Development of a diagnosis application based on artificial neural networks for the detection of brain tumors

Brad Niepceron

Thèse de Doctorat

Mention Informatique

présentée à ***l'Ecole Doctorale Sciences, Technologie, Santé***

de l'Université de Picardie Jules Verne

par

Brad Niepceron

pour obtenir le grade de Docteur de l'Université de Picardie Jules Verne

Développement d'une application d'aide au diagnostic basée sur les réseaux de neurones artificiels pour la détection de tumeurs cérébrales.

Soutenue le 19 Novembre 2021, après avis des rapporteurs, devant le jury d'examen :

M. P. Lorenz, Professeur, Université de Haute Alsace,	**Président**
M. M. Chadli, Professeur, Université Paris-Saclay,	**Rapporteur**
M. L. Navarro, Maître de Recherche, Mines Saint-Étienne,	**Rapporteur**
M. J. Gaber, Maître de Conférences HDR, UTBM, Belfort,	**Examinateur**
M. A. Nait-Sidi-Moh, Professeur, Université de Saint-Étienne,	**Directeur de thèse**
M. F. Grassia, Maître de Conférences, Université d'Amiens,	**Co-directeur de thèse**
Mme. S. Chebi Gamoura, Maître de Conférences, Université de Strasbourg,	**Invitée**

Acknowledgement

I am pleased to express my recognition towards the people who supported me during the last three years. I would first like to express my deepest gratitude to Dr. Filippo Grassia. Working under his supervision has been a great privilege and a very enriching experience. His guidance, optimism and assistance at every stage of this research project have been crucial for its success. I could not thank him enough for all the support and trust he gave me. I would also like to thank as warmly, my thesis advisor Pr. Ahmed Nait-Sidi-Moh, for supporting and trusting me in the making of this work. His benevolence, teaching and willingness to share his experience was an essential source of motivation and an additional important assistance for the success of my work.

I would like to express my gratitude to Dr. Laurent Delahoche and Dr. Jaafar Gaber who both took their time to be part of my thesis monitoring committee and shared insights and valuable comments on my researches. A warm thank you goes to all jury members of my thesis defense, Pr. Pascal Lorenz, Pr. Mohammed Chadli, Dr. Laurent Navarro, Dr. Jafaar Gaber and Dr. Samia Chebi Gamoura. I thank them all for the time they took to take part in this project and for all the discussions and very interesting comments they had about this work, it was a pleasure to exchange with all of them. Thank you to Région Hauts-de-France, Agglomération du Saint-Quentinois and the city of Saint-Quentin for making this project possible by supporting and partially funding it.

Many thanks to Dr. Harold Trannois for letting me discover research during my master degree. Working with him was very important for the choice I made to follow the path that took me to finally write this manuscript. A special thank goes to my friend and former teacher, Florence Métivier, who always supported me and helped me to develop my interest and curiosity for a wide range of domains. Her support and guidance have been very

important in determining the path I felt destined to take. A warm thank you to my friend and former educator, Frédéric Feyt, for teaching me the valuable lesson that sometimes, ”machines and men aren’t that different”. I feel lucky to have crossed both of their paths and can only give them my entire gratitude for everything they have taught me.

I also would like to sincerely thank my dearest friends, Lucas, Kenan, Vincent and Louise. Every second spent with them has been a blessing and an additional crucial help in the making of this work. I am grateful for the impact they had on my life and thank them dearly for the kindness and support they brought me. Thank you to my girlfriend, Lisa, for supporting me from the beginning to the end of my thesis. She has been a great source of motivation and inspiration for the work I produced to write this manuscript. Her positivity, care and trust really helped me to go through these last three years with the greatest optimism.

Last but not least, I want to thank my family, in particular my mother, together with my siblings Micke, Selena and Laurena for being there during each step of my life. Their love and recognition have shaped me and guided me towards the making of this work. Thank you to Jacques, Marie-Claire and their children Laurie, Jimmy and Valentin for making me part of their family and giving me all the support I needed during this thesis.

To all of them and the ones who contributed to this journey, truthfully, thank you.

Contents

List of Figures

List of Tables

Résumé

”Développement d’une application d’aide au diagnostic basée sur les réseaux de neurones artificiels pour la détection de tumeurs cérébrales.”

Au cours de la dernière décennie, l’étude de systèmes de diagnostics de tumeurs cérébrales a attiré une attention particulière compte tenu de la croissance rapide de l’apprentissage profond et du développement de réseaux de neurones artificiels efficaces. Dans le domaine clinique, les algorithmes basés sur l’apprentissage profond sont utilisés pour résoudre des tâches visuelles, telles que la détection ou la segmentation de tissus malsains. Ces méthodes ont notamment prouvé leur efficacité dans le diagnostic de tumeurs agressives telles que les gliomes de haut grade. Néanmoins, contraints par leur important besoin en ressources de calcul, ces modèles ne peuvent être réalistiquement déployés à grande échelle. En effet, leurs architectures devenant plus profondes avec l’amélioration de leurs performances, leur utilisation entraîne des coûts matériels et énergétiques importants qui ne correspondent pas aux exigences du domaine médical.

Objectifs

L’objectif de ce travail de recherche consiste alors en l’étude de nouveaux outils visant à répondre aux pré-requis nécessaires au déploiement d’application d’aide au diagnostic de tumeurs cérébrales, plus spécifiquement des gliomes, basée sur les réseaux de neurones artificiels. Cette étude passe par un besoin d’optimisation ou de remplacement des méthodes déjà utilisées dans l’état de l’art par des solutions moins dépendantes à la disponibilité

de grandes ressources de calcul. Pour répondre à ces problématiques, la compression de réseaux de neurones convolutifs pour la création d'application de segmentation de tumeurs cérébrales sur système embarqué est envisagée.

De plus, bien que de nombreux débats soient apparus sur l'efficacité des modèles d'apprentissage profond, des solutions basées sur les réseaux de neurones impulsionnels doivent encore être examinées afin de construire des methodes d'analyse plus rapides et rentables. L'ordre des contributions exposées dans ce manuscrit permet alors de mettre en avant une transition graduelle entre la deuxième et la troisième génération de réseaux de neurones artificiels.

Par conséquent, l'étude présentée dans ce travail se concentre premièrement sur l'adaptation des réseaux de neurones artificiels à des systèmes possédant des ressources de calculs limitées par le biais de méthodes de compression. Puis, des modèles neuronaux pour l'analyse d'images médicales sont étudiés afin de répondre aux problématiques de coûts posées par l'apprentissage profond. Enfin, une nouvelle méthode de développement de systèmes de diagnostic de tumeurs cérébrales basée sur des modèles de neurones biologiques est proposée.

Chapitre 2

Puisque le développement de systèmes modernes et automatiques de diagnostic de tumeurs nécessite la compréhension de disciplines différentes telles que l'intelligence artificielle, l'imagerie médicale ou le génie neuromorphique, le premier chapitre de ce travail de thèse consiste en une introduction à ces sujets afin de bien reconnaître les défis et enjeux exposés par cette thèse. Des notions importantes concernant le fonctionnement des réseaux de neurones appliqués à la vision artificielles sont premièrement introduites. Pour mieux comprendre le type de donnée utilisé par les méthodes d'analyse d'images présentées dans ce manuscrit, des informations concernant l'acquisition d'images médicales et plus spécifiquement d'images cérébrales sont présentées. Ensuite, une introduction aux neurosciences computationelles est faite afin de comprendre les enjeux que ces outils présentes pour la création de nouveaux systèmes d'aide à la décision basés sur des règles biologiques. Ce chapitre se termine enfin par une introduction à la perception et le fonctionnement des différents acteurs biologiques qui permettent au cerveau humain de reconnaître des formes, des couleurs ou des objects tout en présentant des modèles de filtres numériques permettant de les approximer.

Chapitre 3

Pour qu'un système de diagnostic de tumeurs cérébrales assisté par ordinateur puisse être déployé et utilisé dans le milieu médical, celui-ci doit répondre aux attentes des opérateurs en radiologie mais aussi aux exigences, principalement pécuniaire, du milieu hospitalier. Le chapitre deux présente la première contribution de ce travail de thèse et s'intéresse directement à la problématique de coût engendrée par l'entraînement de réseau de neurones artificiels. Pour cela, l'étude d'une plateforme embarquée est menée et met en avant l'importance de la migration des algorithmes d'apprentissage profond vers des systèmes à ressources limitées. Les caractéristiques de ce système sont alors décrites en détails pour mieux comprendre l'opportunité que présente son usage. Pour que la migration de ces algorithmes ait alors lieu, leur compression paraît cruciale.

Bien que les réseaux de neurones convolutifs soient considérés comme les modèles les plus performants en vision artificielle, ils ne répondent souvent pas aux exigences d'environnements aux ressources limitées tel que celui du médical. En effet, les opérations de convolutions sur lesquelles ils basent leurs performances sont coûteuses en ressource de calcul, en particulier lorsqu'elles sont effectuées sur des données possédant plusieurs dimensions. De plus, suite à la ruée vers de meilleurs scores de précision, ces modèles semblent devenir de plus en plus profonds ce qui augmente davantage le nombre d'opérations qu'ils effectuent. Par conséquent, afin d'être bénéfique à un plus large éventail de domaines, concevoir ce type de réseau doit permettre une réduction des coûts de calcul tout en gardant les performances des modèles aussi bonnes que possible.

Nous étudions donc la compression de l'architecture U-Net (Fig. 1), un réseau de neurones convolutifs construit pour la segmentation d'imagerie médicale afin de concevoir un système d'aide au diagnostic qui répond aux exigences déjà citées. Plusieurs méthodes sont proposées afin de réduire les paramètres du réseaux et de le rendre déployable sur système embarqué.

Chapitre 4

Puisque la détection d'anomalies telles que des lésions ou des tumeurs peut être considérée comme un problème de reconnaissance d'objects, de nombreuses architectures de réseaux de neurones convolutifs différentes ont été proposées pour effectuer la segmentation ou la classification d'imagerie à résonance magnétique (IRM). Ces méthodes ne peuvent néanmoins pas être viables en l'absence d'une grande quantité de données et de ressources de calcul suffisantes. Avec l'indisponibilité des données induite par l'éthique et

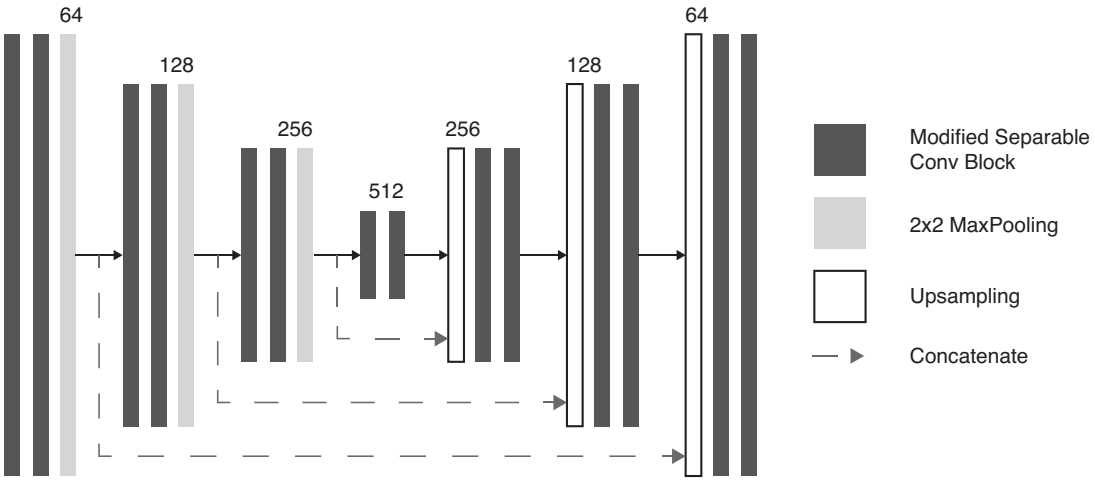

Figure 1: *Compression de l'architecture U-Net pour la segmentation d'IRM.*

la confidentialité dans le domaine médical ainsi que la haute dimensionnalité inhérente aux images médicales qui entraînent de lourdes charges de calcul, il paraît difficile de continuer à baser les systèmes de reconnaissance de lésions sur des réseaux de neurones convolutifs. De plus, ces réseaux sont pour la plupart entraînés de manière supervisée, ce qui induit la construction de jeux de données labellisés en amont, nécessitant des moyens humains qui entraîneraient des coûts supplémentaires pour les hôpitaux.

Le manque d'explicabilité des réseaux de neurones convolutifs entraîne également une pénalité lorsqu'il s'agit de les déployer pour le milieu médical à cause du manque de compréhension des décisions prises par le système. Ce chapitre marque alors le début de la transition vers des réseaux de neurones impulsionnels pour construire un système de reconnaissance de tumeurs cérébrales. Il s'intéresse à l'utilisation de réseaux de neurones à pulsions couplées (Fig. 2) pour effectuer la segmentation et la détection de tumeurs cérébrales. Ici, nous présentons une revue de plusieurs modèles de réseaux de neurones à pulsions couplées et étudions leur capacité à effectuer les tâches de segmentation et de détection de tumeurs ainsi qu'à être utilisé pour fusioner des séquences d'IRM. Pour cela un algorithme basé sur la théorie des signatures d'images produites par ce type de réseau est proposé (Fig. 3). Cet algorithme se base sur une segmentation rapide des images et permet la création d'une application de détection de tumeurs rapide et efficace.

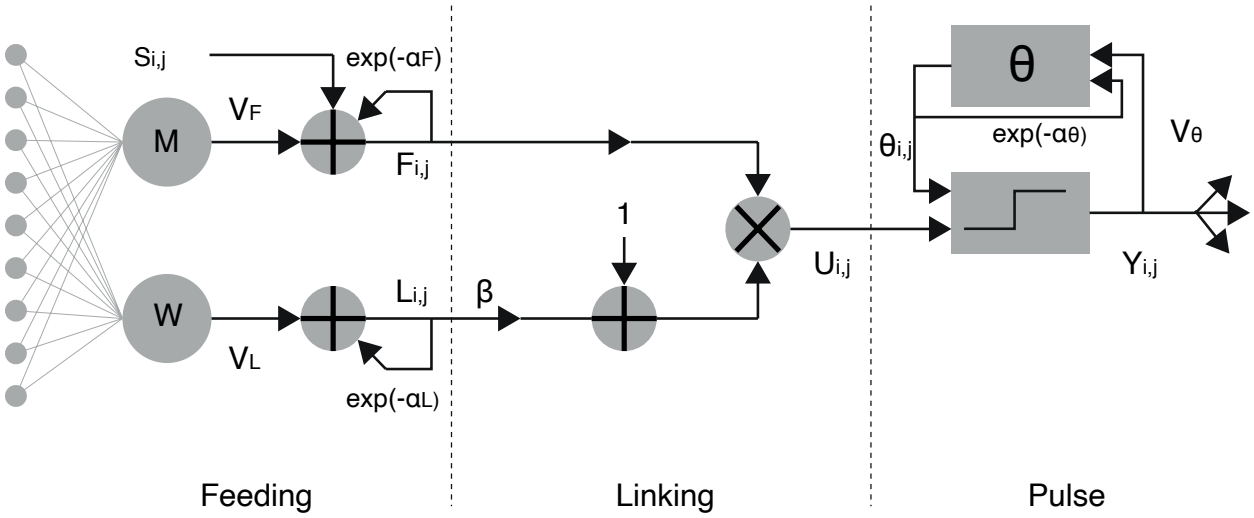

Figure 2: *Un réseau de neurones à pulsions couplées.*

Chapitre 5

Afin d'étendre les travaux précédents ce chapitre et aller plus loin pour prouver l'efficacité du calcul neuronal pour l'analyse d'images cérébrales, nous proposons l'étude d'un réseaux de neurones impulsionnels convolutifs ne possédant qu'une seule couche entraînable pour résoudre la classification de différents types de tumeurs cérébrales. Le type d'encodage utilisé par ce réseau pour représenter la donnée sur le domaine temporelle se base sur des filtres numériques inspirés de modèles de la perception humaine (Fig. 4). Pour evaluer la performance du modèle, deux tâches de classification possédant un niveau de complexité différent sont proposées. Puisque la taille du modèle est la plus basse possible, le but de ce chapitre n'est pas de proposer une méthode qui surpasserait les performances d'algorithmes d'apprentissage profond mais plutôt de poser les bases du développement des reseaux de neurones impulsionnels pour l'étude d'images médicales et prouver les opportunités qu'ils offrent pour la construction de systèmes de diagnostic légers et rentables.

Conclusion

Dans ce travail de thèse, nous avons abordé la modification et le remplacement des réseaux de neurones convolutifs, afin de rendre les systèmes de diagnostic de tumeurs cérébrales, basés sur les réseaux de neurones artificiels, déployables et utilisables dans le milieu médical.

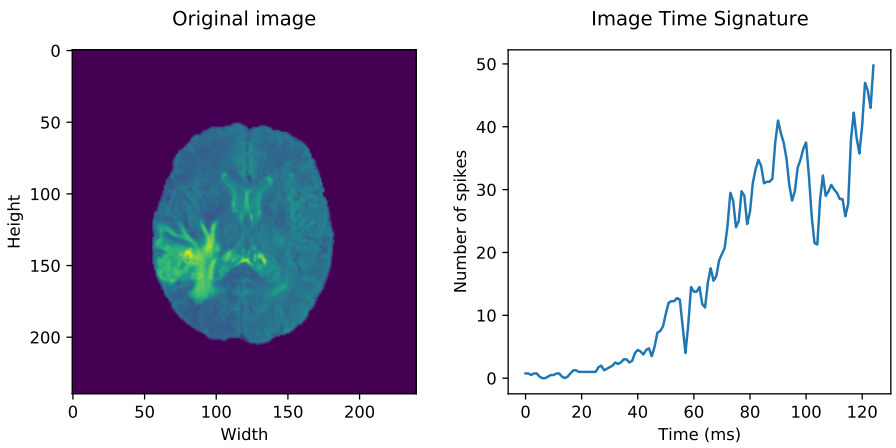

Figure 3: *Extraction de la signature d'une IRM par un réseau de neurones à pulsions couplées.*

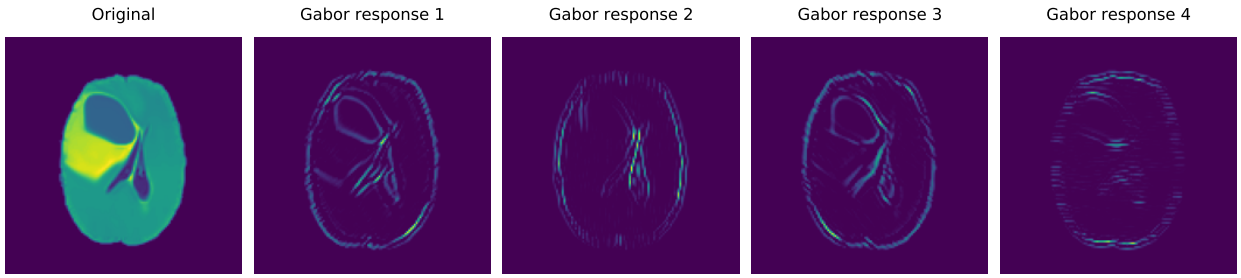

Figure 4: *Réponse de différents filtres de Gabor sur une couche d'IRM.*

Nous avons proposé différentes méthodes afin d'effectuer les trois tâches de diagnostic visuel de tumeurs cérébrales, la classification, la segmentation et la détection. Nous avons exploité les opportunités qu'offrent les modèles de neurones biologiques pour développer des systèmes d'analyse d'images médicales. Sur ce constat, les objectifs cités ont tous été atteints. Néanmoins, le travail proposé dans cette thèse laisse place à certaines améliorations ou opportunités concernant l'entraînement de réseaux de neurones impulsionnels pour la reconnaissance de tumeurs cérébrales.

Mots clés

Réseaux de neurones artificiels, vision artificielles, analyse de tumeur cérébrales, imagerie médicale, neurosciences computationelles, réseaux de neurones impulsionnels.

Chapter 1

Introduction

The malignant, high mortality and shape variety attributes of brain tumors make them one of the most difficult to detect and harmful type of cancer. It is diagnosed by the manual analysis of Magnetic Resonance Imaging (MRI) over a certain period of time necessary for the study of the tumor's evolution. In some severe cases, decreasing this analysis period is crucial for the patient's survival. This goal can be achieved by the development of fully automatic diagnosis systems to support radiologists in achieving the tumor detection task. However, the development of accurate solutions that could outperform the human expertise can be challenging. Nowadays, the progress done in Deep Learning [2], Machine Learning and the countless successful researches on Artificial Neural Networks (ANN) are leading to new horizons in the matter of solving and automating these complex tasks. In fact, computer vision is one of the major topic addressed by ANN, their ability to extract meaningful features in large visual datasets made them particularly popular. Multiple studies on deep learning based methods for MRI analysis have thus proven that ANN can outperform the human brain in segmenting an entire brain tumor. This makes them suitable for the development of end-to-end disease diagnosis applications.

However some of the major drawbacks of this kind of methods are the high cost, the lack of portability and the time needed to train and tune them to a particular task. They induce an important power consumption preventing the large scale deployment of most deep learning based diagnosis applications. Moreover, their applicability in production remains tedious as hospitals and healthcare providers already face important daily cost to treat their patients. In fact, these complex algorithms run on powerful dedicated hardware that can require the consumption of thousands of Watts while the human brain only works

consuming 20 Watts [3] for the the same task. This difference motivates the need to rethink, modify or replace ANN architectures in order to create solutions ready for the clinical field. Another drawback implied for visual diagnosis tasks is that most ANN models need to be supervised and thus require data that have been previously manually labelled by experts. This labelling process can be monotonous and can cost a lot of human and financial resources.

Lately, the issues implied by ANN in the implementation of scalable and powerful computer vision solutions have been building intuitions for deploying Deep Learning applications on low cost and power efficient embedded hardware. Spiking Neural Networks (SNN) are the spike-based equivalent of ANN. They offer a more understandable model of neural computation by aiming to get closer to the real biological behaviour of the human brain. Moreover, the use of SNN based methods implemented on neuromorphic hardware has been investigated to seek for better accuracy, decrease power consumption and develop realistic production ready data analysis systems.

In this chapter, a brief introduction to the application of ANN for Computer Vision tasks is provided. A review on Convolutional Neural Networks (CNN) [4] is also given to build intuition behind the opportunities brought by deep learning to build image analysis systems. A discussion on the applicability of deep learning in the medical field is then presented. We also give an overview of state-of-the-art models build for brain tumor diagnosis systems. Then, we introduce spike-based computation for vision tasks, review key concepts behind SNN and discuss the opportunities they bring for medical image analysis. Finally, an overview of our work is given and details the organization of this thesis.

1.1 Computer vision with Artificial Neural Networks

Artificial neural networks were introduced as a mean to have a better understanding of the computation occuring in the human brain and to attempt to model biological neural networks to capture the structure of this computation. The first theory of an ANN model was proposed by McCulloch and Pitts [5] with a network of neurons implementing a thresholding function. In this model, each neuron takes a binary state set by the weighted sum of all the other neurons' states it is connected to. If this sum exceeds a predefined threshold the neuron's state is set to 1, meaning the neuron is active. Otherwise its state is set to 0 and the neuron stays inactive. This neuron model is defined as follows :

$$\gamma(\sum_{j=1}^{m} w_j x_j - w_0) \tag{1.1}$$

Where γ is the Heaviside function, x are inputs and w synaptic weights. In addition to this previous work, Hebb [6] proposed a learning rule that strengthen the connection between two neurons if they fire or are set to an activate state simultaneously. Thus, the Hebbian learning specifies if the weights of a connection between two neurons has to decrease or increase, influencing the state of other neurons in the network. The weight change Δw_{ij} between pre- and post-synaptic neurons i and j induced by a simple Hebbian learning rule is thus formulated as :

$$\Delta w_{ij} = \eta a_i a_j \tag{1.2}$$

Where a_i and a_j are the activation levels of pre- and post-synaptic neurons, and η represent the learning rate. These previous researches motivated the development of binary classifiers based on an artificial neuronal behaviour like the perceptron [7]. It aimed to solve the limitations of the tresholding neurons by redefining each neuron outputs by activation functions like the sigmoid function. The perceptron was thus defined to solve linear classication tasks and preceded the multilayer perceptron (MLP). The MLP appeared to solve the major drawbacks of the perceptron by increasing the number of output neurons and intermediate layers allowing to solve non-linear classification.

Since the creation of the MLP, ANN have evolved into powerful decision making models. With their computational power increasing over the years they also follow the growth of Big Data and become more and more accurate as they receive more samples to learn on. With the benefits of having good fault tolerance, being able to compute parallel processes or storing input data in the network rather than a remote database, they are now successfully used in a wide variety of domains. In a dynamic of going further with their abilities, researchers have been studying the benefits of such algorithms to solve vision tasks like image segmentation [8], object detection [9] or image reconstruction [10].

Convolutional Neural Networks have been at the center of these studies and confirmed the efficiency of ANN to solve vision tasks. They are commonly used for image processing such as classification, detection and segmentation. Performing on input images, they mostly rely on multiple layers built by convolutional and pooling layers. These operations can capture visual features without supervision, what standard ANN cannot do. They were also developed in the aim to reduce the computational cost of standard ANN when performing on image data. However, although they proved to be particularly efficient for vision tasks, their performance comes with a cost. In fact, most of these models cannot realistically be used outside the scope of a Graphical Processing Unit (GPU). This makes them rely on expensive and dedicated hardware to be trained and used for inference efficiently. Several successful CNN architectures like Inception [11], U-Net [12] or MobileNet [13]

are nowadays being extended and used to solve a large amount of computer vision related problems. They proved to obtain high scores of accuracy on several datasets such as the famous MNIST [14], for digit recognition, or the ImageNet [15] dataset for visual object recognition. These advances motivated studies in a wide number of fields including the medical field in order to adapt ANN for better and faster medical image analysis.

