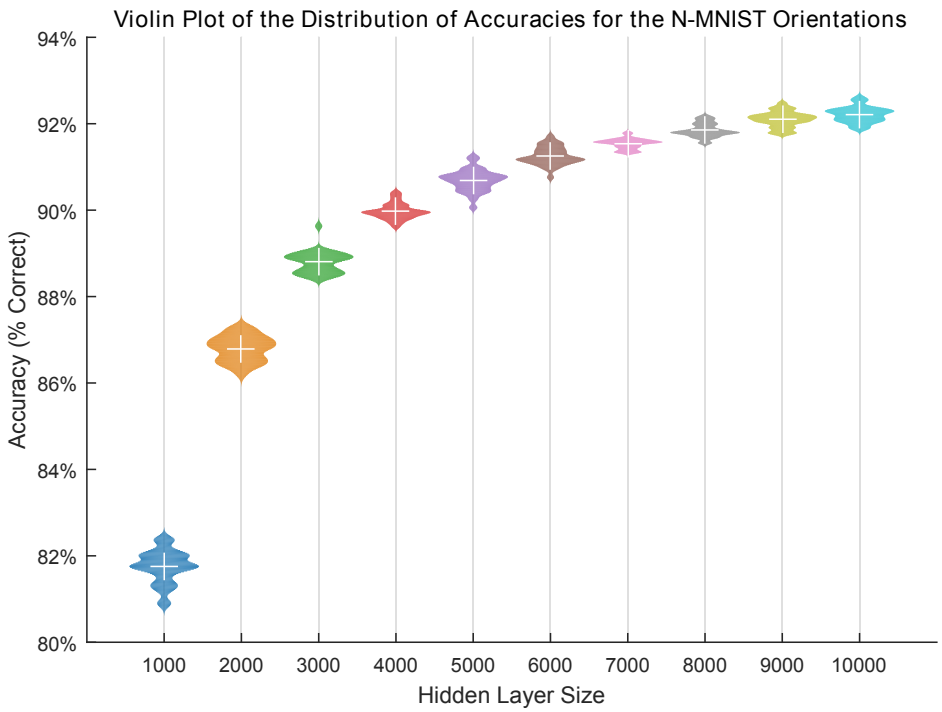



Figure 5.5: **Violin plot of the classification accuracies for networks of varying hidden layer sizes trained and tested against the N-MNIST dataset with 10 output categories**. The violin plot shows the distribution of accuracies for each hidden layer size (containing 30 trials of each). Crosses indicate the mean accuracy and the green square indicates the median value.

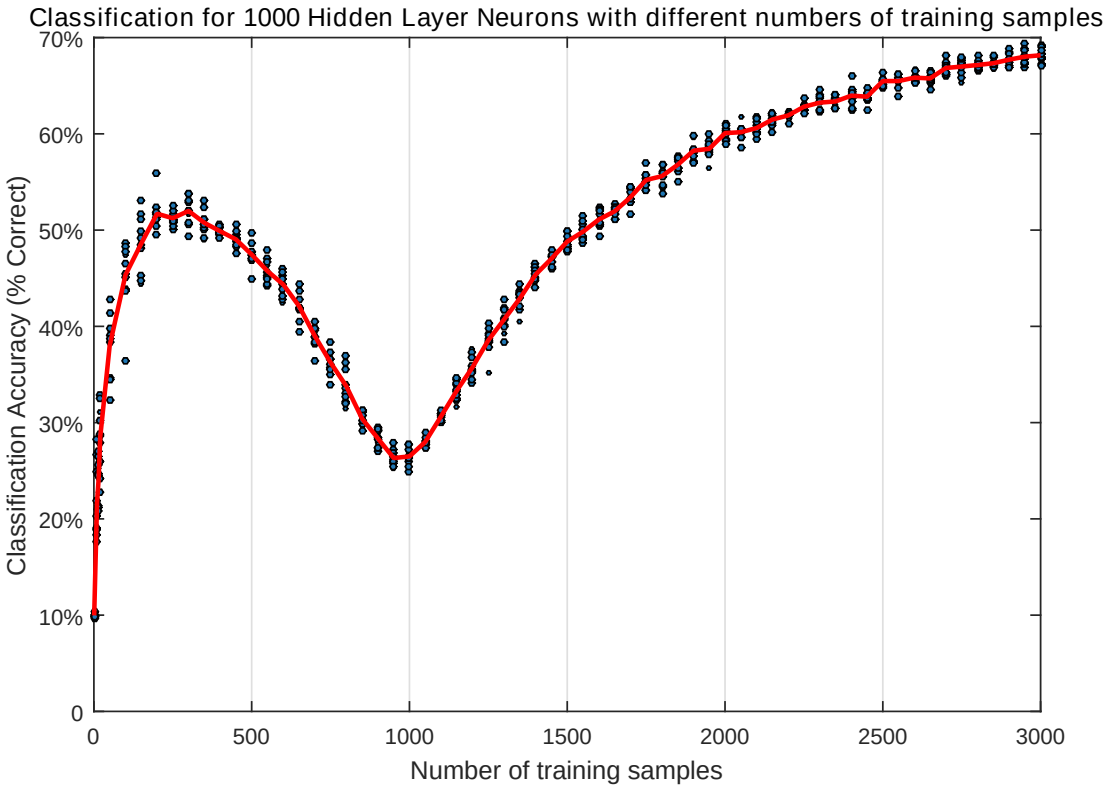



Figure 5.6: **Effect of number of training samples on classification accuracy for the N-MNIST dataset**. Ten trials of each experiment were conducted at 50 sample increments, and are shown as dots. The mean accuracy for each level is shown as the solid line.

samples. After this point, the accuracy paradoxically decreases, bottoming out at just $26.49\% \pm 0.91\%$ at 1000 training samples. After this point, the network accuracy climbs steadily and continues up until levelling off at approximately 81% for this network size. This classification accuracy is reached at approximately 12,000 training samples and remains constant for the rest of the 60,000 training samples.

It is interesting to note that the lowest accuracy occurs when the number of training samples is equal to the hidden layer size. At this point, there are enough hidden layer neurons to learn each input sequence perfectly, which results in no need for generalisation. The characteristic shape of the accuracy at low numbers of training samples provides some insight into the unusual results for the network with 1000 hidden layer neurons in the full 101-way classification task. In that network, the 1000 neuron configuration outperformed the larger hidden layer networks of 2000 and 3000 neurons. As there are 101 object categories with

15 sequences per category, this results in 1515 training sequences, and therefore the results shown in Figure 5.6 indicate that we can expect the lowest training accuracy to occur with a network containing 1515 hidden layer neurons. The network containing 1000 neurons sits above this turning point, whereas the 2000 and 3000 neuron networks sit on the portion of the curve that slowly rises up to the higher accuracies shown for the 101-way classification in Table 5.2.

This result also sheds some light on the marginal improvements shown over existing methods for the full 101-way classification. It is likely that the low number of training samples per category contributes to the small gains in improvement over other methods. As demonstrated by the 5-way classification task, the information contained in the orientation features used in this thesis provides a means on which to separate and classify the data effectively, but the method requires a larger number of samples to achieve a good classification performance.

### 5.3.4 Discussion

The results from both the SKIM and the ELM approaches provide interesting insight into the applicability and suitability of the two approaches. The ELM produced a far superior classification result over the SKIM network, but requires access to the entire pattern and therefore cannot operate in an event-based manner. The SKIM network, on the other hand, performed far worse in terms of accuracy, but it is able to operate on information as it arrives, and therefore does allow for a truly event-based approach.

The SKIM and ELM classifiers share a common methodology, and both operate in an iterative and analytical manner. However, the two methods represent differing paradigms in terms of their operation. The SKIM network is designed to learn spatio-temporal information, whereas ELM networks are intended to learn the relationship between input and output vectors and requires an explicit encoding scheme for time-varying signals. The work in this section is limited to these two classifiers, as their similarity allows for better comparison and investigation of the underlying nature of the output generated by the feature detectors. Although classification accuracy is the metric by which performance is compared, this section focuses primarily on the characterisation of the feature detectors through a real-world task rather than on achieving the highest possible classification performance.

The results suggest that the ELM network is far better suited to the type of pattern used in this section. The use of integer values in place of spikes, and the density of information (as opposed to the relative sparsity of the spatio-temporal patterns used in SKIM) resemble the inputs used by the ELM method when used to learn the original MNIST dataset (as used in Section 4.6.5). In fact, attempting to convert those intensity maps representing the digits into a

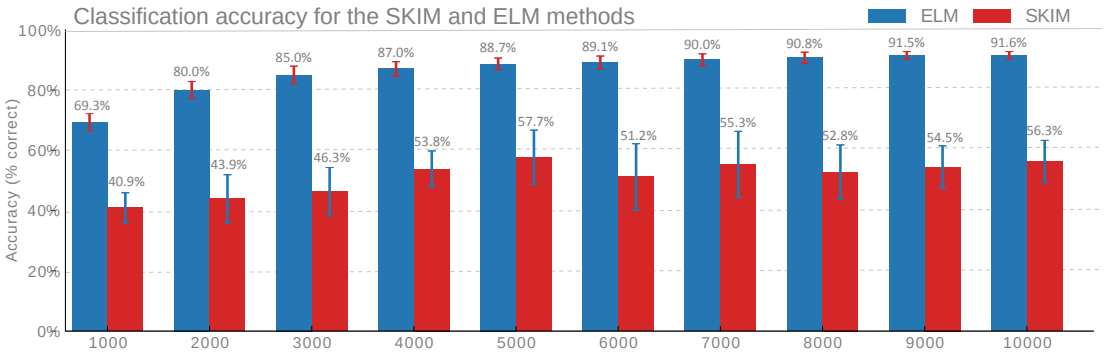



Figure 5.7: **Comparison of accuracies between the SKIM and ELM approach to using orientations as features on the N-Caltech101 dataset for varying hidden layer sizes.** The result presented here show the mean accuracy as a percentage of digits classified correctly. The graph also shows the standard deviations for both network types as error bars. The graph shows the increased accuracy obtained with the ELM solution, and the lower variability it possesses over the SKIM network.

spatio-temporal pattern as done for the SpikingMNIST dataset in Section 4.6.1 yielded results that were lower than the equivalent methods using the ELM and the original MNIST digits.

Figure 5.7 provides a direct comparison of the SKIM and ELM approaches. It shows the accuracy obtained (in percentage correctly classified) for the ELM and SKIM networks over a range of hidden layer sizes. Twelve trials of each network were independently trained and tested and the average accuracy reported in the graph along with error bars indicating the standard deviation. The ELM network outperforms the SKIM network in every network configuration, and also produces a lower standard deviation. The ELM method also demonstrates a positive correlation between hidden layer size and network accuracy, making it consistent with the characteristic network performance presented in Figure 4.9.

The performance of the SKIM network did not display the same level of consistency as the ELM network, and exhibited a larger standard deviation across all trials. The network displays a higher sensitivity to training pattern order and the random weights, as classification accuracy varied by over 20% from trials with the same hidden layer size. Although this result may appear to indicate that the SKIM network performs in an inferior manner to the ELM network, it is important to note the event-based implementation and operating nature of SKIM may be the leading cause.

Whereas the ELM has access retrospectively to the entire pattern, SKIM makes use of the alpha functions in the hidden layer neurons to act as a form of memory and to convey information from the time at which it occurs to the point

during which output determination occurs. The nature of this encoding uses randomly initialised alpha functions with random delays, and therefore can fail to adequately or accurately capture all the information in the input pattern. It is this process that suffers most from the randomness of the training patterns and random hidden weights as they determine how the alpha functions map to the input channels and the initial content presented to those hidden layer nodes. The high density of the patterns used in this section also increases the information loss arising from the nature of the SKIM algorithm, and further suggest that it is better suited to sparse spatio-temporal patterns.

## 5.4 Object Classification using Time Surface Features

The previous section demonstrated the classifying power present in the spatio-temporal orientations described in Section 5.3. Although the results achieved when using the orientations as features were promising, the implementation of the system stretches the overall approach to event-based processing maintained in this thesis. The replacement of a spatio-temporal pattern with real-valued signals breaks the compatibility of such systems with a traditional AER system, and therefore restricts the use of such an approach.

This section presents an alternative means of performing classification using the surfaces of time from Section 3.3 in a manner more aligned with the ethos of event-based systems. Unlike the previous section, the outputs of all components are spikes, either as an AER stream or as a spatio-temporal pattern, and yet it maintains the majority of the benefits of the previous approach, including the reduced sensitivity to input image size and the fixed number of input channels for the SKIM network.

### 5.4.1 Classification Methodology

The surface of time, introduced in Section 3.3, create a 2D representation of the spatio-temporal patterns from the ATIS camera. These surfaces allow for the application of conventional computer vision techniques such as corner and edge detection to identify features of interest, and the descriptors calculated from them can yield a degree of invariance to affine transformations.

This section introduces a method of converting incoming events to a feature space defined by a set of feature clusters on a time surface. An adaptive thresholding algorithm, based on the method underpinning the Synaptic Kernel Adaptation Network (SKAN) [151], provides an online method to determine these

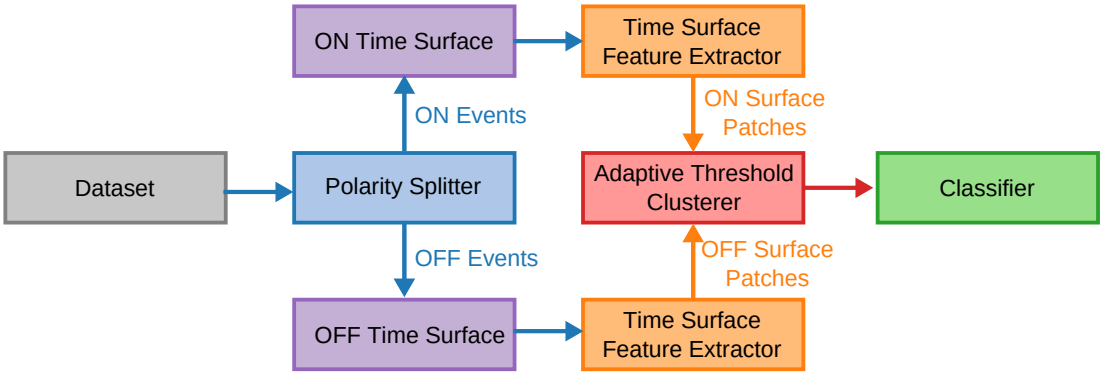



Figure 5.8: **Structure of the time surfaces feature classification system with online learning.** The classification system for the time surface feature detector also makes use of a polarity splitter to handle the ON and OFF events separately. Time-surface features are then extracted from the incoming AER stream and sent to the adaptive threshold clustering algorithm, which identifies and learns features in an unsupervised manner. The output of the clustering algorithm represents AER events in feature space and are then used for classification.

features, and this section presents classification results using both a SKIM and ELM network.

The orientation features introduced in Section 5.1 benefited from the circular and therefore bounded nature of angles and made use of histograms with bins of a fixed size in order to create features. The spatial coordinates are not bounded in the same manner as angles, and therefore the same approach cannot be directly applied. In order to replace the inputs to SKIM with a spatio-temporal pattern as opposed to a real-valued signal, the nature of each input needs to shift from representing a certain orientation range, to a more complex feature for which a single spike at a specific time conveys information. The information must exhibit strong separability in terms of the output classes. It is also desirable to minimise the number of such features presented, yet maintaining enough so as to successfully generalise to the whole dataset.

In this section, each input channel to the SKIM network corresponds to a distinct feature on the surface of time, one that is derived from the incoming data. An online clustering algorithm iteratively determines and refines these features, allowing them to adapt to the nature of the incoming data.

Figure 5.8 shows the structure of the network used to perform classification in a time-surface feature space. All the components operate on, and emit AER data, allowing the network to run in an event-based manner. A noise filter based on the feature detector described in Section 3.3.2 removes the spurious noise events, and

any planar features from the input stream. The noise filter works particularly well in this application, as events that correspond to regions of near-uniform intensity often pose little use in the classification task. It is especially important to remove the dominant noise spikes, as they will invariably train a feature, and produce events representing only noise to the classification system.

As in the previous section, a separate processing stage exists for the two different polarities. This assists in capturing the direction of movement, and is particularly useful in the context of the N-Caltech101 dataset, in which there are often edges parallel to the direction of the camera motion during the conversion process. Clustering the histograms independently for each polarity also allows for the inclusion of the polarity information into the generated features.

The filtered and separated events arrive at the online clustering component, which accepts an AER stream of events and also outputs AER events. Section 5.4.2 presents the specifics and details on the implementation of the adaptive threshold clustering mechanism.

The Adaptive Threshold Clustering represents the point at which the network shifts from a spatially orientated pattern to one in time-surface feature space. The online clustering component receives AER events in the form of $e_s = [x, y, t, p]^T$, where $\mathbf{u} = [x, y]^T$ represents the spatial location of the pixel with reference to the ATIS camera and obtained at time $t$. The output of the online clustering component is in the form of $e_\theta = [F, t]^T$ where $F \in [F_0 ... F_N, N < b]$ corresponds to the feature (out of $b$ features) matched to the current event and the time $t$ represents the time at which the original event occurred, and it remains unchanged from the input.

Examining the nature of the output, which now only contains a feature number $F$ and a timestamp $t$, the benefits of operating in feature space become apparent as this structure maps directly into the spatio-temporal patterns required by SKIM. The conversion discards the spatial information contained in each AER event (namely the $x$ and $y$ coordinate of the incoming event), and the range of the outputs remains independent of the spatial size of the input sequence. This avoids the problems surrounding non-uniform image sizes discussed in Section 4.5.2. The number of feature clusters tracked is a variable parameter in this system, and dictates the number of input channels to the SKIM or ELM network.

As the output preserves the temporal resolution of the input pattern, it can pass through the same temporal downsampling filter presented in Section 4.6.3. This reduces the resolution of each event to 1 ms time-steps.

The following sections examine aspects of this network configuration. The process of comparing and clustering features using time surface features poses certain challenges, and the following sections provide a discussion of these issues and their solutions.

In terms of specific implementation, the network makes use of an exponential

time surface (as introduced in Section 3.3.1) as the effect of the exponential operation serves to suppress the effect of events occurring long in the past, and thereby eliminates the need for an explicit threshold to discard old events. The conversion from a 2D patch to a feature vector is equivalent to the calculation of a descriptor presented in Section 3.3.3, with the normalised descriptor chosen, as the classification system can make use of rotational information and therefore the descriptor process needs to preserve that information.

### 5.4.2 Adaptive Threshold Clustering on the Time Surfaces

The problem of finding good prototype features on which to cluster is a difficult task. An optimal set of features should be orthogonal and distinct given the nature of the data presented. Optimal features are obtainable using techniques such as Principle Component Analysis (PCA) but such methods generally require knowledge of the entire dataset ahead of time, and therefore such an approach is not often possible in an event-based paradigm.

An event-based system requires an online method of determining the optimal prototypical features for the dataset, and the k-means clustering algorithm [152] is perhaps the logical candidate for such an operation. Indeed, k-means clustering allows for a pre-selected number of clusters to be iteratively computed on the arrival of new data.

Unfortunately, the k-means clustering algorithm is not a perfect candidate for clustering on event-based data. It is sensitive to the initial estimates for the clusters [153], and the choice of initial estimates in a high dimensional space is not a trivial problem.

The choice of update rule for k-means clustering is also a concern as there are two main varieties tailored to handling different situations. The first update rule weights each incoming sample by the total number of samples received. This version allows for clusters that encompass a large dataset to be iteratively determined without the loss of any information. Unfortunately, such an approach places emphasis on the initial training sequences, and is poorly suited to a continuous operation (as required by an event-based system) as each subsequent event exerts less influence on the feature clusters.

The second update rule uses a fixed threshold by which to update the clusters. This implements a system similar to a low-pass filter, and allows the clusters to continuously evolve to match the incoming data, making it a better match for an event-based system. However, the sensitivity to the initial choices for the clustering remains a difficult problem to solve, especially as it is not possible to predict the nature of the incoming data in an event-based system.

This thesis introduces a means of performing the clustering online using a method of clustering with an adaptive threshold for each cluster. Under this setup, each cluster implements a unique threshold which determines a matching area about the feature. This threshold is dynamic, and given $i$ feature clusters, the thresholds change based on two rules:

1. If an incoming feature matches the cluster, then decrease the threshold $t_i$ for cluster $i$ by a fixed amount $\Delta I$.

2. If an incoming feature does not match any clusters, then increase all the cluster thresholds by a fixed amount $\Delta E$.

Internally, the system represents features and clusters as vectors, regardless of their interpretation in the larger system (which in the case of this application are 2D patches from time surfaces) and normalising the feature vectors ensures that every feature represents a point on the unit sphere for the dimensionality of the feature space.

Figure 5.9 shows an illustration of the adaptive thresholding as applied to features containing only two dimensions. In practise, the number of dimensions is far larger, and the features are normalised to fall on a unit hypersphere, rather than the unit circle as shown in the figure. The figure depicts two existing clusters $i_1$ and $i_2$, both of which have independent thresholds indicated by the spheres surrounding them[1].

All incoming features require normalising to ensure that they are comparable to the normalised feature clusters. The system makes use of the normalised descriptors described in Section 3.3.3, which already exhibit this property. These normalised features are shown as the green vectors in Figure 5.9.

The distance to each feature is calculated using the cosine distance, and the closest value within the range of the adaptive threshold is chosen as the matching cluster. If successfully matched to a cluster, the feature is then assigned to it, and the cluster is said to have emitted a spike. The feature is then used to update the cluster itself using a fixed mixing rate $\eta$ as follows:

$$i_n = (1 - \eta)i_n + (\eta)v \tag{5.1}$$

In which $i_n$ denotes cluster $n$ to which the incoming feature $v$ is successfully matched. As a point of reference, the mixing rate used to generate the clusters in this work was set as $\eta = 0.001$.

[1]In reality, the normalisation ensures that features only fall on the unit circle in 2D, and therefore the matching region is actually an arc on the unit circle rather than a sphere, but is drawn in the figure as such for illustrative purposes.

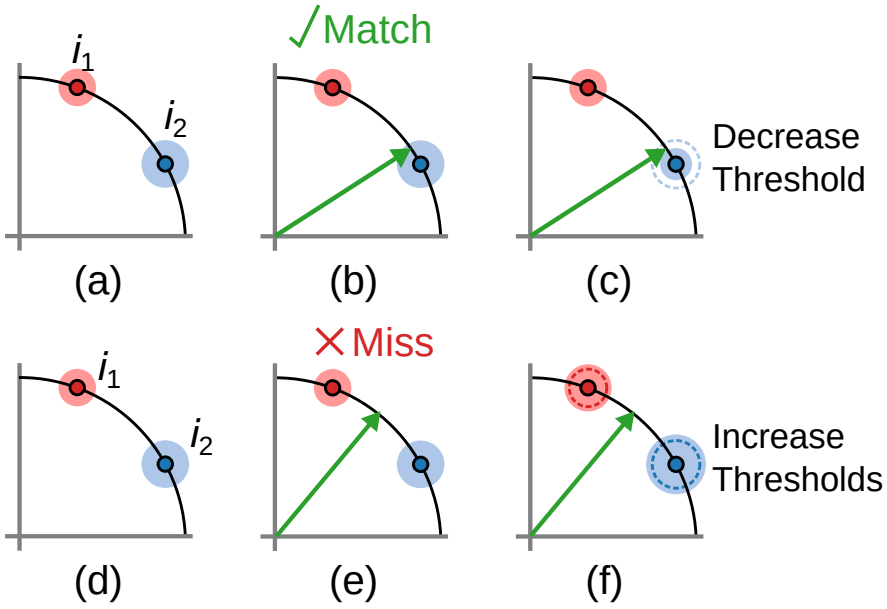



Figure 5.9: **Diagram showing the adaptive threshold clustering in a 2D space.** The diagram above shows a simplified case of the adaptive threshold clustering applied to features consisting of two dimensions. An existing configuration with two normalised clusters is shown in (a) on the unit circle, and with independently configured thresholds. If the next feature to arrive falls within the threshold distance of an existing cluster, as shown in (b), then it is said to have matched and is assigned to the that cluster. The threshold for that cluster is then reduced as shown in (c). If the event does not match any existing clusters, as in (e), the feature is discarded and the thresholds for all clusters are increased as in (f).