1.2 Deep Learning for brain tumor diagnosis

The early diagnosis of a disease plays an important part in determining and enhancing a potential treatment in order to improve a patient's well being or survival [16]. Fast and accurate diagnoses are essential to the survival of patients with severe diseases. Hence, the need for computer-aided diagnosis systems appeared as a way to accelerate the evaluation of medical data and seek for better accuracy [17]. Over the past few years, a lot of studies demonstrated the efficiency of Machine learning and deep learning methods for medical image analysis, bringing faster and better diagnoses [18].

The major interest for deep learning in the medical field has allowed breaking advances for designing fully-automatic diagnosis systems based on the latter tasks. During the last decade, CNN have been drawing a lot of attention in computer vision and an important number of different models have been proposed in the aim to build medical image analysis systems [19, 20, 21]. Additionally to MRI analysis, as diagnosis also aims to predict a treatment or the survival time of a patient, the diagnosis task needs to be addressed as an ensemble. However, as accurate CNN models were with basic object recognition tasks, analysing medical data appears to be more tedious for multiple reasons.

Since, computer-aided medical image analysis often relies on the use of multi-modal 3D images, training CNN requires a large amount of GPU memory. Despite the effort done to reduce the memory usage of these models and to increase GPU power, most modern approaches fail to provide lightweight models. For MRI segmentation, the problem induced by the dimension of the data can be addressed by chunking the slices into patches and classify them to obtain a segmentation map [22, 23]. However, such process relies on heavy data pre-processing operations and always requires to chunk each MRI slice before feeding data to the network. In a scenario when the image analysis system would have to predict a wide amount of segmentation maps, processing the data in this way each time a slice comes in for segmentation can be time-consuming.

It is also crucial to emphasize that when choosing CNN for medical image analysis, one may face limitations with data availability, privacy and ethics concerns. Convolutional Neural Networks were designed to perform supervised learning and thus require the

presence of ground truth labels or segmentation maps according to the performed task. When benchmark datasets such as the BraTS dataset [24] provide several sequences of MRI and ground truth maps for each case, the reality of gathering medical data is not as simple as this in practice.

Finally, training a CNN for tumor segmentation or classification requires the nature of the data to stay the same. This means that if a new type of tumor is given to the CNN to perform a diagnosis, the network will fail if it was not retrained with samples of this new data. Extending the knowledge of the network thus need structural changes. In the medical field, this makes CNN be lesion specific which suggest that one model cannot realistically be used for any type of tumor. The clinical perspective of the brain tumor diagnosis problem exposed in this thesis relies on the detection and the accurate segmentation of a tumor given Magnetic Resonance Images (MRI) to propose an appropriate treatment. Such perspectives would mainly consist in solving three vision tasks, namely detection [25], classification [26] and segmentation [27], each of which can be performed with great accuracy by deep learning methods. The next section exposes other issues related to the development of CNN and bring intuition on how to tackle them using spike-based computation.

1.3 The opportunity of spike-based computation for vision tasks

Capturing the essence of human brain functionality has been an important part of the premises of Artificial Intelligence (AI) researches. Initially based on neuromorphism, AI seeks to mimic the way the mammalian cortex functions to automate tedious tasks. It aims to improve a wide range of different domains such as medicine, security, agriculture or transportation. However, based on Deep Learning algorithms, modern AI does not rely on models inspired by biology and fails to mimic neuron dynamics. This step towards abstract AI lead to the emergence of limits when using and developing deep learning based solutions [28, 29]. In fact, the wide range of applications based on ANNs developed due to the growth of the Internet-of-Things (IoT) and cloud technologies emphasized major drawbacks.

One of most regarded of them is the dependence to powerful and costly hardware that prevents large scale deployment of deep learning solutions. In fact, while they become deeper and more powerful over the years, ANNs also become less power efficient and rely on whether high CPU or GPU resources, which comes with a relatively high purchase and energy cost [30].

Another limitation comes with the lack of explainability of their outputs and the functions they try to approximate [31]. This obstacle, often referred as ”the black-box issue” [32], thus limits the use of deep learning in domains like health-care or banking in which a decision has to be fully explained and understood. This leads us far away from creating models understandable by humans that would speed up the expansion of AI based products.

Finally, as no rule has ever been stated for the creation of ANNs architecture, developing them can be a tedious task. What creates yet another issue that prevents their large scale development and deployment. In fact, having no rigorous way to create these models can lead to consequent workloads when modeling new architectures to solve new tasks.

Spiking Neural Networks are thus reappearing as a way to step closer towards the reality of biologically-plausible brain computation and solve most learning and inference problems induced by ANNs computation [33]. Using models of biological neurons and encoding stimilus to spikes makes these networks more energy efficient [34]. Trying to model phenomena that are present in the nervous system can also prevents the use of deep architectures with hundreds of thousand neurons as they offer the possibility to exploit each neuron dynamics for learning. Indeed, synchronization of neuronal activity is one of these phenomena and can be used for learning.

Therefore, the spike-based computation carried by SNNs is trying to address challenges carried by traditional ANNs by putting neuromorphic computing back to the center of AI applications in order to make them more suitable for production, real-time environments and hardware implementation. These facts then motivate researchers to lead AI towards spike-based models in order to look for better explainability and improve existing solutions by making them more suitable for local daily use in a wide range of domain.

1.4 Thesis overview

The study covered in this manuscript aims to solve visual tasks to perform brain tumor diagnosis, specifically for glioma tumors, by the mean of neural networks while addressing most issues these methods can bring. To this aim, our work focuses on the transition between deep neural networks and spike-based models to build cost and power efficient brain tumor diagnosis applications. The following work material is thus divided in four main chapters as follows :

1. **Chapter 2** presents some knowledge and concepts inherent to the subject exposed in this thesis. Hence, this chapter first gives an introduction to object recognition in Section 2.1. Then, Section 2.2 details properties inherent to brain imaging and

review a benchmark dataset widely used in the state-of-the-art. In Section 2.3, basic computational neuroscience knowledge is introduced by discussing neuron models and learning methods. Finally, the applicability of spike-based models to computer vision is discussed in Section 2.4.

2. **Chapter 3** details the first contribution of this thesis and discuss the importance of cost reduction in Deep Learning based methods in the medical field with the example of a brain tumor segmentation task. In Section 3.2, we introduce an cost-efficient GPU embedded system optimized for the development of deep learning applications and discuss its applicability in the development of CNNs. Section 3.3, provides an overview of CNN compression methods used to move the computation of such models to cost-efficient systems. Finally, Section 3.4 provides details about our implementation and results as well as a comparison of our contribution with other models in the state-of-the-art.

3. **Chapter 4** provides our second contribution and discusses the use of the PCNN for medical image processing. First, Section 4.2 gives an introduction to the computation of several different PCNN models. Secondly, in Section 4.3, an introduction to medical image fusion is provided by emphasizing on the potential of modified PCNNs. Then, Section 4.5 provides details about our experiments on PCNNs for brain tumor segmentation and detection tasks.

4. **Chapter 5** details our third contribution and investigates the development of a single layer Convolutional SNN (C-SNN) for brain tumor classification. Section 5.2 first exposes details on the different classification tasks the model was trained to solve. Then, a data pre-processing pipeline is discussed to prepare MRI slices to be encoded for training. Section 5.3 introduces the structure of the proposed C-SNN by detailing its topology and the learning mechanisms used to train it as well as implementation details. Finally, the results of our experiments on each of the classification task are discussed in Section 5.4. We evaluate the performance of the models by comparing the average classification accuracy to state-of-the-art deep learning models.

Chapter 2

Background and methods

Since the development of modern fully-automatic brain tumor diagnosis systems requires the understanding of several different disciplines such as medical imaging, computer science or neuromorphic engineering, an introduction to these subjects is necessary in order to fully recognize the challenges exposed in this thesis.

Hence, this chapter firstly introduces object recognition in Section 2.1 with a review of recognition methods and Deep Learning models used to complete computer vision tasks. Imaging for brain tumor diagnosis is then introduced in Section 2.2 by discussing brain tumors and imaging as well as reviewing a benchmark dataset of brain scans used in the state-of-the-art. Next, we provide an introduction to Computational Neurosciences in Section 2.3 by reviewing spiking neuron models, neural coding as a data conversion scheme and synaptic learning rules and concepts. Finally, in Section 2.4, we focus on the development of Spiking Neural Networks for image processing and review some of the methods used in the state-of-the-art for computer vision tasks.

2.1 Object recognition

Object recognition is a well-known labeling problem that the human brain can solve with great ease. It aims to match objects perceived in a visual stimulus to some labels already known by the individual. While it appears natural for humans, computer vision systems divides an object recognition task in multiple different sub-tasks (See Fig. 2.1) :

- **Detection** is the task of locating an object within an image and is often performed using bounding boxes.

- **Classification** aims to identify the object and assign it to a label.
- **Segmentation** gives a label to every pixel in an image in order to obtain a segmentation map that divides the image in multiple regions of interest.

When solving these tasks, a large amount of computer vision models are still aiming to approach human performances in terms of recognition. However, they are limited to the amount and nature of data they process and often do not come close to the capabilities of the brain when it comes to generalise. In fact, the high efficiency of the brain to process visual information allows an individual to accurately recognize any object even after only seeing it a couple of times. In order to copy this behaviour, a vision system has to be highly invariant to a wide amount of image properties such as variable lighting [35], visual variation (e.g. scale, orientation, viewpoint) [36] or situations when the object is occluded [37]. This can be daunting as it induces the need for large datasets containing these variations or the use of data augmentation methods. It also means that the extraction of such properties would have to be performed to discriminate the objects in an image. This can have the effect of slowing down the recognition process and does not allow real-time processing which is often demanded for visual systems.

For these reasons, object recognition has been at the core of a wide amount of researches in computer vision over the last decade [9, 38, 39]. These studies highlighted the fact that these systems do not incorporate all the necessary processes to build recognition models that would perform as well as the human brain. In fact, while the brain uses different visual streams composed by a set of neuron groups used to solve distinct tasks (See Section 2.4.1, the common processing pipeline of a recognition model is much less complicated. This pipeline starts with region proposal or selection methods to performed the detection task, then, the extraction of features from the image is computed to finally complete the classification of the object.

2.1.1 Region selection

The ability of the human brain to differentiate objects in a scene can be pictured as a division of the visual stimulus in sub-regions holding discriminating properties. In computer vision, the region selection helps to direct a recognition system towards these sets of pixels referred as Region-of-Interest (ROI). Hence, to have an accurate interpretation of an image, this division is mandatory as it allows to discriminate objects or parts of them. A simple region selection process scans the image using a fixed size window that slides over the image and later sends each region to a classifier to complete the recognition process [40].

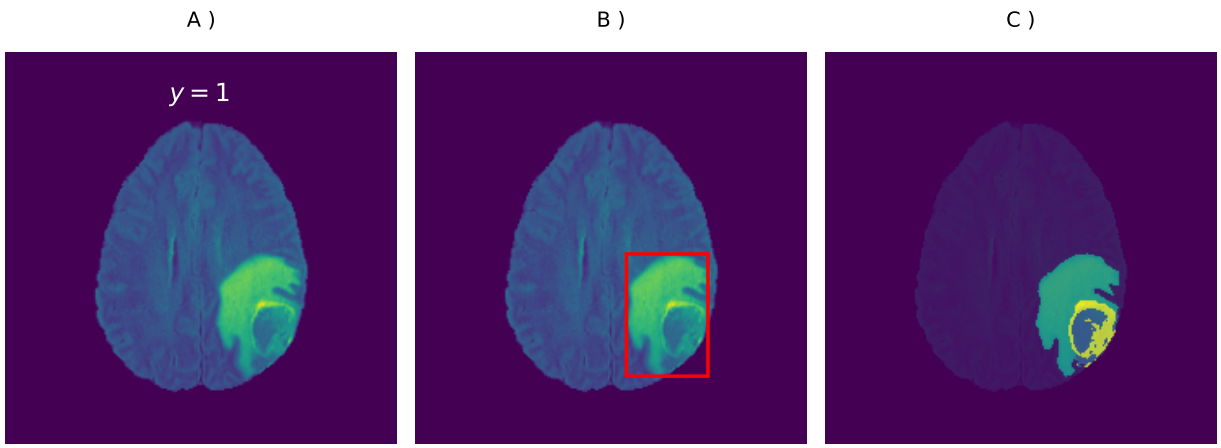

Figure 2.1: *Recognition process for brain tumors. A) Tumor classification, the MRI is assigned to a numerical label y B) Tumor detection using a bounding box. C) Each pixel is classified to perform tumor segmentation.*

This simple method is referred as the Sliding Window algorithm. Although this algorithm performs great to produce small patches of an image, building an efficient and accurate recognition model out of it would induce the generation of a large amount of sub-regions of various sizes and shapes what happens to be computationally costly. Sliding over the entire image to produce overlapping regions can also be counter productive as different areas of the image hold a different amount of information. It is the case for MRI slices that mostly contains background information if they are not cropped.

A way to perform region selection while maintaining its computational cost as low as possible is to base it on a segmentation process. This kind of process attempts to combine similar pixels in an image to produce the ROIs. Early image segmentation works made the assumption that images had to be uniquely divided and attributed a part of this division to each object outlines [41, 42]. However, the fact that images, and object categories they hold, are hierarchical implies that this segmentation has to be done at multiple scales [43]. Typically, this means that relying on one segmentation strategy is not realistic to find all ROIs in an image. Indeed, objects cluttering each other might be separable using different properties like texture, colour or context. The Selective Search [39] algorithm is one of the most popular region proposal method that aims to deal with these limitations. It uses a hierarchical grouping-based segmentation algorithm introduced by F. Felzenszwalb and D. Huttenlocher [44]. In this algorithm, some initial regions are used to start the hierarchical grouping, similar regions are merged to produce larger ones, and this process is repeated until the entire image is represented by a single region. The region proposal list is thus created with each of the found regions during the process. The merging of regions is based

on different similarity metrics which produces multiple segmentation strategies. Proven to be efficient and fast, the Selective Search algorithm was successfully used in deep learning to produce Region Based Convolutional Neural Networks that aimed to perform object detection. This method is later used in Chapter 4 and combined with a spike-based model to create a tumor detection system.

2.1.2 Feature Extraction

Feature extraction is a crucial step to build efficient models for computer vision. In the context of object recognition, it offers a new representation of the original data by extracting meaningful attributes that may discriminate the information it contains [45]. The type of these attributes can differ according to the task to perform and can be referred as global or local features [46]. The algorithms used to retrieve these features are called descriptors. A global feature descriptor aims to quantify an image globally and retrieves meaningful information using all the pixels values. Most commonly retrieved global features in computer vision include textures, global shapes, color histograms and pixel statistics. Since global feature extraction does not enter the details of the data, descriptors have the advantage of being computationally efficient. Local feature extraction does the exact opposite and aims to describe small groups of pixels within the data to highlight small regions of interest like edges or other distinct structures found in image patches. These parts of interest are referred as salient regions or key points. Finding local features also improves recognition models' sensitivity to variance and can help solving some of the issues exposed in Section 2.1. In fact, by only highlighting differences between small parts of the data and their surroundings, local features are robust to occlusion, clutter and variations. Some commonly used local feature descriptors are Scale Invariant Feature Transform, Harris Corner or Binary Robust Independent Elementary Features. An example of both global and local features extraction can be seen in Fig. 2.2.

2.1.3 Classification process

The classification process is the last step in the recognition task pipeline. Algorithms responsible to perform this process are called classifiers. They make use of feature extraction to find bounds in the feature space that best discriminate the classes relevant to the data. Fitting these classifiers to a specific classification task requires to go through three different phases namely, training, validation and testing. Indeed, in order to be efficient, these classifiers are firstly trained on a sub-set of the data to learn to map each sample to an output label. A validation data sub-set is then used to evaluate the performance of the

Figure 2.2: *Example of image feature extraction.*

model during training by the mean of an accuracy rate computed from the classifier's prediction and the true label from the data. Finally, the testing phase performs the total evaluation of the algorithm by sending new samples from a third sub-set and ensures that the classifier is able to generalize.

A classifier's poor ability to generalize means poor classification performances which is mostly caused by two modeling error named under- and over-fitting [47]. Under-fitting mostly occurs when the chosen classifier cannot approximate the function that explains the data. A simple example of under-fitting is the use of a linear classifier on a non-linear. It can also occur if the set of training data was not sufficient to learn the right features. On the contrary, over-fitting happens when the classifier's function is too close to the training data, to the point that it cannot accurately process a sample outside the training scope. A lack of data to train the classifier can also induce over-fitting. Data augmentation [48] is a common method used as a solution to this, which allows the creation of new training samples by performing transformations on the existing ones.

In computer vision, the SVM algorithm has been widely used as an image classification tool. It aims to discover a hyperplane in an N-dimensional space that classifies each data sample. Originally used as a linear classifier, non-linear classifications is made possible by using a high-dimensional space transformation. This is done by changing the kernel functions used by the algorithm. Although SVM proved its efficiency in image classification [49], the advances in deep learning offered new opportunities and possibilities to perform the entire recognition pipeline through the use of Convolutional Neural Networks.

2.1.4 Convolutional Neural Network

In the 1960s, experiments carried by Thorsten Wiesel and David Hubel found that different visual stimuli were processed by different types of cells in the primary visual cortex (V1).

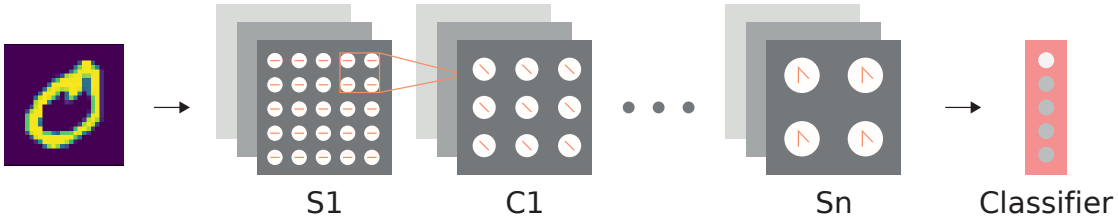

Figure 2.3: *Basic structure of a convolutional neural network. Layers are composed of Simple Cells (S) that perform feature extraction followed by Complex Cells (C) that perform pooling to add translation invariance. The bottom layer of the network is a classifier that receives the transformed representation of the data obtained from intermediate layers.*

This cells, named simple and complex cells, highlighted a filtering-like behaviour happening in the brain. Inspired by this discovery, the first Convolutional Neural Networks appeared in the 1980s [50] and their breakthrough happened with the development of several CNN models for handwritten digit recognition [51]. Taking advantage of the growing amount of visual data to process, CNNs then showed convincing results in a wide range of applications like self-driving cars, event detection or medical image analysis tasks like the one exposed in this thesis. These advances thus highlighted their efficiency and made them the center of attraction in computer vision. In the context of our study, CNNs were used to design powerful medical image analysis systems to solve brain tumor classification and segmentation tasks.

The key concept that makes CNNs particularly efficient as image analysis models is their convolution operations used to apply filters on their inputs. These filters perform feature extraction and thus reduces the dimensionality of the data and drastically decreases the amount of parameters used by the model. This makes CNNs to be highly scalable to large dataset as they can adjust their weights quickly and are more robust to overfitting. Just like the visual cortex, CNNs are hierarchically organized into a sequence of different layers. These layers model a succession of simple and complex cells that are responsible for selecting features and building translation invariance respectively (See Fig. 2.3). In the context of a object recognition, this organization leads to perform a series of alteration to the input data so that the last layer offers a representation of the data that would bring enough information to discriminate it. The architecture of most modern CNNs include four main types of layers namely convolutional layers, non linearity layers, pooling layers and fully-connected layers. Moreover, other types of operations like normalization and regularization can be used to improve CNNs performances.

Convolutional layer

In image processing, convolution operations are used as a technique to alter an image by convolving a value matrix known as kernel over the entire image. The term filter is used instead of kernel when the number of channels in the image is greater than one as filters are simply stacked up kernels. The concept is highly inspired by biological receptive fields (RF) which are regions of the retina that induce neuronal activity on light stimulus. Mimicking the local connectivity in RFs, the convolution operation on an image can be seen as a sliding window over pixels and their local neighbors. The size and values of the kernels influences the transformations applied to the original data. For example, applying Gaussian kernels have the effect of adding blur to the image and thus offers a smoothing transformation. A discrete convolution operation is described as :

$$f(x,y) \cdot k(x,y) = \sum_i \sum_j f(i,j)k(x-i, y-j) \tag{2.1}$$

Where f is the image to transform, k is the convolution kernel and (x, y) the pixel positions. Some common processing techniques performed by convolution operations with different kernels include edge detection, blurring and sharpening as seen in Fig. 2.4

Convolutional layers are created upon the combination of these filtering operations. They aim to model the V1 simple cells discussed above and are responsible for the feature extraction performed by the network. Typically, a CNN's parameters are learnable filters that produce an activation when they passes by visual features like sudden high contrast, edges or shape patterns. These activations produce a new representation of the input data called feature maps. For each filter used in the layer, one feature map is created. The maps are thus stacked to produce a volume that is the output of the layer.

An important concept under the efficiency of convolutional layers is parameter sharing. Taking the example of an RGB image that passes through a layer that gives an output volume of 33x33x64 neurons each of which possesses 9x9x3 filters, the total number of parameters of this layer is of $(33 \times 33 \times 64)(9 \times 9 \times 3) = 169,361,28$. Parameter sharing prevents ending up with this kind of high amount of parameters by constraining each neuron to use the same filter. This comes from the assumption that some features are repeated in the data at different position and there is thus no need to compute a different filter for each channel. Hence, taking the same example, this scheme reduces the amount of parameters to $64 \times 9 \times 9 \times 3 = 15552$ which is extremely lower than the previous number. However, this scheme comes with limitations that could slow down the training process of the network. Indeed, sharing parameters does not serve a great purpose in every task. This is the case in recognition task where the subjects are always located at the same positions

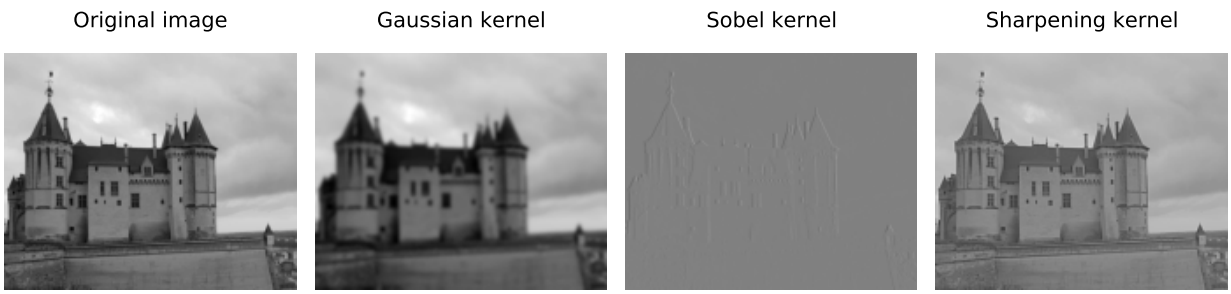

Figure 2.4: *Example of the convolution operation using different kernels.*

in the image but might use different features to be recognized. In that case, using different filters is crucial and locally-connected layers should be used instead.

Non-linearity layer

Convolutional layers are often followed by a layer of non-linearities which expands the capabilities of the model. Indeed, since most natural phenomena do not reply to linear rules, it is crucial that CNNs can make use of non-linearity to deal with more complex tasks. To provide this non-linearity, early deep learning models used functions like the hyperbolic tangent or the Sigmoid function (See Fig. 2.5). However, both were proved to prevent models to be sparse because of their ranges. In fact, as the outputs of such functions have low or non existent likelihood of being zero, small neuron weights will still be considered by the model. Having activation outputs with low values also induces the well-known problem of the vanishing gradient [52].

To improve the previous statements, the Rectified Linear Unit (ReLU) [53] function was introduced and is one of the most commonly used activation function to add non-linearity in modern CNNs. It outputs a zero value for all inputs that are smaller than zero and x to the other ones as follows :

$$f(x) = max(0, x) \tag{2.2}$$

With its zero response on negative values, this function lets neurons that add a poor contribution to the model to be silent which highly benefits the sparsity of it. The vanishing gradient problem is also dealt with has the gradient computed during optimization will always be a constant since the derivative of $f(x) = x$ is one and $f(x) = 0$ is 0. This eases the training process of the model by reducing the time needed for it to converge. Since the computation of ReLU also only relies a on max function, its computationally more efficient than the hyperbolic tangent or Sigmoid. However, if it performs better than the

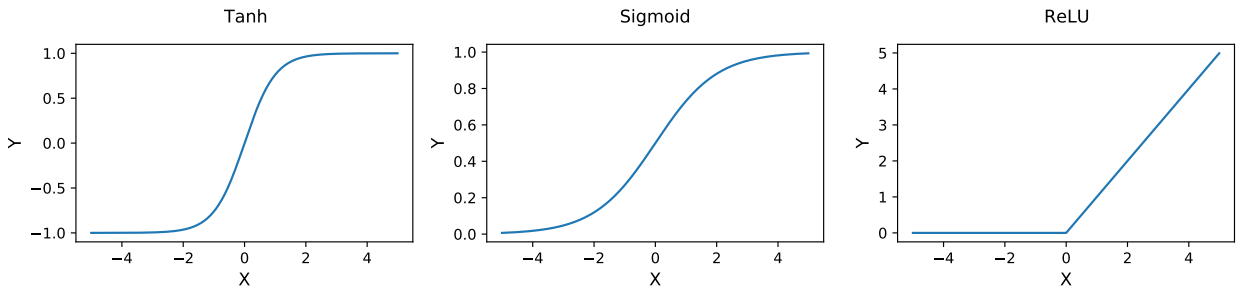

Figure 2.5: *Non-linear activation functions.*

latter functions, ReLU brings its set of issues that need to be considered.

Indeed, keeping the output of low values to zero first causes the dying ReLU problem. This phenomenon mostly occurs when a network is badly initialized as the weights can be drawn to the zero output and never be able to recover. At terms, when a large amount neurons output zero, no more changes are made during training and the network enters a dead state. Another typical problem induced by the use of ReLU is the production of large values as the positive side of the function's domain is limited to infinite. However, in recent works, many models use modified versions of the ReLU or new activation functions to tackle these issues such as the Leaky ReLU [54] or the ELU function [55].

Pooling Layer

Often inserted between convolutional layers, they provide a great spatial size reduction scheme that reduces the amount of parameters of the network and thus ease training while preventing over-fitting from occurring. Designed as models for complex cells, pooling layers also allow to provide translation invariance to the network. The most popular form of pooling in CNNs is the Max-Pooling. It applies a Max operation with a 2x2 filter and a stride of 2 on each of the feature maps slice to down-samples their width and height by 2 as illustrated in Fig. 2.6. Note that, in recent works, pooling operations are often replaced by a series of convolutional layers with larger strides to perform the spatial size reduction.

Fully-connected Layer

The fully-connected layer is the bottom part of a CNN. It is mostly used to learn non-linear patterns from the representation of high-level features obtained from the previous layers and can be seen as a simple multi-layer perceptron. The input x of this layer is a 1-Dimensional volume obtained from flattening the output of the last layer before it. The output i-th output y_i of a fully-connected layer is defined as :

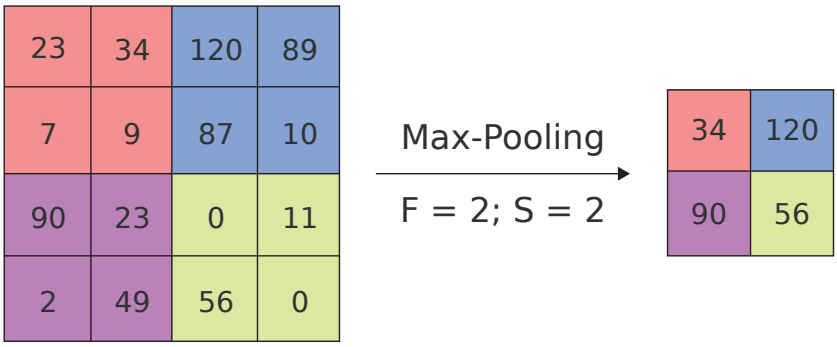

Figure 2.6: *Max-pooling operation with a filter (F) of size 2x2 and a stride (S) of size 2.*

$$y_i = g(\sum_{n=1}^{m} w_n x_n) \tag{2.3}$$

Where w_n are learnable weights and g is a non-linear activation function like the ones previously introduced. After a pass through this layer, the classification process is performed using a Softmax activation function. For a classification problem with N being the number of classes, the output of this Softmax operation is an N-dimensional vector, each value in the vector being the probability that the input belongs to the corresponding class.

2.2 Imaging for brain tumor diagnosis

Medical imaging is a crucial part of medical practices and its development has shaped the modern healthcare industry while offering new tools to better understand the human body. It serves a great purpose in a wide range of applications like the establishment of a diagnosis, the progression monitoring of a disease or the support in surgical procedures. The wide amount of information contained in medical images can also assist healthcare providers to offer better and more comprehensive treatment. In the context of brain tumor diagnosis, the advances in medical imaging have enabled the noninvasive study of tumors and drastically changed the way brain tumors are being treated and diagnosed. Imaging is in general the first tool used to identify infected tissues and can help to determine the present type of tumor according to its shape or location. Different imaging techniques can also be used to proceed to a further examination of the tumor and assist pre-surgical planning as well as the post-treatment patient's care.

In this section, we first define brain tumors and emphasize on gliomas which are the

type of tumors treated by the contributions in this thesis. Then, a presentation of different imaging techniques and their applications to brain tumor diagnosis is given. Finally, we introduce the benchmark dataset of glioma cases used to carry the majority of the experiments presented in this manuscript.

2.2.1 Brain tumors

A tumor is defined as an abnormal growth and division of cells that proliferate and stay alive by ignoring regulation mechanisms. When they grow but do not invade nearby tissues, these tumors are said to be benign and are usually not life-threatening. On the contrary, malignant tumors have the ability to spread to other tissues and organs using the bloodstream or the lymphatic system. This spread is referred as metastasis or secondary tumor as it originally comes from the primary tumor.

As their name implies, brain tumors affect the brain region. The most commonly found types of primary tumors in the brain are gliomas that start from glial cells and represent the majority of malignant cases. Meningioma tumors that originate in the meninges are also frequent but mostly benign. Moreover, the malignancy of gliomas is noticeable by the World Health Organization grading system that offers a criteria to anticipate the consequences of a therapy or the survival time of a patient. Indeed, the overall survival of patients with grade II gliomas is of more than 5 years while its is of about 3 years for patients with grade III tumors [56]. The glioblastoma, being a grade IV glioma, is the most aggressive and infiltrative of them due to its chemotherapy and radiation resistance as well as its reappearance ability after resection [57]. It is defined by a very low survival rate, inaccurate prognosis and restricted therapy options.

Since there are no evidences of an effective medical or surgical treatment for this type of tumor, a diagnosis with an early detection and precise classification is crucial to offer the right treatment [58]. However, mostly relying on medical imaging, diagnosis methods are often performed at a late stage due to an absence of clinical symptoms which does not allow a quick medical response. Reinforcing the medical imaging based diagnosis of glioblastomas is thus critical to gain time and anticipate the growth of the disease. The contributions in this thesis thus focus on the analysis of glioblastomas by using an open-source multi-modal brain image dataset later introduced in Section 2.2.3.

2.2.2 Imaging modalities

Brain Imaging provides a wide range of methods which give enough visual information to study brain structures and help spotting potential lesions or anomalies. The images retrieved

from these methods are substantially used for diagnosis and to monitor the evolution of tumors. Depending on the task and the lesion to find, different types of imaging techniques can be used to ease the diagnosis.

Computerized Tomography

Computerized Tomography (CT) scans are obtained by a series of X-rays taken from different angles. By measuring the absorption of these X-rays by the tissues in the body, a series of 2D images are gathered and used to create a 3D volume by tomographic reconstruction. This method provides images with high spatial resolution and have the ability to give good representations of soft tissues, blood vessels and bones. It also has the advantage to be fast which makes it suitable for short studies when a quick identification of a lesion is needed. However, a major drawback of this type of imaging is that the X-rays it relies on produce ionizing radiation which were found to induce DNA cellular damage and cause cancers.

Positron Emission Tomography

Positron Emission Tomography is a nuclear imaging technique particularly used to study the metabolic activity of body tissues. It is based on an injection of a radioactive tracer in the blood that binds to the cells and allow the observations of their metabolisms and how they are transformed by afflictions. This tracer is made by marking a molecule with a radioactive isotope and can thus be designed to target specific tissues or organs that may be ill. In the context of brain tumors, the Fluorodeoxyglucose is often used to trace brain cells metabolism. Indeed, since the energy and proliferation of brain tumors mostly relies on glucose, the absorption of this tracer will induce a detection of tumorous cells.

This type of imaging is thus often used to monitor the progression of tumors and metastasis as well as the response of a therapy. However, the spatial resolution and the lesion detectability of this method is limited and lower than the other techniques presented in this Section. The PET is often combined with CT scans to retrieve metabolic and anatomical information at the same time.

Magnetic Resonance Imaging

The MRI is a technique that produces three dimensional detailed anatomical images of organs and tissues by using a magnetic field and radio-frequencies. It is based on the detection of signals generated by the magnetic resonance of atoms, mostly produced by hydrogen protons. Typically, the magnetic field pushes protons to align with it and are then stimulated by a radiofrequency current sent through the patient, also called pulse

sequence. This radiofrequency makes the protons lose equilibrium and force them to resist the magnetic field's attraction. By turning off the pulse, the protons realign with the magnetic field and release energy that can be detected by the MRI sensors. The amount of energy released by the protons and the time they took to realign can thus be monitored to differentiate various types of tissues, as it highly depends on the nature of the molecules, and thus detect lesions like tumors.

Different settings in the pulse sequences can lead to the creation of different representations of the studied tissues with various contrasts referred as MRI sequences. For brain tumor imaging, some commonly used MRI sequences include the following :

1. **T2**. Performs a suppression of fat and an attenuation of fluids.
2. **FLAIR**. Performs a suppression of fat and fluids.
3. **T1**. Performs fat suppression enhanced by a gadolinium-based contrast agent.

Using multiple MRI sequences allows to get more information on the studied lesion and can thus ease its analysis as low contrast inherence to some of them often do not help to provide a detail identification. For brain tumors, the use of several sequences offers the opportunity to identify different tumorous tissues like edema or necrosis. MRI is thus well suited for tumor detection and progression monitoring and is more sensitive than other imaging techniques. It is however not capable to give information on the type of lesion making it harder to distinguish tumors from other similar lesions like inflammatory masses. Regarding these facts and the higher availability of MRI scans as open-source datasets, all the experiments presented in this work were carried using MRI data.

2.2.3 BraTS dataset

The important and difficult nature of computed-aided brain tumor segmentation has brought a significant attention over the last 20 years. This interest induced the development of a wide amount of various methods that aimed to perform fully-automated segmentation of tumorous tissues. The comparison of these methods was however not easy to do has each method was evaluated on different private datasets with different evaluation metrics. Indeed, gathering medical data to build computer-aided analysis systems can be a tedious task. Being protected by ethics committees and hospitals, the absence of this type of data slows down the growth of deep learning applications for the medical field and prevents the improvement of existing methods. Hence, providing enough evidences to prove that a model performs better for the given task than another one is not realistic in these settings.

In order to change that, the Multimodal Brain Tumor Image Segmentation Benchmark (BraTS) [24] was firstly introduced in 2012. This challenge made available a dataset of MRI comprising both low- and high-grade glioma cases. The first version of this dataset contained 51 high-grade cases comprising glioblastoma multiforme tumors and anaplastic astrocytomas, and 14 low-grade cases comprising astrocytomas and oligoastrocytomas. The images representing pre- and post-therapy scans were acquired by various institutions, scanners and clinical protocols. They all incorporate 4 different MRI sequences, namely, T1, T1c, T2 and FLAIR. Since the T1c volume had the highest spatial resolution, each sequence was co-registered to it in order to homogenize the data and ease processing.

For each case, a segmentation ground truth was provided and corresponded to a manual segmentation performed by several experts. In this segmentation map, four labels corresponding to different tumoral structures were retained, namely necrosis (label 1), edema (label 2), non-enhancing tumor (label 3) and enhancing tumor (label 4). All other brain tissues were given label 0 corresponding to non-tumorous cells. For the purpose of the recognition process, these labels were grouped in 3 regions, The enhancing tumor region (only containing label 4), the core region (containing all labels except label 2) and the complete tumor (containing all labels).

By defining a standard for the evaluation of brain tumor segmentation methods, the availability of this dataset induced a significant growth in the development of fully-automated diagnosis systems. With time, the dataset grew bigger in the number of cases and data concerning overall survival was added to extend the challenge to survival prediction models. Our work uses different versions of this dataset as it provides enough quality and samples to build powerful tumor recognition systems.

2.3 Computational Neurosciences

Computational neuroscience (CN) is a domain that aims to provide a better understanding of the brain by modeling neural mechanisms that underpin cognitive skills. It expresses the possibilities of exploiting computational researches to discover what roles brain structures have, how they perform different tasks or simply how information is processed by the nervous system.

While the field originally provided theory about neuronal computation to build models that replicated the dynamics of brain structures, its implication in Artificial Intelligence has recently brought a significant interest. Indeed, regarding the success of ANNs, which computation was firstly inspired by biology, it appears obvious that incorporating computational neurosciences to AI based researches can lead to new opportunities in terms of

efficiency. Compared to deep learning algorithms, that are predominant in modern AI, biologically plausible models outputs are explainable which makes them perfectly suited to build non-abstract AI systems. If early researches in computational neurosciences only focused on building mathematical models of neurons, recent approaches to this domain extended these works to build SNNs with the aim to solve cognitive tasks. Multiple works also investigated how learning was processed in the nervous system by providing trainable synapse models.

This section thus provides an overview of some of the key concepts of computational neurosciences. From basic neuron modeling and encoding to an overview of synaptic plasticity models and basic learning rules, we introduce the methods needed to understand the work exposed in Chapter 4 and 5.

2.3.1 Spiking neuron models

Neurons are the principal computational elements in the brain. Their main functions are to encode and transmit information by the mean of action potentials (spikes). These spikes occur when a neuron was sufficiently excited by other surrounding neurons. Hence, a spiking neuron can be defined by the integration of a current over time expressed as the sum of incoming i^{th} spikes as :

$$I(t) = \sum_{s \in S} v_i g(t - t_i) \tag{2.4}$$

Where v_i is the potential of the spike that drives the amplitude of the pulse, t_i is the spike timing and S the set of spikes. g is a pulse function, it often uses the Dirac's delta to produce short impulses defined as :

$$\delta(x) = \begin{cases} 1 & \text{if } x = 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.5}$$

Building mathematical models of biological neurons dynamics has been, and still is, at the center of researches in computational neurosciences. A lot of research works hence provide a substantial range of models and discuss their applications and utility [59]. This section gives an introduction to some of the most prominent neuron models used in the state-of-the-art.

Hodgkin-Huxley model

Conductance-based neuron models marked the premises of spiking neurons, the Hodgkin-Huxley model [60] is amongst the most important ones of them. In their experiments on the giant axon of the squid, Hodgkin and Huxley investigated ionic flow by introducing an electrode into the cell and applying a current to see how the flow of ions and the cell's membrane changed. They showed that the conductance of the cell's membrane to potassium (K) and sodium (Na) ions depends on the membrane potential and were able to construct precise equations to describe the voltage and time dependence of these ions conductance. The dynamics of the membrane potential $v(t)$ of the Hodgkin-Huxley model is thus defined as :

$$C\frac{dv}{dt} = -I_L(v) - I_{Na}(v) - I_K(v) + I_{syn}(t) \tag{2.6}$$

With I_L being a leak current as $I_L(v) = g_L(v - v_L)$ where v_L and g_L are the leak potential and conductance respectively. The potassium current I_K and sodium current I_{Na} are defined by :

$$I_K(v) = \bar{g}_K n^4 (v - v_K) \tag{2.7}$$

$$\tau_n(v)\frac{d_n}{dt} = -n - n_\infty(v) \tag{2.8}$$

$$I_{Na}(v) = \bar{g}_{Na} m^3 h (v - v_{Na}) \tag{2.9}$$

$$\tau_m(v)\frac{d_m}{dt} = -m + m_\infty(v) \tag{2.10}$$

$$\tau_h(v)\frac{d_h}{dt} = -h + h_\infty(v) \tag{2.11}$$

Integrate-and-fire model

Derived from the Hodgkin-Huxley neuron as a simplification of the model, the Integrate-and-fire (IF) neuron is easily the simplest and most used neuron model in neuronal computation. It considers every spike as an individual event described by its timing. As its name states, this model starts with the integration of the membrane potential v until it reaches a threshold v_θ to produce a spike. Once the neuron fired, v is reset to a resting potential v_{rest} and the neuron is not allowed to fire again during a refractory period τ_{ref}. This model is defined by

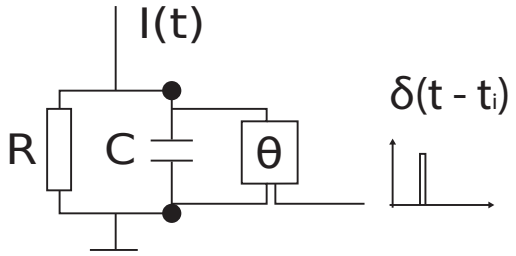

Figure 2.7: *Simple circuit of an Integrate-and-Fire neuron.*

:

$$C_m \frac{dv}{dt} = I(t) \tag{2.12}$$

$$\text{if } v \geq v_\theta, \text{ then } v \leftarrow v_{rest} \tag{2.13}$$

One of the most popular IF model is the Leaky Integrate-and-fire (LIF). It steps closer to biological activity by forcing the neuron to go back to a resting potential when no spike has been received. It does so by introducing a leak to the membrane potential that consists of a capacitor C in parallel with a resistor R driven by an input current $I(t)$. A LIF neuron can be expressed as :

$$\tau_m \frac{dv}{dt} = (v(t) - v_{rest})RI(t) \tag{2.14}$$

$$\text{if } v \geq v_\theta, \text{ then } v \leftarrow v_{rest} \tag{2.15}$$

Izhikevich Model

The model of spiking cortical neuron proposed by Izhikevich [61] is more complex than the models exposed above. Unlike the Hodgkin-Huxley model introduced above, the Izhikevich model does not take the biophysics of neurons into consideration.

$$\frac{dv}{dt} = 0.04v^2 + 5v + 140 - u + I(t) \tag{2.16}$$

$$\frac{du}{dt} = a(bv - u) \tag{2.17}$$

$$\text{if } v \geq 30mV, \text{then} \begin{cases} v = c \\ u = u + d \end{cases} \tag{2.18}$$

Where v is the membrane potential, u represents the membrane recovery variable and a, b, c and d are dimensionless parameters allowing to model different neuronal behaviour and firing pattern. This model is able to compute a wide range of different spiking patterns of cortical neurons, all obtainable by modifying the values of a, b, c and d. There are two substantially used types of spiking patterns obtained by tuning this set of parameters, namely Regular Spiking (RS) neurons with $\{a, b, c, d\} = \{0.02, 0.2, -65.0, 8.0\}$ which is mostly used to model excitatory neurons and Fast Spiking (FS) obtained with $\{a, b, c, d\} = \{0.1, 0.2, -65.0, 2.0\}$ to model inhibitory ones.

2.3.2 Neural coding

Neurons in the nervous system transform stimuli into sequences of neural spikes. The exact timing of these spikes is key for processing information. While conventional artificial neuron networks lack the ability of encoding information as a temporal code, neural coding [62] is one of the most important mechanism in SNNs. It is at the core of their power efficiency. Advances on neural coding led to the development of neuromorphic visual capturing devices such as Dynamic Visual Sensors (DVS). These devices can receive visual inputs and output their spike train representation. This drastically reduces the workload of machine perception. However, when data cannot be directly transformed into spikes, a conversion as to be implied with the right coding strategy. Different works on SNNs development debate the use of three main types of coding stategies, namely temporal coding, population coding and rate coding.

Rate coding

Rate coding is the most popular encoding method used in the state-of-the-art. The method is based on observations made on biological neurons' spiking activity [63] and let a neuron fire at a frequency that is proportional to its input intensity. A high intensity thus leads to a high neuron firing rate. This method also often relies on Poisson distributions to generate random spike firing times with respect to the coding frequency. In the context of image processing, pixels are represented by neurons with firing frequencies that are proportional to the pixel's value. When using this method, images are exposed for a certain duration to make sure that later neurons will receive enough spikes emit post-synaptic spikes. After

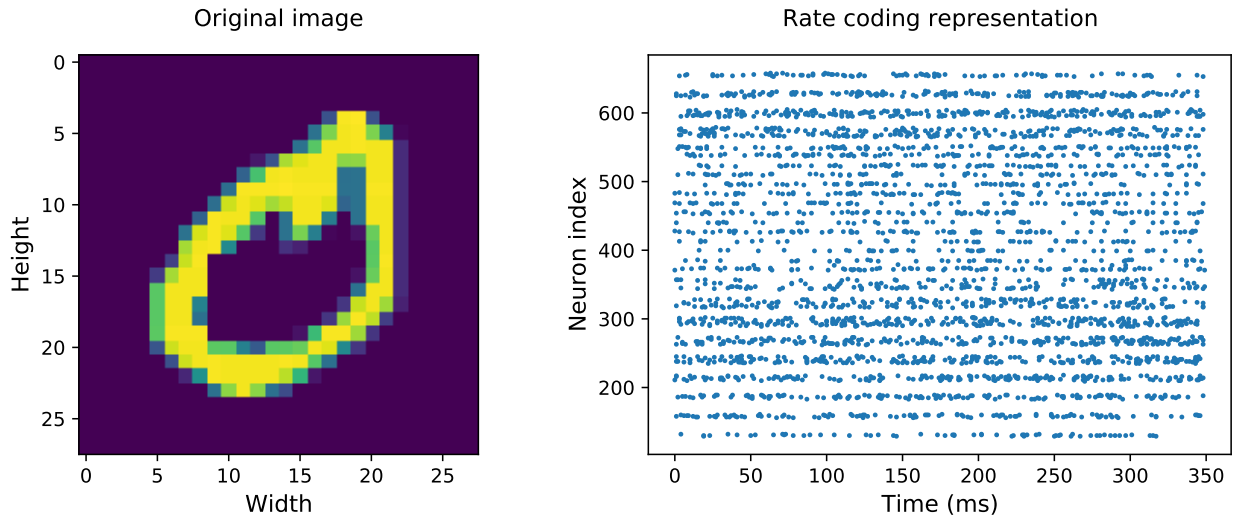

Figure 2.8: *Example of the rate coding method on an MNIST image over a 350 milliseconds duration using the Poisson process.*

each image exposition, a resting period is often used [64] to conduct the neurons to their resting state a prevent them from firing again before a new image is presented. An example of an image, from the MNIST dataset, encoded into spike trains with a maximum frequency of 63.75 Hz can be seen in Fig. 2.8.

Temporal coding

In temporal coding, information is assumed to be carried by the precise firing time of neurons. With this method, a value is encoded using an individual neuron that only fires once, meaning that one spike is enough to represent the value. Hence, compared to rate coding, this method has the advantage of being able to bring a lot of information with a drastically reduced number of spikes. In image processing, TTFS is a common temporal coding method and is mostly employed to encode gray pixels into spikes using a sigmoid function as follows :

$$T(p) = \frac{T_{max}}{1 + exp(-\sigma(128 - p)} \tag{2.19}$$

Where p is the current pixel intensity being encoded and σ is a non-linear coding variable set to 0.05. T_{max} is the maximum firing time. This method is often referred as latency coding [65] since lower values are encoded to late spikes and higher values to early spikes. Note that a positive σ as the affect of reversing the function and thus reverses the timing of spikes. Rank-order coding [66] is another type of temporal coding that does not attribute any importance to the exact firing time but to the order of the arrival of spikes. Although

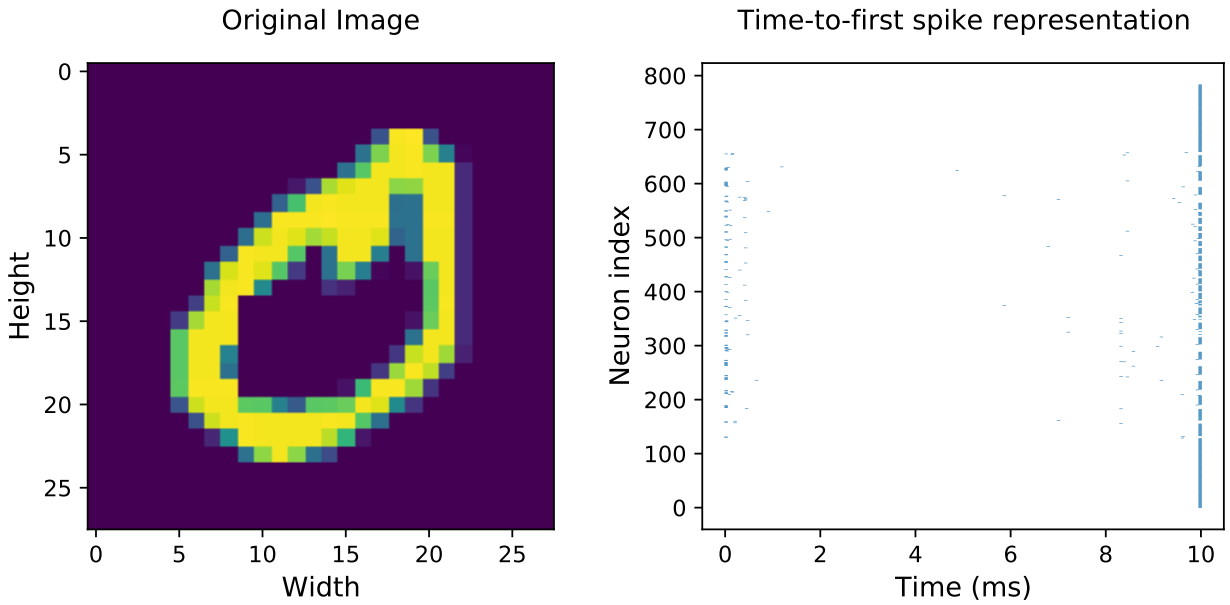

Figure 2.9: *Example of a temporal coding method over an image taken from the MNIST dataset.*

temporal coding provides a good alternative to rate coding by fixing some of its limitations, it can suffer from noise sensitivity. Indeed, even very small temporal latencies can alter the information delivered by the spikes issued from the coding method. An example of TTFS encoding on an MNIST image is shown in Fig. 2.9.