The threshold for that cluster is then decreased by $\Delta I$ as shown in Figure 5.9c. If the feature does not match any existing clusters, as in Figure 5.9e, the feature is discarded and the thresholds for all features is increased by $\Delta E$.

The effect of this dynamic thresholding serves to ensure that the rate of firing of for all clusters is roughly equal, as decreasing the threshold on matching serves to damper and specialise each cluster. If the clusters are coding poorly for the incoming data (if, for instance, they have become too specialised), then the global threshold increase serves to expand the range of input features to which they will respond.

This learning process is dynamic and responsive to the nature of the incoming data, and relies on the abundance of events in the ATIS dataset. The clusters are initialised to random points on the unit circle, and the learning can be disabled by fixing the thresholds for each cluster. Once the system is no longer learning, the thresholds give an indication of the specificity of the given feature, with a large threshold indicating a more general feature. Often the feature exhibiting the largest threshold is a noise spike and can be discarded. Due to the pre-filtering performed using the feature detectors, this was not performed for the classification systems presented in this section.

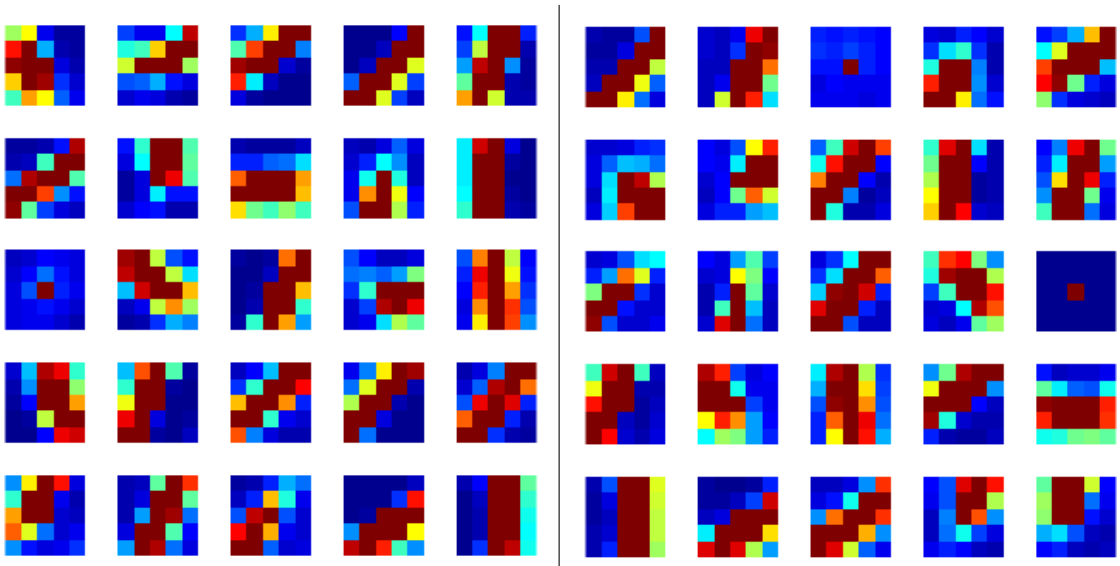
For the purposes of these experiments, the above method was used to generate clusters on the N-MNIST dataset. Only the training samples were used to generate the clusters, and made use of a $\Delta I = 0.001$ and a $\Delta E = 0.003$ which were derived empirically. Two cluster configurations were tested:

- 200 Clusters (100 for ON events, 100 for OFF events) with $11 \times 11$ feature patches
- 50 Clusters (25 for ON events, 25 for OFF events) with $5 \times 5$ feature patches

The number of clusters used depends greatly on the nature of the data and the size of the feature patches. When using 100 clusters with the $5 \times 5$ patches, it was observed that the majority of the features were not stable and were alternatively coding for noise events and therefore contributed little benefit to the system. 25 clusters were found to represent a stable number of clusters and were used instead.

Figure 5.10 shows the resulting 50 clusters of $5 \times 5$ image patches from the N-MNIST dataset. These features represents the normalised clusters against which the incoming features from the time surface are matched.

The features for the 200 cluster versions were also generated and can be found in the Appendices. Figure 14 shows the 100 clusters generated for the ON events in the MNIST training dataset, whilst Figure 15 shows the corresponding OFF features.



(a) (b)

Figure 5.10: $5 \times 5$ **feature clusters learnt from the MNIST dataset for ON events (a) and OFF events (b)**. Each feature represents a normalised vector reshaped to match the size of the incoming feature patches. The ordering of the features conveys no significance.

### 5.4.3 Surface Feature Classification with ELM

In a similar fashion to the feature orientations explored in Section 5.3.3, an ELM network was used to verify that the extracted features provide a suitable basis for which to perform classification.

ELM networks do not operate on time-steps, and therefore the input sequences from the N-MNIST dataset cannot be directly inputted into an ELM network. The most direct approach requires the use of a separate input channel for each pattern at each time-step, and when dealing with the original N-MNIST sequences, the image size of $34 \times 34$ pixels and the 316 time-steps results in a required input size of 365,296 per digit. Networks with such large input layers are not practical to implement.

The conversion to a feature domain as defined by a set of clusters, determined using the adaptive thresholding method outlined in Section 5.4.2 reduces the size of the input layer to one per cluster per time-step, and for a network containing 100 feature clusters, this results in an input pattern size of 31,600 which is similar in size to those used in the ELM classifier for the classifiers operating in feature space. Additionally, unlike the orientation ELM method presented in Section 5.3.3, the patterns generated from these clusters are sparse by comparison.

The testing with the ELM classifier involved the two different set of feature clusters introduced in Section 5.4.2, and the same conversion method was used to convert the events streams into the feature spaces. The conversion also made use of the same adaptive thresholds determined alongside the clusters.

Table 5.3 presents the results of the ELM classifier for the two different cluster configurations. Eight different hidden layer sizes were tested, and ten trials of each experiment performed. The results show that the $11 \times 11$ features outperform the $5 \times 5$ features at every tested hidden layer size, and by a significant percentage, achieving an overall accuracy of 94% with 8000 hidden layer neurons. This represents one of the highest accuracies achieved on the N-MNIST dataset to date.

Figure 5.11 shows a direct comparison of the accuracies achieved using both sets of feature clusters. As shown in Table 5.3, the standard deviations across all configurations never exceeds 0.40%. These results exceed those achieved using the same number of hidden layer neurons with the SKIM algorithm, with the $11 \times 11$ configuration outperforming the equivalent SKIM classifiers from Section 4.4.2 for each hidden layer network size.

The $11 \times 11$ configuration also outperformed the ELM classifiers based on orientations introduced in Section 5.3.3, which achieved a best-case result of 91.6% accuracy with a 10,000 hidden layer configuration, which the above classifier exceeded with only 4000 hidden layer neurons, and achieved an accuracy of 92.39%.

The reduction in the number of inputs to the ELM network serves to increase

Table 5.3: **Classification accuracy for ELM networks trained using the features events generated from the $11 \times 11$ and $5 \times 5$ pixel clusters.** The classification accuracy is reported in terms of percentage of digits correctly identified. The $11 \times 11$ configuration made use of 100 clusters per polarity, whilst the $5 \times 5$ clusters used only 25 per polarity. Ten trials of each configuration were performed.

| Hidden Layer | $5 \times 5$ Clusters | | $11 \times 11$ Clusters | |
|---|---|---|---|---|
| | Mean | $\sigma$ | Mean | $\sigma$ |
| 1000 Neurons | 71.76% | 0.25% | 84.93% | 0.31% |
| 2000 Neurons | 78.58% | 0.30% | 89.56% | 0.28% |
| 3000 Neurons | 81.24% | 0.38% | 91.45% | 0.22% |
| 4000 Neurons | 83.00% | 0.17% | 92.39% | 0.18% |
| 5000 Neurons | 84.01% | 0.16% | 92.87% | 0.16% |
| 6000 Neurons | 84.85% | 0.24% | 93.30% | 0.20% |
| 7000 Neurons | 85.17% | 0.13% | 94.06% | 0.13% |
| 8000 Neurons | 85.64% | 0.30% | 94.00% | 0.12% |

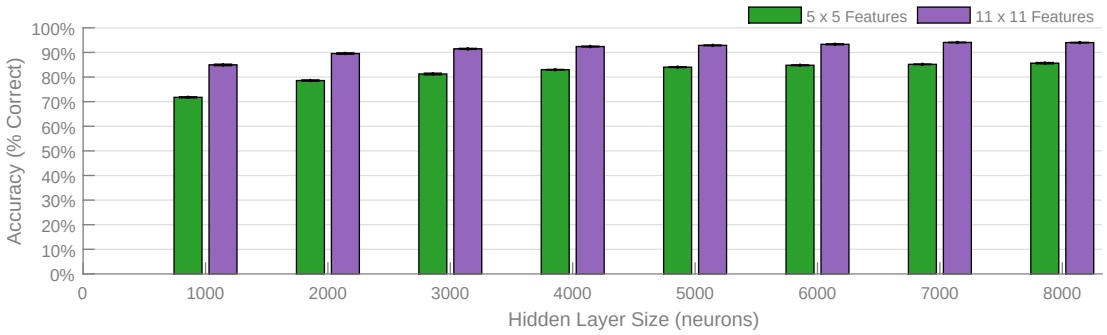



Figure 5.11: **Comparison of the ELM results with $11 \times 11$ feature clusters and $5 \times 5$ feature clusters with an ELM network for increasing hidden layer sizes.** The figure shows a comparison between the 200 $11 \times 11$ pixel features and the 50 $5 \times 5$ pixel features when used to generate feature events for an ELM classifier. Ten trials of each experiment were conducted, and a standard deviation under 0.5% was achieved for each hidden layer size. It is clear from the above graph that the $11 \times 11$ configuration performed better in every case.

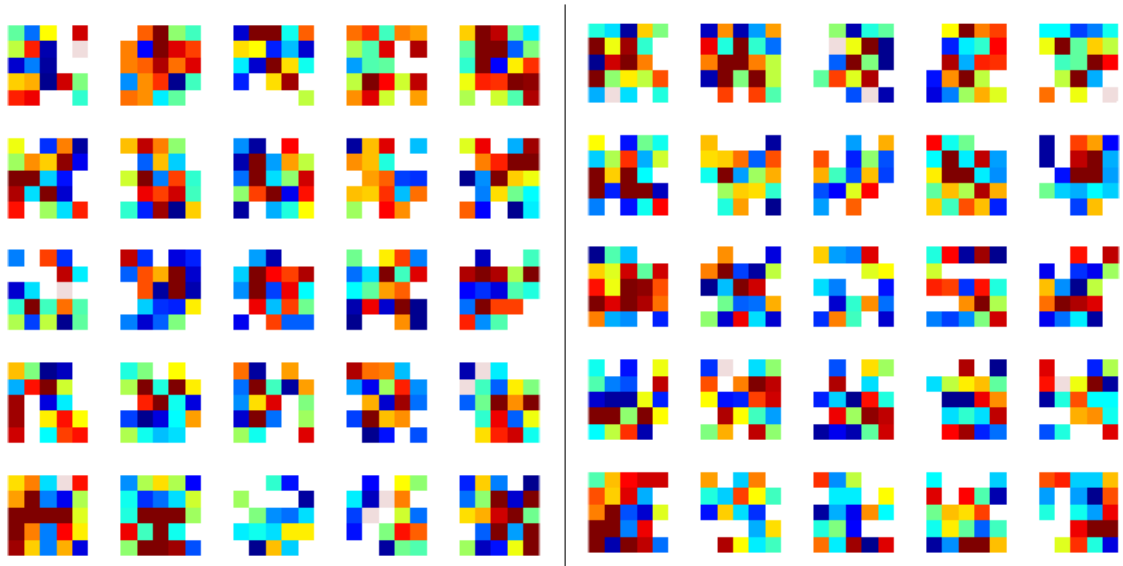



Figure 5.12: **The $5 \times 5$ pixel random features generated for the ON events (a), and the OFF events (b) used to generate feature events for the N-MNIST dataset.** The random features are initially generated, normalised and then fixed. The use of random features such as these allows for efficient hardware implementations which do not require explicit feature learning, and can even exploit device mismatch to generate the random features.

the effective fan-out to the higher dimensional space, and each feature event encodes more information than a single event from the raw N-MNIST sequences. These two factors may be the contributing factor in the classification accuracies achieved.

### 5.4.4 Classification using Random Feature Clusters

To validate that the clusters generated by the given method are providing a good set of basis functions on which to generate feature events, it was necessary to run a classification system that made use of entirely randomised clusters. The results from these tests can serve as a benchmark for comparison, as they represent an arbitrary conversion from the spatial coordinates to a feature space, whilst still maintaining much of the timing information from the original system.

To test this configuration, 25 random clusters were generated for features of size $5 \times 5$, and are shown in Figure 5.12. These features comprised a randomised vector of uniformly distributed random numbers, which was then normalised so

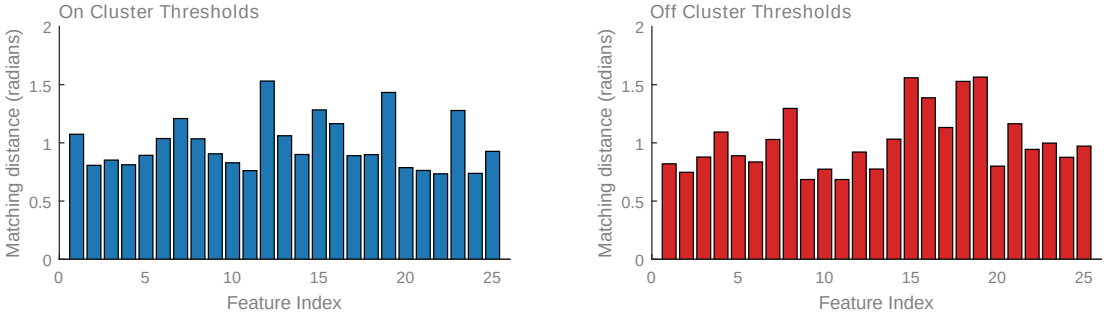



Figure 5.13: **Thresholds for the random features after learning on the training set of the N-MNIST dataset.** These graphs show the final threshold values for the features provided in Figure 5.12 after training on the N-MNIST dataset. The adaptive thresholds determined by the algorithm differ greatly from the initialised value of 0.5 for both the ON and OFF clusters. It is these determined thresholds that produce the strong classification performance of the random clusters.

as to place it on the unit hypersphere. These clusters are identical to those used to initialise the clusters before learning based on the N-MNIST dataset under the adaptive thresholding technique described in Section 5.4.2.

For the first set of tests, the random feature clusters were initialised with a fixed threshold value of 0.5 for each cluster. As the algorithm makes use of the cosine distance as a metric to match features, the range is inherently limited to $[-\pi, \pi)$, and therefore any value within that is an appropriate threshold. The threshold is therefore measured in radians, as it defines the range of angles over which the cluster will match an incoming feature.

When attempting to match features to these clusters, the algorithm produced hardly any features per input sequence, and the resulting classifier did not produce any classification accuracy above chance.

Given that the random clusters with a fixed threshold failed to produce feature events with any separability under the ELM classifier, the adaptive thresholding mechanism was implemented without the update to the clusters from Equation (5.1). This fixes the clusters, and simply adjusts their thresholds according to the input. In essence, the adaptive threshold seeks to ensure that the firing rate of each cluster is roughly equal, by penalising clusters that match too often, and expanding the threshold of clusters that do not.

Figure 5.13 shows the thresholds learnt through a single pass over the N-MNIST dataset. These thresholds and clusters were then used to convert the N-MNIST digits into feature events for classification, and then classified using an ELM classifier. The classifier operated in the same manner as presented in Section 5.4.3, except making use of the randomly generated features and their

Table 5.4: **Classification accuracies obtained with $5 \times 5$ random clusters and adaptive thresholding on the N-MNIST dataset using an ELM classifier**. These results made use of the random clusters shown in Figure 5.12 and the learnt adaptive thresholds from Figure 5.13. Ten trials of each configuration were tested.

| | Hidden Layer Sizes | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 1000 | 2000 | 3000 | 4000 | 5000 | 6000 | 7000 | 8000 |
| Mean | 74.17% | 80.90% | 83.16% | 84.95% | 85.94% | 86.99% | 86.84% | 87.23% |
| Max | 74.64% | 81.09% | 83.56% | 85.05% | 86.32% | 87.28% | 86.98% | 87.35% |
| Min | 73.79% | 80.67% | 82.80% | 84.82% | 85.55% | 86.70% | 86.69% | 87.11% |
| STD | 0.35% | 0.17% | 0.31% | 0.10% | 0.38% | 0.29% | 0.14% | 0.12% |

Table 5.5: **Classification accuracies obtained with $11 \times 11$ random clusters and adaptive thresholding on the N-MNIST dataset using an ELM classifier**. Ten trials of each configuration were tested.

| | Hidden Layer Sizes | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 1000 | 2000 | 3000 | 4000 | 5000 | 6000 | 7000 | 8000 |
| Mean | 75.53% | 83.03% | 86.37% | 87.92% | 89.01% | 89.74% | 90.27% | 90.50% |
| Max | 75.79% | 83.24% | 86.81% | 88.27% | 89.34% | 89.98% | 90.75% | 90.69% |
| Min | 75.01% | 82.56% | 85.98% | 87.72% | 88.83% | 89.30% | 89.90% | 90.38% |
| STD | 0.28% | 0.25% | 0.28% | 0.24% | 0.17% | 0.23% | 0.27% | 0.11% |

associated thresholds.

Table 5.4 presents the results of the classification using the randomly-generated feature clusters. Interestingly, these results actually outperform the $5 \times 5$ clusters generated by the adaptive thresholding method, as shown in Figure 5.14. The random clusters outperformed the trained clusters by no more than 3%, but consistently, implying that the limited number of clusters is insufficient to cover the feature space of the input data, and that the random features were better able to cover the range of inputs in the higher dimensional input space.

The same methodology was then applied to the $11 \times 11$ clusters, with 100 clusters generated for each event polarity. Each cluster was randomly generated and normalised in the same manner as the $5 \times 5$ clusters, with the thresholds determined using the same iterative learning method. Figure 14 and Figure 15 in the Appendices show the random clusters generated for the ON and OFF events respectively.

Table 5.5 shows the classification results obtained using the $11 \times 11$ random

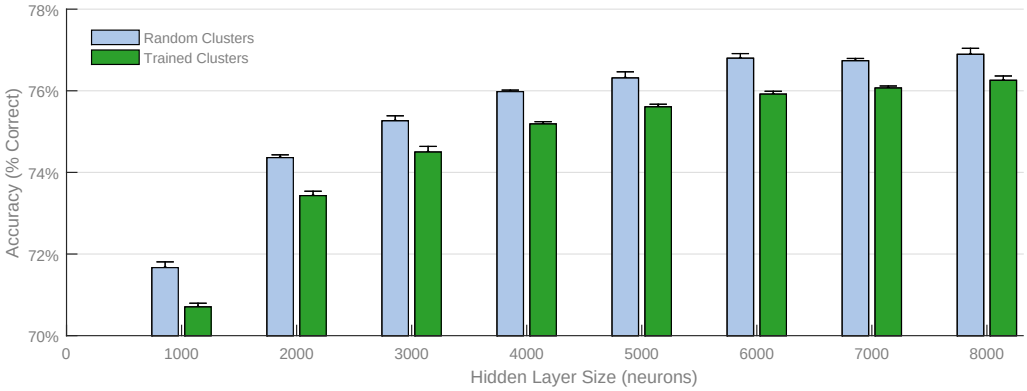



Figure 5.14: **Comparison of the classification accuracy between the trained $5 \times 5$ features and the random $5 \times 5$ features.** Each network contained 50 clusters, evenly split between ON and OFF clusters. Ten trials of each experiment were performed.

clusters. Eight hidden layer sizes were tested and ten trials of each experiment were performed. The $11 \times 11$ features outperformed the $5 \times 5$ random features for each hidden layer size, and achieved a maximum mean accuracy of 90.27% with 8000 hidden layer neurons.

Unlike the $5 \times 5$ random clusters, the $11 \times 11$ clusters did not outperform their trained equivalents. As shown in Figure 5.15, the $11 \times 11$ trained clusters outperformed the random clusters for each hidden layer configuration, with a strikingly higher performance when using networks of smaller hidden layer sizes.

This result is interesting as it validates the use of the trained clusters, and suggests that the $5 \times 5$ features do not encode enough specific information to produce good separability for the ELM classifier. The $11 \times 11$ clusters allowed for more clusters to be implemented (100 per polarity as opposed to 25), and each cluster encodes more spatial information per event, which allows the ELM classifier to achieve higher accuracies.

The performance of the random clusters provides additional insight into the nature of the information encoding in this type of network, as it shows that much of the information is contained within the temporal information, as the random features perform well without the explicit spatial structures that the adaptive clustering seeks to extract, and further upholds the theory that the motion of the camera serves to encode the spatial information into the temporal aspect of the spatio-temporal pattern.

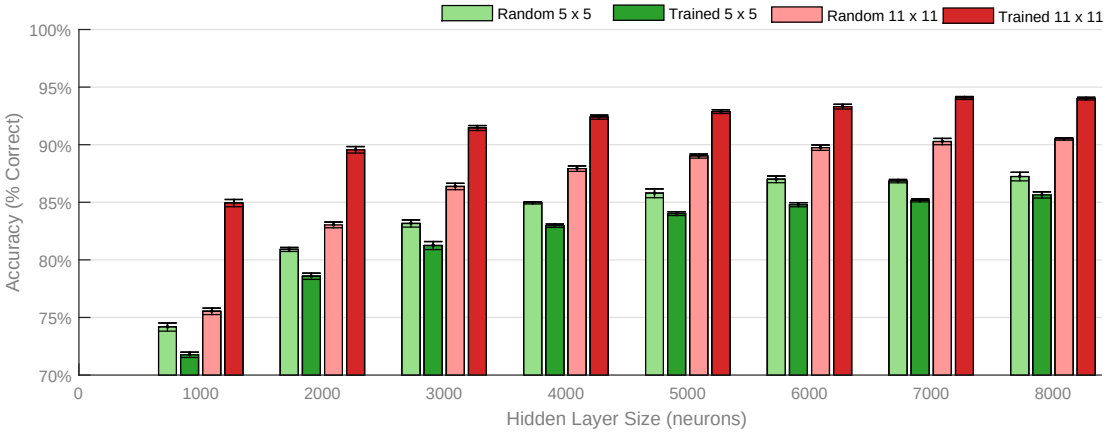



Figure 5.15: **Comparison of the classification accuracy achieved with the random clusters and trained clusters for varying hidden layer sizes**. The results for the two cluster configurations are presented for both the randomly generated clusters and the clusters learnt using the adaptive thresholding technique.