Population coding

Compared to rate and temporal coding, population coding relies on multiple neurons to encode a value. The representation of that value is thus shared amongst a group of neurons and not given by only one individual neuron. In a population coding scheme, neurons are responsible for different representation of the input data which makes this type of coding more complex than temporal or rate coding. A common technique to perform population coding is the use of overlapping Gaussian receptive fields in which a neuron corresponds to a Gaussian distribution as seen in Fig. 2.10. This method is often referred as Rank order population coding and is an extension of the regular Rank-order coding discussed above. Hence, if a set of N neurons is chosen to encode a variable x, the Gaussian receptive field of a neuron i is defined by its width σ_i and its center μ_i as :

$$\sigma_i = \frac{1}{\beta} \cdot \frac{I^x_{max} - I^x_{min}}{N - 2} \tag{2.20}$$

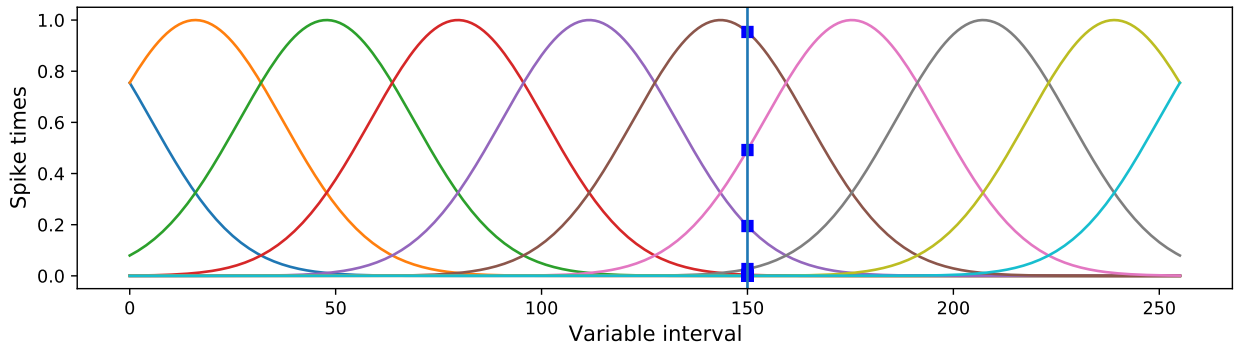

Figure 2.10: *Rank-order population coding with Gaussian Receptive Fields. The value $x = 150$ is encoded by 10 neurons with the spike times represented by the blue squares intersecting each of the distributions.*

$$\mu_i = I^x_{min} + \frac{2i-3}{2} \cdot \frac{I^x_{max} - I^x_{min}}{N-2} \tag{2.21}$$

Where $[I^x_{min}, I^x_{max}]$ is the domain of the variable to encode. β is a parameter that controls the width of the receptive field and respects $1 \leq \beta \leq 2$.

2.3.3 Synapses and learning process

Just like in a biological brain, neurons in SNNs are connected by synapses that act as information transmitters. They are core components of SNNs topology and their main role is to carry and regulate spikes sent from a neuron to another. Strengthening or weakening the influence of an input to an output neuron is done by the presence of a synaptic weight w that helps to increase or decrease the voltage of a neuron. The strongest the synaptic weight is, the highest the voltage of transmitted spikes to the output neuron will be, leading to the emission of post-synaptic spikes. If w is weak, the contrary effect happens and as the voltage of transmitted spikes is low, the output neuron does not fire.

The changes of synaptic weights is called synaptic plasticity and is at the core of learning processes. It can induce short or long term effects. Typically, short-term plasticity defines a rapidly induced and non persistent synaptic strength modulation happening on a millisecond scale that mostly occurs when only the pre-synaptic neuron is active. On the contrary, long-term plasticity induces a persistent weight change that can last minutes or hours and is triggered by pre- and post-synaptic neurons' mutual activities. Inducing long-term potentiation (LTP) or depression (LTD) is highly related to the notion of learning and led to the formulation of learning processes like the well-known STDP [67] rule.

Spike-timing dependent plasticity

As mentioned in Section 1.1, one of the first learning rule introduced was the Hebbian learning rule that aims to increase the synaptic weight of interconnected neurons that fire at the same time. The STDP is probably the most popular implementation of this rule. Following STDP consists in inducing LTP on a synapse if a pre-synaptic neuron fired before a post-synaptic one. Reversely, the synapse follows a LTD if the pre-synaptic neuron fired after the post-synaptic one. The synaptic weight changes between a pre-synaptic neuron i and its post-synaptic neuron j in STDP is described as :

$$\Delta w_j = \sum_{m=1}^{N} \sum_{n=1}^{N} W(t_j^n - t_i^m) \tag{2.22}$$

Where t_i^m and t_j^n are the firing times of pre-synaptic and post-synaptic neurons respectively. $W(x)$ is a function that dictates the order in which weights are decreased or increased depending on the synchronization of pre and post-synaptic neurons. This function is often defined as :

$$W(x) = \begin{cases} A_+ exp(-\frac{x}{\tau_+}) & x > 0 \\ A_- exp(\frac{x}{\tau_-}) & \text{otherwise.} \end{cases} \tag{2.23}$$

The parameters A_+ and A_- are constants that define the amplitude of the weight changes. Their initialization depends on the value given to the initial synaptic weights. τ_+ and τ_- are time constants that drive the exponential weight change decay, they are often initialized as $\tau_+, \tau_- = 10ms$. Fig. 2.11 depicts the evolution of synaptic weights as a function of the pre- and post-synaptic spike timings.

Competitive learning

In competitive learning, neurons compete with each other to respond to specific input stimuli. By getting the right to fire when the same inputs are presented and preventing other neurons to fire, these winner neurons become more sensitive to inputs with the same characteristics and can thus be used for classification purposes. This process relies on a biological mechanism called inhibition.

Inhibition is the process of restricting neuronal activity and is mostly achieved by inhibitory neurons that release neurotransmitters like the gamma amino butyric acid (GABA). The role of these neurons is to hyper-polarize post-synaptic neurons and thus reduce their membrane potential in order to prevent them from firing. This hyper-polarized

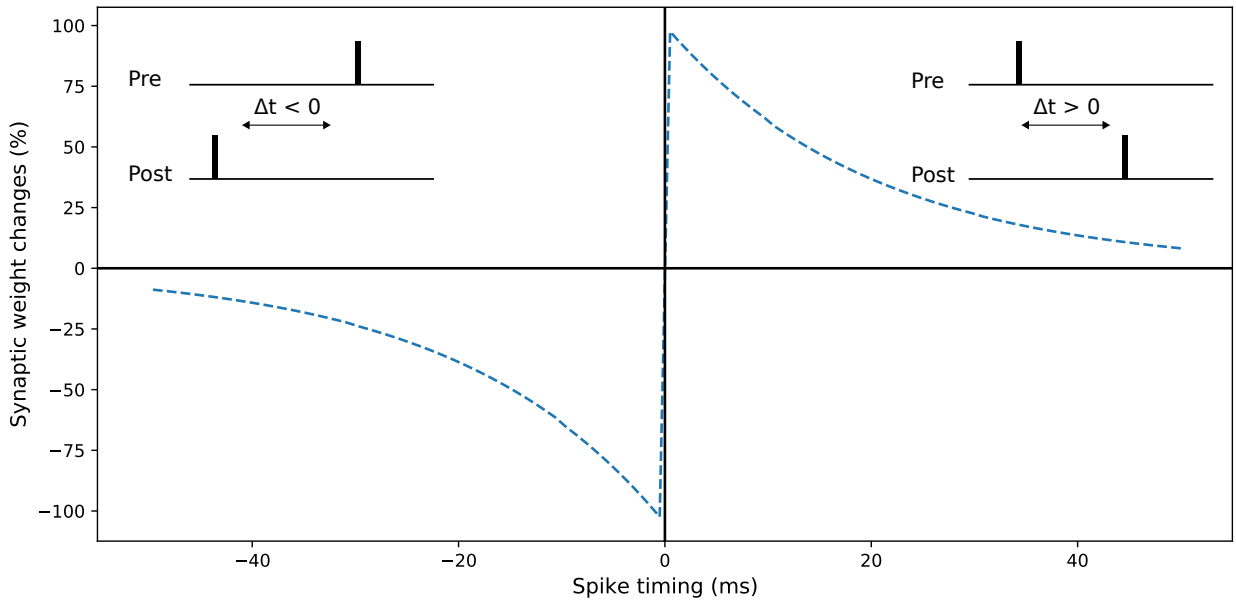

Figure 2.11: *STDP learning window adapted from Bi and Poo [1].*

potential is called inhibitory post-synaptic potential (IPSP). Inhibition thus allows to control the amount of excitation and regulates the transmission of information done by excitatory neurons. In opposition to inhibitory neurons, excitatory neurons release neurotransmitters like glutamic acid or the AMPA that open sodium ions in post-synaptic neurons which in terms eases their depolarization and leads them to fire. This is referred as the excitatory post-synaptic potential (EPSP). The distribution of inhibitory neurons in the mammalian cortex is of about 20 % leaving the last 80 % neurons to account for excitation.

A well-known implementation of competitive learning based on inhibition is the WTA strategy. In a WTA competition, when a neuron is allowed to fire, all other neurons are kept silent. In a network of neurons following the WTA principle, that means that a particular class of input should trigger one output neuron only.

Homeostasis

In biology, homeostasis is a phenomenon that enables cells and organisms to maintain and control the balance and stability they need to effectively operate. A simple example of homeostasis happening in the human body is the regulation of blood pressure that prevents the organs from failing. In the context of neuronal activity in SNNs, homeostasis plays a crucial role as it aims to prevent excessive firing from the same neurons by controlling their synaptic strengths or modulating their intrinsic excitability. When performing learning with STDP, the continuous stimulation of a post-synaptic neuron by a pre-synaptic neuron

induces a lasting growth in their synaptic strengths and leads to an increase of the post-synaptic firing rate. This enhances the correlation between pre- and post-synaptic activity which promotes LTP from all pre-synaptic input in an unrestricted positive feedback cycle. In terms this has the effect of making synapse-specificity disappear. Indeed, without bounds, this process will infinitely increase the synaptic strength and make some neurons fire significantly more than others what also makes the stability of the learning process difficult to maintain. Homeostasis mechanisms are thus necessary to counteract this behaviour.

Synaptic scaling and intrinsic plasticity are amongst the most widely studied homeostasis mechanisms [68] observed in biology. Synaptic scaling is an adaptive mechanism that neurons can use to ensure stability in synaptic strengths. It aims to adjust the weights of a neuron's excitatory synapses to stabilize firing. A biologically plausible implementation of such mechanism can be obtained by applying a scaling factor to every synapse as :

$$\frac{dw_i}{dt} = \alpha \cdot w_i(R_{target} - R) \tag{2.24}$$

Where α is the synaptic strength scaling factor, R is the real post-synaptic firing rate and R_{target} is the target firing rate of the post-synaptic neuron. While synaptic scaling modulates the neuronal activity at the synaptic level, the intrinsic plasticity mechanism aims to do it at the neuron level by modifying neurons' electrical properties in order to alter their excitability. In the work of Zhang and Li [69], an intrinsic plasticity rule is proposed and applied to the LIF model to maximize the entropy of each neuron's output firing rate distribution. To do so, the output firing rate distribution was tuned to a specified exponential distribution. Their online intrinsic plasticity rule was thus defined by :

$$R_m = R_m + \eta_1 \frac{2R\tau_m v_\theta - w - v_\theta - \frac{1}{R_{target}}\tau_m V_\theta R^2}{R_m w} \tag{2.25}$$

$$\tau_m = \tau_m + \eta_2 \frac{\tau_{ref} R - 1 - \frac{1}{R_{target}}(\tau_{ref} R^2 - R)}{\tau_m} \tag{2.26}$$

$$w = \frac{v_\theta}{e^{(\frac{1}{\tau_m}(\frac{1}{R}^{-\tau_{ref}}))} - 1} \tag{2.27}$$

Where R_m and τ_m are the membrane resistance and the time constant of the membrane potential respectively. η_1 and η_2 are learning rates. v_θ is the threshold of the membrane, τ_{ref} is the refractory period of the neuron. R and R_{target} are the real output firing rate and

the average expected firing rate respectively.

2.4 Spike-based Image Processing

Although ANNs have been able to tackle a large amount of visual problems and proved to outperform most methods used before their ascension, they still bring several issues that need to be addressed. Indeed, being far away from the brain's computation performances, they suffer from high memory and energy cost and induce a long training period consequent to their great level of performance. Moreover, their outputs lack explainability. Their usage is thus limited to domains in which high computational resources are available and where relying on abstract decision making is not a problem.

In fact, if ANNs were originally inspired by biology, the way they function and are taught to respond to specific stimuli greatly differs from neural computation. Researches in computational neurosciences highlighted the ability of SNNs to overcome the limits of ANNs. To recover biological-plausibility, SNNs carry information within their units by the means of spike timing, rates and latencies. This difference makes the computation of SNNs faster and allow easy hardware implementations. However, relying on spikes to carry information induces the use of different learning algorithms since methods like backpropagation cannot directly be applied to SNNs since the neuron's transfer function is non-differentiable.

Even though SNNs did not prove to be more accurate than ANNs, the theory on their computational efficiency suggests that they could however outperform them. In the state-of-the-art, the transition between ANNs and SNNs computation appears to happen by gradually removing levels of abstraction, starting by a full model conversion, from the spike-based implementation of learning algorithms to the implementation of fully biologically-plausible networks. Motivated by the speed of biological vision in decision making and the aforementioned statements, the development of SNNs for computer vision offers new promising opportunities in terms of computational efficiency and performances. In the next section, we discuss biological vision and perception by introducing methods to model receptive fields of the visual cortex.

2.4.1 Biological vision and perception

In the case of human perception, visual stimuli are captured by the retina, encoded and transmitted to the brain using the optic nerve. Connected to the thalamus, the optic nerve relays the information to the Lateral Geniculate Nucleus (LGN) that transmits it further to the visual cortex where visual processing happens. This cortex is divided in several visual

areas. The beginning of perception in the visual cortex happens in the area called primary visual cortex or V1. The two-streams hypothesis of visual processing [70] discusses the presence of two distinct systems in the visual cortex. When an individual captures visual information, these streams carry it and aim to perform different tasks. The ventral stream, consisting of V1 and areas of the extrastriate cortex (V2, V4, inferior temporal cortex (IT)) carries information inherent to object and space recognition. The purpose of V1 cells is not only to encode a visual stimulus to carry information to other areas in the cortex but also to provide a good representation of the stimulus. In those terms, these cells are the first interpreters of the visual stimulus and excel in object and pattern recognition which motivates the development of SNNs for computer vision.

Model of a retinal receptive field

Retinal receptive fields are visual space areas that are responsible for the modification of the firing patterns of retinal ganglion cells in response to specific stimuli. These receptive fields were experimentally found to have center-surround structures. Dividing them into two types, namely On-center/off-surround if the central region is excitatory and Off-center/On-surround if it is inhibitory. "On" receptive regions get excited when exposed to high light intensity and "Off" regions are stimulated by darker stimuli. These two types of receptive fields thus transmit different representations of the input stimulus as illustrated in Fig. 2.12. These cells are believed to manage the contrast range of visual stimuli and their shape was often modeled as a Difference-of-Gaussians (DoG). For 2-dimensional Gaussian functions, the DoG is expressed by the following :

$$G_{\sigma_n}(x,y) = \frac{1}{\sqrt{2\pi\sigma_n^2}} e^{\left(-\frac{x^2+y^2}{2\sigma_n^2}\right)} \tag{2.28}$$

$$DoG = G_{\sigma_1} - G_{\sigma_2} \tag{2.29}$$

In image processing, DoG kernels can be used to create filters in order to apply the transformation induced by center-surround cells. The shape of this filter is illustrated in Fig. 2.13.

Model of a V1 cell receptive field

After being processed by the retina and transmitted through the LGN, a visual stimulus arrives to V1 cells. Simple-cells in V1 have an orientation selection property, meaning that

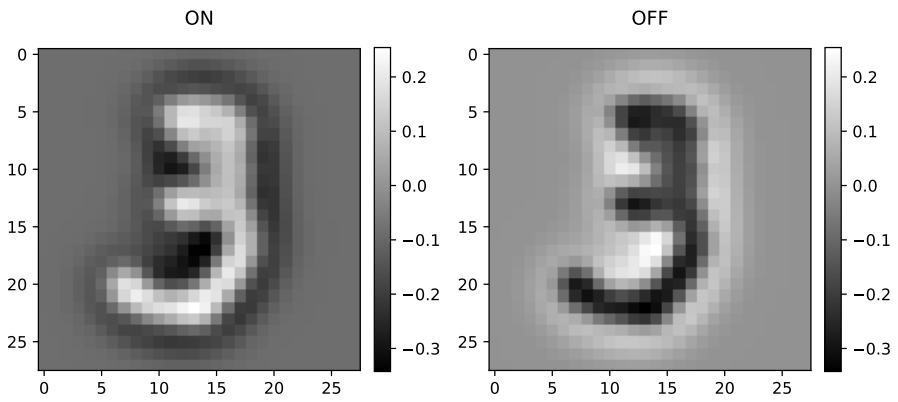

Figure 2.12: *ON and OFF receptive fields responses to an image from the MNIST dataset.*

they are exclusively sensitive to variations in spatial intensity along a specific orientation [63]. It was shown that the receptive field of these cells could be modeled by Gabor filters which are simply Gaussian functions modulated by sinusoids [71, 72]. These receptive fields were also found to take place in quadrature pairs, implying that the orientation of adjacent cells is 90 degrees out of phase. A 2D Gabor filter is thus defined as :

$$G(x, y) = exp\left(-\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2}\right) \cos\left(2\pi\frac{x'}{\lambda} + \phi\right) \tag{2.30}$$

With θ the orientation of the filter, λ the wavelength of the sinusoidal factor, σ the standard deviation of the Gaussian function, γ the spatial aspect ratio, ϕ the phase offset, $x' = x\cos\theta + y\sin\theta$ and $y' = -x\sin\theta + y\cos\theta$. Filter banks of Gabor filters (See Fig. 2.14) are often used as local feature extraction methods.

2.4.2 Pulse-coupled neural networks

Pulse-Coupled Neural Networks are biologically inspired models that mimic the dynamics of cortical neurons. Firstly introduced in the work of Eckhorn et al. [73], the PCNN was proposed as a model of a cat visual cortex and aimed to perform fast image processing. The model has been successfully explored for its application to segmentation [74] and detection tasks [75]. Compared to Deep Learning methods, a PCNN does not include a weight learning process but relies on constant synaptic weights instead. This lack of a training step, makes it particularly appealing for applications that prioritize quick output production and allows to solve a variety of vision task without the need for several models with different structures.

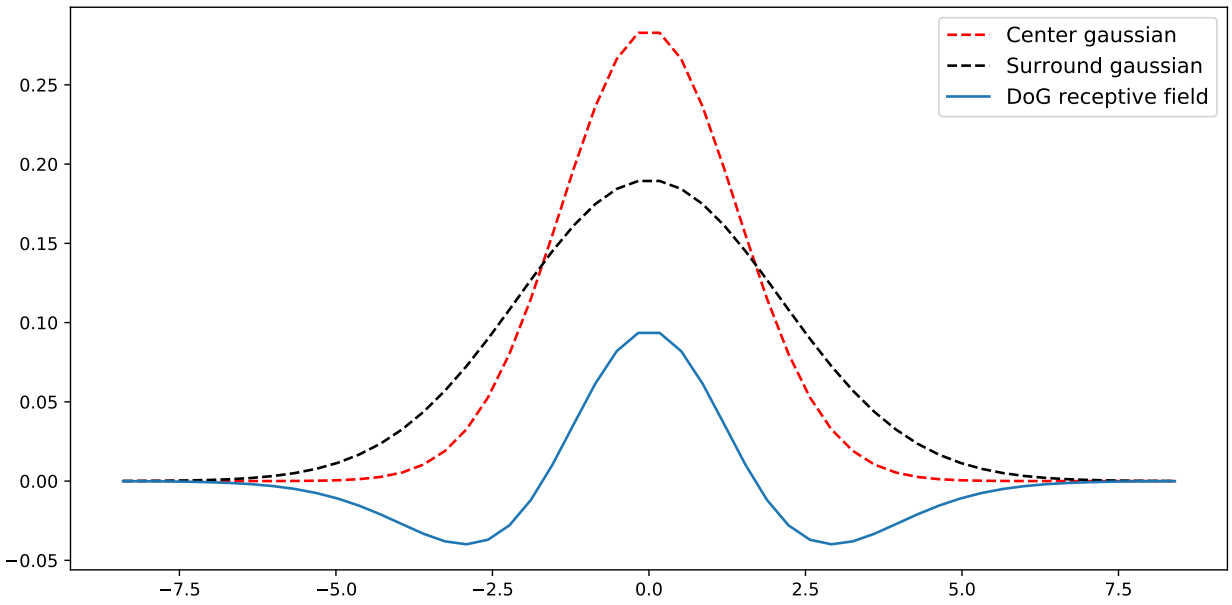

Figure 2.13: *The Difference-of-Gaussians.*

In a standard PCNN model, neurons are laterally connected and match pixels from the input image. The intensity of these pixels being the major drive for each neuron's membrane potential integration, the neuronal activity of the PCNN allows the creation of clusters of pixels with similar intensity. This behaviour makes this model a perfect fit for segmentation processes. Regarding its structure, a PCNN neuron is composed of three main parts namely, the feeding channel, the linking channel and the pulse generator. The feeding channel is the start of the neuron's computation and is responsible for the reception of two inputs, the feeding input F_{ij} and the linking input L_{ij}. Both receive a grayscale normalized pixel value at location (i, j) as a stimulus S_{ij}. Since each neuron in a PCNN computes iteratively, the feeding channel is described by the following equations at the n^{th} iteration :

$$F_{ij}[n] = \exp^{-\alpha_f} F_{ij}[n-1] + V_F \sum_{k,l} M_{ij,k,l} Y_{ij}[n-1] + S_{ij} \tag{2.31}$$

$$L_{ij}[n] = \exp^{-\alpha_L} L_{ij}[n-1] + V_L \sum_{k,l} W_{ij,k,l} Y_{ij}[n-1] \tag{2.32}$$

Where M and W are the constant synaptic weight matrices and are set as kernels of a Gaussian filter. $Y_{ij}[n-1]$ is a binary value corresponding to the previous output pulse of the neuron and (k, l) refers to neighboring neurons. The voltage potential and the attenuation time constants of the feeding and linking channels are V_F, α_F and V_L, α_L respectively.

The feeding and linking inputs are then merged in the linking channel part of the neuron

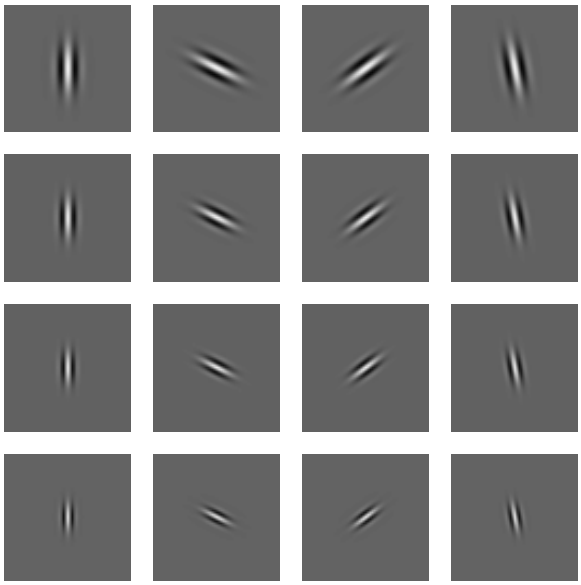

Figure 2.14: *Gabor filter bank.*

to compute the internal neuronal activity U_{ij} as :

$$U_{ij}[n] = F_{ij}[n](1 + \beta L_{ij}[n]) \tag{2.33}$$

Where β is the positive linking coefficient. In the pulse generator, the neuron is finally triggered if the internal activity U_{ij} is greater than the dynamic threshold θ_{ij} as follows :

$$Y_{ij}[n] = \begin{cases} 1, & U_{ij}[n] > \theta_{ij}[n] \\ 0, & \text{otherwise} \end{cases} \tag{2.34}$$

$$\theta_{ij}[n] = \exp^{-\alpha_\theta}\theta_{ij}[n-1] + V_\theta Y_{ij}[n] \tag{2.35}$$

Where the time attenuation constant and the voltage potential of the threshold are referred as α_θ and V_θ respectively. The overall structure of a PCNN neuron is shown in Fig. 2.15.

While this simple PCNN proved its efficiency as an image processing tool [76], the large number of parameters it embarks makes it tedious to use and optimize. Some attempts to reduce the number of parameters in the network were then done to accelerate the computation of each neuron dynamics and reduce the overall complexity of the model. Such models are discussed in more details in Chapter 4 and used to build a tumor recognition system.

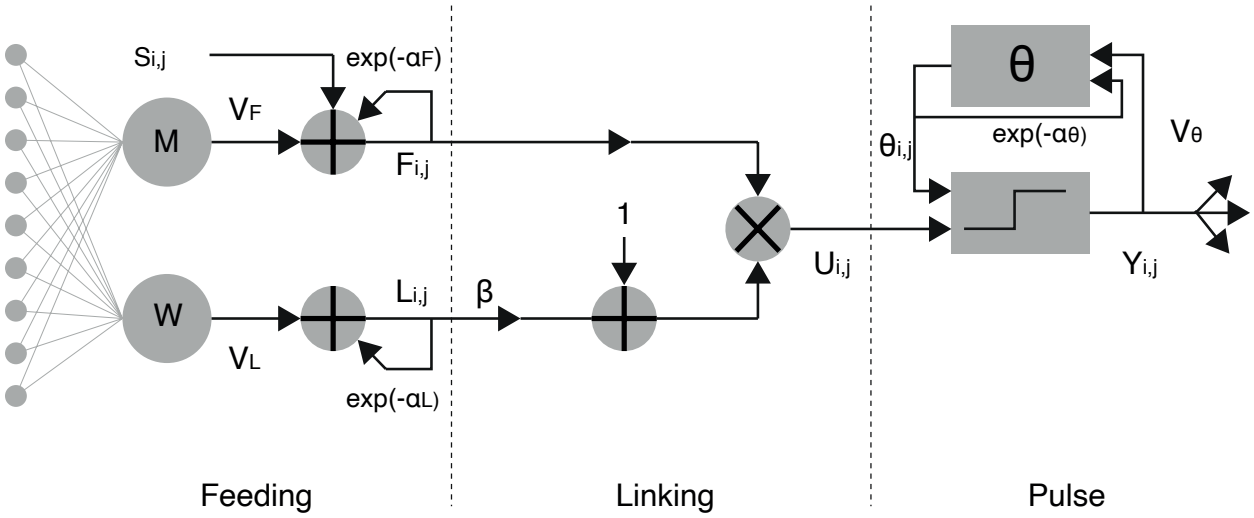

Figure 2.15: *A PCNN neuron model.*

2.4.3 Spiking Neural Networks for computer vision

In the aim to have a better understanding of biological perception and seek for computational efficiency, SNNs are being highly investigated in computer vision. Several studies have already proved that SNNs could be successfully used for pattern recognition [64, 77], what motivated the exploration of the transition between ANNs and SNNs. If the use of ANNs is still dominant in the state-of-the-art, SNNs offer several advantages that motivate further studies in computer vision.