Table 5.6: **Summary of the accuracies using SKIM for the MNIST dataset using clustered features on the surface of time.**

| | Hidden Layer Sizes | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 1000 | 2000 | 3000 | 4000 | 5000 | 6000 | 7000 | 8000 |
| Mean | 85.97% | 90.47% | 91.88% | 92.75% | 93.43% | 93.73% | 94.11% | 94.25% |
| Max | 86.82% | 90.99% | 92.34% | 93.06% | 93.66% | 93.85% | 94.66% | 94.56% |
| Min | 85.01% | 89.95% | 91.50% | 92.39% | 92.87% | 93.58% | 93.92% | 93.97% |
| STD | 0.69% | 0.45% | 0.34% | 0.21% | 0.25% | 0.08% | 0.24% | 0.15% |

### 5.4.5 Surface Feature Classification with SKIM

Combining the online clustering with the SKIM network creates a fully event-based network from end-to-end. The network operates on each spike, updating the clusters and learning in a feed-forward manner. The SKIM network is also particularly well suited to the nature of the events produced by the adaptive threshold clustering, as they are inherently sparse spatio-temporal patterns.

Where the ELM required the vectorisation of the resulting spatio-temporal pattern in feature space, the SKIM network can operate on it directly, and therefore has only a single input channel for each feature cluster, allowing a far smaller input layer than in any previous SKIM network presented in this work.

Table 5.6 shows the results from the SKIM classifier using the $11 \times 11$ clusters

on the N-MNIST dataset. It is immediately clear that the SKIM network performs exceeding well, and achieves the highest overall accuracy for the N-MNIST dataset of any work presented in this thesis.

The SKIM network also outperforms the other networks and classifiers at all hidden layer sizes, achieving an accuracy of 85.97% with only 1000 hidden layer neurons, outperforming the best ELM result of 75.53% achieved with 1000 hidden layer nodes and the $11 \times 11$ clusters. The SKIM implementation also outperforms the ELM results achieved using the feature orientations in Section 5.3.3, with the orientations-based classifier achieving 81.75% with 1000 hidden layer nodes.

### 5.4.6 Discussion

The results presented in this section demonstrate that features extracted from the time surfaces introduced in Section 3.3 provide a good means to convert the classification task from a spatial domain to a feature domain. In contrast to the results obtained with the orientation features in Section 5.3, the entire system presented in this section operates in an event-based paradigm, and retains the sparse event-based representations generated by the ATIS camera.

Accordingly, the SKIM classifier also outperformed the ELM classifier. When using the orientation features, which diverged somewhat from a fully event-based methodology, the ELM classifier produced a better classification result. The sparse spatio-temporal patterns generated by the feature detection and extraction process outlined in this section are more consistent with the types of patterns for which the SKIM classifier was designed. The resulting online system produced better results with fewer neurons, and achieved the highest classification accuracy to date with the N-MNIST dataset.

The adaptive threshold clustering approach presented in this section also illustrated a number of interesting properties of event-based visual classification. The performance of the networks with random clusters, although not on par with the trained clusters for the $11 \times 11$ feature sizes, still produced good classification results, comparable to those achieved using the raw event streams for digit classification presented in Section 4.4.2.

The strong classification results achieved using random features leads to a number of important directions for future research. Using random clusters removes the need to explicitly learn features, reducing the complexity and complications for a hardware implementation. These features can be hard-coded and fixed, removing the need to store or transmit their values. It is even possible to make use of device heterogeneity to generate these clusters in hardware directly.

The use of random clusters also allows for the use of fixed clusters, which greatly simplifies the hardware implementation of such a system, allowing the use of the same clusters with multiple scenes. This also opens up other potential

research avenues regarding the use of orthogonal random clusters and other types of fixed clusters.

The random features also highlight the importance of camera motion in encoding spatial information into the temporal aspect of the event-based data. The conversion to time surface features effectively discards much of the spatial information in the event-stream. Each feature represents a specific spatial structure, but the conversion to a spatio-temporal pattern only indicates the presence of such a feature at a given time, rather than the specific spatial location of it.

The SKIM networks used to do the initial classification on the N-MNIST dataset in Section 4.4.2 mapped pixels directly to inputs to the SKIM network, and the delayed alpha functions allowed the effects of a spike on one input channel to affect the system at a later point in time. The results achieved using that framework relies on the spatial data which is no longer present in the feature spaces introduced in this section.

The strong performance of the purely random clusters indicates the importance of this temporal information, as it shows that the structural information attached to each cluster through training contributes only a small portion of the classification accuracy. This result serves to provide further evidence of the importance of both the spatial and temporal information in event-based vision, and is consistent with the effects underlying the improvements in accuracy resulting from the spatial downsampling presented in Section 4.6.

# Chapter 6

# Conclusions

## 6.1 Validation of the Neuromorphic Datasets

Good benchmarks are important as they serve to provide a means of comparing and contrasting algorithms and serve to spur interest and further development in the field of research surrounding them. The lack of consistent and well-adopted benchmarks in the neuromorphic community complicates this process of comparing and contrasting performances and figures of merit for algorithms and systems. This thesis introduced two new neuromorphic datasets, created from existing and well-established computer vision datasets and intended to address this shortfall. These datasets, based on the MNIST dataset and the Caltech101 datasets, maintain the structure and methodology of the originals, allowing for comparison and contrast to existing conventional computer vision systems.

This thesis includes a detailed analysis of these two datasets, in terms of the conversion process and by providing a number of important benchmark results. An initial set of statistical classifiers demonstrate a baseline classification accuracy, and the work in later chapters serves to set further accuracy benchmarks on the datasets. This work also serves to demonstrate the viability and usefulness of the two benchmark datasets in comparing, contrasting and characterising the network performance of event-based classifiers. The ability to compare to conventional computer vision approaches proved useful in the analysis of the results as it provided an intuitive means of understanding of the input data, which often is not easily accessible from the event-based output of the silicon retina.

The N-MNIST dataset, the neuromorphic conversion of the MNIST dataset, retains all the benefits of the original dataset in terms of the large corpus of available samples, the clear training and testing splits and the homogeneous and centred digits in each class. The N-Caltech101 dataset, representing the converted Caltech101 dataset, provides a more challenging classification task as does

the conventional computer vision equivalent. The numerous and diverse set of categories and the variety within create a far more difficult classification task more reminiscent of a real-world task.

As demonstrated in this thesis, these two datasets together provide a powerful means of characterising and evaluating feature detectors, AER filtering components and full classification systems. Despite the noise and artifacts introduced into the dataset information through the physical conversion process, the results achieved on the datasets come close to those achieved with similar techniques in conventional computer vision, especially in regard to the N-MNIST dataset.

## 6.2 Viability of Event-Based Object Classification

The work in this thesis deals primarily with the classification of objects from the event-based output of a silicon retina. This work explores two primary means of performing such classification. Chapter 4 presents the first approach which operates on the event-streams directly in order to perform classification, making use of both the spatial and temporal information in the patterns directly.

Chapter 5 describes the second approach to classification on event-based visual data, and makes use of extracted features from the event-based data in order to perform classification. Chapter 3 explores two classes of such features, providing a detailed description and analysis of the nature of these features, the means of extraction in an event-based manner and methods to assess the quality and usefulness of such features. A corollary to the detection of features of interest is the removal of noise and features that do not contribute information to the classification system, and the analysis of the feature detectors includes details on their noise filtering abilities. This is particularly relevant as it is possible to implement these systems as stand-alone noise filters for use in any event-based visual processing system.

The results from both approaches demonstrates the viability of an event-based classification system using silicon retinas and biologically-inspired learning mechanisms. The event-based nature of the majority of the systems presented in this work serves to maintain the data-driven ethos of event-based vision and computation. Although no explicit hardware implementations were undertaken in this work, it serves to verify the efficacy of such an approach and validates the benefits of event-based systems.

## 6.3 Applicability of SKIM and OPIUM to Event-Based Classification

This work explores the use of the SKIM and OPIUM learning methods with both event-based vision systems and large datasets. This work makes use of SKIM networks applied to the largest datasets to date, implementing the largest hidden layer sizes and simultaneously training the largest number of output neurons. The success of the classifiers built using these SKIM networks validates both the underlying SKIM algorithm and its applicability to event-based tasks such as those presented in this work. The further work on optimisation of the SKIM network for parallel GPU implementation serves as a first-step toward the faster hardware-based implementations required for real-time operation of such systems.

A number of experiments in this thesis also made use of ELM networks trained using the iterative OPIUM method, and made extensive use of the iterative nature of the algorithm to handle event-based data. The increased cost per update found in the OPIUM approach is offset by the ability to train and inspect the system iteratively, and to respond in an event-based manner to newly arriving information from the silicon retina. This work makes use of large three-layer networks trained with OPIUM in both a fan-in and fan-out configuration, and demonstrates the effectiveness of the method to large event-based techniques.

A further finding in this work is in the context in which to apply the OPIUM and SKIM methods. The work on detection using features presented in Chapter 5 serves to highlight the applicability of the two networks. Although both feature extraction methods operated on event-based data and produced spatio-temporal patterns, the nature of the two patterns differ, as do the results obtained using the SKIM and OPIUM methods. The orientation features produced dense spatio-temporal outputs on which the OPIUM-based ELM network produced a far superior performance, whereas the spatio-temporal patterns produced by the adaptive thresholding on the time surfaces were sparse by contrast, and resulted in better performance from the SKIM network.

The SKIM network is therefore better suited to effectively learning sparse spatio-temporal patterns, whereas the OPIUM-based ELM produces better results on denser spatio-temporal patterns. The SKIM method also performed better when operating in feature-space, rather than on the spatial information from the raw events from the silicon retina. This suggests that the SKIM method works optimally with less input channels that encode more information.

Another important distinction highlighted in this work is that the SKIM network operates in an event-based manner, whereas the OPIUM-based ELM method requires the vectorisation of the entire output prior to classification. For

this reason, the ELM networks were primarily used to validate the ability for classification, rather than to provide a viable means of performing on-line classification. The SKIM networks implemented in this thesis provided an on-line approach to both learning and recognition, with the final classification system presented using the time-surface features performing both the extraction of template features and the learning simultaneously in an event-based manner. As a result, the best performing classifier in this thesis is also the one closest to the fully event-based and iterative ethos of event-based vision and computation.

## 6.4 The Importance of Motion in Event-Based Classification

The best classification results achieved in this thesis involved either the spatial and temporal downsampling of the event-streams, or the extraction of features at the expense of spatial information, with the former producing the best results on the N-Caltech101 dataset and the latter producing the highest classification accuracy on the N-MNIST dataset. Both these techniques serve to highlight the importance of motion in the encoding of spatial information into the temporal aspect of the event-based data.

The spatial and temporal downsampling on the two datasets produced an increase in accuracy, with the spatial downsampling effectively creating small receptive fields on the input sequences, thereby providing some additional spatial information on each channel. The temporal downsampling also produced a boost in accuracy, particularly on the N-Calteech101 dataset. A candidate explanation for this increase in accuracy relates to the means by which the motion of the camera encodes the spatial structure of the scene in the relative timing of spatially adjacent events and the temporal downsampling aggregates this information for the classifier. Bolstering this theory is the increased accuracy obtained using the clustered features on the time surface, in which there is no direct spatial information provided to the classifier.

The conversion process used to create the two datasets made use of a fixed motion pattern for the camera, and the consistency of the motion across the conversion of each element in the datasets serves to make the classification task easier. It is likely that any consistent motion of the camera will result in a similar ability to perform classification. It may also be possible to make use of different random movements of the camera, especially if the nature of the movement is available to the classification system.

## 6.5 Future Work

The work presented in this thesis intends to provide a framework and investigation into the applicability and efficacy of event-based classification and object recognition using silicon retinas. The design of the methods placed more emphasis on exploring the nature of the problem and potential solutions than on efficient and hardware implementations. As the results of this work demonstrate the potential of such systems, the practical and real-time implementation of the components forms a portion of the future directions for this work. This work includes the implementation of the feature detectors as on-board hardware for the ATIS camera, FPGA and analogue implementations of the SKIM and OPIUM methods and use of the hardware-optimised SKAN method instead of the real-valued variant used in the adaptive threshold clustering.

An important future direction for this work is the extension of the principles to the realm of tracking in event-based data. The high-speed and event-based nature of the silicon retinas provides the optimal conditions for high-speed tracking, and much of the feature detection and processing presented in this thesis applies directly to the task of tracking, and can ultimately apply to the development of a truly event-based Simultaneous Localisation and Mapping (SLAM) algorithm. The two feature detectors presented in this work serve as candidate feature detectors for a SLAM algorithm, along with the ability to automatically extract relevant clusters using the adaptive thresholding technique. The classification and recognition tasks, along with the invariance achievable using the descriptors presented in this thesis, serve as potential building blocks in the recognition portion of a SLAM implementation.

# References


[1] Teresa Serrano-Gotarredona, Andreas G. Andreou, and Bernabé Linares-Barranco. AER image filtering architecture for vision-processing systems. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, 46(9):1064–1071, 1999. ISSN 10577122. doi: 10.1109/81.788808. vii, 11

[2] Christoph Posch, Daniel Matolin, and Rainer Wohlgenannt. A QVGA 143 dB Dynamic Range Frame-Free PWM Image Sensor With Lossless Pixel-Level Video Compression and Time-Domain CDS. *IEEE Journal of Solid-State Circuits*, 46(1):259–275, jan 2011. ISSN 0018-9200. doi: 10.1109/JSSC.2010.2085952. vii, 11, 14

[3] Garrick Orchard, Jie Zhang, Yuanming Suo, Minh Dao, Dzung T Nguyen, Sang Chin, Christoph Posch, Trac D Tran, and Ralph Etienne-Cummings. Real time compressive sensing video reconstruction in hardware. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 2(3):604–615, 2012. vii, 15

[4] Jonathan Tapson and André van Schaik. Learning the pseudoinverse solution to network weights. *Neural Networks*, 45:94–100, sep 2013. ISSN 08936080. doi: 10.1016/j.neunet.2013.02.008. viii, 23, 26, 27, 184

[5] José Antonio Pérez-Carrasco, Bo Zhao, Carmen Serrano-Gotarredona, Begoña Acha, Teresa Serrano-Gotarredona, Shouchun Chen, and Bernabé Linares-Barranco. Mapping from frame-driven to frame-free event-driven vision systems by low-rate rate coding and coincidence processing - Application to feedforward convnets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(11):2706–2719, 2013. ISSN 01628828. doi: 10.1109/TPAMI.2013.71. x, 22, 58, 59

[6] Garrick Orchard, Cedric Meyer, Ralph Etienne-Cummings, Christoph Posch, Nitish Thakor, and Ryad Benosman. HFirst: A Temporal Ap-

proach to Object Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8828(c):1–1, 2015. ISSN 0162-8828. doi: 10.1109/TPAMI.2015.2392947. x, 58, 59, 64, 83, 84, 87, 124

[7] Horace B. Barlow. Possible Principles Underlying the Transformations of Sensory Messages. *In Sensory Communication*, pages 217–234, 1961. 10

[8] Frank Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958. ISSN 0033-295X. doi: 10.1037/h0042519. 10

[9] Carver Mead. Neuromorphic electronic systems. *Proceedings of the IEEE*, 78(10):1629–1636, 1990. ISSN 00189219. doi: 10.1109/5.58356. 10

[10] Giacomo Indiveri and Timothy K. Horiuchi. Frontiers in neuromorphic engineering, 2011. ISSN 16624548. 10

[11] Vincent Yue Sek Chan, Craig T. Jin, and André van Schaik. Neuromorphic Audio-Visual Sensor Fusion on a Sound-Localizing Robot. *Frontiers in Neuroscience*, 6(February):1–9, 2012. ISSN 1662453X. doi: 10.3389/fnins.2012.00021. 10

[12] Kwabena Boahen. Communicating Neuronal Ensembles between Neuromorphic Chips. In *Neuromorphic Systems Engineering*, pages 229–259. Springer US, Boston, MA, 1998. ISBN 0-7923-8158-0. doi: 10.1007/978-0-585-28001-1{_}11. 10

[13] Eugenio Culurciello, Ralph Etienne-Cummings, and Kwabena Boahen. High dynamic range, arbitrated address event representation digital imager. *The 2001 IEEE International Symposium on Circuits and Systems*, 2:505–508, may 2001. doi: 10.1109/ISCAS.2001.921358. 10

[14] Massimo Barbaro, Pierre-Yves Burgi, Alessandro Mortara, Pascal Nussbaum, and Friedrich Heitger. A 100 x 100 pixel silicon retina for gradient extraction with steering filter capabilities and temporal output coding. *IEEE Journal of Solid-State Circuits*, 37(2):160–172, 2002. ISSN 00189200. doi: 10.1109/4.982422. 11

[15] Kwabena Boahen. A retinomorphic vision system. *IEEE Micro*, 16(5):30–39, 1996. ISSN 02721732. doi: 10.1109/40.540078. 11

[16] Charles M. Higgins and Shaikh A. Shams. A biologically inspired modular VLSI system for visual measurement of self-motion. *IEEE Sensors Journal*, 2(6):508–528, dec 2002. ISSN 1530-437X. doi: 10.1109/JSEN.2002.807304. 11

[17] Shih-Chii Liu, Jörg Kramer, Giacomo Indiveri, Tobi Delbrück, Thomas Burg, and Rodney J. Douglas. Orientation-selective aVLSI spiking neurons. *Neural Networks*, 14(6-7):629–643, 2001. ISSN 08936080. doi: 10.1016/S0893-6080(01)00054-5. 11

[18] Stephen R. Deiss, Rodney J. Douglas, and Adrian M. Whatley. A Pulse-Coded Communications Infrastructure for Neuromorphic Systems. *Pulsed Neural Networks*, pages 157–178, 1999. 11

[19] David H. Goldberg, Gert Cauwenberghs, and Andreas G. Andreou. Analog VLSI spiking neural network with address domain probabilistic synapses. In *ISCAS 2001. The 2001 IEEE International Symposium on Circuits and Systems*, volume 2, pages 241–244, 2001. ISBN 0-7803-6685-9. doi: 10.1109/ISCAS.2001.921292. 11

[20] Shih-Chii Liu and Rodney J. Douglas. Temporal coding in a silicon network of integrate-and-fire neurons. *IEEE Transactions on Neural Networks*, 15 (5):1305–1314, 2004. ISSN 10459227. doi: 10.1109/TNN.2004.832725. 11

[21] André van Schaik. Building blocks for electronic spiking neural networks. *Neural Networks*, 14(6-7):617–628, 2001. ISSN 08936080. doi: 10.1016/S0893-6080(01)00067-3. 11

[22] F. Gomez-Rodriguez, Alejandro Linares-barranco, L Miro, Shih-Chii Liu, André van Schaik, Ralph Etienne-Cummings, and M. A. Lewis. AER Auditory Filtering and CPG for Robot Control. In *2007 IEEE International Symposium on Circuits and Systems*, pages 1201–1204. IEEE, may 2007. ISBN 1-4244-0920-9. doi: 10.1109/ISCAS.2007.378268. 11

[23] Shih-Chii Liu, André van Schaik, B. Minch, and Tobi Delbrück. Event-Based 64-Channel Binaural Silicon Cochlea with Q Enhancement Mechanisms. In *IEEE International Symposium on Circuits and Systems*, Paris, 2010. 11

[24] Hisham Abdalla and Timothy K. Horiuchi. An ultrasonic filterbank with spiking neurons. *Proceedings - IEEE International Symposium on Circuits and Systems*, pages 4201–4204, 2005. ISSN 02714310. doi: 10.1109/ISCAS.2005.1465557. 11

[25] Nagendra Kumar, Wolfgang Himmelbauer, Gert Cauwenberghs, and Andreas G. Andreou. An analog VLSI architecture for auditory based feature extraction. In *1997 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 5, pages 4081–4084. IEEE Comput. Soc. Press, 1997. ISBN 0-8186-7919-0. doi: 10.1109/ICASSP.1997.604843. 11

[26] John Lazzaro and John Wawrzynek. A multi-sender asynchronous extension to the AER protocol. In *Proceedings Sixteenth Conference on Advanced Research in VLSI*, pages 158–169. IEEE Comput. Soc. Press, 1995. ISBN 0-8186-7047-9. doi: 10.1109/ARVLSI.1995.515618. 11

[27] Vincent Chan, Shih-Chii Liu, and André van Schaik. 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, 2007. ISSN 10577122. doi: 10.1109/TCSI.2006.887979. 11

[28] K. Fukushima, Y. Yamaguchi, M. Yasuda, and S. Nagata. An electronic model of the retina. *Proceedings of the IEEE*, 58(12):1950–1952, 1970. ISSN 0018-9219. doi: 10.1109/PROC.1970.8066. 11

[29] Misha Mahowald. *An analog VLSI system for stereoscopic vision*. Kluwer Int. series in engineering and computer science. Kluwer Academic Publishers, 1994. ISBN 9780792394440. 11

[30] Kareem A. Zaghloul and Kwabena Boahen. Optic Nerve Signals in a Neuromorphic Chip I: Outer and Inner Retina Models. *IEEE Transactions on Biomedical Engineering*, 51(4):657–666, 2004. ISSN 00189294. doi: 10.1109/TBME.2003.821039. 11

[31] Kareem A. Zaghloul and Kwabena Boahen. Optic Nerve Signals in a Neuromorphic Chip II: Testing and Results. *IEEE Transactions on Biomedical Engineering*, 51(4):667–675, 2004. ISSN 00189294. doi: 10.1109/TBME.2003.821040. 11

[32] Pierre-François Rüedi, Pascal Heim, François Kaess, Eric Grenet, Friedrich Heitger, Pierre-Yves Burgi, Stève Gyger, and Pascal Nussbaum. A 128 128 Pixel 120-dB Dynamic-Range Vision-Sensor Chip for Image Contrast and Orientation Extraction. *IEEE Journal of Solid-State Circuits*, 38(12): 2325–2333, 2003. ISSN 00189200. doi: 10.1109/JSSC.2003.819169. 11

[33] Pierre-François Rüedi, Pascal Heim, Stève Gyger, François Kaess, Claude Arm, Ricardo Caseiro, Jean Luc Nagel, and Silvio Todeschini. An SoC combining a 132dB QVGA pixel array and a 32b DSP/MCU processor for vision applications. In *Digest of Technical Papers - IEEE International Solid-State Circuits Conference*, pages 47–48, 2009. ISBN 1424434580. doi: 10.1109/ISSCC.2009.4977300. 11

[34] J. A. Leñero-Bardallo, T. Serrano-Gotarredona, and B. Linares-Barranco. A mismatch calibrated bipolar spatial contrast AER retina with adjustable

contrast threshold. In *Proceedings - IEEE International Symposium on Circuits and Systems*, number 2, pages 1493–1496, 2009. ISBN 9781424438280. doi: 10.1109/ISCAS.2009.5118050. 11

[35] Udayan Mallik, Matthew Clapp, Edward Choi, Gert Cauwenberghs, and Ralph Etienne-Cummings. Temporal change threshold detection imager. In *ISSCC. 2005 IEEE International Digest of Technical Papers. Solid-State Circuits Conference*, volume 39, pages 362–364. IEEE, 2005. ISBN 0-7803-8904-2. doi: 10.1109/ISSCC.2005.1494019. 11

[36] Jörg Kramer. An on/off transient imager with event-driven, asynchronous read-out. In *2002 IEEE International Symposium on Circuits and Systems. Proceedings*, volume 2, pages 165–168, 2002. ISBN 0-7803-7448-7. doi: 10.1109/ISCAS.2002.1010950. 11