Indeed, since they are able to be coupled with neuromorphic sensors, building spike-based recognition systems first comes with a drastic power consumption reduction [78] that deep learning based vision systems lack. Moreover, SNNs are more computationally efficient than ANNs and can provide easier hardware implementation. However, if the accuracy of SNNs in solving vision tasks, such as MNIST or CIFAR-10, tackled ANNs performances and gave intuition that spike-based models could take over artificial neural networks, dealing with complex data remains a challenge. In fact, the data encoding step required to perform image processing through SNNs can induce information loss or bad feature representations. Another major problem to consider when developing SNNs is the lack of training algorithms especially to perform supervised learning that most state-of-the-art recognition systems rely on. This is mostly due to the discontinuous nature of spikes and a lack of understanding of biological learning mechanisms that makes the design of such algorithms complicated. In an attempt to overcome these difficulties, many research works have emerged to transfer

the performances of ANNs to SNNs, to match commonly used learning algorithms to spike computation or develop fully biologically-plausible alternatives to ANNs.

The conversion of pre-trained ANNs to SNNs was proposed [79, 80] as a mean to make the inference benefit from the energy efficiency provided by SNNs. By using trained weights and replacing rate-coded neurons to spiking neurons, these methods however only provide a solution for inference hardware implementation. Hence, although they allow to base the decision making on spike-based computation, they do not permit to prove the efficiency of SNNs in computer vision and do not give any insight on the brain's computation when dealing with visual stimuli. Some adaptations to the backpropagation algorithm have also been investigated to tackle the lack of learning methods in SNNs [81]. If new spike-based learning rules are thus derived from ANNs learning algorithms, most of them do not search for biological-plausibility what still leaves some of the mentioned issues untackled.

Some further investigations to design SNNs and train them for computer vision using synaptic plasticity rules like the one introduced in Section 2.3.3 have thus been carried. The design of these SNNs, their learning methods, the data encoding schemes, and the neuron models employed vary depending on the visual task to perform. For image classification, the study of Diehl and Cook [64] introduced an SNN made of 2 layers. The first layer encodes images into spike trains using a rate coding scheme. The other layer is responsible for the network's processing and includes excitatory and inhibitory LIF neurons. Learning is made in an unsupervised way by STDP and the classification task is solved by analyzing the firing rate of each neuron regarding the class of which the presented pattern belonged to. For image segmentation and edge detection, Meftah et al. [82] proposed a 3 layers feedforward SNN. The first layer receives RGB values and transmits them to the second layer that encodes the values to temporal codes by the mean of Gaussian radial basis functions. The last layer outputs the class of the RGB pixel value presented to the network. As they did not use ground truth segmentation maps, the learning process was unsupervised and relied on a WTA learning rule that potentiated the synaptic weights between input neurons and firing output neurons. Another SNN architecture inspired by the primate visual cortex was introduced by Wu et al. [83, 84] for image segmentation and edge detection. Their network performed a color feature extraction using ON/OFF receptive fields and an error-backpropagation training method. A general SNN architecture for recognition tasks is illustrated in Fig. 2.16.

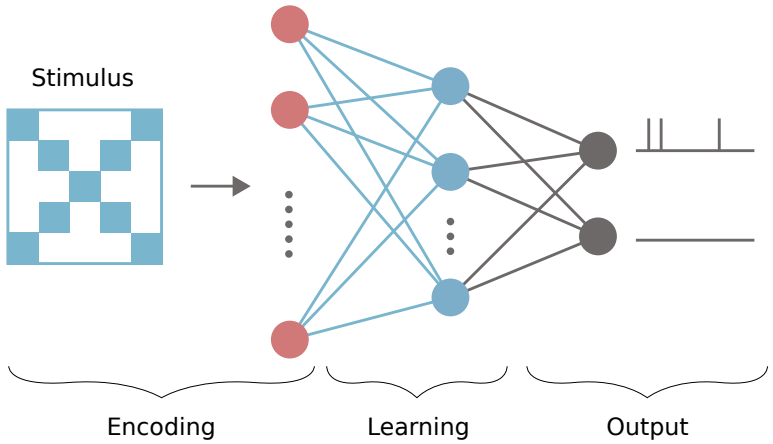

Figure 2.16: *A general Spiking Neural Network architecture.*

2.5 Conclusion

Medical image analysis has brought a particular interest in deep learning research over the last decade as it exposes challenges to improve the way we diagnose and treat illnesses. Building fully-automatic brain tumor diagnosis systems is one of the most difficult challenges brought by this attraction. Many studies have investigated the development of deep CNNs to solve the tumor recognition task, most of which obtained results that were comparable to the human accuracy or outperformed it.

Deep learning algorithms are, however, failing at several levels including energy and cost efficiency, interpretability, high dependence to large datasets that induces long training time and a dependence to powerful hardware. All of these issues limit their usage and large scale deployment. The emergence of low-cost and power efficient GPU embedded systems is thus encouraging deep learning research towards the compression of ANNs. The structures and operations of Convolutional Neural Networks, being the most commonly used deep learning methods in computer vision, thus need to be redefined to fit low-cost deployment requirements. Simultaneously, spike-based computation by the mean of SNNs is offering new opportunities that aim to address these problems. Being totally bio-inspired, these neural networks have the advantage to be particularly energy-efficient which makes their hardware implementation easier than their artificial counterparts. Since they carry information as neural spikes, they also provide fast processing and memory footprint reduction. However, while they appear to be very appealing as comparable candidates to conventional ANNs, training these methods to solve complex cognitive tasks and approach deep learning performances still remains a challenge. Even with the wide amount of

different SNN architectures proposed in the state-of-the-art, proving the efficiency of spike-based model for computer vision is yet to be done as the number of benchmark datasets used stays relatively small. Although it can highly benefits from the cost-efficiency and fast inference provided by spiking models, the brain tumor diagnosis task exposed in this thesis has also not been, to our knowledge, investigated in the spike computation domain.

The work detailed in this manuscript mainly focuses on the investigations of cost-efficient alternatives to Deep Learning approaches to build medical image analysis systems. The goal of this study is to move brain tumor visual diagnosis tasks to more affordable, fast and powerful systems to meet the requirements of the clinical field. The contribution in Chapter 3 provides a framework for the compression of CNNs in order to adapt a brain tumor segmentation method to devices with limited resources. Following, an investigation on the ability of Pulse-Coupled Neural Networks to perform glioblastoma segmentation and detection tasks is carried in Chapter 4. Finally, Chapter 5 proposes an attempt to perform the visual diagnosis of brain tumors by the mean of different SNNs trained using the unsupervised and supervised STDP learning rules.

Chapter 3

Deploying tumor diagnosis on cost-efficient embedded systems

3.1 Introduction

Brain tumor diagnosis relies on the analysis of data obtained by MRI and is often performed by one or several radiologists. The complexity and high variation in shapes and locations of certain types of tumor, however, produces challenges regarding their accurate segmentation. With the increase in the number of medical image data to process, healthcare providers are thus investigating the use of computer-aided analysis systems that would tackle this issue. However, to be useful in the medical field, these systems have to meet several requirements such as cost efficiency, automaticity or the ability to produce fast outputs. Indeed, with the aim to always decrease expanses, building computer-aided diagnosis system has to come with awareness in terms of energy consumption. This induce, using low-power hardware to compute the diagnosis as fast as possible.

As mentioned in the previous chapter, the rise of Deep Learning over the last decade induced the development of a wide range of models to assist the decision making in hospitals, specifically for diagnosis tasks. Mostly based on CNNs [12, 22], these solutions aim to meet most of the above requirements but their poor adaptation capability to devices with limited resources is preventing their full deployment and usability. Indeed, the efficiency of CNNs in computer vision is not new [85], they proved to be very powerful to solve recognition tasks by obtaining outstanding levels of accuracy. In the context of brain tumor detection, CNNs were tested and applied with success to perform MRI slice classification or segmentation.

Although they appeared to prove excellent results in detecting a large amount of different types of tumors, most of the models developed do not meet the cost efficiency requirement. In fact, in a run for better accuracy and performances, these models are growing bigger and so do their memory footprints. This leads to the use of very powerful GPU based hardware that comes with high cost in both purchase and energy consumption, which also makes Deep Learning have a meaningful carbon footprint. Hence, building smaller models or optimizing larger ones appears crucial to implement computer-aided diagnosis systems on limited computational resource devices [86] and step towards more profitable solutions.

Discussions about model compression thus emerged in an attempt to confront this situation as energy-efficiency is not only crucial in the medical field. These studies show an importance in the modification of existing deep neural networks or provide motivations to construct new light-weight models. In parallel, in the aim to provide GPU computing while dealing with cost and power efficiency, new embedded platforms designed as deep learning accelerators are emerging to offer new perspectives in the development of deep learning based applications.

In this chapter, we propose mechanisms to optimize an existing neural network architecture in order to perform a brain tumor segmentation task. To do so, we first discuss the advantages of a deep learning ready embedded system and its applicability in the development and deployment of CNNs in Section 3.2. Then, in Section 3.3 we review the use of CNNs compression methods in order to reduce the number of parameters in the U-Net [87] architecture. Finally, Section 3.4 provides details on the implementation of an end-to-end tumor segmentation system deployed on a cost-efficient embedded system and exposes the results of our compression methods while comparing the performances of the network to other state-of-the-art CNNs.

3.2 A GPU Embedded system for Deep Learning applications

The growth in the development of powerful embedded computing systems is leading to redefine the way neural networks are being conceived in order to fit requirements in energy-efficiency in domains like healthcare [88] or autonomous transport [89]. In the context of computer vision, research challenges appeared to discuss the transformation of these applications to build low-power solutions without sacrificing their performance in the tasks they are solving [90]. In fact, they highlight the necessity of a balance between accuracy and computational cost. Only with this balance would a deep learning solution be

considered applicable on a large scale. Taking advantage of these opportunities, NVIDIA launched the JAX developer kit [91], one of the most popular cost-efficient embedded system for machine learning. It comes with deep learning accelerators and aims to simplify the development of deep learning applications. It was specifically designed to run deep learning workloads as it embeds a powerful GPU with CUDA and cuDNN supports as well as common libraries for neural network implementations like Tensorflow and TensorRT.

	Jetson AGX Xavier Module
CPU	8-core ARM v8.2 64-bit CPU, 8MB L2 + 4MB L3
GPU	512-core Volta GPU with Tensor Cores
Memory	16GB 256-Bit LPDDR4x — 137GB/s
Weight	630 grams
Dimension	100mm x 87mm x 16mm
Storage	32GB eMMC 5.1

Table 3.1: *Nvidia Jetson AGX Xavier specifications.*

Taking advantage of both CPU and GPU embedded in the platform, the JAX also appears to be a powerful competitor to other embedded platform investigated for deep learning applications development such as Field Programmable Gate Arrays (FPGAs) [92]. There are several reasons that thus motivates moving deep learning workloads to the JAX :

1. Its low weight and dimension feature. Size and weight can be factors of choices for applications in small or flying embedded systems [93] as well as to improve the mobility of recognition system in computer vision. In the research field, it also provides research teams with the ability to process data anywhere at anytime or to try new models without being connected to powerful servers.

2. Its low power consumption. It allows to drastically reduce energy consumption and thermal problems inherent to a lot of systems with heavy workloads. Both power consumption and weight features can be seen in Table 3.1.

3. Its modular scalable architecture. The platform comes with a Deep Learning Accelerator (deep learningA), as seen in Fig. 3.1, which was specifically designed to simplify the deployment of deep learning applications and optimize the energy efficiency of the kit when running these applications. It also perfectly fits any requirement in terms of CNNs implementation as it supports common operations found in this type of network like convolution, pooling or normalization.

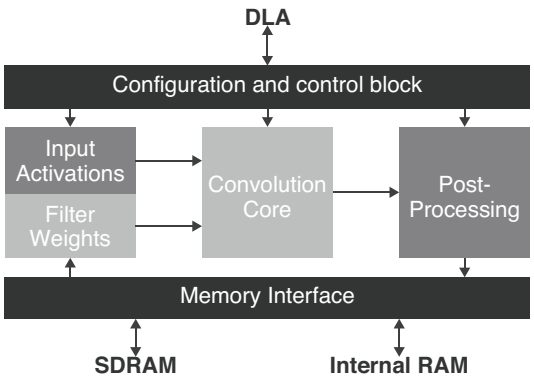

Figure 3.1: *The Nvidia Deep Learning Accelerator architecture. Nvdla.org.*

4. The JetPack SDK. JetPack embeds an Operating System image as well as all the developer tools and APIs necessary for Computer Vision, Accelerated Computing and Deep Learning. It also provides documentation and sample applications for testing.

5. Its available memory slot. The JAX comes with an empty memory slot ready to receive an NVMe SSD drive. This features allows the availability of a large storage unit to process and store data locally which makes it interesting when remote services are not available or when the platform lies in a closed network. Indeed, processing data directly on the platform gives the opportunity to not rely on a third-party server and therefore prevent any connection disruption, latency and security flaw when writing or fetching data.

However, regarding these listed advantages, some modifications in the conception of CNNs for computer vision have to be made to make the most of the JAX platform. Indeed, if the JAX provides a power efficient environment it does not perform compression on deep learning models. There is thus a need to redefine modern CNN architectures to make them usable on the embedded system.

3.3 Compressing the U-net architecture

While CNNs are regarded as one of the most powerful models being used in computer vision, they also often do not fit the requirements in a limited resources environment. In fact, convolutions are computationally expensive, especially when they are performed on a high number of feature maps for representation learning. Moreover, following the rush for better accuracy, these networks appear to be deeper over the years, what increase the number

of operations they perform on the data and the number and size of weight matrices to be modified during training. Hence, in order to be beneficial to a wider range of domains, the way we conceive CNN architecture has to evolve in the opposite direction to decrease the computational cost of computer vision systems while keeping them as powerful as possible. The U-Net architecture was design to perform medical image segmentation and the model can be seen as an autoencoder [94] with an encoding part that contracts an image to a feature vector and a decoding part that uses the feature maps learnt during the encoding process to expand the feature vector in order to create a segmented image. In this work, we propose a compressed version of this model by modifying some of its native operations.

3.3.1 Group Normalization

Normalization methods are widely used in deep learning to reduce the number of epochs necessary to train a deep neural network. It aims to standardize inputs, stabilizes the learning process, ease the optimization and convergence of ANNs [95]. Performing a global feature normalization on the batch dimension as in Eq. 3.1, Batch Normalization [96] (BN) is one of the most well-known of theses methods. It is also often used as a regularizer that prevents deep learning models from overfitting as a result of the stochastic uncertainty of the batch statistics.

$$\mu_i = \frac{1}{m}\sum_{k\in S_i} x_k, \qquad \sigma_i = \sqrt{\frac{1}{m}\sum_{k\in S_i}(x_k - \mu_i)^2 + \varepsilon} \tag{3.1}$$

Although BN was successfully used to optimize the training of ANNs, the method suffers from inaccuracy on batch statistics estimation with small sizes of batches. This phenomenon increases the model's error and prevents the use of BN on devices with limited resources as large size of batches are necessary to avoid it. It is thus crucial to adapt normalization methods to smaller batches in order to tackle this issue. Solving the batch statistics estimation on smaller batches has been attempted by several methods like the Batch Renormalization [97] and the Synchronized Batch Normalization [98]. While they performed better than or as well as BN, both methods do not, however, entirely respond to the computational problem exposed here.

In the last few years, another normalization method appeared and offer great opportunities for dealing with the problems induced by BN, it's the Group Normalization (GN) [99]. This new method aims to divide input channels in groups and perform the normalization within each of them. By doing so, the batch statistic computation is avoided and the method does not rely on the batch dimension, which makes it a good fit for ANNs deployed on

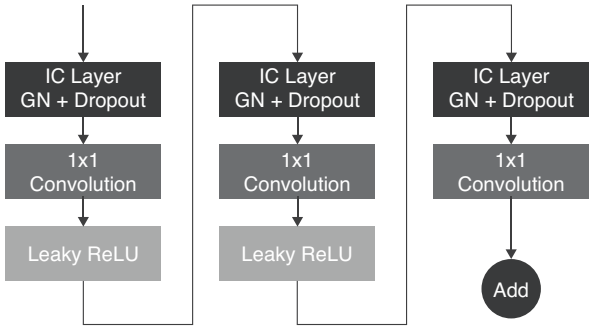

Figure 3.2: *Detailed convolution block replacing each convolution operation of the U-net model.*

resource-constrained systems.

With respect to the computation of μ and σ in Eq. 3.1, and with G being the number of groups to divide the input in, N the batch axis, C the number of input channels and C/G the number of channels per group, GN is defined as :

$$S_i = k|k_N = i_N, \lfloor \frac{k_C}{C/G} \rfloor = \lfloor \frac{i_C}{C/G} \rfloor \tag{3.2}$$

Following this definition, different values of G can lead to different types of known normalization. An Instance Normalization [100] is obtained with $G = C$ and if $G = 1$, a Layer Normalization [101] will be applied. Obtaining lower training error by replacing BN in the ResNet-50 network [102], GN gives confidence to train neural networks without relying on large batches and thus solve the memory contraints imposed by other normalization methods. In this work, GN was used in Independent-Component (IC) layers [103] as shown in Fig. 3.2. Each of these layers combines Dropout [104] and GN as a regularization method and optimize training by reducing the correlation between pairs of neurons.

3.3.2 Depthwise Separable Convolution

As convolutions are the most computationally expensive operations in the network, our first compression task is to replace them by a cost-efficient process while keeping their powerful feature extraction capacities. Depthwise separable convolutions (DSC) [105] were developed in order to enhance AlexNet and seek for convergence speed, accuracy and computational cost gains. This type of convolution was also used in different architectures like Xception [106], MobileNets [13] or Inception [96] to decrease their size. The aim of DSC is to factorize convolutions and decompose them in two different operations namely,

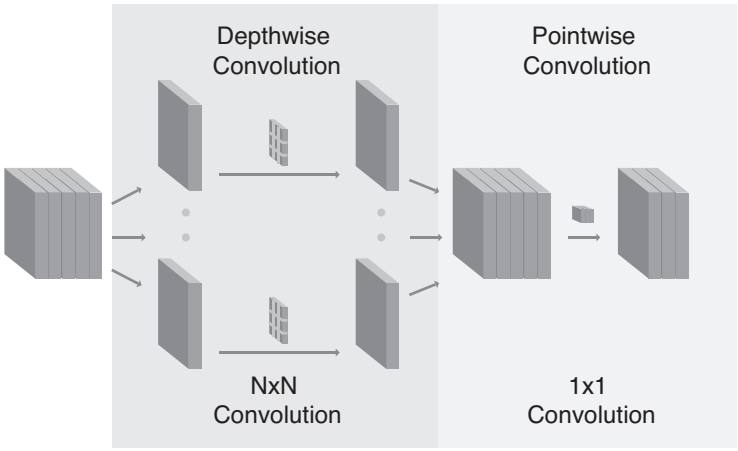

Figure 3.3: *Separable convolution block.*

the depthwise convolution and the pointwise convolution as shown in Fig. 3.3. In the depthwise operation, the convolution kernel is applied on each input channel individually. The pointwise convolution is a regular convolution with a kernel of 1x1 that merges the features built by the depthwise part. This process breaks down the input and kernel dimensions and drastically reduces the computational cost of the convolution. Indeed, assuming an input of shape (W, H, D_{in}), a convolution kernel of size K and an output dimension D_{out}, the computational cost λ and the number of trainable parameters ω for a regular convolution layer is described as :

$$\lambda = W * H * D_{in} * K^2 * D_{out} \tag{3.3}$$

$$\omega = D_{in} * K^2 * D_{out} \tag{3.4}$$

In contrast, performing the convolution on independent depth gives a DSC layer the following cost and parameters :

$$\lambda = W * H * D_{in} * (K^2 + D_{out}) \tag{3.5}$$

$$\omega = D_{in} * (K^2 + D_{out}) \tag{3.6}$$

The excellent results of DSC in terms of model size and computational cost reduction motivated the integration of this type of convolution in modern CNN architectures like MobileNet [13] and ShuffleNet [107]. Both of the latters achieved comptetitve results in

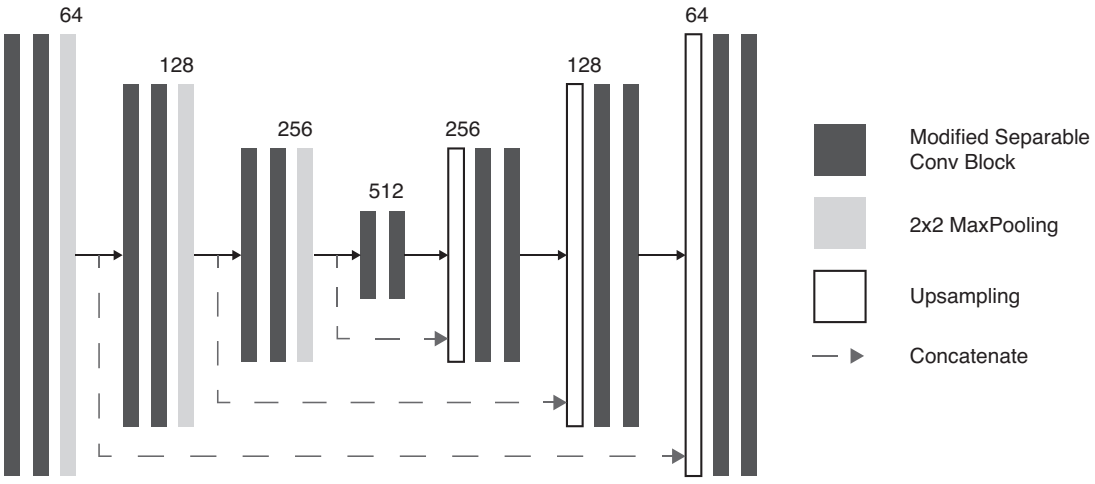

Figure 3.4: *The compressed U-net architecture.*

computer vision tasks like ImageNet [15]. Encouraged by these advances, we thus applied DSC on the U-net architecture in both encoder and decoder part. In this modified architecture, every convolution layer is replaced by DSC convolution blocks like the ones used in MobileNetV2 [108]. Hence, each of the DSC block used in the compressed architecture is composed of a 1x1 convolution followed by the depthwise and the pointwise convolution as previously seen in Fig. 3.2. Each convolution except for the pointwise is activated by a LeakyReLU function and regularization is performed in each layer using the IC component described above. To aim for more size reduction, two addition modifications are made in the encoder and decoder part of the network. In the encoder, we set the maximum feature maps number in the bottleneck to 512, drastically reducing the computational cost of the entire network. In the decoder part, Bilinear Interpolation [109, 110] was used for upsampling. The obtained compressed U-net can be seen in Fig. 3.4.

3.3.3 Model quantization

To go further in the compression scheme and respond to the computational cost problems induced by CNNs, quantization can also be used to reduce the memory footprint of the model. It refers to methods used for reducing the precision of computations and values stored by the model. One of the most popular type of quantization in Deep Learning is the Int8 Quantization that uses 8-bit integers instead of floating points values and operations in the network. This had the effect of reducing the overall computational cost of the network and its memory requirements. If quantization implies the presence of errors, the resilience

of neural networks to noise does not effect their computations as the changes in weight matrices would be too small to affect their performances. Neural network quantization [111, 112] was thus used for inference in the proposed compressed U-net without any penalty on prediction accuracy. The network uses a post-training 8-bit quantization that has the effect of accelerating predictions and reduce segmentation time.

3.4 Experiments and results

3.4.1 Dataset

Each of the experiment carried to prove the efficiency of the compression applied to U-net were run using the 2015 version of the BraTS dataset mentioned in Section 2.2.3. Compared to more recent versions, the 2015 dataset contains a training set with 2 different classes of tumors namely high-grade glioma (HGG) and low-grade glioma (LGG). We used a set of 220 cases of HGG and 54 cases of LGG. For each case, a segmentation is provided by corresponding 4 different types of brain tissues to 5 numerical labels as follows : necrosis (label 1), edema (label 2), non-enhancing tumor (label 3), enhancing tumor (label 4) and non-tumorous tissue (label 0). For the purpose of the segmentation performance quantification these structures form 3 different groups, namely the complete tumor which contain labels from 1 to 4, the enhancing tumor that is only composed of label 4 and the tumor core which is composed of all labels except label 2. Fig. 3.5 shows a sample of 3 cases taken from the along with their attributed ground truth map.

Before being used by the network, the entire dataset went through a pre-processing pipeline. First, sub-datasets for training and validation were created with a ratio of 0.8 / 0.2 respectively. Both contains scan from HGG and LGG cases. Then, N4ITK bias field correction [113] was applied to both T1 and T1C sequences using the SimpleITK library [114]. After this correction, normalisation was also applied to the entire dataset. To be able to fit and process the data on the JAX, empty early and late slices were removed for each case, the remaining slices were then cropped down to 128x128 images avoiding computation in background areas. To prevent the model from over-fitting and improve generalisation, an augmentation process then follows by picking 40% of the cases and randomly applying different variation operations like flipping, shifting or elastic transformation. Finally, every case is encoded and saved individually into a TFrecord file. This format uses the Protocol Buffer Format and allows to save memory and reading time, two important properties that allows the storage and the analysis of the data on the JAX platform. The entire data pipeline is illustrated in Fig. 3.6.