[37] Patrick Lichtsteiner, Christoph Posch, and Tobi Delbrück. A 128 X 128 120db 30mw asynchronous vision sensor that responds to relative intensity change. *2006 IEEE International Solid State Circuits Conference - Digest of Technical Papers*, pages 2004–2006, 2006. ISSN 0193-6530. doi: 10.1109/ISSCC.2006.1696265. 11

[38] Vincent Chan, Craig Jin, and André van Schaik. An Address-Event Vision Sensor for Multiple Transient Object Detection. *IEEE Transactions on Biomedical Circuits and Systems*, 1(4):278–288, 2007. ISSN 1932-4545. doi: 10.1109/TBCAS.2007.916031. 11

[39] Christoph Posch, Daniel Matolin, Rainer Wohlgenannt, Thomas Maier, and Martin Litzenberger. A microbolometer asynchronous dynamic vision sensor for LWIR. *IEEE Sensors Journal*, 9(6):654–664, 2009. ISSN 1530437X. doi: 10.1109/JSEN.2009.2020658. 11

[40] Xavier Arreguit, F. André Van Schaik, François V. Bauduin, Marc Bidiville, and Eric Raeber. A CMOS motion detector system for pointing devices. *IEEE Journal of Solid-State Circuits*, 31(12):1916–1921, 1996. ISSN 00189200. doi: 10.1109/4.545813. 11

[41] Patrick Lichtsteiner, Christoph Posch, and Tobi Delbrück. A 128x128 120 dB 15 us Latency Asynchronous Temporal Contrast Vision Sensor. *IEEE Journal of Solid-State Circuits*, 43(2):566–576, 2008. ISSN 0018-9200. doi: 10.1109/JSSC.2007.914337. 11, 13

[42] Tobi Delbrück, Bernabe Linares-Barranco, Eugenio Culurciello, and Christoph Posch. Activity-driven, event-based vision sensors. In *Pro-*

*ceedings of 2010 IEEE International Symposium on Circuits and Systems*, pages 2426–2429. IEEE, may 2010. ISBN 978-1-4244-5308-5. doi: 10.1109/ISCAS.2010.5537149. 12

[43] Eugenio Culurciello, Ralph Etienne-Cummings, and Kwabena Boahen. A biomorphic digital image sensor. *IEEE Journal of Solid-State Circuits*, 38 (2):281–294, 2003. ISSN 0018-9200. doi: 10.1109/JSSC.2002.807412. 12

[44] C Brandli, R Berner, M Yang, S.C. Liu, V Villeneuva, and Tobi Delbrück. Live demonstration: The DAVIS Dynamic and Active-Pixel Vision Sensor. In *2014 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 440–440. IEEE, jun 2014. ISBN 978-1-4799-3432-4. doi: 10.1109/ISCAS.2014.6865163. 13

[45] Botond Roska and Frank Werblin. Rapid global shifts in natural scenes block spiking in specific ganglion cell types. *Nature Neuroscience*, 6(6): 600–8, jul 2003. ISSN 1097-6256. doi: 10.1038/nn1061. 13

[46] Kwabena Boahen. Point-to-point connectivity between neuromorphic chips using address events. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 47(5):416–434, may 2000. ISSN 10577130. doi: 10.1109/82.842110. 13

[47] Garrick Orchard, Daniel Matolin, Xavier Lagorce, Ryad Benosman, and Christoph Posch. Accelerated frame-free time-encoded multi-step imaging. In *2014 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 2644–2647. IEEE, jun 2014. ISBN 978-1-4799-3432-4. doi: 10.1109/ISCAS.2014.6865716. 13

[48] Tobi Delbruck, Bernabe Linares-Barranco, Eugenio Culurciello, and Christoph Posch. Activity-driven, event-based vision sensors. In *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, pages 2426–2429. IEEE, may 2010. ISBN 978-1-4244-5308-5. doi: 10.1109/ISCAS.2010.5537149. 14

[49] Christoph Posch, Teresa Serrano-Gotarredona, Bernabe Linares-Barranco, and Tobi Delbruck. Retinomorphic event-based vision sensors: Bioinspired cameras with spiking output. *Proceedings of the IEEE*, 102(10):1470–1484, 2014. ISSN 00189219. doi: 10.1109/JPROC.2014.2346153. 14

[50] Hanxuan Yang, Ling Shao, Feng Zheng, Liang Wang, and Zhan Song. Recent advances and trends in visual tracking: A review. *Neurocomputing*, 74 (18):3823–3831, 2011. ISSN 09252312. doi: 10.1016/j.neucom.2011.07.024. 16

[51] D. M. Gavrila. Pedestrian Detection from a Moving Vehicle. In *ECCV European Conference on Computer Vision*, volume 2, pages 37–49, 2000. doi: 10.1007/3-540-45053-X-3. 17

[52] Zhe Lin, Larry S. Davis, David Doermann, and Daniel DeMenthon. Hierarchical part-template matching for human detection and segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1–8, 2007. ISBN 978-1-4244-1631-8. doi: 10.1109/ICCV.2007.4408975. 17

[53] Vittorio Ferrari, Tinne Tuytelaars, and Luc Van Gool. Object Detection by Contour Segment Networks. In *9th European Conference on Computer Vision, Graz, Austria, May 7-13, 2006, Proceedings, Part III*, pages 14–28, 2006. ISBN 3-540-33836-5. doi: 10.1007/11744078{_}2. 17

[54] Mohammad Anvaripour and Hossein Ebrahimnezhad. Accurate object detection using local shape descriptors. *Pattern Analysis and Applications*, 18(2):277–295, 2013. ISSN 1433-7541. doi: 10.1007/s10044-013-0342-x. 17

[55] Bo Wu and Ram Nevatia. Detection of Multiple, Partially Occluded Humans in a Single Image by Bayesian Combination of Edglet Part Detector. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 1, pages 90–97, 2005. 17

[56] David G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110, nov 2004. ISSN 0920-5691. doi: 10.1023/B:VISI.0000029664.99615.94. 18, 29, 37

[57] C. Harris and M. Stephens. A Combined Corner and Edge Detector. In *Procedings of the Alvey Vision Conference 1988*, pages 147–151, 1988. doi: 10.5244/C.2.23. 18, 36

[58] Tony Lindeberg. Scale-space theory: A basic tool for analyzing structures at different scales. *Journal of applied statistics*, 21(2):225–270, 1994. 18

[59] Yan Ke and Rahul Sukthankar. PCA-SIFT: a more distinctive representation for local image descriptors. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 2, pages 506–513. IEEE, 2004. ISBN 0-7695-2158-4. doi: 10.1109/CVPR.2004.1315206. 18

[60] Alaa E. Abdel-Hakim and Aly Farag. CSIFT: A SIFT descriptor with color invariant characteristics. In *Proceedings of the IEEE Computer Society*

*Conference on Computer Vision and Pattern Recognition*, volume 2, pages 1978–1983, 2006. ISBN 0769525970. doi: 10.1109/CVPR.2006.95. 18

[61] Paul Scovanner, Saad Ali, and Mubarak Shah. A 3-Dimensional SIFT descriptor and its application to action recognition. In *Proceedings of the 15th international conference on Multimedia*, pages 357–361, New York, 2007. ACM Press. ISBN 9781595937025. doi: 10.1145/1291233.1291311. 18

[62] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. SURF: Speeded-Up Robust Features. *9th European Conference on Computer Vision*, 110(3): 346–359, 2008. doi: 10.1007/11744023{_}32. 18

[63] Michael Calonder, Vincent Lepetit, Mustafa Ozuysal, Tomasz Trzcinski, Christoph Strecha, and Pascal Fua. BRIEF: Computing a local binary descriptor very fast. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(7):1281–1298, nov 2011. ISSN 1939-3539. doi: 10.1109/TPAMI.2011.222. 18

[64] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. ORB: An efficient alternative to SIFT or SURF. In *2011 International Conference on Computer Vision*, pages 2564–2571. IEEE, nov 2011. ISBN 978-1-4577-1102-2. doi: 10.1109/ICCV.2011.6126544. 18

[65] Jie Chen, Shiguang Shan, Chu He, Guoying Zhao, Matti Pietikäinen, Xilin Chen, and Wen Gao. WLD: A robust local image descriptor. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1705–20, sep 2010. ISSN 1939-3539. doi: 10.1109/TPAMI.2009.155. 18

[66] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Proceedings - 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2005*, volume I, pages 886–893, 2005. ISBN 0769523722. doi: 10.1109/CVPR.2005.177. 19, 41

[67] Navneet Dalal, Bill Triggs, and Cordelia Schmid. Human detection using oriented histograms of flow and appearance. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 3952, pages 428–441, 2006. ISBN 3540338349. doi: 10.1007/11744047{_}33. 19

[68] Alejandro Linares-barranco, F Gomez-Rodriguez, A. Jimenez-Fernandez, Tobi Delbrück, and P Lichtensteiner. Using FPGA for visuo-motor control with a silicon retina and a humanoid robot. In *2007 IEEE International*

*Symposium on Circuits and Systems*, pages 1192–1195. Ieee, may 2007. ISBN 1-4244-0920-9. doi: 10.1109/ISCAS.2007.378265. 20

[69] Jorg Conradt, Raphael Berner, Matthew Cook, and Tobi Delbrück. An embedded AER dynamic vision sensor for low-latency pole balancing. In *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*, pages 780–785. IEEE, sep 2009. ISBN 978-1-4244-4442-7. doi: 10.1109/ICCVW.2009.5457625. 20

[70] Tobi Delbrück and Manuel Lang. Robotic goalie with 3 ms reaction time at 4% CPU load using event-based dynamic vision sensor. *Frontiers in Neuroscience*, 7(7 NOV):1–7, 2013. ISSN 16624548. doi: 10.3389/fnins.2013.00223. 20

[71] AN Belbachir, M. Litzenberger, S. Schraml, M. Hofstatter, D. Bauer, P. Schon, M. Humenberger, C. Sulzbachner, T. Lunden, and M. Merne. CARE: A dynamic stereo vision sensor system for fall detection. In *2012 IEEE International Symposium on Circuits and Systems*, pages 731–734, Seoul, may 2012. IEEE. ISBN 978-1-4673-0219-7. doi: 10.1109/ISCAS.2012.6272141. 20

[72] Ryad Benosman, Sio Hoï Ieng, Paul Rogister, and Christoph Posch. Asynchronous event-based Hebbian epipolar geometry. *IEEE Transactions on Neural Networks*, 22(11):1723–1734, 2011. ISSN 10459227. doi: 10.1109/TNN.2011.2167239. 20

[73] Luis Camuñas-Mesa, Teresa Serrano-Gotarredona, Sio Hoï Ieng, Ryad Benosman, and Bernabé Linares-Barranco. On the use of orientation filters for 3D reconstruction in event-driven stereo vision. *Frontiers in neuroscience*, 8(March):48, 2014. ISSN 1662-4548. doi: 10.3389/fnins.2014.00048. 20

[74] Stephan Schraml, Ahmed Nabil Belbachir, Nenad Milosevic, and Peter Schon. Dynamic stereo vision system for real-time tracking. In *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, pages 1409–1412. IEEE, may 2010. ISBN 978-1-4244-5308-5. doi: 10.1109/ISCAS.2010.5537289. 20

[75] João Carneiro, Sio Hoï Ieng, Christoph Posch, and Ryad Benosman. Event-based 3D reconstruction from neuromorphic retinas. *Neural Networks*, 45: 27–38, 2013. ISSN 08936080. doi: 10.1016/j.neunet.2013.03.006. 20

[76] Ryad Benosman, Sio Hoï Ieng, Charles Clercq, Chiara Bartolozzi, and Mandyam Srinivasan. Asynchronous frameless event-based optical flow.

*Neural networks : the official journal of the International Neural Network Society*, 27:32–7, mar 2012. ISSN 1879-2782. doi: 10.1016/j.neunet.2011.11.001. 20

[77] Ryad Benosman, Charles Clercq, Xavier Lagorce, Sio Hoï Ieng, and Chiara Bartolozzi. Event-based visual flow. *IEEE transactions on neural networks and learning systems*, 25(2):407–17, feb 2014. ISSN 2162-2388. doi: 10.1109/TNNLS.2013.2273537. 20, 39

[78] Garrick Orchard, Ryad Benosman, Ralph Etienne-Cummings, and Nitish V Thakor. A spiking neural network architecture for visual motion estimation. In *IEEE Biomedical Circuits and Systems Conference, BioCAS*, number 1, pages 298–301, 2013. ISBN 9781479914715. 20

[79] Zhenjiang Ni, C. Pacoret, Ryad Benosman, Sio-Hoi. Ieng, and Stéphane Régnier. Asynchronous event-based high speed vision for microparticle tracking. *Journal of Microscopy*, 245(3):236–244, mar 2012. ISSN 00222720. doi: 10.1111/j.1365-2818.2011.03565.x. 20

[80] Zhenjiang Ni, Aude Bolopion, Joël Agnus, Ryad Benosman, and Stéphane Régnier. Asynchronous event-based visual shape tracking for stable haptic feedback in microrobotics. *IEEE Transactions on Robotics*, 28(5):1081–1089, 2012. ISSN 15523098. doi: 10.1109/TRO.2012.2198930. 20

[81] David Reverter Valeiras, Xavier Lagorce, Xavier Clady, Chiara Bartolozzi, Sio Hoï Ieng, and Ryad Benosman. An Asynchronous Neuromorphic Event-Driven Visual Part-Based Shape Tracking. *IEEE Transactions on Neural Networks and Learning Systems*, (99):1–1, 2015. ISSN 2162-237X. doi: 10.1109/TNNLS.2015.2401834. 20

[82] Xavier Lagorce, Cedric Meyer, Sio-Hoi Ieng, David Filliat, and Ryad Benosman. Asynchronous Event-Based Multikernel Algorithm for High-Speed Visual Features Tracking. *IEEE transactions on neural networks and learning systems*, pages 1–12, sep 2014. ISSN 2162-2388. doi: 10.1109/TNNLS.2014.2352401. 20

[83] Jun Haeng Lee, Tobi Delbrück, Michael Pfeiffer, Paul K. J. Park, Chang-Woo Shin, Hyunsurk Ryu, and Byung Chang Kang. Real-Time Gesture Interface Based on Event-Driven Processing from Stereo Silicon Retinas. *IEEE Transactions on Neural Networks and Learning Systems*, 25(12):2250–2263, 2014. ISSN 2162237X. doi: 10.1109/TNNLS.2014.2308551. 21

[84] Hanme Kim, Ankur Handa, Ryad Benosman, Sio Hoï Ieng, and Andrew J. Davison. Simultaneous Mosaicing and Tracking with an Event Camera. In *BMVC*, pages 1–12, 2014. 21

[85] Xavier Lagorce, Sio Hoï Ieng, Xavier Clady, Michael Pfeiffer, and Ryad Benosman. Spatiotemporal features for asynchronous event-based data. *Frontiers in Neuroscience*, 9(February):1–13, 2015. ISSN 1662-453X. doi: 10.3389/fnins.2015.00046. 21

[86] Eustace Painkras, Luis A Plana, Jim Garside, Steve Temple, Francesco Galluppi, Cameron Patterson, David R Lester, Andrew D Brown, and Steve B Furber. SpiNNaker: A 1-W 18-Core System-on-Chip for Massively-Parallel Neural Network Simulation. *IEEE Journal of Solid-State Circuits*, 48(8):1943–1953, aug 2013. ISSN 0018-9200. doi: 10.1109/JSSC.2013.2259038. 21, 84

[87] Francesco Galluppi, Xavier Lagorce, Evangelos Stromatias, Michael Pfeiffer, Luis A. Plana, Steve B. Furber, and Ryad Benosman. A framework for plasticity implementation on the SpiNNaker neural architecture. *Frontiers in Neuroscience*, 8(January):1–16, jan 2015. ISSN 1662-453X. doi: 10.3389/fnins.2014.00429. 21

[88] Evangelos Stromatias, Daniel Neil, Michael Pfeiffer, Francesco Galluppi, Steve B. Furber, and Shih-Chii Liu. Robustness of spiking Deep Belief Networks to noise and reduced bit precision of neuro-inspired hardware platforms. *Frontiers in Neuroscience*, 9(July):1–14, 2015. ISSN 1662-453X. doi: 10.3389/fnins.2015.00222. 21

[89] Ben Varkey Benjamin, Peiran Gao, Emmett McQuinn, Swadesh Choudhary, Anand R. Chandrasekaran, Jean-Marie Bussat, Rodrigo Alvarez-Icaza, John V Arthur, Paul A Merolla, and Kwabena Boahen. Neurogrid: A Mixed-Analog-Digital Multichip System for Large-Scale Neural Simulations. *Proceedings of the IEEE*, 102(5):699–716, may 2014. ISSN 0018-9219. doi: 10.1109/JPROC.2014.2313565. 21

[90] Robert F. Service. The brain chip. *Science*, 345(6197):614–616, 2014. ISSN 0036-8075. doi: 10.1126/science.345.6197.614. 21

[91] Eugene M Izhikevich. Polychronization: computation with spikes. *Neural computation*, 18(2):245–82, feb 2006. ISSN 0899-7667. doi: 10.1162/089976606775093882. 21

[92] Runchun Wang, Gregory Cohen, Klaus M Stiefel, Tara Julia Hamilton, Jonathan Tapson, and André van Schaik. An FPGA implementation of a

polychronous spiking neural network with delay adaptation. *Frontiers in neuroscience*, 7, 2013. 21

[93] Arindam Basu, Sun Shuo, Hongming Zhou, Meng Hiot Lim, and Guang Bin Huang. Silicon spiking neurons for hardware implementation of extreme learning machines. *Neurocomputing*, 102:125–134, 2013. ISSN 09252312. doi: 10.1016/j.neucom.2012.01.042. 22

[94] Robert Gütig and Haim Sompolinsky. The tempotron: a neuron that learns spike timing-based decisions. *Nature neuroscience*, 9(3):420–428, 2006. ISSN 1097-6256. doi: 10.1038/nn1643. 22

[95] Timothée Masquelier and Simon J. Thorpe. Unsupervised learning of visual features through spike timing dependent plasticity. *PLoS Computational Biology*, 3(2):0247–0257, 2007. ISSN 1553734X. doi: 10.1371/journal.pcbi.0030031. 22

[96] Filip Ponulak and Andrzej Kasinski. Supervised Learning in Spiking Neural Networks with ReSuMe : Sequence Learning , Classification and Spike-Shifting. *Neural Computation*, pages 1–31, 1999. 22

[97] Ammar Mohemmed, Stefan Schliebs, Satoshi Matsuda, and Nikola Kasabov. SPAN: spike pattern association neuron for learning spatio-temporal spike patterns. *International Journal of Neural Systems*, 22(04): 1250012, aug 2012. ISSN 0129-0657. doi: 10.1142/S0129065712500128. 22

[98] Peter O'Connor, Daniel Neil, Shih Chii Liu, Tobi Delbrück, and Michael Pfeiffer. Real-time classification and sensor fusion with a spiking deep belief network. *Frontiers in Neuroscience*, 7(7 OCT):1–13, 2013. ISSN 16624548. doi: 10.3389/fnins.2013.00178. 22

[99] Emre Neftci, Srinjoy Das, Bruno Pedroni, Kenneth Kreutz-Delgado, and Gert Cauwenberghs. Event-driven contrastive divergence for spiking neuromorphic systems. *Frontiers in Neuroscience*, 7(January):1–14, 2014. ISSN 1662-453X. doi: 10.3389/fnins.2013.00272. 22

[100] Chris Eliasmith and Charles H. Anderson. *Neural Engineering: Computation, Representation, and Dynamics in Neurobiological Systems*. MIT Press, 2004. ISBN 0262550601. 22, 23, 26, 84

[101] Dan Goodman and Romain Brette. Brian: a simulator for spiking neural networks in python. *Frontiers in neuroinformatics*, 2(November):5, 2008. ISSN 1471-2202. doi: 10.3389/neuro.11.005.2008. 22

[102] Fabio Stefanini, Emre Neftci, Sadique Sheik, and Giacomo Indiveri. PyNCS: a microkernel for high-level definition and configuration of neuromorphic electronic systems. *Frontiers in Neuroinformatics*, 8(5):1–14, 2014. ISSN 1662-5196. doi: 10.3389/fninf.2014.00073. 22

[103] Andrew P. Davison, Daniel Brüderle, Jochen Eppler, Jens Kremkow, Eilif Muller, Dejan Pecevski, Laurent Perrinet, and Pierre Yger. PyNN: A Common Interface for Neuronal Network Simulators. *Frontiers in neuroinformatics*, 2(January):11, 2008. ISSN 1662-5196. doi: 10.3389/neuro.11.011.2008. 22

[104] Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew. Extreme learning machine: Theory and applications. *Neurocomputing*, 70(1-3):489–501, dec 2006. ISSN 09252312. doi: 10.1016/j.neucom.2005.12.126. 23, 24

[105] André van Schaik and Jonathan Tapson. Online and adaptive pseudoinverse solutions for ELM weights. *Neurocomputing*, 149:233–238, feb 2015. ISSN 09252312. doi: 10.1016/j.neucom.2014.01.071. 24, 25, 26, 124, 182

[106] Guang-Bin Huang. Learning capability and storage capacity of two-hidden-layer feedforward networks. *IEEE Transactions on Neural Networks*, 14(2): 274–281, mar 2003. ISSN 1045-9227. doi: 10.1109/TNN.2003.809401. 24

[107] Guang Bin Huang, Dian Hui Wang, and Yuan Lan. Extreme learning machines: A survey. *International Journal of Machine Learning and Cybernetics*, 2(2):107–122, 2011. ISSN 18688071. doi: 10.1007/s13042-011-0019-y. 25

[108] T. N. E. Greville. Some Applications of the Pseudoinverse of a Matrix. *SIAM Review*, 2(1):15–22, jan 1960. ISSN 0036-1445. doi: 10.1137/1002004. 25

[109] Noah Snavely, Steven M Seitz, and Richard Szeliski. Skeletal graphs for efficient structure from motion. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, jun 2008. ISBN 978-1-4244-2242-5. doi: 10.1109/CVPR.2008.4587678. 29

[110] Jun Yang, Yu-Gang Jiang, Alexander G Hauptmann, and Chong-Wah Ngo. Evaluating bag-of-visual-words representations in scene classification. In *Proceedings of the international workshop on Workshop on multimedia information retrieval - MIR '07*, page 197, New York, New York, USA, 2007. ACM Press. ISBN 9781595937780. doi: 10.1145/1290082.1290111. 29

[111] Ivan Laptev and Tony Lindeberg. Local Descriptors for Spatio-temporal Recognition. In WJ MacLean, editor, *Spatial Coherence for Visual Motion Analysis*, pages 91–103. Springer Berlin Heidelberg, 2006. ISBN 978-3-540-32534-5. doi: 10.1007/11676959{_}8. 29

[112] Piotr Dollar, V. Rabaud, G. Cottrell, and Serge Belongie. Behavior Recognition via Sparse Spatio-Temporal Features. In *2005 IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, pages 65–72. IEEE, 2005. ISBN 0-7803-9424-0. doi: 10.1109/VSPETS.2005.1570899. 29

[113] S. Se, D. Lowe, and J. Little. Mobile Robot Localization and Mapping with Uncertainty using Scale-Invariant Visual Landmarks. *The International Journal of Robotics Research*, 21(8):735–758, aug 2002. ISSN 0278-3649. doi: 10.1177/027836402761412467. 29

[114] Tinne Tuytelaars and Krystian Mikolajczyk. Local Invariant Feature Detectors: A Survey. *Foundations and Trends in Computer Graphics and Vision*, 3(3):177–280, 2007. ISSN 1572-2740. doi: 10.1561/0600000017. 30

[115] H Moravec. Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover. Technical report, 1980. 36

[116] Jianbo Shi and Carlo Tomasi. Good features to track. *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR '94., 1994 IEEE Computer Society Conference on*, 1994. ISSN 1063-6919. doi: 10.1109/CVPR.1994.323794. 38

[117] Pedro E. Maldonado, Imke Gödecke, Charles M. Gray, and Tobias Bonhoeffer. Orientation selectivity in pinwheel centers in cat striate cortex. *Science*, 276(5318):1551–1555, 1997. ISSN 00368075. doi: 10.1126/science.276.5318.1551. 44

[118] J. Bowers, I. Morton, and G. Mould. Directional statistics of the wind and waves. *Applied Ocean Research*, 22(1):13–30, 2000. ISSN 01411187. doi: 10.1016/S0141-1187(99)00025-5. 44

[119] Thomas Kubiak and Cornelia Jonas. Applying circular statistics to the analysis of monitoring data: Patterns of social interactions and mood. *European Journal of Psychological Assessment*, 23(4):227–237, 2007. ISSN 10155759. doi: 10.1027/1015-5759.23.4.227. 44

[120] Jennifer a. Mooney, Peter J. Helms, and Ian T. Jolliffe. Fitting mixtures of von Mises distributions: A case study involving sudden infant death

syndrome. *Computational Statistics and Data Analysis*, 41(3-4):505–513, 2003. ISSN 01679473. doi: 10.1016/S0167-9473(02)00181-0. 44, 48, 50

[121] John A Long and Paul C Stoy. Peak tornado activity is occurring earlier in the heart of Tornado Alley. *Geophysical Research Letters*, 41(17):6259–6264, 2014. ISSN 00948276. doi: 10.1002/2014GL061385. 44

[122] S Rao Jammalamadaka and Ambar Sengupta. *Topics in circular statistics*, volume 5. World Scientific, 2001. ISBN 9812779264. 44

[123] Philipp Berens and Marc J. Valesco. The Circular Statistics Toolbox for Matlab. *Technical Report, 184*, 1(184):1–7, 2009. 44

[124] Kanti V. Mardia and Peter E. Jupp. *Directional Statistics*. John Wiley & Sons, 2009. ISBN 0470317817. 47

[125] Suvrit Sra. A short note on parameter approximation for von Mises-Fisher distributions: And a fast implementation of Is(x). *Computational Statistics*, 27(1):177–190, 2012. ISSN 09434062. doi: 10.1007/s00180-011-0232-x. 47

[126] Arindam Banerjee and Inderjit Dhillon. Clustering on the unit hypersphere using von Mises-Fisher distributions. *Journal of Machine Learning Research*, 6:1345–1382, 2005. ISSN 15324435. 47

[127] N. I. Fisher. *Statistical Analysis of Circular Data*. Cambridge University Press, 1995. ISBN 0521568900. 47, 51

[128] Yannis Agiomyrgiannakis and Yannis Stylianou. Stochastic modeling and quantization of harmonic phases in speech using wrapped gaussian mixture models. *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, 4:1121–1124, 2007. ISSN 15206149. doi: 10.1109/ICASSP.2007.367271. 50

[129] Yannis Agiomyrgiannakis and Yannis Stylianou. Wrapped gaussian mixture models for modeling and high-rate quantization of phase data of speech. *IEEE Transactions on Audio, Speech and Language Processing*, 17(4):775–786, 2009. ISSN 15587916. doi: 10.1109/TASL.2008.2008229. 50

[130] Arindam Banerjee, Inderjit Dhillon, Joydeep Ghosh, and Suvrit Sra. Generative model-based clustering of directional data. In *ACM SIGKDD international conference on Knowledge discovery and data mining*, page 19, 2003. ISBN 1581137370. doi: 10.1145/956755.956757. 50

[131] B. Ajne. A Simple Test for Uniformity of a Circular Distribution. *Biometrika*, 55(2):343, 1968. ISSN 00063444. doi: 10.2307/2334875. 51

[132] Edward Batschelet. *Circular Statistics in Biology*. Academic Press, 1981. ISBN 0120810506. 51

[133] G. S. Watson. Goodness-Of-Fit Tests on a Circle. *Biometrika*, 48(1/2):109, jun 1961. ISSN 00063444. doi: 10.2307/2333135. 51

[134] Jerrold H. Zar. *Biostatistical Analysis (5th Edition)*. Prentice-Hall, Inc., aug 2007. ISBN 0131008463. 51

[135] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2323, 1998. ISSN 00189219. doi: 10.1109/5.726791. 57, 58

[136] Thomas Serre, Lior Wolf, Stanley Bileschi, Maximilian Riesenhuber, and Tomaso Poggio. Robust object recognition with cortex-like mechanisms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(3): 411–426, 2007. ISSN 01628828. doi: 10.1109/TPAMI.2007.56. 57, 64, 83, 96, 176, 178

[137] P Grother. NIST special database 19 handprinted forms and characters database, 1995. URL `http://www.nist.gov/srd/upload/nistsd19.pdf`. 58

[138] M.J. Dominguez-Morales, P. Inigo-Blasco, A. Linares-Barranco, G. Jimenez, A. Civit-Balcells, and J.L. Sevillano. Performance study of synthetic AER generation on CPUs for Real-Time Video based on Spikes. *2009 International Symposium on Performance Evaluation of Computer & Telecommunication Systems*, 41, 2009. 59

[139] M. R. López-Torres, F. Diaz-Del-Rio, M. Domínguez-Morales, G. Jimenez-Moreno, and A. Linares-Barranco. AER spiking neuron computation on GPUs: The frame-to-AER generation. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 7062 LNCS(PART 1):199–208, 2011. ISSN 03029743. doi: 10.1007/978-3-642-24955-6{_}24. 59

[140] José Antonio Pérez-Carrasco, Carmen Serrano-Gotarredona, Begoña Acha-Piñero, Teresa Serrano-Gotarredona, and Bernabé Linares-Barranco. Advanced vision processing systems: Spike-based simulation and processing. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 5807 LNCS, pages 640–651, 2009. ISBN 3642046967. doi: 10.1007/978-3-642-04697-1{_}60. 59

[141] T. Serrano-Gotarredona and B. Linares-Barranco. MNIST-DVS Database. URL `http://www2.imse-cnm.csic.es/caviar/MNISTDVS.html`. 59

[142] Ralf Engbert. Microsaccades: a microcosm for research on oculomotor control, attention, and visual perception. In *Progress in Brain Research*, volume 154, pages 177–192. 2006. ISBN 978-0-444-52966-4. doi: 10.1016/S0079-6123(06)54009-9. 60

[143] Teresa Serrano-Gotarredona and Bernabé Linares-Barranco. Poker-DVS and MNIST-DVS. Their History, How They Were Made, and Other Details. *Frontiers in Neuroscience*, 9(December):1–10, 2015. ISSN 1662-453X. doi: 10.3389/fnins.2015.00481. 60

[144] Garrick Orchard, Ajinkya Jayawant, Gregory Cohen, and Nitish Thakor. Converting Static Image Datasets to Spiking Neuromorphic Datasets Using Saccades. *TBA*, pages 1–15, 2015. 63, 65, 97, 123

[145] Jonathan Tapson, Gregory Cohen, Saeed Afshar, Klaus Stiefel, Yossi Buskila, Runchun Mark Wang, Tara J Hamilton, and André van Schaik. Synthesis of neural networks for spatio-temporal spike pattern recognition and processing. *Frontiers in neuroscience*, 7:153, jan 2013. ISSN 1662-4548. doi: 10.3389/fnins.2013.00153. 65, 68, 182

[146] Jonathan Tapson, Gregory Cohen, and André van Schaik. ELM solutions for event-based systems. *Neurocomputing*, 149:435–442, feb 2015. ISSN 09252312. doi: 10.1016/j.neucom.2014.01.074. 65

[147] Wang Wei Lee, Haoyong Yu, and Nitish V Thakor. Gait event detection through neuromorphic spike sequence learning. In *5th IEEE RAS/EMBS International Conference on Biomedical Robotics and Biomechatronics*, pages 899–904. IEEE, aug 2014. ISBN 978-1-4799-3128-6. doi: 10.1109/BIOROB.2014.6913895. 65

[148] Jr. Frank J. Massey. Kolmogorov-Smirnov Test for Goodness of Fit. *Test*, 46 (253):68– 78, 2011. ISSN 0162-1459. doi: 10.1080/01621459.1951.10500769. 74

[149] Hubert W Lilliefors. On the Kolmogorov-Smirnov Test for Normality with Mean and Variance Unknown. *Journal of the American Statistical Association*, 62(318):399–402, jun 1967. ISSN 0162-1459. doi: 10.1080/01621459.1967.10482916. 75, 120

[150] Garrick Orchard, Jacob G Martin, R Jacob Vogelstein, and Ralph Etienne-Cummings. Fast neuromimetic object recognition using FPGA outperforms

GPU implementations. *IEEE Transactions on Neural Networks and Learning Systems*, 24(8):1239–1252, 2013. 91, 96, 120, 176

[151] Saeed Afshar, Libin George, Jonathan Tapson, André van Schaik, and Tara Julia Hamilton. Racing to learn: statistical inference and learning in a single spiking neuron with adaptive kernels. *Frontiers in Neuroscience*, 8(November):1–18, nov 2014. ISSN 1662-453X. doi: 10.3389/fnins.2014.00377. 129

[152] J. MacQueen. Some Methods for classification and Analysis of Multivariate Observations. In *5th Berkeley Symposium on Mathematical Statistics and Probability 1967*, volume 1, pages 281–297, 1967. ISBN 1595931619. 132

[153] Aristidis Likas, Nikos Vlassis, and Jakob J. Verbeek. The global k-means clustering algorithm. *Pattern Recognition*, 36(2):451–461, 2003. ISSN 00313203. doi: 10.1016/S0031-3203(02)00060-2. 132

[154] Li Fei-Fei, Rob Fergus, and Pietro Perona. Learning generative visual models from few training examples: An incremental Bayesian approach tested on 101 object categories. *Computer Vision and Image Understanding*, 106 (1):59–70, 2007. ISSN 10773142. doi: 10.1016/j.cviu.2005.09.012. 174

[155] Jiaoyan Chen, Huajun Chen, Xiangyi Wan, and Guozhou Zheng. MR-ELM: a MapReduce-based framework for large-scale ELM training in big data era. *Neural Computing and Applications*, 2014. ISSN 09410643. doi: 10.1007/s00521-014-1559-3. 182

[156] Fei Han, Hai-Fen Yao, and Qing-Hua Ling. An improved evolutionary extreme learning machine based on particle swarm optimization. *Neurocomputing*, 116:87–93, 2012. ISSN 09252312. doi: 10.1016/j.neucom.2011.12.062. 182

[157] Mark Van Heeswijk, Yoan Miche, Erkki Oja, and Amaury Lendasse. GPU-accelerated and parallelized ELM ensembles for large-scale regression. *Neurocomputing*, 74(16):2430–2437, 2011. ISSN 09252312. doi: 10.1016/j.neucom.2010.11.034. 182

[158] Richard Vuduc, Aparna Chandramowlishwaran, Jee Choi, Murat Guney, and Aashay Shringarpure. On the limits of GPU acceleration. In *Proceedings of the 2nd USENIX conference on Hot topics in parallelism*, page 13, 2010. 183

[159] André van Schaik and Jonathan Tapson. Online and Adaptive Pseudoinverse Solutions for ELM Weights. *International Conference on Extreme*

*Learning Machines*, 149:1–9, 2013. ISSN 09252312. doi: 10.1016/j.neucom.2014.01.071. 185

# Appendix A: Detailed Analysis of N-MNIST and N-Caltech101

## Analysis of the N-MNIST Dataset

The MNIST dataset contains a total of 70,000 presentations of the handwritten digits in the range of 0 to 9. These are divided into a training set of 60,000 images, and a further 10,000 images reserved for testing purposes. The N-MNIST contains the same number of testing and training items, and preserves the distribution of images between them. Numbering, ordering and labelling match those in the MNIST dataset exactly.

As the nature of each training and testing item is no longer a static image, but rather an event sequence from a physical device, the specific characteristics relating to events of each sequence varies slightly. This is partially due to slight variations in the motor timing and responses, but primarily due to the nature of the digits and the event-based nature of the ATIS device. For a more thorough overview of how these cameras work, please refer to Section 2.2.3 and Section 2.2.4.

Figure 1 shows the characteristics of the sequences in the training and testing datasets respectively. It can be seen that the structure of both the testing and training set are similar. The pattern lengths are consistent between the testing and training sets, and based on this data a fixed pattern length of 315 ms was chosen to fully contain all digit sequences. A summary of the overall statistical properties for the entire dataset is given in Table 1, shown separately for the training and testing dataset.

Each digit consists of an AER recording from a $34 \times 34$ pixel window centred on the digit. As Figure 4.5 shows, this larger spatial resolution was required to keep the full $28 \times 28$ pixel digit in the frame during the motion of the camera.

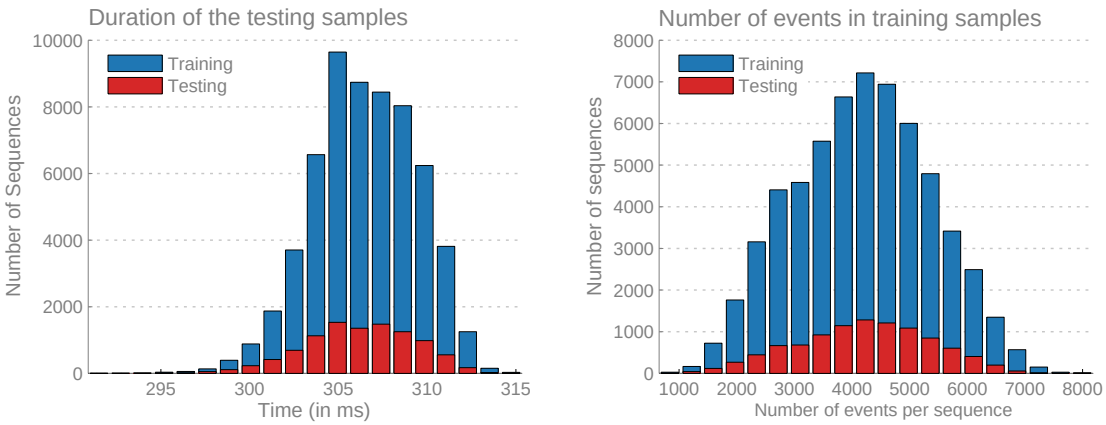



Figure 1: **Histograms showing the number of events and the duration of the sequences for the training and testing datasets.** The testing and training histograms are overlaid to illustrate the similar nature of the testing and training sets. There are 60,000 training samples, and 10,000 testing samples and the same bin-widths were used for both the training and testing data.

Table 1: **Statistical summary of the N-MNIST dataset**. Results are shown separately for the training set of 60,000 digits, and the testing set comprising 10,000 digits.

| | Training Set | | Testing Set | |
|---|---|---|---|---|
| Statistic | Mean | $\sigma$ | Mean | $\sigma$ |
| Duration of Recording (ms): | 306.5 | 2.898 | 306.2 | 3.05 |
| Number of Events: | 4171.89 | 1196.2 | 4203.6 | 1157.4 |
| Number of ON Events | 2083.69 | 573.59 | 2087.4 | 557.28 |
| Number of OFF Events | 2088.3 | 622.75 | 2116.19 | 600.4 |
| $x$ Addresses (pixels) | 17.66 | 5.05 | 17.66 | 4.97 |
| $y$ Addresses (pixels) | 18.1 | 6.38 | 18.11 | 6.36 |

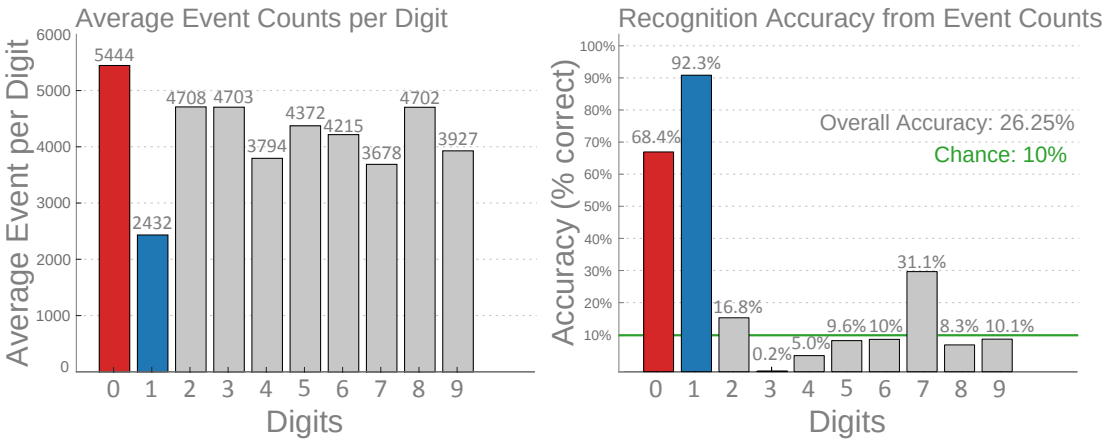



Figure 2: **Average number of events per digit in the dataset (left), and accuracy of classification per digit when attempting to classify using the number of events alone**. The classification method determined the digit with the closest mean number of events to the current test sample. It is interesting to note that digit 0 has the largest number of events, whilst digit 1 has the fewest.

The average length of each recording is 306 ms, and contains an average of 4172 events. However, the number of events varies from digit to digit. Figure 2 shows the average number of events per digit on the right. It is interesting to note that the digit 0 contains the largest number of spikes on average, whilst the digit 1 contains the fewest.

In order to explore the predictive power of the number of events in each pattern, a simple classifier was constructed to classify each testing sequence based on the digit with the closest mean to the current number of events for that training sample. The results of this classifier are shown in Figure 2. Overall, the classifier produced a recognition accuracy of just 26.25%, which is well above the chance level of 10%.

Individual digit results are presented in Figure 2, where it can be seen that the digits 0 and 1 displayed the highest recognition accuracy. This is to be expected, considering that they occupy the two ends of the range of event numbers. This result may be significant in explaining the tendencies of poor classifiers to favour these two digits.

The above approach takes the simplest and most direct means of constructing a classifier by examining the digit means. A more robust approach uses a k-nearest neighbour (kNN) method to classify each digit. This has the disadvantage of being computationally more costly during recall as it cannot be as readily precomputed as in the digit means method above. A kNN approach was implemented and tested for 9 different statistical properties calculated across the each

individual training and testing sample. For each pattern, the following statistical properties were used:

1. The total number of events in the pattern
2. The duration of the pattern in milliseconds
3. The number of ON events in the pattern
4. The number of OFF events in the pattern
5. The ratio of ON events to OFF events in the pattern
6. The mean $x$ address calculated across all the events in the pattern
7. The mean $y$ address calculated across all the events in the pattern
8. The standard deviation of the $x$ addresses in the pattern
9. The standard deviation of the $y$ addresses in the pattern

The values shown in Table 1 give the mean value and standard deviation across each individual sequence in the training and testing sets. These same parameters were calculated for each individual training sequence, and then used as the source dataset for a kNN classifier. For each testing sequence, the same statistics were then found, and the closest $k$ neighbours extracted. Values of $k$ up to 1000 were tested, and the results stabilised with a $k \geq 10$.

A vector containing the $k$ neighbouring digits was then created, and the mode extracted to serve as the final classification output. Due to the sheer size of the training set, it is often possible to have multiple training sequences that are exactly the same distance from the testing sample in question, resulting in a tie. The classifier was tested with, and without including ties, will no significant difference seen in performance.

Figure 3 shows the results of the kNN-classifier when applied to the 9 different statistics from the N-MNIST dataset. The classifiers based on duration of sequence, and the means of duration and event coordinates (shown in grey in the figure) yielded statistically insignificant classification results. As the original MNIST digits were centred using their centre of mass within the $28 \times 28$ pixel window, and as the N-MNIST dataset is based on a scaled and centred version of those digits, it holds that the N-MNIST dataset exhibits this same property. It is therefore to be expected that the mean $x$ and $y$ position should hold no statistical value, as visible in Figure 3.

As the movements and timings were all controlled, the duration of the pattern should ideally be equal for all sequences. Due to noise and real-world effects

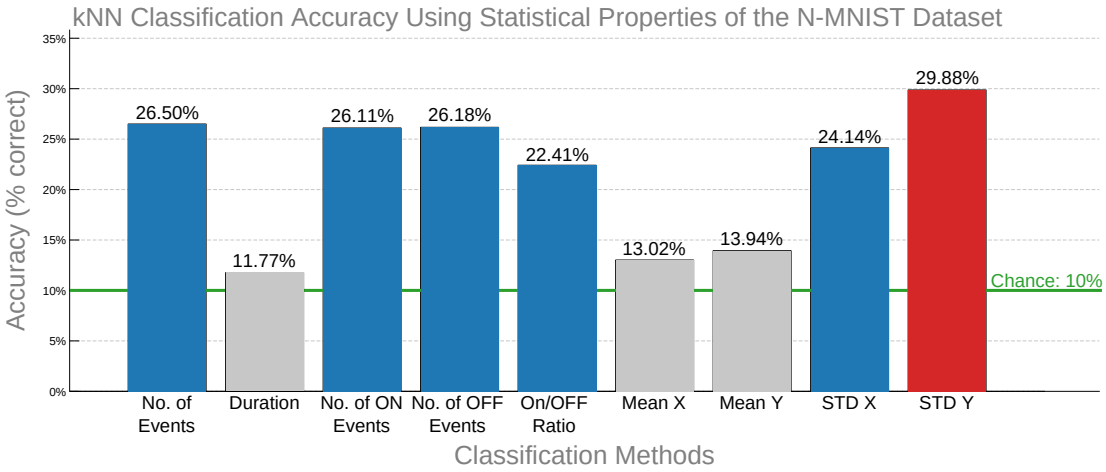



Figure 3: **Classification Results for a kNN Classifier based on Statistical Properties of the N-MNIST dataset**. The resulting accuracy of the kNN classifiers with a k = 10 for 9 statistical measures on the dataset. Statistically insignificant results are shown in grey.

resulting from timing and accuracies in the hardware and acquisition process, there is a small variation (as seen in Table 1) in the duration of each pattern. The performance of the classifier based on the duration of the sequence, as shown in Figure 3, demonstrates that this small offset contains no significant classifying power.

The classifiers based on the number of events produced a classification accuracy well above the level of pure chance, and achieved similar performance when applied to the number of ON and OFF events, in line with expectations as the number of ON and OFF events in the dataset is well balanced.

Finally, the standard deviation of $y$ produced the best classification result. The standard deviation of the $x$ values also performed well, but did not yield the same level of performance. The overall standard deviation in $y$ across the whole dataset is larger than that of the $x$ pixels, which arises from the fact that the third saccade motion is a pure rotation, and therefore produces fewer events for features that exhibit strong vertical gradients.

# Analysis of the N-Caltech101 Dataset

The Caltech101 dataset, originally collated to provide an assortment of images to test online object recognition algorithms [154], contains images of objects from 101 categories and a separate background class. Intended to provide a more

challenging classification problem than the MNIST dataset, it contains far fewer training samples per category and far more classes.