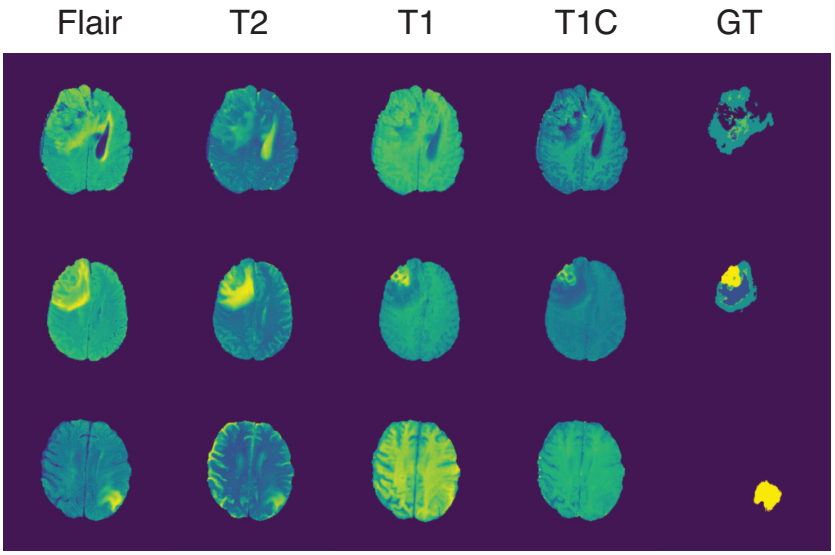

Figure 3.5: *Four sequences and ground truth map of three cases taken from the BraTS 2015 dataset. The first row is a case of LGG and the next 2 rows are HGG cases.*

3.4.2 Experimental protocol

The experiments consisted in training a neural network to perform a brain tumor segmentation task on a cost-efficient platform. This network is an optimized version of the U-net architecture obtained by reducing its depth and modifying convolution and normalization operations as well as using quantization to speed up inference. This entire experimental protocol was implemented using the Tensorflow framework and used the full potential of the JAX during both training and inference by setting its configuration the the MAXN mode. This mode brings increased GPU Frequency, the use of all CPU Core available and lead to speed up training by reducing each epoch time to an average of 20 minutes. Each of the modes stated in Table 3.2 were also tested and proved the efficiency of our compression method to run training and inference successfully on the platform. [115].

The input of the network is a mini-batch composed of 2 scan slices to prevent the JAX from running out of memory during training. The size of the batches was motivated by the replacement of BN layers by Group Normalization. In the intermediate layers, the weight kernels were initialised using the glorot normal initialisation [116] and biases were all set to zeros. We trained the network using a 0.0001 learning rate with an Adam Optimizer [117]. To record the performance of the network during training and lead the weight changes, we used a combination of the Dice loss function and the Binary Cross-Entropy

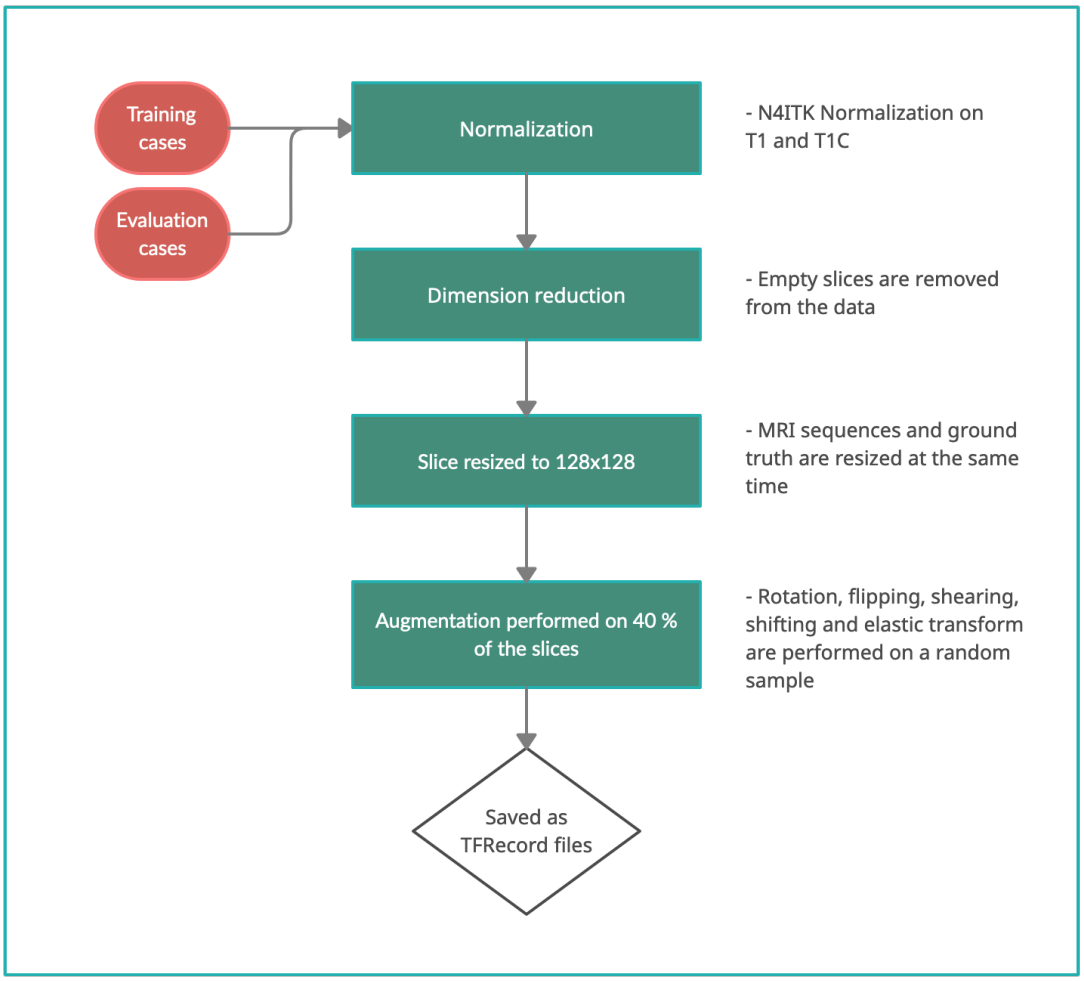

Figure 3.6: *Data pipeline applied to the BraTS 2015 dataset before getting fed to the compressed U-Net.*

(BCE) as follows :

$$DiceLoss(y, \hat{y}) = 1 - \frac{2\sum y \cdot \hat{y}}{\sum y + \hat{y}} \tag{3.7}$$

$$BCE(y, \hat{y}) = -\frac{1}{N}\sum_{i=1}^{N}[y_i log(\hat{y}_i) + (1 - y_i) log(1 - \hat{y}_i)] \tag{3.8}$$

$$TrainingLoss(y, \hat{y}) = BCE(y, \hat{y}) + DiceLoss(y, \hat{y}) \tag{3.9}$$

Where $\hat{y}$ and y are the model predictions and the ground truth respectively and N is the number of cases in the dataset. Recording the loss at each iteration, an early stopping

	MAXN	10W	15W	30W_ALL
CPU Cores	8	2	4	8
CPU Max Freq. (MHz)	1377	520	6700	900
GPU Max Freq. (MHz)	2265.6	1200	1200	1200
Memory Max Freq. (MHz)	2133.6	1066	1333	1600

Table 3.2: *Nvidia Jetson AGX Xavier power modes.*

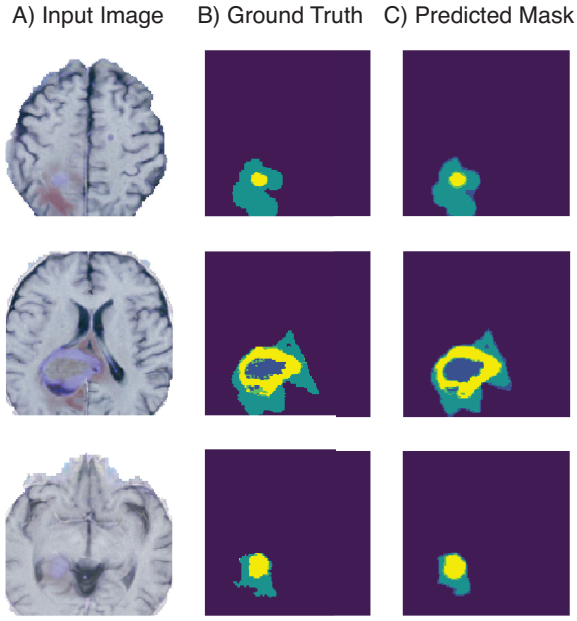

Figure 3.7: *Results of the compress U-Net taking a 128x128x4 input slice in A). B) is the ground truth attached to the input and C) the predicted segmentation map.*

scheme was also used to prevent over-fitting and was set to stop the training after 2 iterations if the network error was not decreasing. Early stopping also allows not to set the number of training epoch accurately.

Finally, a sigmoid activation sets the output of the network to a pixel-wise probabilistic mask wherein pixel values are in the range [0, 1]. A threshold is then applied to the mask as a post-processing operation to obtain a prediction mask composed by the original label values.

3.4.3 Results

The compressed U-Net proposed in this work was successfully trained to segment both HGG and LGG cases from the BraTS dataset. The compression method allowed us to obtain a

parameter reduction of about 93 % with a total number of roughly 2.1 millions parameters compared to the 31 millions parameters of the original U-Net model. This considerable reduction was key to deploy the model on the JAX as it highly reduces its memory footprint. To prove the similarity between the ground truth and the model's predictions, and evaluate the segmentation performances on each of the brain tissue groups described in Section 3.4.1, we computed the Dice Coefficient on the validation dataset as follows :

$$DiceCoefficient = \frac{2|y \cap \hat{y}|}{|y| + |\hat{y}|} \tag{3.10}$$

As seen in the results showed in Table 3.3 the reduction process did not cause important penalties to the overall performances of the network. Although our method obtained lower segmentation scores compared to other methods in Table 3.3, it still approaches these performances and stay comparable, especially regarding the high parameter reduction achieved. An example of the results obtain by our network is shown is Fig. 3.7.

	Dice Score		
Method	Complete	Core	Enhancing
Proposed [118]	**0.81**	**0.77**	**0.53**
Zhao [119]	0.82	0.72	0.62
Pereira [120]	0.84	0.72	0.62
Dong [12]	0.86	0.86	0.65

Table 3.3: *Results of our proposed approach compared to other similar Deep Learning based brain tumor segmentation methods.*

While the methods proposed in this chapter to compress a deep CNN permitted significant parameter reduction and the implementation of a successful light weight MRI segmentation system, more investigations could be carried to build an even lighter model without performance penalty. Knowledge Distilliation [121] is one of these methods and direct the training of a small model to copy a pre-trained larger model. It is done by minimizing a loss function that aims to learn the behavior of the larger model by trying to reproduce its outputs. Parameter Pruning is another popular type of model compression method and is used to avoid parameter redundancy. It performs on evaluation of the importance of all the model's parameters and rank them to remove the ones with small important. This process results in a much smaller and faster model but can cause major penalties if the network is not trained even more after the pruning. Note that, however, we did not consider these methods as they are in both cases time consuming as Knowledge Distillation

requires the training of a teacher model and the Parameter Pruning runs iteratively and implies multiple training stages. Using more model compression methods would have thus taken us far from our primary goal to offer a fast and efficient training scheme.

3.5 Conclusion

In this chapter, a review of some CNN compression and optimization methods was given with a proof of their efficiency to transform a large model into a much smaller one without performance penalties. This transformation led to the development of a light and fast medical image analysis model, able to perform tumor segmentation on a GPU embedded developer platform with limited resources.

We discussed a step-by-step CNN layer replacement using Depthwise Separable Convolutions and Independent-Component layers composed of Group Normalization and Dropout to decrease the number of trainable parameters. Moreover, the performances of the compression model were compared to networks performing the same brain tumor segmentation task and proved to be comparable or better on some of the tumor region segmentation. This reinforced the efficiency of neural network compression methods and encourages to move their computation on affordable embedded systems to make the deployment of end-to-end computer-aided diagnosis realistic in the medical field. However, despite the promising results demonstrated in this chapter and because model compression is limited, other image processing methods based or inspired by neural computation can be investigated. In the next chapter, we study the efficiency of a neural model developed for biomimetic image processing by addressing the segmentation and detection task of the brain tumor diagnosis. We will focus on the use of this model for glioblastoma analysis by proposing an easy detection framework supported by an evolutionary algorithm for parameter tuning.

Chapter 4

Glioblastoma diagnosis using pulse-coupled neural networks

4.1 Introduction

The glioblastoma is one of the most common primary malignant tumor in adult cases. It emerges in supporting glial cells and is extremely infiltrative and aggressive. A recent work also highlighted that they are shape-shifting, what prevents the use of targeted drugs to treat them [122] and make their detection tedious. Some improvements in diagnosis technology and the growth of the world's population have induced an increasing number of cases regarding this type of tumor. It thus becomes crucial to improve diagnosis systems to detect them earlier for treatment anticipation and hence increase a patient's life expectancy. However, mostly based on MRI analysis, the diagnosis of such brain tumors can be tedious and most importantly time consuming.

Advances in computer vision along with the appearance of medical image analysis challenges [24] induced a rise in the interest for computer-aided diagnosis systems development. In fact, by making large medical image datasets available to the research community, these challenges brought a rapid growth of deep learning based applications for the medical field. Mostly based on ANNs and more specifically CNNs, these methods proved to be particularly efficient for tumor segmentation, detection and classification [12, 22, 123]. They brought new opportunities to assist health care providers and improve the establishment of the diagnosis task.

Since the detection of anomalies like lesions or tumors can be seen as an object recogni-

tion problem, many different CNN architectures were proposed to perform the segmentation or classification of medical images [12, 124, 125]. These methods can however not be viable in the absence of a large amount of data and enough computational resources. With the unavailability of data induced by ethics and privacy in the medical field as well as the high dimensionality inherent to medical images that causes heavy computational workloads, training CNNs to build computer aided diagnosis systems can therefore be challenging. The use of labelled data is also crucial as most CNNs are trained in a supervised way, which makes it even more difficult to develop them for the medical field as building these datasets would require human resources and lead to additional cost for hospitals. Moreover, as seen in the previous chapter, compressing these CNNs always comes with limits that do not make their hardware implementation easier. Finally, the lack of explainability of this type of neural networks causes a penalty when it comes to large scale deployment. Indeed, choosing a treatment by relying on a decision making system can be sensitive, especially if the origin of its outcomes cannot be fully understood or explained.

Although deep learning image processing methods showed convincing results for brain tumor analysis, the aforementioned issues still have to be addressed. In this work, we thus study the use of the PCNN, a biologically inspired type of neural network to process medical image data. Since PCNN models can perform image segmentation by the use of spiking neurons and do not need to be trained they provide a light and fast alternative to deep learning methods. Moreover they mark a transition between ANNs and SNNs, and let us investigate the power of neural computation for medical image analysis.

This chapter fist gives an introduction to such model and reviews two versions of modified PCNNs in Section 4.2. Secondly, Section 4.3 provides a review of medical image fusion methods to let PCNN models process brain images as they cannot deal with high dimensionality. Then, we exposes the use of PCNNs as brain tumor feature extraction methods by recording their output firing activity in Section 4.4. Finally, our experiments and results on a segmentation and a detection task are given in Section 4.5 where we provide a framework to obtain the best performances out of the models.

4.2 Modified PCNN models

While the PCNN model introduced in Section 2.4.3 was widely studied for its performances in image segmentation, the great amount of parameters it embarks makes it tedious to tune for different tasks. In an attempt to tackle this issue and ease PCNNs usage in computer vision, some studies discussed the development of modified PCNNs with reduced parameters and faster outputs.

In this section we review two of the most commonly used type of modified PCNNs and detail their neuron dynamics.

4.2.1 Unit-Linking PCNN

The ULPCNN [126] represents one of the attempts to accelerate the computation of the standard PCNN model and to reduce its number of parameters. In this model, most of the dynamics of the neuron stay unchanged, however, to simplify the initialization of the feeding channel, the input F_{ij} is set to :

$$F_{ij} = S_{ij} \tag{4.1}$$

Then, a concept of unit-linking is also added to the linking channel and redefine it as follows :

$$L_{ij} = \begin{cases} 1, & \sum_{(k,j)\in N(i,j} Y_{k,l} > 0 \\ 0, & \text{otherwise} \end{cases} \tag{4.2}$$

This modification has the effect of setting the linking channel of a neuron to 1 if neighboring neurons fired what makes the linking inputs uniform. Moreover, the ULPCNN threshold θ is now updated following the next equation :

$$\frac{d\theta(t)}{dt} = -\alpha_{ij}^{\theta} + V_{ij}^{\theta} Y_{ij}(t) \tag{4.3}$$

Finally, as seen in Fig. 4.1, each neuron in the network is connected to 8 neighboring neurons by the linking channel. All these modifications lead to a drastic reduction in the amount of parameters as ULCPNN now only needs the initialization of the linking coefficient β as well as the attenuation constant α_θ and voltage potential V_θ of the dynamic threshold.

4.2.2 Fast-Linking Spiking Cortical Model

The FLSCM [127] is a type of PCNN based on the neuronal activity of cortical neurons. It combines a synaptic modulation coming from post-synaptic neighboring neurons to an external stimulus in order to integrate the neuron's membrane potential. To produce faster outputs, the feeding and linking channel of the standard PCNN neurons are not computed in this model. The rest of neuron's dynamics are described as :

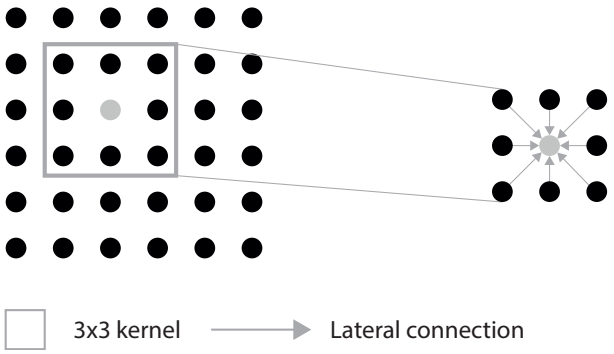

Figure 4.1: *Connectivity of neurons in the ULPCNN.*

$$U_{ij}[n] = \alpha_U U_{ij}[n-1] + S_{ij}(1 + \gamma \sum_{kl} W_{ijkl} Y_{kl}[n-1]) \tag{4.4}$$

$$\theta_{ij}[n] = \alpha_\theta \theta_{ij}[n-1] + hY_{ij}[n] \tag{4.5}$$

$$Y_{ij}[n] = \begin{cases} 1, & U_{ij}[n] > \theta_{ij}[n-1] \\ 0, & \text{otherwise} \end{cases} \tag{4.6}$$

$$\theta_{ij}[n] = \theta_{ij}[n-1] - \delta \tag{4.7}$$

Where, as in the PCNN models above, α_θ and α_U are the attenuation constants of the threshold and the neuron's membrane potential. These parameters are set so that α_U $(0 < \alpha_U < 1)$ and α_θ $(0 < \alpha_\theta < 1)$. γ is the linking coefficient and is commonly set as a factor of a Laplacian operator. h corresponds to the absolute refractory period of the cortical neuron and δ is the threshold decay factor that allows small intensity pixels to induce firing. By bringing the synaptic modulation, this model permits a faster firing of neurons corresponding to pixels with similar intensities whats brings the creation of homogeneous regions after a few iterations.

4.3 Medical image fusion

Identifying a tumor or any other abnormal cluster of cells in the human body often necessitates the analysis of a series of scans with varying aspects. These series, also known as MRI sequences, are created by a varying radio frequency pulses and gradients when performing the scan. Using these sequences is crucial to provide a detailed study of the body portion

that is being scanned as clinical specialists may make use of this amount of information by examining each sequence slice separately. Most open-source datasets provided to build tumor segmentation, like the one described in Section 2.2.3, provide multimodal MRI. This comes particularly useful when building deep learning models for tumor diagnosis systems, as more data make the network extract every meaningful feature that would help to discriminate brain lesions. However, the high computational workload caused by the use of this high dimensional data has to be considered. It appears crucial to whether build light image processing models that can perform on large data or reduce the dimension of the data itself. Here, we adopt the second option and propose an introduction to medical image fusion. Image fusion is a method used to fasten the analysis of multi-channel data. It allows the creation of a single image containing information merged from other channels. Note that a good fusion technique will keep the amount of information in the image as high as possible. This information can be measured by several metrics like the Entropy or the Mutual Information.

4.3.1 Discrete Wavelet Transform

The DWT is one of the most commonly used technique for medical image fusion [128, 129]. It performs a decomposition of a signal into a higher and a lower frequency band. The results of this process on a multi-channel image is a set of detail and approximation coefficients that corresponds to vertical, diagonal and horizontal directions of the input image. DWT thus leads to the creation of four different intermediate images called high-high (HH), low-low (LL), low-high (LH) and high-low (HL) bands.

In order to fuse multi-modal medical images, the algorithm is computed on image pairs to obtain intermediate fused images that will be fused together again until the fusion leads to one single image corresponding to the results of the whole fusion process. Note that pre-processing can be needed to obtain the best fusion. Methods like histogram equalization and normalization are often needed to avoid a potential bias induced by a higher contrast image within the fusion. To that end, a reference sequence is chosen to match the histograms of all the others. The decomposition of a multi-modal MRI scan slice from the BraTS dataset using this method is illustrated in Fig. 4.2

Although it allows an efficient fusion of medical image data, DWT can induce a heavy workload and be time-consuming because of redundant representation of the input image. This prevents its use on a large amount of high dimensional data. To tackle this issue, a modified version of the PCNN, called m-PCNN, was proposed and designed for Computerized Tomography (CT) scans and MRI fusion [130, 131].

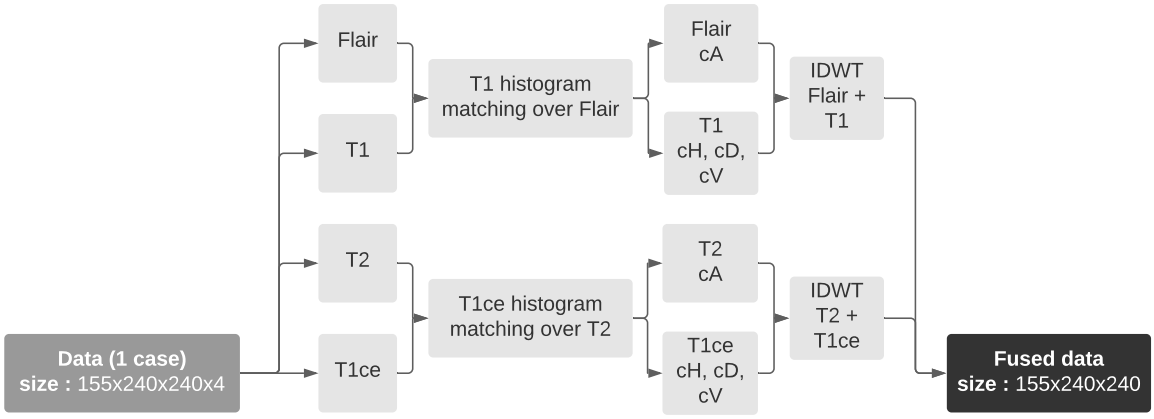

Figure 4.2: *Fusion of a 4 sequence scan using the DWT method.*

4.3.2 Multi-channel PCNN

The m-PCNN is a modified version of the standard PCNN and was designed to perform multi-channel image fusion. In this model, an information fusion layer is added and replaces the linking channel of the PCNN. This layer receives the high dimensional stimulus and is responsible for merging each dimension into the internal state of the neuron. This modification allows the model to receive multiple inputs at the same time, which is different from the standard PCNN that can only receive a single 2-D image. Taking the example of an MRI with $k = 4$ sequences as in the BraTS dataset with the T1, T1c, FLAIR and T2 modes, this new fusion channel $H_{ij}[n]$ is defined by :

$$H_{ij}[n] = \prod_{k=4}^{K}(1 + \beta^k H_{ij}^k[n]) + \sigma \tag{4.8}$$

Where σ is a factor that controls the level of the internal activity. β^k is a weighting factor that modulates the importance of one sequence during the fusion, a higher value of k thus leads to a stronger representation of the corresponding sequence in the fused image. While the dynamics of the pulse generator and the threshold are stay unchanged, the internal activity of the neuron is here changed to :

$$U_{ij}[n] = M^k(Y[n-1]) + S_{ij}^k \tag{4.9}$$

The absence of a linking channel in the m-PCNN reduces its amount of parameters

compared to the standard PCNN, making it computationally interesting. However, in the context of MRI fusion, it is important that the scans are skull-stripped and registered to be able to use this method. Indeed, a wrong layering of the sequences can lead to the creation of visual anomalies during the fusion process. An result example of the m-PCNN applied to the 4 sequences of BraTS dataset is shown in Fig. 4.3. The m-PCNN was initialized with $\{\beta^1, \beta^2, \beta^3, \beta^4\} = \{0.8, 0.3, 0.3, 0.8\}$, where 1, 2, 3 and 4 represents the FLAIR, T1, T1CE and T2 sequences respectively.