Unlike the MNIST dataset, there is no specific, logical or defined split between testing and training data and, due to the complexity of the task, a number of different methodologies to divide the dataset exists. Issues arise from the varying number of image samples in each category, requiring either the use of only a subset of the available images, or the post-classification adjustment of results to reflect the imbalanced number of training samples per category.

The Caltech101 dataset images were converted using the same process used to create the N-MNIST dataset (detailed further in Section 4.3.3). The same saccade timings were used to generate the patterns, and the distance from the LCD monitor determined for the N-MNIST dataset was maintained. To compensate for the differing image sizes, each image was resized to fit within the frame of the ATIS sensor whilst maintaining its aspect ratio. As before, no noise filtering was performed either in hardware or software.

Table 2: **Statistical Summary of the N-Caltech101 Dataset**. The statistics for the N-Caltech101 dataset are presented separately for the 101 object classes and the background class as shown below. As the N-Caltech101 images are different sizes, the range of $x$ and $y$ addresses are also included in the statistics.

| | Object Categories | | Background Category | |
|---|---|---|---|---|
| Statistic | Mean | $\sigma$ | Mean | $\sigma$ |
| Duration of Recording (ms) | 300.16 | 4.60 | 300.92 | 5.26 |
| Number of Events | 115117 | 57968 | 140620 | 71924 |
| Number of ON Events | 56936 | 28039 | 69023 | 34577 |
| Number of OFF Events | 58180 | 30021 | 71597 | 37443 |
| $x$ Range (pixels) | 198.53 | 43.079 | 189.96 | 45.11 |
| $y$ Range (pixels) | 155.88 | 26.76 | 166.82 | 12.37 |
| $x$ Addresses (pixels) | 100.72 | 57.784 | 95.83 | 57.78 |
| $y$ Addresses (pixels) | 81.23 | 46.15 | 84.51 | 46.15 |

Table 2 shows a statistical summary of the N-Caltech101 dataset, with separate statistics for the object classes and the background class. As with the N-MNIST dataset, the duration of each sequence is close to the expected duration of the three 100 ms saccades. Unlike the MNIST dataset, the number of events is far greater and exhibits a far greater variability. It is likely that this difference is the result of the increased complexity and heterogeneity of the real-world objects and scenes in the Caltech101 dataset.

Despite the varying image sizes, in which the widths ranges from 41 pixels to 223 pixels and the heights from 55 pixels to 173 pixels, the mean $x$ and $y$ addresses (analogous to the concept of the image centre of mass) are squarely within the mean $x$ and $y$ ranges, indicating that the images are well centred within the camera frame. The table also shows that, similarly to the N-MNIST dataset, there is also a good balance between ON and OFF events on average. There is a large variance around the mean for the number of events, which is indicative of the wide range of image sizes, and the varying content of each image.

It is important to consider the nature of the sensor in the image conversion process, especially with respect to real-world scenes. As each pixel operates independently, and responds only to change in illumination, it is not always possible to reliably correlate the image size to the number of events, as an image with large areas of uniform intensity will generate fewer spikes. Section 2.2.3 contains a full treatment of the nature of these imaging devices.

In a similar vein to the approach taken the with the N-MNIST dataset, statistical measures were used to build a simple kNN classifier in order to determine a base level for classification accuracy. As the images sizes vary for the Caltech101 dataset, two new statistics based on the $x$ and $y$ sizes of the images were also included.

As an initial test, the binary classification task proposed by Serre et al. [136] and more recently by Orchard et al. [150], was attempted as it provides benchmarks by which the results can be compared.

For each category, the task is to determine if the object is present or not. The background image class is included in the training to provide the non-object training data. The classifier is trained on equal numbers of samples from the class under test and the background image class, with the remaining images from both sets used for testing. The results of each class were not weighted to reflect the number of samples available for testing, and therefore the results for each class should be considered independently.

Although the two papers mentioned above make use of six categories, only four of these are contained in the Caltech101 dataset (airplanes, cars, faces and motorbikes) and only those four were used in these experiments. Each category was tested independently, and the results of the binary classification task are shown in Figure 4.

The binary separation task is one of the least challenging classification tasks for the Caltech101 dataset, and most algorithms achieve accuracies in the range of 96% to 99%, and it is clear from Figure 4 that the statistical approach does yield some predictive power. Performing a sweep through all $k$ values in the range of $k \in [1, 1000]$ found that a value of $k = 10$ produced good results with a reasonable computational load, and only a small increases in accuracy ($\pm 0.5\%$) occurred with higher values of $k$, but at a higher computational cost. This value

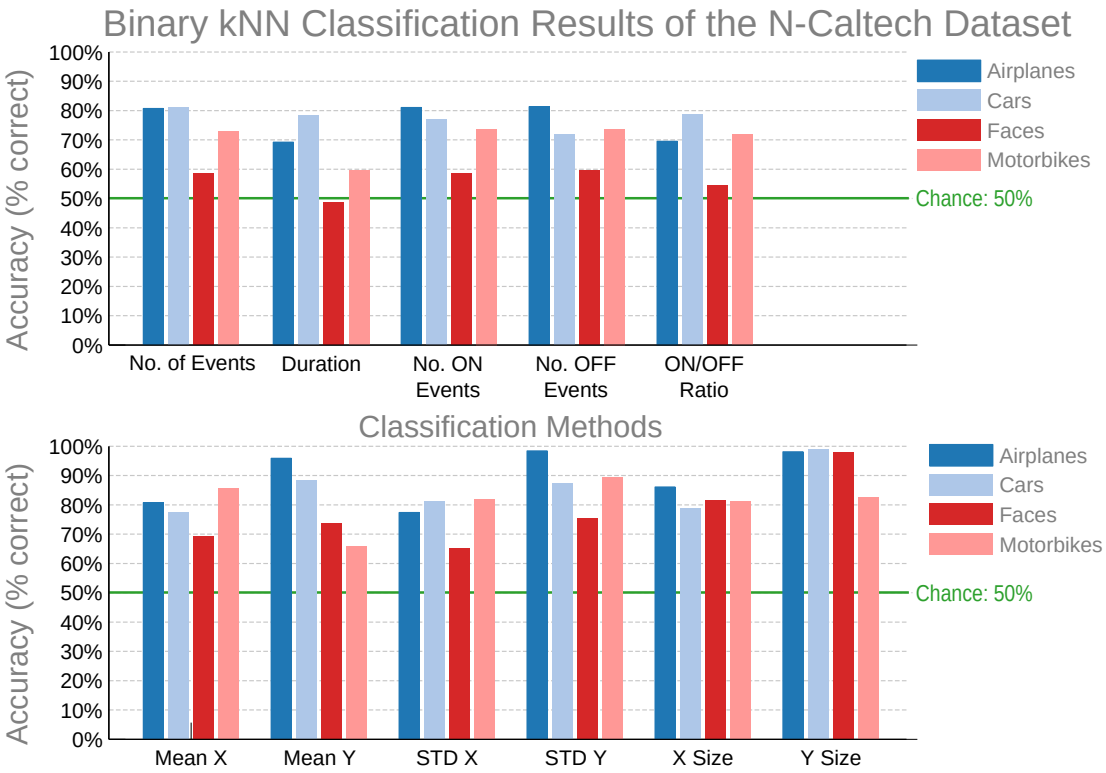



Figure 4: **Results for kNN Classifiers based on statistical properties of the N-MNIST dataset**. The classifiers were trained with $k = 10$, which was empirically found to produce marginally better results than any other value such that $k > 4$. The four categories were tested independently from each other. As this is a binary task, the probability due to chance is 50%.

of $k$ is also consistent with the value used in the statistical classifiers previously for the N-MNIST dataset.

The number of events in each sequence proved to have less classification power than it did against the N-MNIST dataset. The similarity between each sequence within each class, and the stark contrasts between items of different classes in the N-MNIST dataset (as highlighted by the blue and red bars shown in Figure 2) may serve to explain the better result on N-MNIST.

It is immediately apparent that the two additional classifiers based on sequence size ($x$ size and $y$ size) provide the best results. This is a factor that needs consideration when the using the data in a learning algorithm. The specifics of the learning system, and the means by which data is inputted, are the primary factors that dictate the handling of images of non-uniform size. As an example, Serre et al [136] restricted the image width to 140 pixels and scaled the height accordingly, but their implementation used a multi-layer network with pooling, and the difference in size did not have an impact on their system or on its structure. When dealing with the N-Caltech101 dataset, this difference in image size manifests as a change in the range of the $x$ and $y$ dimensions of the event stream for each sequence.

Using the same subset of data in a full 5-way classification task, with the same statistical measures, yields the results shown in Figure 5, which also shows the full confusion matrices for all 11 classifiers and the overall classification accuracy achieved. The 5-way classification task considers all the samples from the different categories and labels them according to their class. The kNN classifiers for the binary separation task made use of only two classes, whereas there are five classes available in this experiment.

The dataset required division into an equal training and testing set, with half the available samples from each category included in each. There is no inherent training with a kNN classifier, but rather a calculation step which generates the labelled data. The testing process performs the actual nearest neighbour calculation, and allows for the varying of the number of neighbours (the $k$ value) during the testing. This also inherently allows for multiple and parallel testing, allowing for the wide sweeps of k-values performed. As the entire process is deterministic, there is no need to run multiple trials. The order of training and testing also plays no role, and there are no effects arising from them.

The results in Figure 5 demonstrate a number of interesting properties of the dataset. As expected, the classifier based on the duration of the sequences provided almost no classifying power as the duration is uniform across all samples. In addition the classifier based on the ratio of ON to OFF events also produced a result close to chance, which is in line with the balanced nature of the number of ON and OFF events shown in Table 2. The classifiers based on the number of events produced similar results for the number of ON events and the number

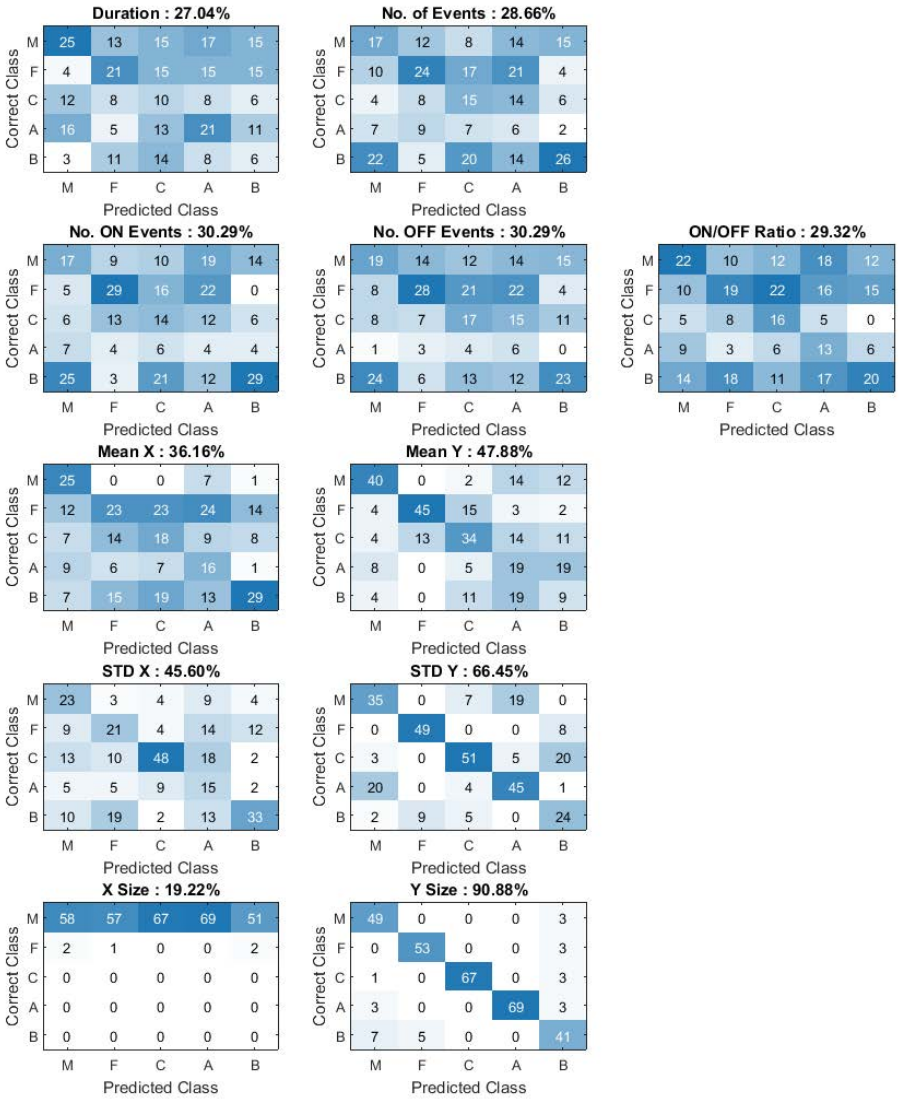



Figure 5: **Confusion matrices for the 5-category classification problem using a kNN classifier trained with statistical information.** For each of the eleven statistical classifiers, the full 5-way confusion matrix is presented, along with the overall accuracy achieved. The matrices are coloured to highlight the distribution of classification results.

of OFF events, but slightly better for the total number of events. The complex nature of the objects and the dissimilarity between objects within a single category may serve to explain why the number of events does not perform well as a classification statistic.

The classifiers based on the mean and standard deviation of the $x$ and $y$ addresses of the events in each sequence produced much better results than those based on the number of events. These relate to the balance and distribution about the centre of the image, and demonstratively possess some classification ability. As mentioned in the discussion on conversion methodology in Section 4.3.3, the conversion process attempted to maintain the aspect ratio of each sequence and also ensured that each image was centred within the camera frame. It is important to note that this centre is not necessarily the same as the centre of mass of the image itself, but rather the central pixel of the sequence. It assumes that the image is already centred within the input image during the conversion process. It is likely that the object structure affects the distribution of pixels around the central point, and the variability around it gives rise to the ability to classify with the standard deviations about the means.

The results from the $x$ and $y$ size classifiers are also interesting. The almost uniform response from the classifier based on the $x$ size arises from the resizing action of the conversion process, which limits the width of the image, and adjusts the height accordingly to maintain a constant aspect ratio. The high accuracy attained with the $y$ size is indicative of this, and demonstrates that there is a lot of information encoded in the aspect ratio of the image. This is likely due to the shape of the objects having a direct impact on the framing of the images, and therefore the aspect ratio.

The same statistical classification approach can also be applied to the full 101-way classification task. The number of images per category varies in the N-Caltech101 dataset, and therefore only 30 images from each category could be used in order to ensure equal numbers of images from each category. The same eleven classifiers were used, and trained using the same methodology as in the 5-way classification task.

Figure 6 shows the results of the 101-way classification task using the eleven statistical classifiers. The probability due to chance in this task is below 1%. As before, the classifiers based on $x$ and $y$ size produced the highest accuracies, but it is clear from the plots that this is only achieved by correctly identifying a handful of categories and defaulting to a single class for the rest. The classifiers based on the means achieved the second best results, with a more even spread across the object classes.

These results are intended to serve as a benchmark representing the lowest expected accuracy from any given classification technique.

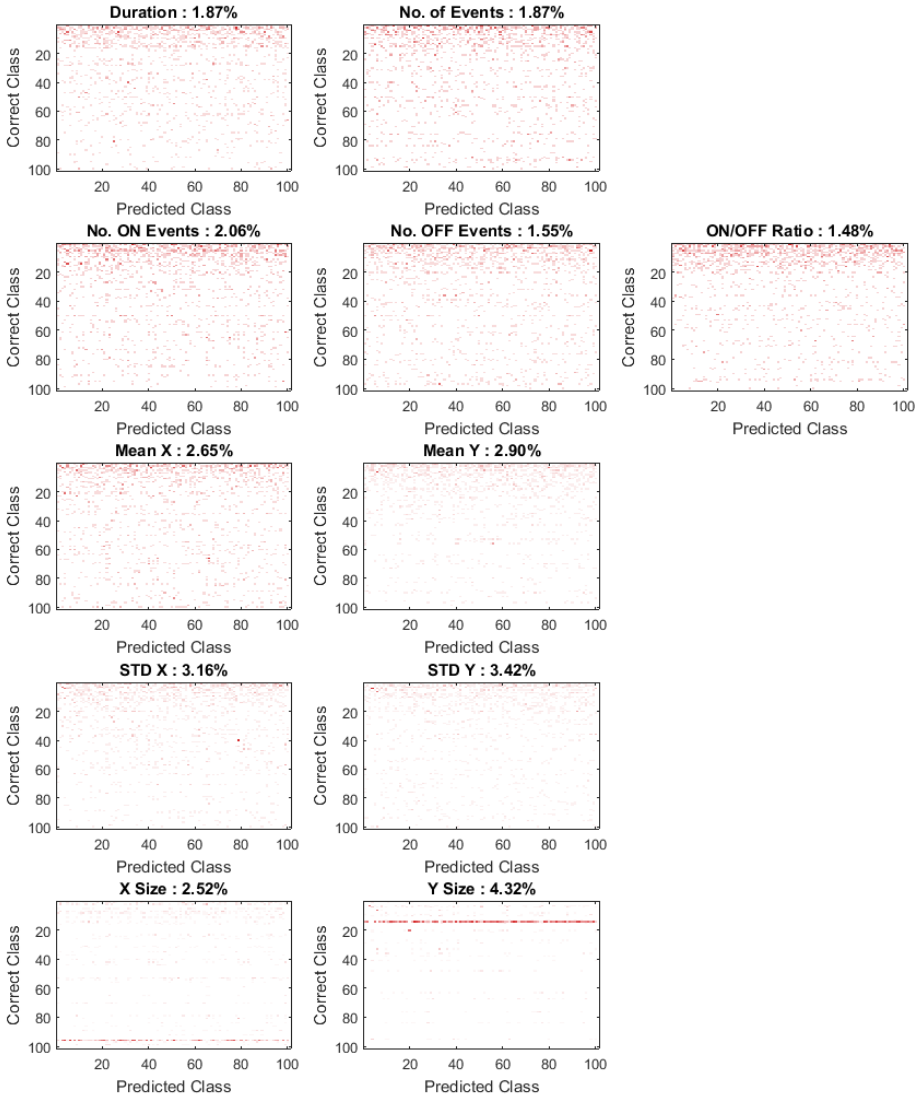



Figure 6: **Classification accuracy and confusion matrices for the 101-way classification tasks performed using the kNN classifier.** The confusion matrices for all 101 classes are shown with the colours indicating the relative number of samples falling into that classification. In the case of the 101-way classification task, the accuracy due to chance is less than 1%. These graphs show that the statistical classifiers perform poorly and favour only a small subset of the available classes. The graphs also lack the large values on the diagonal corresponding to correct predictions which are present in later classifiers.

# Appendix B: Optimisation Methods for LSHDI Networks

## Introduction

This section introduces and describes the optimisation methods used throughout this thesis. These methods were instrumental in allowing for the breadth and depth of testing performed with regard to the SKIM and OPIUM methods. These techniques enabled the exploration of large networks and provided the means to perform multiple trials of experiments to produce statistically rigorous results. The methods detailed in this section relate primarily to the optimisation of the existing SKIM and OPIUM techniques outlined in [145] and [105] respectively.

There have been a number of different approaches used to improve the speed and efficiency of the ELM algorithm. These have included recasting the problem as a MapReduce problem [155] to allow for distributed processing, making use of particle-swarm optimisation to optimally choose optimal input weights [156] and a variety of methods making use of the GPU to accelerate the processing [157]. Whereas these methods focus on improving or accelerating the conventional ELM approach, the techniques discussed in this section extend this work to the iterative methods of determining the pseudo-inverse weights used in the OPIUM and SKIM algorithms.

The iterative implementation can even benefit more directly from the increased processing capabilities of a GPU than the conventional ELM algorithm, as the iterative algorithm forgoes computational efficiency for the ability to iteratively train. This allows methods such as OPIUM to operate in an online manner, relieving the need to wait until all data is present before processing. As a result, each update requires a computationally expensive update step, each of which can benefit directly from the GPU optimisation.

## Contributions

This section introduces the following contributions:

- Provides an adaptation of the OPIUM method for use on a GPU, characterising the performance gains and exploring the effect on memory utilisation and classification accuracy.
- Extends the GPU approach to the SKIM method and demonstrates the performance gains achieved using the GPU.
- Explores the nature of iterative solutions to the pseudo-inverse method for over-defined systems, highlighting the potential pitfalls and optimisations possible using the iterative algorithms discussed in this thesis.

## OPIUM Optimisation with a GPU

The Graphical Processing Unit (GPU) has gained popularity over the past five years, due to a sudden increase in the availability and accessibility of the technology. This has created an entire field of research focused on crafting, adapting and reconstituting existing algorithms and systems to utilise this computational paradigm. Although a wide range of problems can directly benefit from the sheer parallelisation offered by a GPU, it is not always possible to optimise every solution. There have also been a number of cases in which the impressive performance gains can be misleading, and biased toward the GPU solution [158]. Often, the comparison systems used in those experiments makes use of old algorithms and out-of-date code, and applying a similar amount of time and effort to optimising these systems with a conventional CPU-based approach would yield a similar performance boost.

The fundamental nature of a GPU differs from that of a conventional processor, and certain algorithms lend themselves particularly well to the slower clock speed but immensely parallel nature of these devices. In these cases, the performance boost attained through a GPU device can be orders of magnitude faster than a standard CPU approach. Matrix manipulations are an example of a set of operations where performing the calculations in a parallel and distributed manner is trivial. SKIM and OPIUM both make use of an iterative approach to calculating the pseudo-inverse solution to a large ELM network and this iterative update algorithm naturally lends itself to a GPU optimisation. These algorithms are also computationally costly, and are often limited by computational time rather than memory or storage limitations. Therefore, there is a strong need to find

methods to speed up the calculation of these networks to allow for larger and more intricate networks, and to move toward a real-time online learning system.

This section explores the benefits of utilising a GPU in OPIUM-based techniques through a number of experiments designed to demonstrate the practical and theoretical improvements that are achievable, followed by a discussion of the problems and limitations of the such systems.

## Implementation

At the core of the OPIUM method is an iterative update of the inverse correlation matrix used to calculate to update the linear output weights on each iteration. This allows for the method to handle any number of training samples, but at the cost of increased processing per training sample (as the correlation matrix requires an update for each sample, and is not updated in batch as would be the case when using a non-iterative means of calculating the inverse correlation matrix). The nature of this problem lends itself particularly well to optimisation on a GPU as is evident when one examines the update equation for the pseudo-inverse of the correlation matrix $\theta_k$ from [4] and introduced in Section 2.5.2:

$$\theta_k = \theta_k - \theta_k a_k b_k^T \tag{1}$$

When dealing with large systems of equations, the size of the vector $\theta_k$ in 1 can grow large in size (often having between 1 and 10 million elements), causing the computational requirements for calculating the transpose with itself to grow exponentially. Computing the dot product of two matrices or vectors is an operation that is particularly well suited to a parallel implementation, and it is this equation that benefits from a significant speed boost from a GPU implementation.

However, a common problem that complicates the use of GPUs is the differing architectures in which the GPU and CPU operate. Beyond the fundamental differences in computing organisation, each device also has its own separate physical memory. Utilising both the GPU and CPU in an algorithm requires careful consideration of the costs involved in moving data between the two systems, especially as these transfers incur significant overhead. Often these overheads can serve to nullify any increase gained in computational speed.

The structure of the OPIUM method, and all the deriving methods that follow, can avoid the worst of these overhead issues when taking special considering in calculating and applying the updates to the inverse correlation matrix. By examining the implementation of the algorithm, it is clear that the inverse correlation matrix $\theta_k$ is only used in two operations; the determination of the error weighting for a new update of the linear output weights, and then a update to

itself based upon the resultant error. The salient point is that both of these operations can take place entirely within the architecture of the GPU device, and the large $\theta_k$ matrix can remain entirely on the GPU, with only trivially small matrices (the training input matrix, and the output weights matrix) needing transfer back and forth from the CPU domain. All of the high intensity computation with respect to the calculating of the pseudo-inverse can take place on the GPU with no costly memory transfer overheads.

## Results

When using OPIUM to iteratively solve the weights of an ELM system, the implementation of the update on the GPU yields a significant increase in performance. The inverse correlation matrix $\theta_k$ is inherently the largest matrix in an OPIUM implementation and grows exponentially in size with the number of hidden layer neurons, which in turn grows with the size of the input when using a fixed fan-out. The update of this matrix, as shown in (1), is the most computationally intensive step in each training update (approximately 76% of the CPU time).

Therefore, moving this computation to the GPU should yield a boost in performance, specifically in terms of computational speed. The GPU implementation was compared to the original ELM implementation in order to characterise and compare these performance gains. Recreating the network configuration used in the OPIUM paper by van Schaik and Tapson [159] and making use of the same datasets provided a means to place the results of these tests in context.

The tests made use of a GeForce GTX 650 GPU from NVIDIA, running in a PC with an Intel Core i7-3820 CPU running at 3.6 GHz and containing 64 GB of memory. This machine represents a high-end desktop workstation, making it a suitable device on which to run the CPU comparison tests.

Both networks received the full set of 60,000 training samples in the same random order. Both networks made use of the same sets of random hidden layer weights, with no additional pre-processing performed on the input data with the full resolution of $28 \times 28$ pixels used to form the 784 inputs for each training sample.

Figure 7 shows the training and testing times for classifying the MNIST dataset iteratively using the OPIUM method for the normal implementation and the GPU-optimised variant. The training time represents the total time taken to train the entire network, including all overheads and to mitigate any external effects surrounding the implementation of these algorithms; both methods used a common framework to load input data and process output results.

Both methods were written in Matlab with a focus on ensuring that the two implementations were as similar as possible. The non-GPU code is entirely vec-

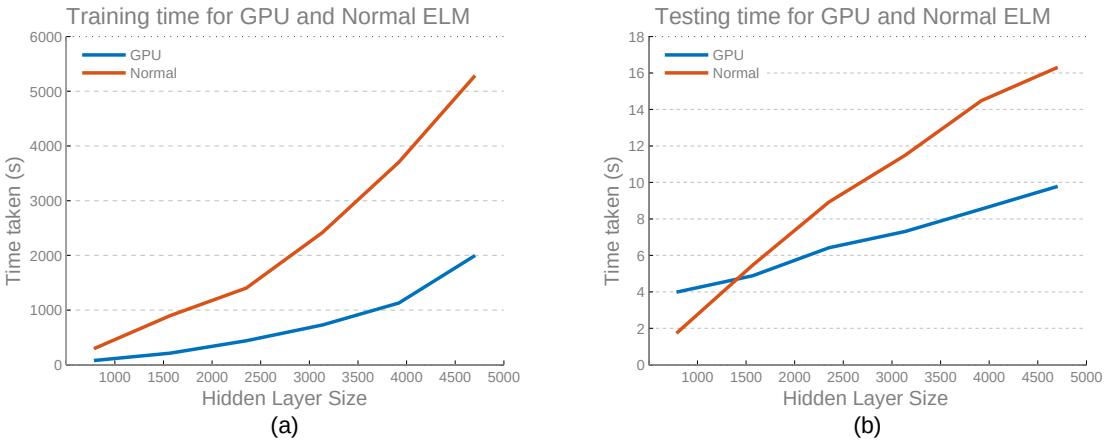



Figure 7: **Comparison of the Training and Testing Times of an ELM Classification of the MNIST digits with and without GPU Acceleration:** The clear speed improvement in training times resulting from the GPU optimisation is shown in (a), whereas (b) shows the more moderate improvements in testing time when using the GPU acceleration. The graph in (b) shows how the overhead from the GPU memory transfers negatively affect the testing times for lower numbers of hidden layer neurons.

torised, and therefore is able to make use of all the optimisations provided by Matlab, which also include multi-core optimisations. The methodology and design of the experiment was such as to minimise the difference between the two implementations wherever possible..

The results show that the GPU optimised algorithm outperforms the standard implementation in both training times and testing times. The difference in speed becomes more apparent as the number of hidden layer neurons increases and the size of the underlying correlation matrix accordingly. This is a direct result of the benefits of the parallelised cores in the GPU.

The testing does not make use of the correlation matrix at all, as recall requires no updates to the pseudo-inverse matrix. The testing can only make use of the GPU optimisation during the projection of the inputs to the hidden layer through the random input weights. This is not performed during training as it occupies memory that would further reduce the number of supported hidden layer neurons.

Implementing the project on the GPU is possible as the random input weights used by the ELM method are constant throughout all operations, and therefore can be loaded into the memory of the GPU prior to all recall operations.

The testing results in Figure 7b show that the GPU is slower than the conventional ELM method when the number of hidden layer neurons is small (typically fewer than 1400 neurons). This is due to the additional overhead of transferring

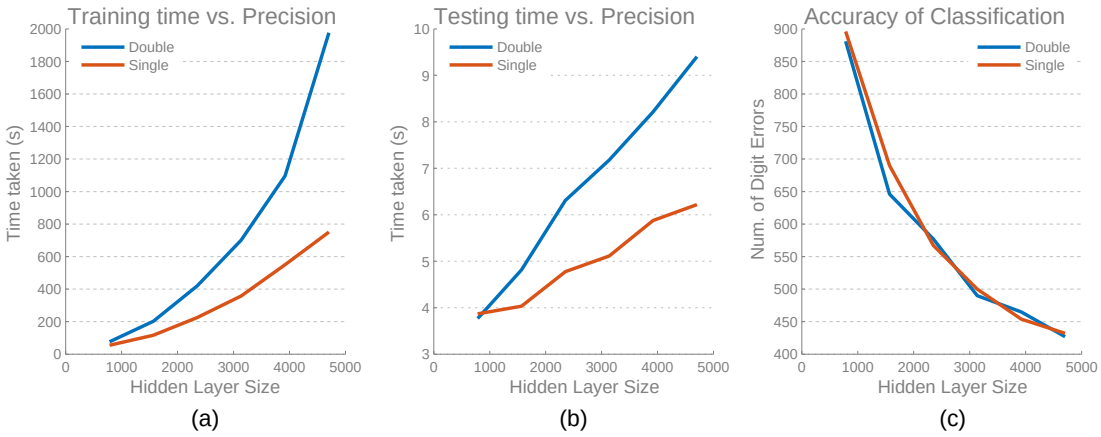



Figure 8: **Effect of Differing Levels of Floating Point Precision on Accuracy and Training Time For GPU Optimisation.** The training times (a) and testing times (b) for the same GPU optimisations running at *single* and *double* precision, showing that using *single* precision yields a significant speed advantage. The accuracy in terms of incorrect digits when classifying against the MNIST dataset is shown in (c), and demonstrates that there is no loss of accuracy associated with the lower float point precision.

data to and from the GPU memory. When the size of the hidden layer increases beyond this point, the significance of this overhead diminishes.

The amount of memory available on the GPU devices became the limiting factor in the testing of the above algorithm, in turn limiting the maximum number of hidden layer neurons. On the GPU used for these tests, this resulted in a maximum hidden layer size of 5550 hidden layer neurons when using the standard Matlab *double* precision. Later testing on larger GPUs with more memory allowed the size of this hidden layer to extend well past 12,000 hidden layer neurons.

## Effect of Floating Point Precision on Accuracy

One potential means of overcoming the memory limit imposed by the GPU is to reduce the precision of the calculations by using fewer bytes to store each numerical value. In effect, this is analogous to changing the underlying type of the matrix in Matlab from a *double* to a *single*. As it happens, GPU hardware is more adept at performing calculations on *single* data types than larger ones such as *doubles*. This results in a noticeable improvement in performance in addition to the lower memory footprint. Figure 8 shows the results of training an ELM classifier with varying hidden layer sizes for the same MNIST dataset used in the previous examples.

Each test made use of the same set of random weights for the *single* and *double* precision experiments, and trained using the same random order of the full set of training samples. Testing made use of the complete testing set, and the reported time represents the total time taken to train the network including all overhead. Figure 8c shows the error rate in terms of digits incorrectly classified.

The results in Figure 8c show that there is no significant difference in accuracy between using *single* versus *double* precision. However, there is a clear improvement in performance for both training and testing when using *single* precision. The OPIUM method approximates the pseudo-inverse solution for the ELM system, making the inverse correlation matrix the most susceptible point to a change in floating point precision. The effective error rate of the system is a practical means of measuring the effect of this change in precision, as it represents the overall performance of the classification system.

The change in precision allows the GPU implementation to support a larger number of hidden layer neurons. Given the same hardware as before, the same device was able to implement a fan-out of 8, requiring 6272 hidden layer neurons. This configuration achieved just 385 errors (96.16%).

Repeating these tests on a machine with a slower CPU but containing a far more powerful NVIDIA Tesla S2050 GPU, was able to support a fan-out of 11 (8624 hidden layer neurons) at *double* precision and produced 357 errors (96.45% accuracy). Dropping the floating point precision to *single* allowed for a network with a fan-out of 17 (13,328 neurons) and produced only 292 errors (97.08% accuracy).

# SKIM Optimisation with a GPU

Although the SKIM method can make use of any gradient-descent algorithm in order to determine the linear output weights, the methods used in this work make use of the OPIUM method as a means to perform optimisations in an iterative fashion. It therefore lends itself well to benefit from a GPU implementation as it relies on the same underlying pseudo-inverse update.

In fact, the nature of the SKIM algorithm makes heavy use of OPIUM as it performs an update on every time step of the input pattern. Therefore, even a short pattern requires multiple pseudo-inverse calculations and throughout all the training (inter- and intra-pattern), the associated inverse correlation matrix $\theta_k$ can remain in GPU memory.

In order to validate the performance improvements, tests were run using the same methodology as applied to the standard ELM approach outlined in Section 6.5. SKIM differs from OPIUM in that it operates on spatio-temporal patterns

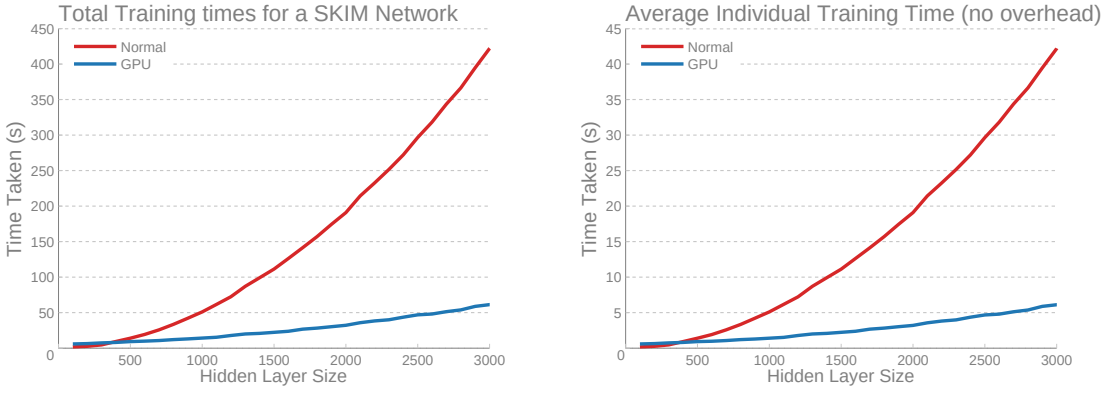



Figure 9: **Comparison between the GPU SKIM implementation and the Normal SKIM Implementations for varying hidden layer sizes.** The training time to train all ten trials is shown in (a) and shows a dramatic improvement for the GPU implementation. The average across time for all ten trials is shown in (b) and represents just the training time itself without any overhead. It is clear from both plots that the GPU implementation greatly outperforms the standard implementation.

as opposed to static inputs, and therefore cannot make use of the MNIST dataset for characterisation. In place of the MNIST data set, the experiments made use of the N-MNIST dataset from Section 4.3.

The testing made use of the full 60,000 training samples, and the same framework implemented both the normal SKIM algorithm and the GPU-optimised SKIM version. Only the internal OPIUM update method differed between the two SKIM implementations.

For each test, ten randomly selected digits were selected from the training samples and converted into a spatio-temporal pattern consisting of 360 time steps and 1156 channels. Both systems were trained independently of one another, with each test including the initialisation of all matrices and objects and the time taken to remove any matrices from memory (both on the CPU and the GPU).

Figure 9 shows the difference in training and testing times between the two methods for a variety of hidden layer sizes. Figure 9a shows the overall training time, which measures the total time taken to complete all ten trials, including all overheads such as initialising the dataset and all necessary variables. The time taken for each update to the SKIM network (one for each time step) was also recorded, and the average of those taken and displayed in Figure 9b. This measurement therefore captures the time taken to perform just the update step, excluding the overheads of transferring to and from the GPU.

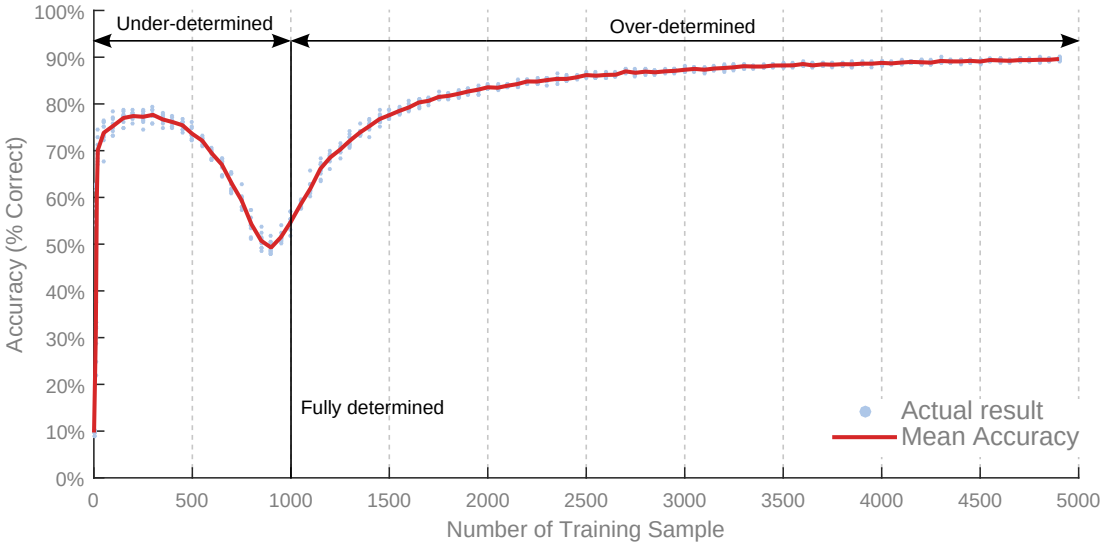



Figure 10: **Illustration of the under-determined and over-determined region of an LSHDI network of 1000 hidden layer neurons and the performance characteristic for an MNIST classifier trained on varying numbers of training samples.**

# Iterative Solutions to Under-Determined Problems

The OPIUM method provides a means of iteratively training an LSHDI network such as ELM through an iterative means of calculating and updating the Moore-Penrose pseudoinverse. It is this ability to iteratively train such a network that gives it the power to tackle problems of any number of training samples without placing an increasing demand on available memory on the computing device.

When an iteratively trained system is initialised, it has not received any training samples on which to train and the matrices need to be initialised to logical values. The work in this thesis has so far initialised the output weights $\beta = \mathbf{0}$ and the inverse correlation matrix $\theta_k = \mathbf{0}$.

If the network contains $L$ hidden layer neurons, this equates to solving a system with $L$ unknowns with the pseudoinverse. As a result, when iteratively training an LSHDI system from the start, the entire system is under-determined until the number of training samples $l$ is equal to $L$. As a result, the system of equations underpinning the network will have infinitely many solutions (as there will always be at least one free parameter).

Figure 10 illustrates the three regions in which an LSHDI network operates.

Once $l = L$, the network is fully determined, and the network will have an exact solution if one exists. Once $l > L$, the network becomes over-determined and the network shifts from an exact solution to a generalisation to minimise least-squares error. This is the usual configuration in which an ELM network operates.

In most classification tasks, the under-determined region is of little significance, as the number of training samples greatly exceeds the number of hidden layer neurons. MNIST, for example, contains 60,000 training samples, which would require the same number of hidden layer neurons for it to remain in the under-determined state after training.

The N-Caltech101 dataset, used extensively in this thesis, poses a different problem. Although it contains many thousands of images in total, when selecting subsets for training on the 101-way classification problem, only 1515 images are available for training. As networks in this thesis range in size from 100 hidden layer neurons to over 10,000 in size, the potential to operate in the under-determined region becomes a distinct possibility.

Figure 10 also shows the performance of a classifier consisting of 1000 hidden layer neurons and trained with varying numbers of training samples. The results shown were generated using the OPIUM method, but an ELM network calculating the pseudoinverse directly generates the same results. What is interesting and immediately apparent is that that the network performance reaches a local minimum around the point at which the network is fully determined. Before that peak, there is a local maximum accuracy achieved once the network has been trained on approximately 350 training samples.

This effect is also visible when using other datasets, such as the orientation features generated in Section 5.3.3 and is shown again in Figure 5.6. The same effect may also play a role in the performance on the N-Caltech101 dataset as visible in Figure 5.4, where the results for a network of 1000 hidden layer neurons outperforms those for a network of 2000 and 3000 neurons.

Interestingly, this effect disappears when the LSHDI network is initialised using random output weights instead of zero, as shown in Figure 11. In this case, the initial rise in accuracy disappears, and the values slowly climb instead. This may be a result of an incorrect error output during the initial training patterns, causing the iterative solution to deviate from the true pseudoinverse value until the system becomes fully determined.

This result highlights an important aspect of designing and tuning an LSHDI network, as having a larger number of hidden layer neurons is not always advantageous when dealing with a small training set. Understanding the nature of the results during the under-determined region can shed insight into the performance of vastly under-determined systems and can serve to explain some of the erratic results obtained.

It is therefore possible that the results achieved on the full 101-way classifica-

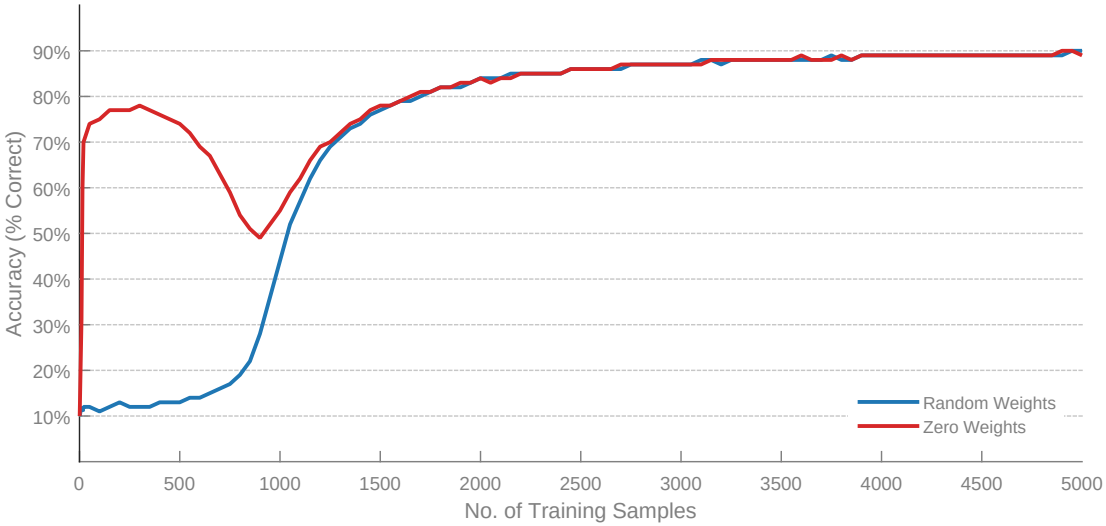



Figure 11: **Effect of initial output weights on classifier performance on the MNIST dataset with iterative training.** The graph shows the results of training an iteratively trained network with 1000 hidden layer neurons on varying number of training inputs from the MNIST dataset using both randomly initialised output weights and output weights initialised to zero. Note that these results show the mean accuracy achieved over 10 trials of each number of training samples.

tion on the N-Caltech101 datasets as reported in Section 4.5.6, and which made use of large hidden layer sizes, are operating in this under-determined region and therefore could potentially achieve much better performances with a larger number of training samples.

# Appendix C: Additional Tables and Figures

This appendix contains additional tables and figures that either supplement the work presented in the above chapters, or were too large to include within the chapters themselves.