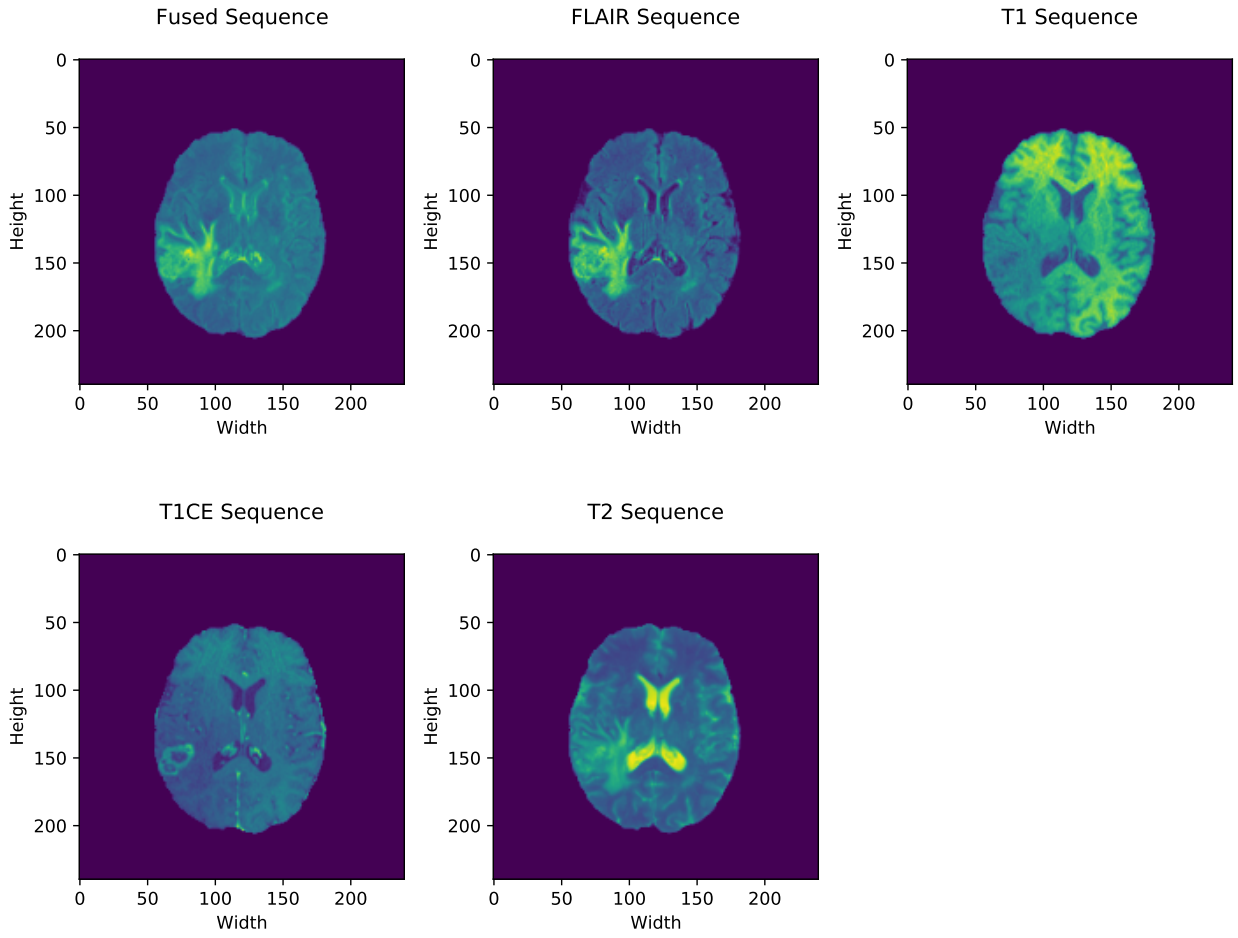

Figure 4.3: *Sequences of one MRI slice of the BraTS 2020 dataset and the sequence obtain by m-PCNN fusion.*

4.4 PCNN for brain tumor feature extraction

The effectiveness of spike-based models for computer vision drew attention to the use of PCNN as an efficient image processing method. Its ability to create clusters of firing

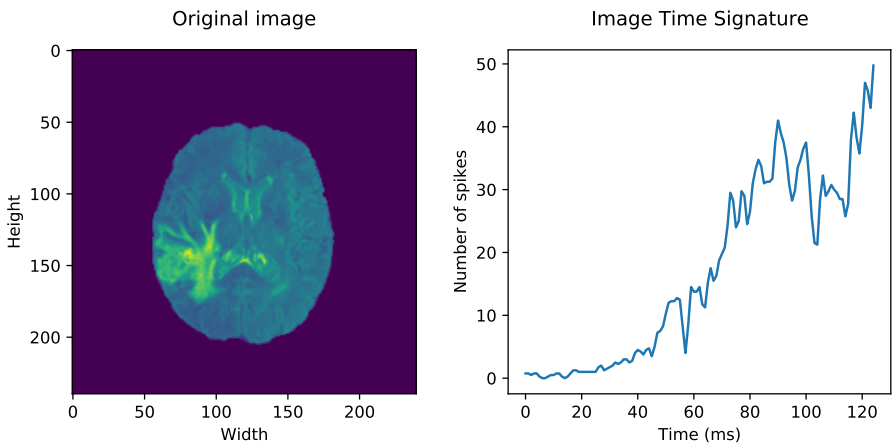

Figure 4.4: *Example of the PCNN signature extraction on an MRI slice from the BraTS dataset.*

neurons is particularly interesting to perform image segmentation. Although, PCNN models have primarily shown their effectiveness in handling basic binary segmentation tasks, some changes to their core processes can allow an extension of their capabilities. The first idea concerning this process was introduced by McClurkin et al. [132]. In their work, they studied a macaque's neural response to patterns and colour stimuli individually. They found the creation of small patterns in the subject's brain that reflected the input stimulus. When presented to a new stimulus containing both patterns and colours, the response appeared to be the multiplication of both previous individual pattern and colour responses.

Following these findings, Johnson [133] proposed the use of a PCNN to encode images into a univariate time series called Image Time Signature (ITS). This ITS is simply obtained by plotting the sum of spikes at each PCNN iteration. Hence, for any type of PCNN and at iteration n, this time series S is defined as :

$$S[n] = \sum_{ij} Y_{ij}[n] \tag{4.10}$$

The retrieved signature is then specific to the shapes found in the image. The main advantages of this method are its fast computation and the fact it provides a new representation of an image that is significantly smaller in size and memory. This makes it particularly interesting to build cost-efficient image analysis systems.

We thus assume that PCNN can be used to retrieve meaningful information in MRI slices (See Fig. 4.4) in order to build a brain tumor detection system. One important quality of an ITS is that it contains sub-signatures of objects held in the image it represents. Typically, the ITS of an image displaying an object on a background is the summation of the ITS of

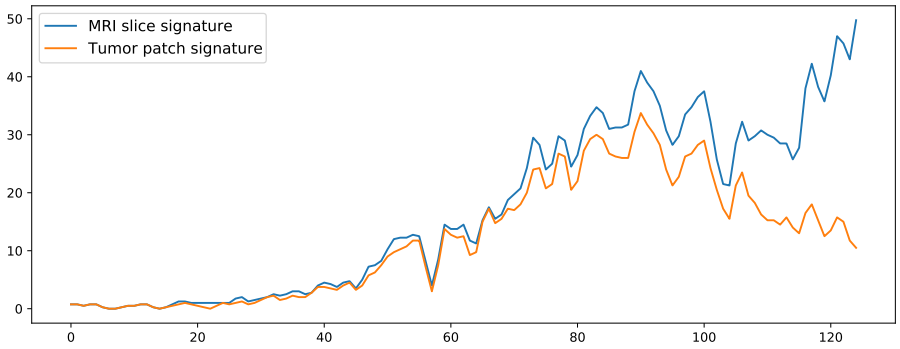

Figure 4.5: *Comparison of the entire MRI slice signature with to the signature of the tumor ground truth patch.*

the object and the ITS of the background. This means that although the method is sensitive to complex background, retrieving the sub-signature that defines the tumor area can still be done using the MRI slice signature.

This can be seen in Fig. 4.5 in which the signature of an MRI slice from the BraTS dataset is compared with the signature of a patch extracted using the ground truth maps also provided in the dataset. The figure shows that although the patch does not include as much information as the MRI slice, the signature obtained from the patch is clearly identifiable and thus motivates the use of ITS in a brain tumor detection system.

4.5 Experiments and results

4.5.1 Dataset

The 2020 version of the BraTS dataset presented in Section 2.2.3 was used to carry all of the experiments exposed in this chapter. As the PCNN models we used take 2-D images as inputs, fusion was applied to all cases in the dataset using the m-PCNN model. The initialization of each β being the most important process for the fusion to provide a good representation of all 4 sequences, the entropy can be computed for each sequence to find which ones carried the greatest amount of information. Our experiments showed that the FLAIR and T2 sequences carried significantly more information than T1 and T1Gd, leading to set β_{FLAIR} and β_{T2} to high values. This is also reinforced by the fact that performing an accurate tumor extraction only using T1 and T1Gd is a tedious task even for powerful methods like ANNs because they mostly hold low-contrast.

However, setting the parameters manually can be tedious, we thus relied on an optimization scheme to find these parameters automatically. Using the same dataset for

a segmentation and a detection task separately, this optimization was based on the DE algorithm and several fitness functions were tested to seek for better performances. More details about this optimization are given in the next sections discussing the segmentation and detection tasks. After being fused, each image was normalized so that pixel values were in the range $[0, 1]$. Finally, for the segmentation task only, the ground truth maps were flattened to binary masks in order to compare them to the PCNN segmentation output and evaluate the models performances.

4.5.2 Brain tumor segmentation

To study the efficiency of the PCNN model for the segmentation of glioblastoma, we tested each of the different PCNN introduced in Section 4.2. The feeding channel F, the linking channel L, the internal activity U and the output map Y were all initialized as zero matrices of the same size as the input stimulus S. The threshold T was initialized as a one matrix of size S as well. All of the other parameters except the number of iterations n were found using the DE algorithm. The list of the optimized parameters can be seen in Table 4.1. Since an optimal value for n can not be easily and automatically determined, the DE process was set to find optimal results within 10 iterations. This allows to keep the computation of PCNN as fast as possible and taking advantage of it to obtain good segmentation results faster. This optimization was carried using a Dice Loss function based on the Dice coefficient that computes the similarity between a ground truth $\hat{Y}$ and the PCNN segmentation output Y. This loss function is defined as :

$$DiceCoef(Y, \hat{Y}) = \frac{2|Y \cap \hat{Y}|}{|Y| + |\hat{Y}|} \tag{4.11}$$

$$DiceLoss(Y, \hat{Y}) = 1 - DiceCoef(Y, \hat{Y}) \tag{4.12}$$

Doing so, we can evaluate the segmentation performance and fine tune the parameters at the same time by minimizing false segmentation. We recorded the development of the coefficient after each iteration to create an early stopping strategy to halt the segmentation process when the loss did not decreased after two iterations, to ensure that we always obtained the best segmentation. This method is frequently used while training Deep Neural Networks to ensure that the training process does not continue after the model has attained its peak performance. This thus allows the model to always output a segmentation map that is as close as possible to the ground truth.

Model	Standard PCNN	ULPCNN	FLSCM
β	0.47	0.47	0.44
α_θ	0.0125	0.015	-
α_F	0.96	-	-
α_L	0.81	-	-
V_θ	20	20	20
V_F	0.21	-	-
V_L	0.36	-	-
W	3x3 gaussian kernel	-	-
M	W	-	-
α_U	-	-	0.49

Table 4.1: *Optimized parameters of the PCNN models.*

4.5.3 Brain tumor detection

It was proven that ITS could be used to build a fast object recognition model by comparing local ITS, obtained by computing a PCNN over an image sub-region, to a global ITS representing the spiking activity of neurons from the entire image [134, 135]. While this method proved to be efficient for small object recognition, some room was left for improvements as the study presents two major drawbacks. First, the method was only tested to detect objects that were already distinct from the background they laid on. No clutter or complex patterned background was used to prove the robustness of the model. Secondly, when the model was pushed to detect multiple objects, overlapping was not considered and the model was only proven efficient in a situation when objects stood alone. Finally, the algorithm used to find sub-regions to build local IS was defined as a moving window of size N that increments after each iteration if the best fit was not found. This method can be computationally.

The right choice of image sub-region is thus crucial to decrease the complexity of the model. In fact, relying on incrementing moving windows does not allow to build an efficient glioblastoma detection model because of the wide variety of shape and size these tumors can take. Region proposal algorithms have been widely investigated in deep learning and their capability to choose regions of interest was successfully applied in CNN to perform semantic segmentation [136]. They allow to avoid the computing of a multitude of detection windows with different sizes. In this work we use the Selective Search algorithm discussed in Section 2.1.1. This algorithm is particularly interesting for glioblastoma detection as it provides an efficient way to retrieve regions of interest with different width and height. In fact, as mentioned above, as glioblastoma tumors can take a wide range of shapes and dimensions, it is important to rely on an algorithm that can fastly produce bounding boxes

with divers sizes.

Combining Selective Search to the PCNN feature extraction discussed in Section 4.4 we propose the following algorithm to perform a complete tumor detection :

Step 1 : Compute Selective Search on M to create B_{ij} bounding boxes
Step 2 : Remove boxes with area greater than threshold θ
Step 3 : Extract image patches P_{ij} from the boxes
Step 4 : Convert each P_{ij} to signature S_{ij}
Step 5 : Compute the Euclidean distance between each S_{ij} and the signature M_{sign} obtained from M
Step 6 : Retain B_{min} the box that gives the smallest distance D_{min} as the complete tumor detection box

Where the value of θ is obtained from experimental observations regarding the average volume of the tumors available in the dataset. This threshold permits to fasten the region selection process by removing abnormal large boxes. To evaluate the performances of this algorithm, we conducted several experiments regarding the initialization of the PCNN parameters for the feature extraction and the fusion process. Tuning the parameters for both processes allowed to analyse their contribution to the detection task. To prove the efficiency of our algorithm we first started to extend the BraTS dataset by automatically creating bounding boxes using the ground truth maps. These ground truth bounding boxes were then used to compute the accuracy of the proposed model by the mean of the Intersection over Union (IoU) defined as :

$$IoU = \frac{B \cap G}{B \cup G} \tag{4.13}$$

Where G is the ground truth bounding box and B is the one selected by our algorithm. The fusion process performed by the m-PCNN is crucial for our algorithm to perform well. Getting a new representation of the original multi-modal data that gives enough information to identify the tumor's edges is an important step before computing the Selective Search. It can let the right segmented areas to stand-out when computing the over-segmentation with the Felzenszwalb algorithm. Some pre-processing step can also be added before computing the algorithm to further enhance edge detection and thus optimize the over-segmentation. Fig. 4.6 shows the effect of a badly tuned m-PCNN on the performance of the detection algorithm. It also shows that the amount of box candidates is greater when the image was badly fused as the pixels intensities around the tumor region were not enhanced. Therefore, since all the candidates are compared to the main signature M_{sign}, a bad initialization of

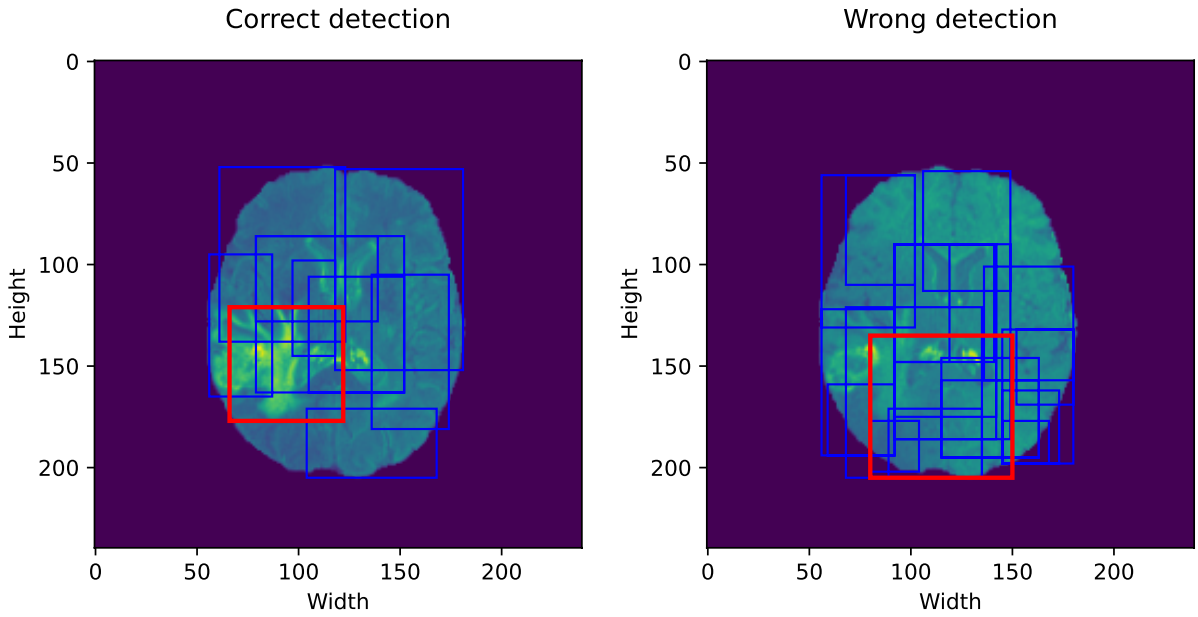

Figure 4.6: *Correct and wrong detection using different fusion parameters.*

the fusion parameters induces a higher computational cost for the detection algorithm.

It hence appears crucial to optimize the fusion parameters initialization. The first strategy to do so is to use a weight factor over the data sequences that show high contrasts such as the FLAIR and T2. However, when weighting the β coefficient of the m-PCNN for these two sequences we observed that global low contrast was key to the creation of local ITS that could be found in global ITS. We thus employed the DE algorithm to find the parameters automatically.

We tested the optimization process with two different fitness functions. The first one aimed to minimize the standard deviation of the fused image to decrease its contrast. The second fitness function searched to minimize the entropy of the image to induce a maximization of the information gain. In our experiments, minimizing the standard deviation appeared to give better results and induce a higher β for the T1CE sequence. Nevertheless, a minority of samples in the dataset responded negatively to the reduction of contrast which caused inappropriate ITS creations.

4.5.4 Results and discussions

The experiments carried for both glioblastoma segmentation and detection tasks proved the efficient of PCNN models for medical image analysis.

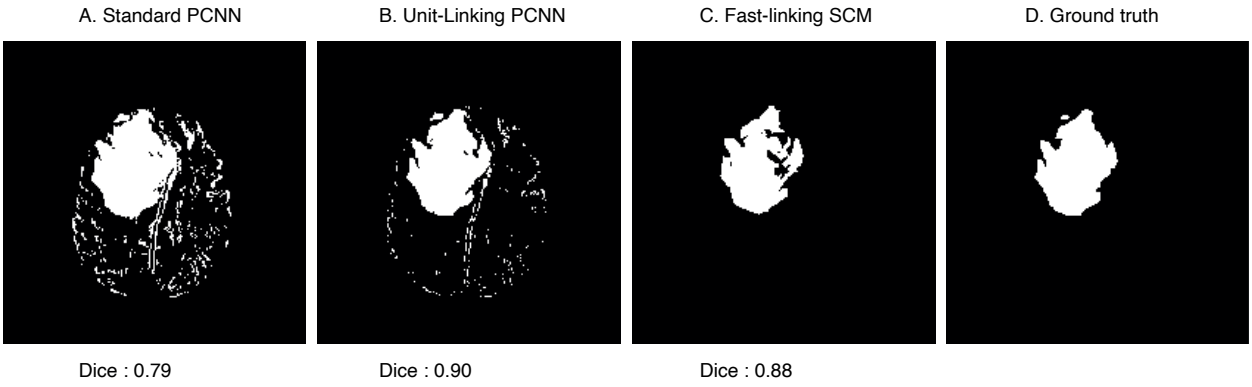

Figure 4.7: *Results of three PCNN models on a brain tumor segmentation task.*

Segmentation

In the segmentation process, each of the PCNN models presented in this chapter were tested and succeeded in segmenting cases from the BraTS dataset. Their performance was evaluated using the Dice Coefficient to compare the ground truth maps to the models segmentation maps. Computation time was also retrieved to reinforce the proof of their ability to perform this task.

Fig. 4.7 shows the results of a segmentation performed on an scan slice along with the manually segmented ground truth. Note that, during all the evaluation process, the FLSCM and the ULPCNN performed better than the standard PCNN in both Dice score and computation time. Ran on CPU, the standard PCNN had an average segmentation time of about 60 seconds for roughly 100 slices corresponding to one BraTS scan. Both FLSCM and ULPCNN were able to segment the same amount of slices in 17 seconds on average. However, the FLSCM was the only model that could handle small contrast variations between tumorous and non-tumorous tissues. Its segmentations appeared clearer and well-defined.

Some limitations regarding this method were however found. The spiking of individual neuron kept when building the final segmentation map induce the presence of noisy pixels as seen in B of Fig. 4.7. The only way we found to deal with this problem was to apply post-processing operations like morphological image processing but this includes an additional operation and can lead to heavier workloads.

Moreover, note that we kept the segmentation task as a binary problem. Some modifications in the PCNN models could be further investigated to provide a multi-label segmentation

map.

Detection

Our proposed method was found to be effective in detecting the tumor in the majority of our experiments. Ran on CPU, our algorithm was able to detect tumors in each MRI slice in 8 seconds on average when the maximum number of proposal was not specified. In other trials, limiting the maximum number of boxes created by Selective Search to ten reduced the algorithm's computing time to an average of 3 seconds per slice. These results prove the efficiency of the proposed algorithm as well as its independence to GPU. However, the selected bounding boxes were often not pixel perfect. This can be due to the lack of a pre-processing step that could help the Selective Search algorithm in discriminating meaningful regions. We indeed decided not to apply any pre-processing operation to the original data in order to keep the computational workload of the entire process as low as possible. With this strategy we also have a proof of the performance of PCNN models without data transformation. Some results of our detection method are shown in Fig. 4.9. Although the average IoU score obtained over the entire dataset proved the efficiency of our method, some limitations and enhancements have to be considered.

If our method was successful on slices with large tumors, it was often limited when trying to detect smaller regions in the early and late slices of the scans. This is due to the particularly small sizes of these tumors, mostly made of a few pixels that can hardly be identified. Due to the fact that early and late slices only exposes small spatially sparse groups of voxels, comparing the signature of the whole slice to any of these groups would result in short euclidean distances, causing the detection procedure to be incorrect. This also had an impact on the segmentation performed during the Selective Search, since variations in all similarities computed by the algorithm resulted in the production of incorrect bounding boxes, as seen in the Fig. 4.8. As a result of the lack of information surrounding the tumor, the algorithm failed to detect it.

Regarding this issue, the model was only tested on slices containing enough contextual information to detect tumorous tissues. Our method was thus applied to every scan in the BraTS dataset, yielding an average IoU score of 0.78. Some further work can thus investigate pre-processing steps to increase the intensity of these small tumorous region as well as a tuning process for the Selective Search algorithm to let it propose very small regions. Enhancing the fusion process with a multi-criteria optimization can also be explored to provide a balance between contrast and information gain.

By rerunning the algorithm on the predicted picture patches and fine-tuning the fusion process to increase contrast on certain tumorous areas, this work may be extended to a

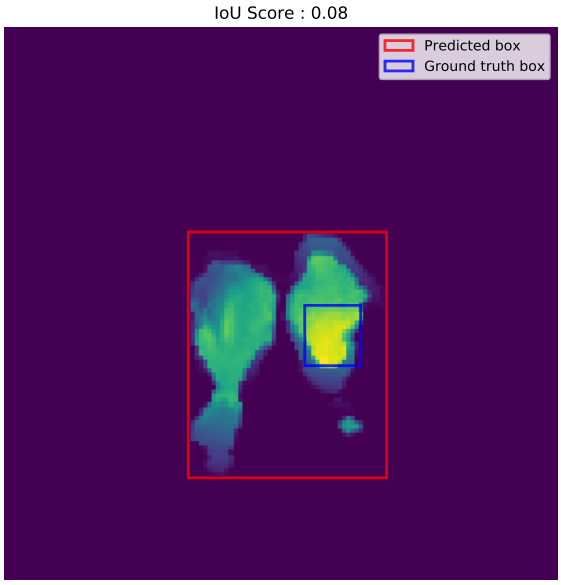

Figure 4.8: *Detection of a tumor in the last slice of a scan.*

multi-label detection approach. For instance, the algorithm was able to recognize the edges of the core tumor region when the fusion process was heavily impacted by the T1c sequence. Allowing the FLAIR and T2 sequences to affect the fusion more than other sequences, on the other hand, improved the accuracy of recognizing the whole tumor. Running the algorithm on the patches contained in the found bounding boxes could also lead to a tumor sub-region detection process. However, some changes would have to be made in the computation of the time signatures to make sure each sub-regions can be find in the global ITS. Note that this process can be time consuming as the PCNN models used for fusion and feature extraction would have to be fine-tuned again. Finally, to build a complete recognition system, a classifier like SVM can be employed to label the sub-regions.

4.6 Conclusion

In this chapter, the use of PCNNs was investigated to solve both glioblastoma segmentation and detection tasks. Firstly, we employed PCNNs to perform a binary segmentation of scan slices taken from the BraTS dataset. Our experiments proved the efficiency of all the models tested even with a simple parameter setting scheme. The differential evolution algorithm was used in order to get optimal results by finding the best parameters possible for each PCNN. It was based on a fitness function using the Dice Coefficient as a similarity metric between the ground truth segmentation and the one obtained by the model. The same

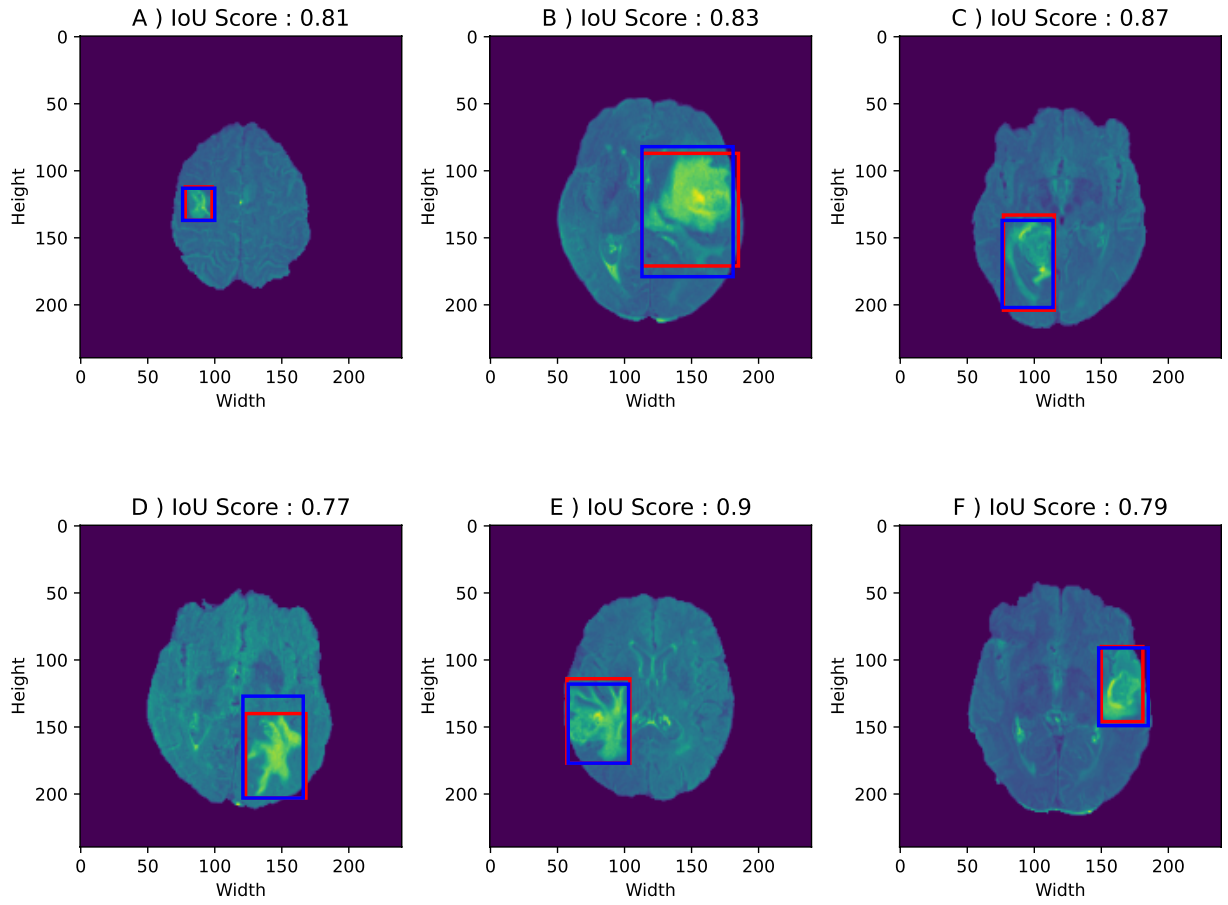

Figure 4.9: *Examples of brain tumor detection obtained by the proposed algorithm. The red boxes are the predicted detections and the blue boxes are real bounding boxes created from the ground truth maps.*

metric was used to build an early-stopping scheme and make sure that the models output the best segmentation map possible.