Table 3: **ATIS Bias configuration values for the N-MNIST and N-Caltech101 recordings**. The following biases were chosen for the ATIS camera device when performing the conversion process described in Section 4.3.3. The biases were empirically chosen to provide a balanced number of ON and OFF events.

| Bias | Value |
|---|---|
| APSvrefL | 3050mV |
| APSvrefH | 3150mV |
| APSbiasOut | 750mV |
| APSbiasHyst | 620mV |
| CtrlbiasLP | 620mV |
| APSbiasTail | 700mV |
| CtrlbiasLBBuff | 950mV |
| TDbiasCas | 2000mV |
| CtrlbiasDelTD | 400mV |
| TDbiasDiffOff | 620mV |
| CtrlbiasSeqDelAPS | 320mV |
| TDbiasDiffOn | 780mV |
| CtrlbiasDelAPS | 350mV |
| TDbiasInv | 880mV |
| biasSendReqPdY | 850mV |
| TDbiasFo | 2950mV |
| biasSendReqPdX | 1150mV |
| TDbiasDiff | 700mV |
| CtrlbiasGB | 1050mV |
| TDbiasBulk | 2680mV |
| TDbiasReqPuY | 810mV |
| TDbiasRefr | 2900mV |
| TDbiasReqPuX | 1240mV |
| TDbiasPR | 3150mV |
| APSbiasReqPuY | 1100mV |
| APSbiasReqPuX | 820mV |

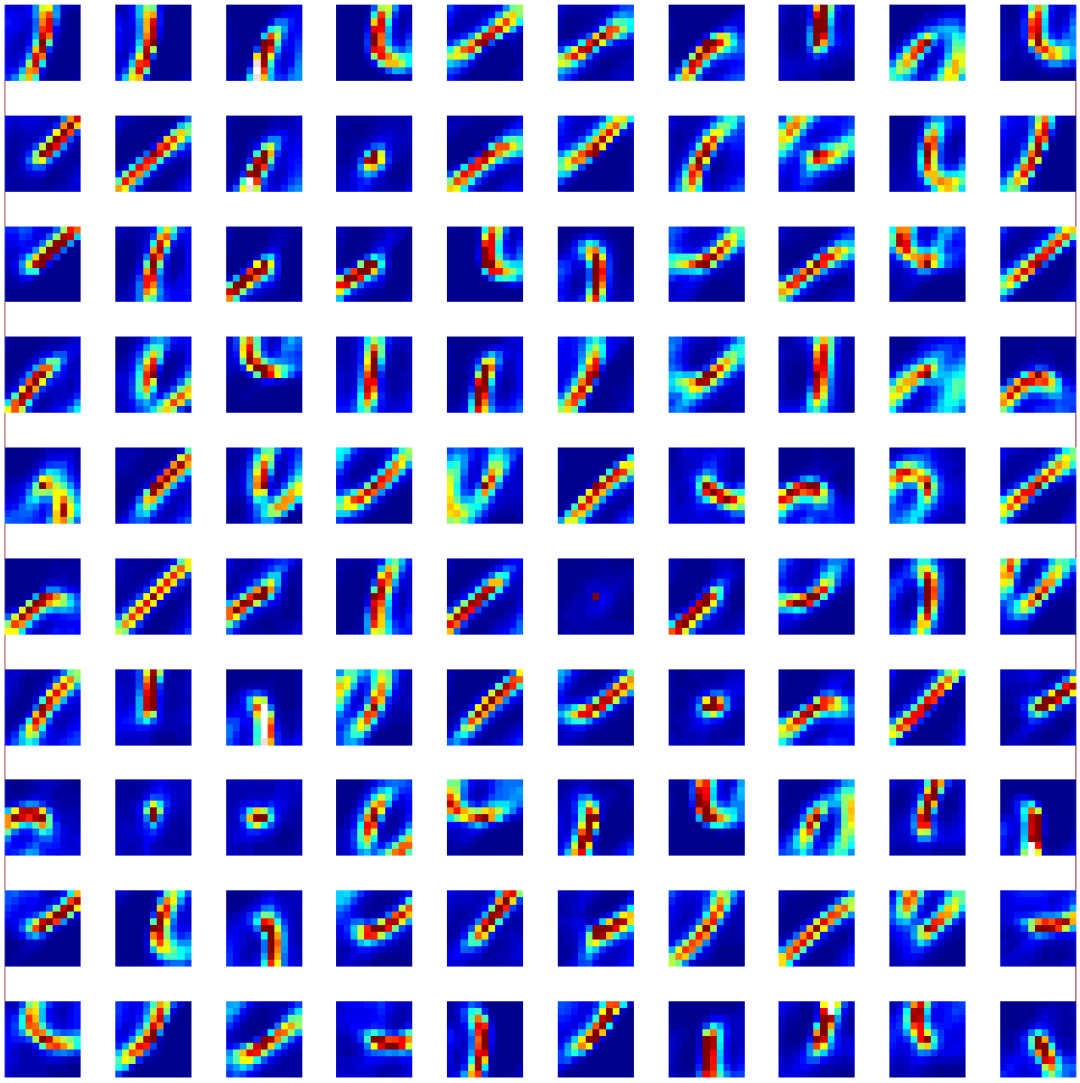


Figure 12: **The 100 ON Clusters for the N-MNIST dataset for** $11 \times 11$ **pixel features generated by the Adaptive Threshold Clustering**. The resulting 100 feature clusters after training on all 60,000 training digits in the N-MNIST dataset. The order conveys no particular significance.

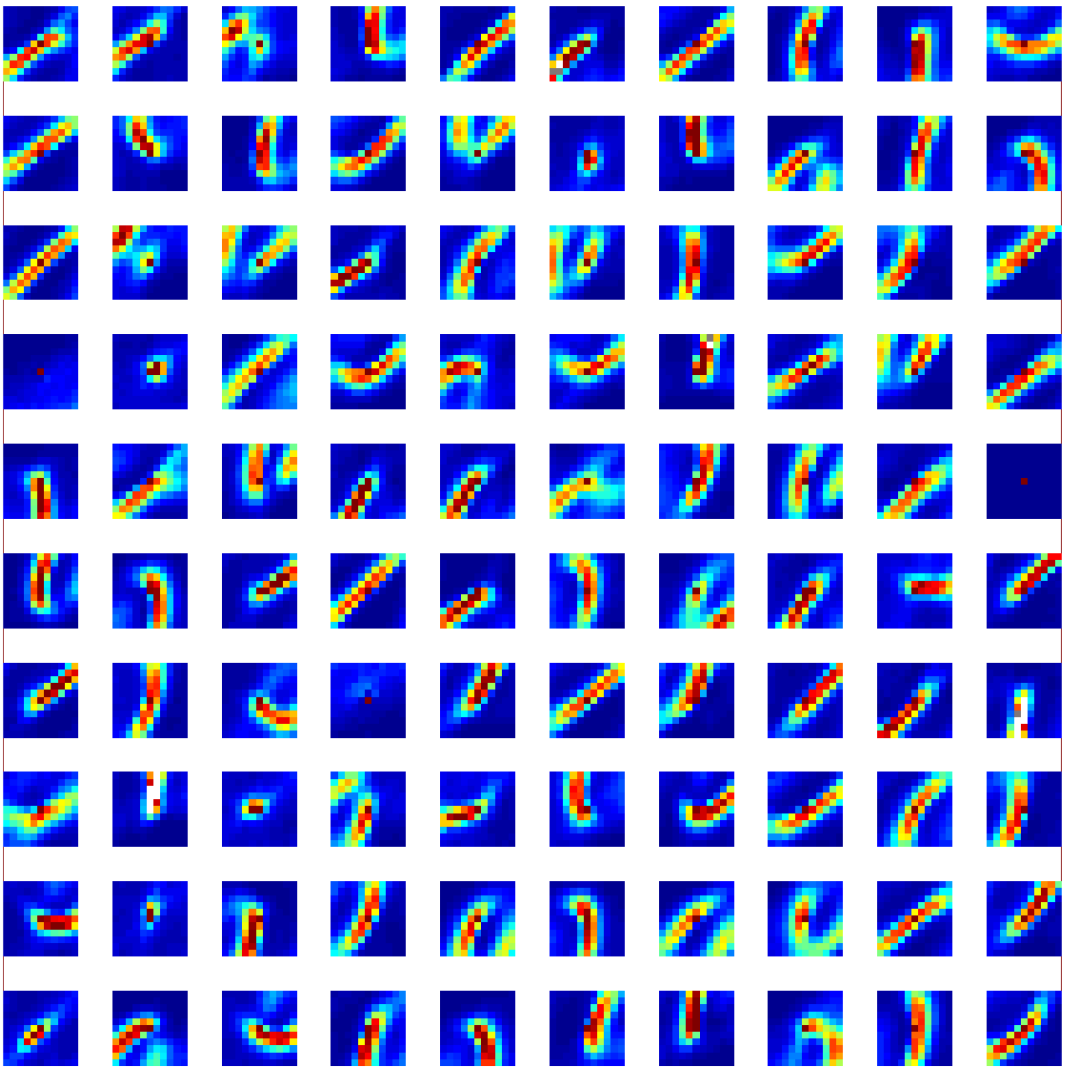


Figure 13: **The 100 OFF Clusters for the N-MNIST dataset for** $11 \times 11$ **pixel features generated by the Adaptive Threshold Clustering**. The resulting 100 feature clusters after training on all 60,000 training digits in the N-MNIST dataset. The order conveys no particular significance.

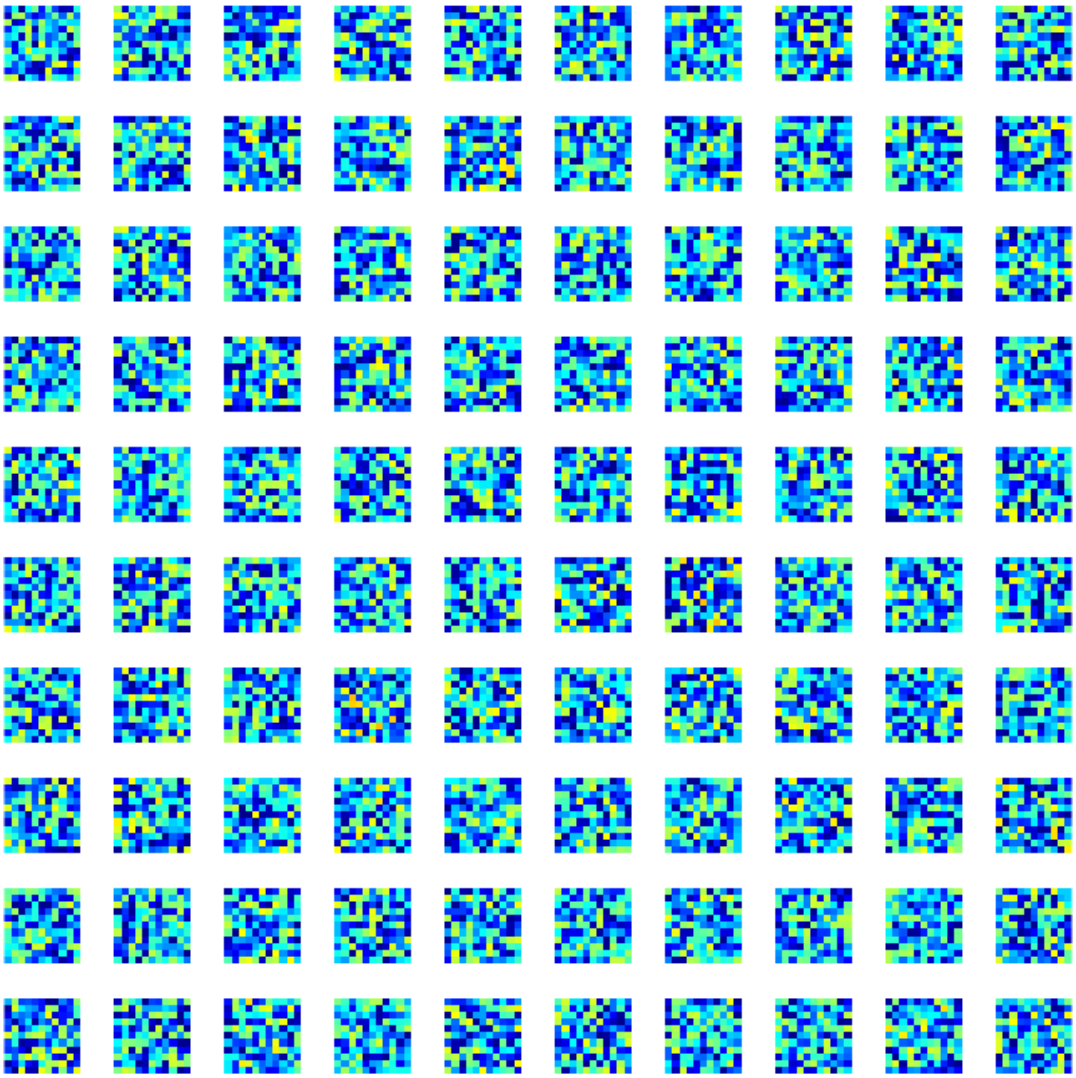


Figure 14: **The 100 random ON Clusters for the N-MNIST dataset for $11 \times 11$ pixel clusters**. These feature clusters are normalised, and the adaptive thresholding technique was used to determine the individual thresholds for each cluster.

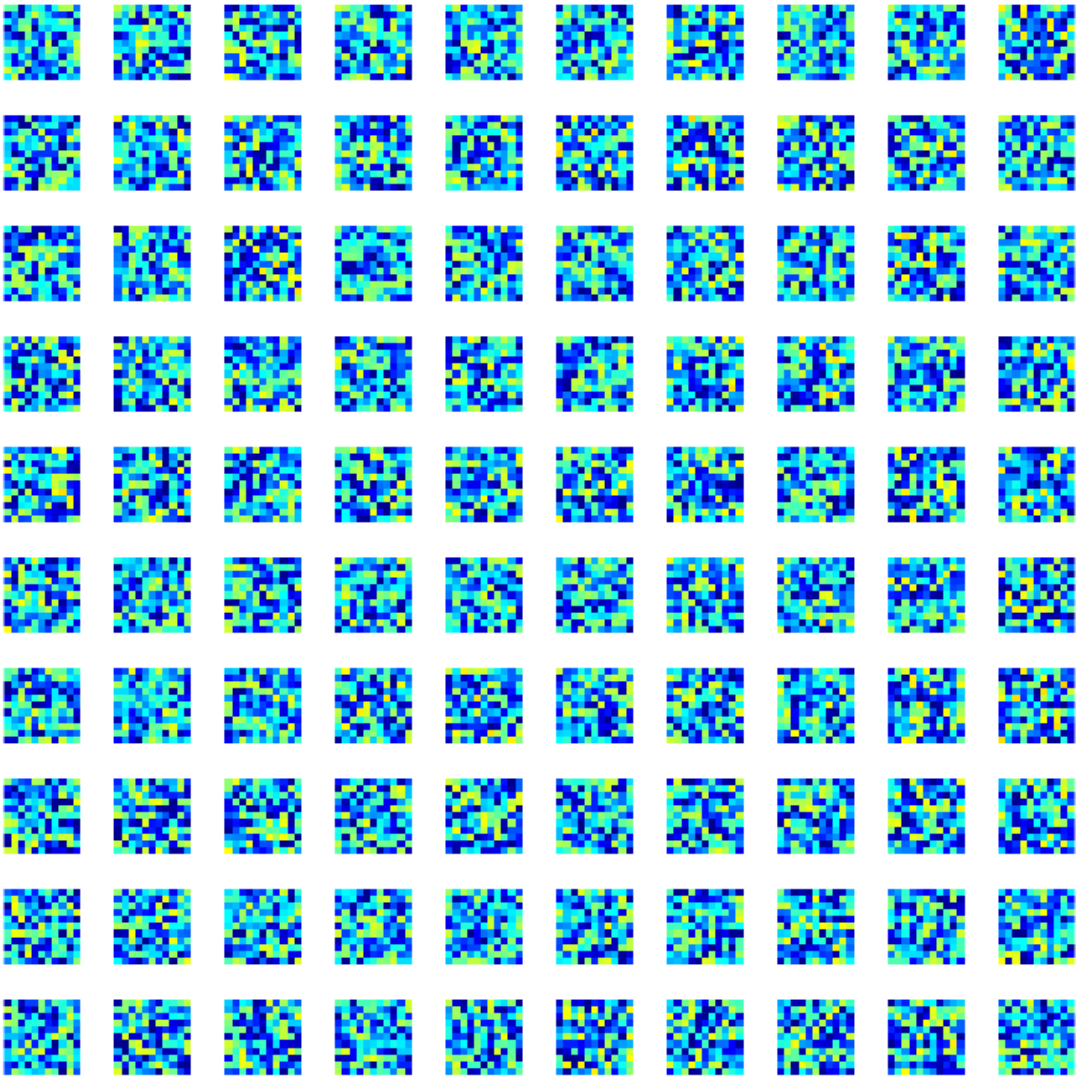


Figure 15: **The 100 random OFF Clusters for the N-MNIST dataset for $11 \times 11$ pixel clusters**. These feature clusters are normalised, and the adaptive thresholding technique was used to determine the individual thresholds for each cluster.

Table 4: **Spatial and temporal downsampling accuracy results for the 5-way classification problem on the N-Caltech101 dataset using SKIM.** Each value represents the percentage of correctly classified sequences in the 5-way classification task. Each network trained on a random subset of 254 sequences drawn in equal proportions from the five classes, and tested with the remaining objects. Each network was configured with 1000 hidden layer neurons.

| | Temporal Downsampling (millisecond resolution) | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1ms | 2ms | 3ms | 4ms | 5ms | 6ms | 7ms | 8ms | 9ms | 10ms | 11ms | 12ms | 13ms | 14ms | 15ms | 16ms | 17ms | 18ms | 19ms | 20ms |
| 223 × 173 | 56.45% | 58.47% | 59.27% | 62.90% | 62.50% | 70.97% | 66.94% | 72.18% | 78.23% | 74.19% | 78.23% | 70.97% | 79.03% | 73.79% | 81.86% | 81.45% | 79.44% | 82.66% | 75.00% | 81.05% |
| 111 × 86 | 53.63% | 54.44% | 55.65% | 61.29% | 65.32% | 62.10% | 63.31% | 65.32% | 68.15% | 69.36% | 71.77% | 73.39% | 78.23% | 71.77% | 72.18% | 71.37% | 78.23% | 75.81% | 79.84% | 75.40% |
| 74 × 57 | 49.19% | 57.66% | 58.07% | 58.07% | 63.31% | 65.32% | 68.55% | 64.11% | 68.95% | 70.16% | 72.18% | 71.37% | 73.79% | 69.36% | 74.19% | 77.82% | 77.02% | 72.18% | 79.03% | 77.42% |
| 55 × 43 | 48.79% | 54.44% | 55.65% | 58.87% | 60.48% | 66.94% | 64.92% | 69.36% | 71.37% | 74.60% | 73.39% | 75.40% | 75.00% | 73.79% | 74.60% | 73.39% | 78.23% | 73.39% | 68.95% | 78.63% |
| 44 × 34 | 49.19% | 53.63% | 51.61% | 60.08% | 60.08% | 61.29% | 66.53% | 62.10% | 62.10% | 69.36% | 72.58% | 75.81% | 71.37% | 65.73% | 72.18% | 73.79% | 79.03% | 74.60% | 77.82% | 75.81% |
| 37 × 28 | 52.82% | 45.97% | 54.03% | 60.48% | 57.66% | 57.26% | 61.69% | 62.50% | 64.52% | 64.11% | 72.98% | 69.36% | 72.98% | 72.98% | 69.76% | 76.21% | 76.61% | 74.19% | 79.84% | 76.21% |
| 31 × 24 | 45.97% | 45.57% | 47.58% | 56.05% | 53.23% | 54.84% | 61.29% | 64.92% | 60.89% | 61.69% | 64.92% | 65.32% | 70.57% | 69.76% | 67.74% | 68.15% | 72.58% | 72.58% | 76.21% | 75.81% |
| 27 × 21 | 41.53% | 40.32% | 49.19% | 46.77% | 44.76% | 55.24% | 55.65% | 57.66% | 60.08% | 66.13% | 64.92% | 61.69% | 68.95% | 74.19% | 66.94% | 69.36% | 70.57% | 65.73% | 66.53% | 70.97% |
| 24 × 19 | 43.15% | 45.16% | 41.53% | 50.00% | 53.23% | 58.47% | 54.84% | 64.52% | 59.68% | 56.05% | 64.92% | 70.16% | 76.61% | 59.27% | 62.90% | 73.39% | 69.76% | 65.73% | 66.94% | 68.95% |
| 22 × 17 | 46.77% | 53.63% | 50.81% | 58.07% | 55.65% | 65.32% | 56.45% | 67.34% | 60.89% | 63.31% | 63.31% | 60.48% | 66.94% | 67.34% | 70.57% | 68.15% | 72.18% | 68.55% | 68.55% | 69.36% |
| 20 × 15 | 39.92% | 49.19% | 43.15% | 53.23% | 56.86% | 54.03% | 54.84% | 59.68% | 59.68% | 66.53% | 72.98% | 57.26% | 70.57% | 64.92% | 70.16% | 69.36% | 67.74% | 71.37% | 72.18% | 65.32% |
| 18 × 14 | 46.37% | 44.76% | 51.21% | 47.18% | 54.03% | 51.61% | 58.47% | 62.10% | 56.05% | 69.76% | 67.74% | 67.74% | 65.73% | 66.53% | 71.77% | 64.52% | 69.36% | 65.73% | 68.15% | 67.34% |

Table 5: **Spatial and temporal downsampling accuracies for the full 101-way classification problem using SKIM.** Each result represents the percentage of sequences correctly identified for the 101-way classification task using 1000 hidden layer neurons. A randomly selected 15 sequences from each category were used for training, and another randomly drawn 15 used for testing.

| | Temporal Downsampling | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1ms | 2ms | 3ms | 4ms | 5ms | 6ms | 7ms | 8ms | 9ms | 10ms | 11ms | 12ms | 13ms | 14ms | 15ms | 16ms | 17ms | 18ms | 19ms | 20ms |
| 223 × 173 | 6.37% | 7.55% | 9.10% | 9.78% | 9.59% | 10.27% | 12.44% | 11.57% | 12.25% | 14.73% | 14.48% | 13.92% | 15.96% | 15.96% | 16.03% | 15.66% | 13.86% | 15.96% | 15.47% | 15.72% |
| 111 × 86 | 6.06% | 7.18% | 8.54% | 8.66% | 9.47% | 10.46% | 11.14% | 12.07% | 12.75% | 12.44% | 12.44% | 12.13% | 14.79% | 13.74% | 13.80% | 13.68% | 14.73% | 13.74% | 14.97% | 14.97% |
| 74 × 57 | 7.05% | 6.68% | 8.35% | 9.16% | 9.53% | 9.22% | 11.08% | 11.82% | 11.32% | 12.75% | 11.45% | 13.24% | 14.79% | 12.13% | 14.85% | 14.05% | 15.53% | 13.99% | 14.85% | 15.41% |
| 55 × 43 | 5.94% | 7.74% | 7.49% | 8.23% | 8.85% | 8.54% | 10.52% | 10.52% | 12.38% | 11.94% | 12.93% | 12.19% | 13.86% | 13.18% | 12.75% | 14.60% | 14.42% | 14.91% | 15.04% | 15.66% |
| 44 × 34 | 5.75% | 6.19% | 7.49% | 8.04% | 7.98% | 9.22% | 10.40% | 10.46% | 11.45% | 11.45% | 12.69% | 12.19% | 12.38% | 12.13% | 11.57% | 13.43% | 14.79% | 14.05% | 13.68% | 13.18% |
| 37 × 28 | 5.57% | 6.62% | 6.13% | 8.17% | 6.68% | 7.98% | 9.84% | 9.16% | 10.03% | 13.12% | 10.95% | 12.07% | 10.95% | 12.50% | 12.31% | 12.56% | 12.75% | 13.37% | 12.44% | 13.80% |
| 31 × 24 | 5.14% | 5.26% | 6.99% | 6.93% | 6.99% | 8.17% | 8.23% | 9.03% | 9.22% | 10.89% | 10.46% | 11.14% | 11.14% | 11.63% | 12.69% | 12.38% | 13.12% | 13.06% | 13.00% | 12.19% |
| 27 × 21 | 4.52% | 5.20% | 6.31% | 6.74% | 7.49% | 7.12% | 9.22% | 8.85% | 9.84% | 9.59% | 9.53% | 9.03% | 10.58% | 11.32% | 11.63% | 12.87% | 11.70% | 11.51% | 12.25% | 12.19% |
| 24 × 19 | 6.93% | 5.07% | 6.06% | 7.12% | 7.05% | 6.37% | 9.34% | 9.65% | 8.73% | 10.33% | 9.84% | 10.64% | 11.20% | 10.58% | 10.77% | 12.13% | 13.68% | 11.08% | 12.50% | 11.88% |
| 22 × 17 | 5.69% | 5.32% | 6.81% | 6.81% | 7.18% | 8.29% | 7.92% | 9.22% | 10.52% | 9.65% | 10.09% | 10.58% | 12.07% | 10.77% | 12.31% | 12.75% | 10.58% | 12.81% | 12.62% | 12.25% |
| 20 × 15 | 5.69% | 5.01% | 5.57% | 7.49% | 7.98% | 7.80% | 9.90% | 10.15% | 9.34% | 9.53% | 10.40% | 9.78% | 11.01% | 10.77% | 11.32% | 11.14% | 11.32% | 13.12% | 11.39% | 13.86% |
| 18 × 14 | 4.46% | 4.70% | 6.93% | 5.69% | 7.67% | 7.43% | 10.09% | 9.59% | 8.97% | 9.72% | 10.77% | 11.63% | 12.13% | 11.88% | 10.77% | 12.62% | 12.50% | 12.19% | 12.01% | 12.69% |