Secondly, we discussed the application of PCNNs to build a glioblastoma detection system by using them as image fusion and feature extraction methods. Coupled with the Selective Search algorithm, the use of Image Time Signatures produced by PCNN allowed to build an efficient complete tumor detection framework. To obtain optimal results, the differential evolution algorithm was used as a mean to optimize the fusion process which is crucial to increase detection accuracy.

Finally, our investigation showed that a tumor recognition system could be built without the use of heavy deep learning algorithms. Indeed, the results obtained encourages the use of simple neural model based image processing methods and build confidence in the

creation of fast, light and scalable medical image analysis systems. In the next chapter, we go further in investigating brain inspired computation to address the computational cost, power consumption and explainability issues inherent to deep learning models. We will discuss the development of a single trainable layer C-SNN for tumor classification and compare its performances with state-of-the-art deep and machine learning models.

Chapter 5

Training an SNN for brain tumor classification

5.1 Introduction

As discussed in the previous chapter, the promising opportunities offered by spike-based computation inspired by biological neurons have been getting more attention in recent years [137, 138]. With the power to solve some of the issues carried by deep learning methods like energy consumption, the need for large datasets or the dependence to powerful hardware, brain inspired computation methods can provide a good alternative to deep neural networks. If we proved that PCNN models could be use for simple brain tumor segmentation and detection tasks, they were not designed to perform classification, which is necessary to complete a diagnosis, and they also lack of a strong feature extraction step. More complex models of spiking neurons can thus be investigated to have further proofs of brain inspired computation's efficiency in computer vision.

Inspired by biology and motivated by the speed of visual processing in the human brain, SNNs are exploiting the properties of neurons and synapses to carry information and now appear as powerful competitors to ANNs. The last few years have seen an increase in the development of these models that aimed for a level of performance that would come close to deep learning methods. However, although they do solve the computational and energy cost problems of their artificial counter parts and provide an easier hardware implementation, SNNs have not been able to outperform ANNs in most vision tasks. This is mostly due to the lack of efficient training methods and the lossy encoding schemes they induce to represent

data.

To extend the work exposed in the previous chapter and go further with applying neural computation to brain image analysis, which as never been done to our knowledge, we propose a study of a single trainable layer C-SNN to solve brain tumor classification by using biologically-plausible plasticity rules. Regarding how small this model is, this work does not aim to outperform deep CNN but rather gives bases to develop C-SNNs for medical image analysis and prove the opportunities they bring in building lightweight and cost-efficient diagnosis systems.

In this chapter, we thus describe the implementation of such model. In Section 5.2 we first describe the classification tasks our proposed approach aims to solve. The data used for this purpose is also presented along with a pre-processing pipeline designed to facilitate the feature extraction performed by the C-SNN. Then, in Section 5.3, the structure of the proposed model is discussed along with the learning mechanisms used. Finally, our experiments and results are described in Section 5.4. A comparison to deep learning methods is also provided to place the performance of C-SNNs in the state-of-the-art.

5.2 Classification tasks and data processing

To prove the efficiency of the models proposed in this chapter, we define two classification tasks with different levels of complexity. The first one (CT1) is a binary classification that uses the 2015 version of the BraTS dataset to categorize high- (HGG) and low-grade (LGG) gliomas. For this task, only the FLAIR scans of the dataset were used as they often offer a greater amount of information to extract tumorous tissues. The other classification task (CT2) our model aim to solve relies on a dataset composed of 3064 slices from 233 patients containing 3 types of tumors, namely glioma, meningioma and pituitary tumors. The data was presented in the work of Cheng et al. [139].

Since SNNs convert input data to spike trains, pre-processing operations can be very useful to ease the encoding process by reducing the complexity of the data. The MRI slices from all the datasets mentioned were pre-processed following the same pipeline. To use the same type of data throughout our experiments we used the method proposed by Shattuck et al. [140] to perform skull-stripping when needed. This work employed a combination of processes including Marr-Hildreth edge detection and morphological operations in order to separate the brain region from the skull surrounding it. Appart from the BraTS dataset, other MRI slices used were not skull-stripped. Since we used different datasets, the histograms of all images were matched using a FLAIR scan from the BraTS dataset as template.

The MRI slices were then converted to grayscale images and then normalized. To decrease the computational workload of our experiments each slice was also downsized to a 128x128 image. To account for the fact that brain images are often noisy and that the dataset used for CT-2 presents some artifacts, Anisotropic Diffusion Filtering (ADF) [141] was used. This method has been widely investigated in medical image processing [142, 143, 144] since it allows to conserve the edges of an image while smoothing it out to remove any outlier or artifact that might be present (See Fig. 5.2). The discrete form of ADF is described by :

$$I_{p_1}^{t+1} \approx I_{p_1}^{t} + \frac{\lambda}{|\eta_{p_1}|} \sum_{p_2 \in \eta_{p_1}} f(\left|\nabla I_{p_1,p_2}^{t}\right|, \gamma) \nabla I_{p_1,p_2}^{t} \tag{5.1}$$

Where $I_{p_1}^{t}$ corresponds to the intensity of pixel p_1 at time t from an image I, η_{p1} is a set neighboring pixels from p_1, λ corresponds to a scalar associated to the diffusion rate, γ is a positive constant that sets the smoothing level of the filter, $\nabla I_{p_1,p_2}^{t}$ indicates the magnitude of the directional gradient from pixel $_1$ to p_2 and f is an the edge-stopping function that performs the smoothing process. These pre-processing pipeline was ran on each slice right before they were fed to the network and is summed up in Fig. 5.1.

5.3 Convolutional SNNs with synaptic plasticity learning rules

Convolutional neural networks have been widely investigated in computer vision for their ability to extract meaningful features to build highly accurate decision making systems. In parallel, the recent breakthrough of SNNs used for pattern recognition showed the efficiency of spike-based models to process signals with temporal dependencies. However, their representation of data in the temporal or rate domain often lacks the ability to perform complex feature extractions such as the ones found in CNNs. In computer vision, this explains the gap between the accuracy of CNNs and SNNs. Hence, investigations to build models that use the best of both types of neural networks led to the development of C-SNNs.

As their name implies, C-SNNs are the spiking equivalent of CNNs and aim to bring powerful feature extraction to SNNs while taking advantage of the spike-based computation to reduce computational cost issues involved in CNNs. A simple and typical C-SNN architecture consists in three main layers that perform encoding, feature extraction and decision making respectively. The encoding layer is responsible for the conversion of visual stimuli to spike times and can use any of the coding scheme discussed in Section 2.3.2

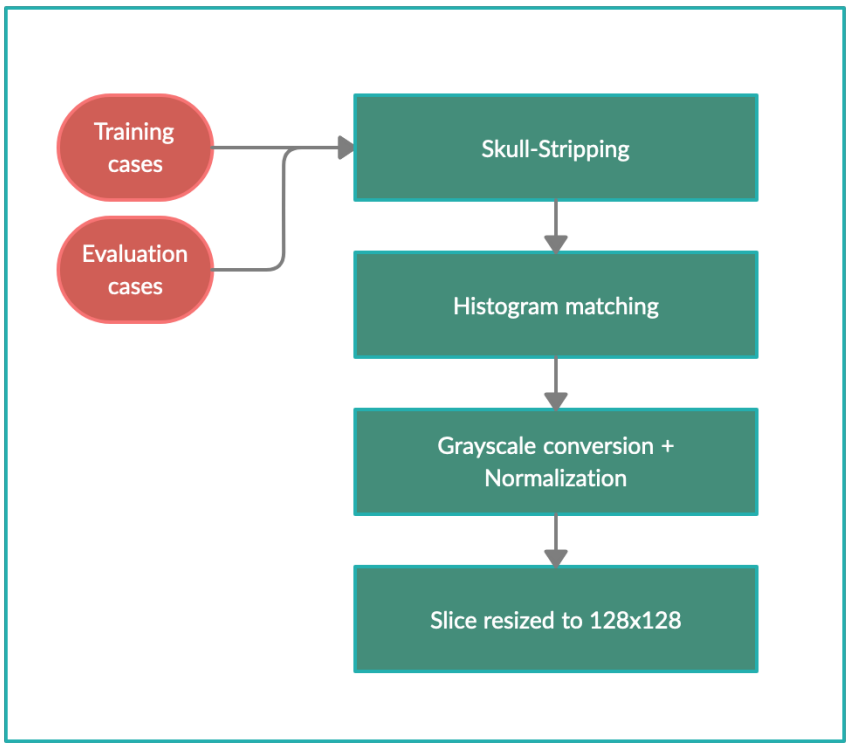

Figure 5.1: *Data pre-processing pipeline before training the C-SNN.*

and can also apply some filters on the data to simulate real perception processes. In the feature extraction layer, CNN common operations like convolution and pooling are used to build a new representation of the input data by the mean of feature maps. As the data is propagated in the network under the form of temporal codes, this layer also contains sets of spiking neurons that convert the features to spike times. In bio-plausible settings the connections between the two previous layers are represented as plastic synapses that build the learning process of the entire network. The output layer often holds groups of neurons which activities are associated with a decision. Being driven by the activity of neurons in the feature extraction layer, the decision is often attributed regarding the timing of spikes so that the first neuron to fire in the output layer indicates the decision. In this section, we describe the development of such model in details and discuss its application to brain tumor classification.

5.3.1 Network topology

The network used in this work was created following the architecture of the HMAX [145] model. It is thus composed of four layers of simple and complex cells. Designed for object

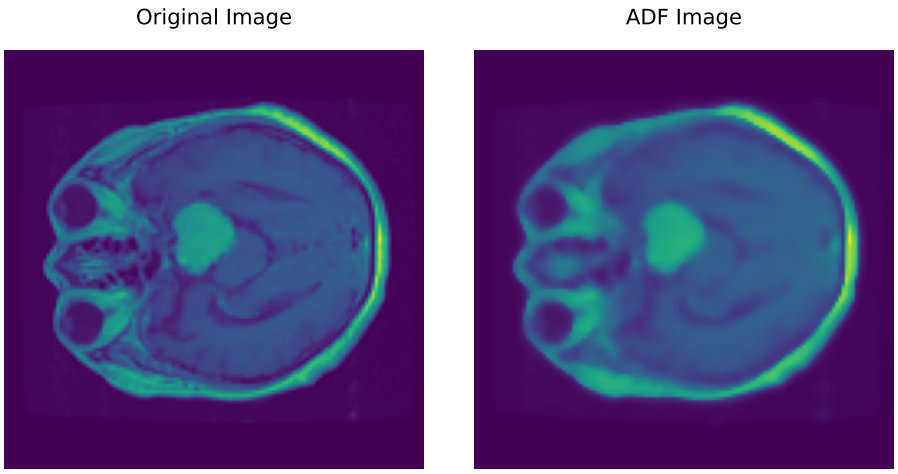

Figure 5.2: *Anisotropic Diffusion Filter (ADF) applied to an MRI slice.*

recognition on natural images, this structure simulates the invariance and complexity observed in the ventral stream. Here, simple cells are used for feature selectivity and complex cells add the invariance needed for the recognition task.

V1 Simple Layer 1 - S1

Before passing through the first layer and as mentioned in the previous section a denoising operation was performed on each MRI slice to avoid the detection of features in background areas cause by possible MRI artifacts or bad acquisition. In the input layer, a DoG filter was then applied to the brain images to mimic the LGN encoding of visual stimuli. This transformation is followed by V1 simple-cells as the ones discussed in Section 2.4.1. Since these cells are sensitive to orientation, they are modeled as Gabor filters that are known to perform edge detection. In our experiments, we used 4 different filters that were 45 degrees out of phase to extract all oriented edges. The filtered stimuli are then converted to spike times using the TTFS encoding leading to the creation of four feature maps of the same size as the input image so that each position in the map corresponds to a pixel firing time. This allows salient edges to spike earlier and have a stronger effect on later neurons. An example of this layer output can be seen in Fig. 5.3.

V1 Complex Layer 1 - C1

Complex cells are known for their spatial phase invariance properties. They extend the features of simple cells by reacting to oriented grating stimuli with a wide variety of spatial phases. The first layer of complex cells in our network performs a pooling operation on each

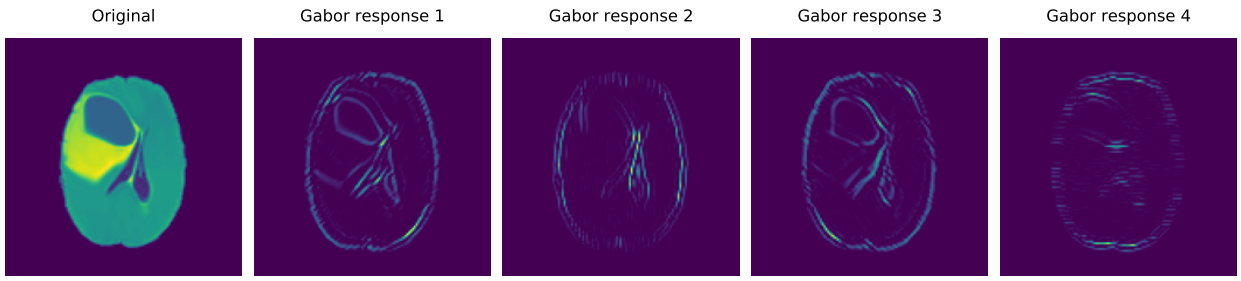

Figure 5.3: *Output example from the S1 layer.*

of the feature maps retrieved from the S1 layer. It is thus composed of the same amount of maps as in S1. Just like pooling operations in CNNs, this layer allows to reduce the dimension of the incoming stimulus by downsizing the number of neurons and consequently increases computational speed while adding spatial invariance on the detection of edges. This is done by keeping the earliest spike times in each pooling window. To provide sparse feature representations, lateral inhibition is also used in this layer. For each feature map, an inhibition kernel first increases the spike timing of the neurons around the ones that fired earlier. Moreover, when a neuron spikes in one of the feature maps, the neuron at the same position in all other maps remains silent while the image is being processed. This helps to avoid feature redundancy in the layer by making sure each feature map offers a different representation of the input.

V1 Simple Layer 2 - S2

The second layer of simple cells is responsible for most of the processing happening in the network and is the only trainable layer. It is composed of sets of IF neurons that integrates by receiving inputs as kernels of C1 neurons activity and fire when their membrane potential reaches a threshold value. A refractory period is used to make sure every neuron can only fire once per image. A WTA mechanism is also triggered by each spike using lateral inhibition. The synaptic weights between C1 and S2 are updated according to the learning rule used. With a simple learning rule such as STDP, the order of pre- and post-synaptic spikes defines the potentiation or depression of weights between the layer. The strengthening of weights thus carries information about features found in the data to the next layer.

V1 Complex Layer 2 - C2

C2 is the output layer of the network and is responsible for the classification process. Each neuron in this layer is connected to one set of S2 neurons and transmits the first spike from

it. To classify images, C2 neurons are then assigned to a class label and decision is made according to the group of neurons that fired earlier.

5.3.2 Supervised learning

Many studies investigated the role of the reward system in the human brain and suggested that it highly influenced decision making. In such mechanism, rewards are sent to motivate a state or behaviour to repeat itself. On the contrary, penalty signals can be sent to avoid triggering these behaviors. When the brain is exposed to the reward signals, it answers by intensifying the production of dopamine, a neurotransmitter that was found to reinforce beneficial behaviors. For learning, it was shown that neurotransmitters like dopamine could have an impact on synaptic plasticity. This kind of system can thus be interpreted as a supervised learning scheme with reinforcement.

The Reward-Modulated STDP (R-STDP) [146] is an extension of the STDP learning rule that incorporates this reward mechanism and can be used to train SNNs in a supervised manner. The idea is to regulate the effect of STDP using the reward mechanism that can thus change the weight in positive or negative ways according to specific stimuli and not only the order of pre- and post-synaptic spike timings. The timing of the reward signal is also important in biological settings. Indeed, if a reward or punishment signal is sent long after the neural activity that triggered it, it does not have any use as many other events would have occured. This is discussed in the work of Izhikevich [147]. To avoid this, the reward system we use employs synaptic traces that store the outcome of STDP to let the reward be consumed and change the weights at the right time. The weight changes ΔW_{ij} between pre-synaptic neuron j and post-synaptic neuron i in layers $l1$ and $l2$ respectively driven by R-STDP under a reward signal is then defined as :

$$\Delta W_{ij} = \begin{cases} A_{+} W_{ij} \cdot (1 - W_{ij}) & t_{l1}(j) - t_{l2}(i) > 0, \\ A_{-} W_{ij} \cdot (1 - W_{ij}) & t_{l1}(j) - t_{l2}(i) \leq 0 \end{cases} \tag{5.2}$$

Where A_{+} and A_{-} are the magnitude of weight change and t the firing time of a neuron. In the context of the proposed model, this reward mechanism is thus set to induce weight changes when the network's output neurons that fired represent the right tumor label of the data being processed.

5.4 Experiments and results

5.4.1 Implementation and experiment

The C-SNN discussed in this work was originally implemented using the SpykeTorch simulator [148]. Compared to other SNN simulator, SpykeTorch was specifically designed to simulate C-SNNs with at most one spike per neuron and provides the necessary tools to train them in supervised and unsupervised manner using synaptic plasticity rules. Based on the PyTorch library, it also has the potential to exploit GPU resources and allows deep learning developers to use a friendly interface. Although this framework pretends to make shallow and deep C-SNNs implementation easy, building C-SNNs with more than 3 layers can quickly become tedious as the networks' forward passes have to be coded for each layer in both training and testing phases. The training loops also have to be defined individually and were not automated considering the networks structure and learning methods used. To ease the development of our experiments and provide a new way to define C-SNNs, we thus extended the SpykeTorch framework and made it easier to use, especially for common deep learning framework users. To offer a new interface for creating C-SNNs we provided a $Network$ class to the framework. This class allows to define C-SNNs in a way similar to the well-known Keras API and only defines them by convolutional and pooling layers as well as the learning rule used in each layer. A $SupervisedEstimator$ class was also added to solve the hard definition of training loops by automating the forward passes for the layers. This extension along with the implementation of our work can be found at `https://github.com/bniepce/csnn-brain-tumor-classification` and is discussed in Appendix A.5.

The proposed C-SNN was ran for 500 epochs for each of the task. All MRI slices from both datasets were used and fed to the network by batches of 64 samples. We recorded the average accuracy at the end of each step and used testing samples to evaluate the performances of the model. To prove C-SNNs do not need GPU resources to be efficiently trained, we did not use the CUDA support provided by the framework and ran every experiment on an Intel Xeon Bronze 3106 CPU. Using this resource, training the network took 20 and 45 minutes for the CT1 and CT2 tasks respectively.

5.4.2 Results and discussions

Our proposed method was successfully able to classify the different types of tumors provided by both datasets while obtaining competitive results to state-of-the-art CNNs. The average accuracy scores of 0.868 and 0.828 were recorded for the CT1 and CT2 tasks respectively.

These performances were compared to other machine learning and deep learning methods found in the literature as seen in Table 5.1 and 5.2. The tables show that our model mostly performed slightly worse than other works but still provides suitable results according to the fact that the model was only composed of a single trainable layer and requires significantly less computational power than the methods we compared it to. The results also demonstrate that simple bio-plausible learning rules can be used to learn complex features such as the ones inherent to brain scans.

Also note that, while the machine learning models from Polly and Cui exposed in Table 5.1 obtained a better average accuracy than our model, the length of the set of data they used was relatively small and do not prove the efficiency of the algorithm on a large amount of cases or their ability to generalize and be less affected to data variations. Outperforming all of other methods, the CNNs of Banerjee et al. and Díaz-Pernas et al. proved the superiority of deep learning models in general. They are both however deep or multi-pathways CNN that do not meet cost efficiency requirements exposed by health care providers. Moreover, in CT2, our model also closely approached the CNN proposed by Abiwinanda et al. and outperformed the one proposed in the work of Pashaei et al. This reinforces the promising abilities of SNNs to be a good alternative to heavy deep learning models.

Authors	**Method**	**Total classification accuracy**
Banerjee et al. [149]	Deep CNN	0.971
F. P. Polly et al. [150]	DWT + SVM	0.914
Mzoughi et al. [151]	Deep CNN	0.96
Ge Cui et al. [152]	Random Forest	0.913
Proposed model	**Single layer C-SNN**	**0.868**

Table 5.1: *Comparative results on the CT1 task.*

5.5 Conclusion

In this chapter, we discussed the development of a C-SNN for brain tumor classification. The aim of our work was to prove the efficiency of SNNs in medical image analysis to address some issues inherent to deep learning models such as computation cost, power consumption and explainability. Our study showed promising results and reinforced the

Authors	Method	Total classification accuracy
Díaz-Pernas et al. [153]	Multiscale CNN	0.973
Anaraki et al.[154]	CNN + Genetic algorithm	0.942
Abiwinanda et al.[155]	CNN	0.841
Proposed model	**Single layer C-SNN**	**0.828**
Pashaei et al.[156]	CNN	0.810

Table 5.2: *Comparative results on the CT2 task.*

opportunities brought by bio-plausible models in computer vision.

Our proposed approach addressed two different brain tumor classification tasks that were introduced along with the MRI datasets used to train the network. We provided a data pre-processing pipeline to ease the feature extraction performed during training and supply a clean representation of the data before the time encoding step required by the C-SNN. The architecture of the network was then detailed layer by layer and the use of the R-STDP as a supervised reinforcement learning rule was discussed. We provided an extended version of the SpykeTorch framework to ease the creation of our experiments and successfully trained the C-SNN without GPU support. The results we obtained for both classification tasks proved that the network could converge under 500 epochs and be trained relatively fast. However, although the accuracy scores the network obtained were acceptable, comparison to other methods showed that simple C-SNNs could in most cases not outperform deep CNNs. Our model could however approach state-of-the-art accuracy and was even slightly better than a CNN. The work exposed in this chapter thus enlightens possible opportunities to build visual tumor diagnosis tools using brain inspired computation.

Chapter 6

Conclusion

Building computer-aided diagnosis systems based on Artificial Neural Networks has been getting increasing attention due to the outstanding performances of deep learning models in a wide range of domains. While they were already proven to be efficient to perform recognition tasks on medical images, these models do not meet clinical setting requirements. They indeed suffer from high memory and energy cost, and are dependent to expensive powerful hardware. Moreover, they require the use of large datasets to generalize, what can be tedious to rely on in the medical field because of data privacy inherent to this domain. The goal of this thesis is thus to search for new methods to build computer-aided diagnosis systems while addressing the aforementioned limitations of ANNs. We proposed a gradual transition from Deep CNN to lightweight and cost efficient spiking network models. To this aim, we investigated different ways to deal with common issues found in applied Deep Learning and tested them on brain tumor diagnosis visual tasks. This conclusion thus discusses the contributions and results exposed in this work. We then conclude by addressing future works and perspectives that can extend our work to go further in building powerful and cost efficient brain tumor diagnosis applications.

6.0.1 Contributions

The work proposed in this thesis aimed to provide frameworks to build brain tumor diagnosis systems while accounting for the requirements exposed by health-care providers on the use of such tools. The proposed methods thus intent to redefine or replace powerful computer vision methods based on ANNs in order to reduce the memory, energy and material cost

they imply. Our work is structured to study the transition between heavy deep learning methods to lightweight networks of spiking neurons for brain tumor analysis.

In Chapter 3, we studied the deployment of a CNN performing brain tumor segmentation on a device with limited resources. This contribution addresses the computational and material cost issues inherent to the use of CNNs. Firstly, we discussed the usage of new GPU embedded platforms for the development and training of deep neural networks. We investigated the opportunities of such hardware to reduce the cost of existing computer vision methods. Secondly, we reviewed a compression framework for CNNs that aimed to drastically reduce their number of parameters. This work was based on redefining convolutional operations, replacing batch-normalization and using quantization to reduce the precision of the network's operations. Finally, our method was applied to the U-Net neural network to produce a fast and lightweight multi-label brain tumor segmentation system. The BraTS 2015 dataset was thus used to evaluate the performances of the network to differentiate 3 tumor sub-regions. While our compression method lead to a parameter reduction of roughly 93 %, our experiments showed that even if the segmentation scores were lower than other state-of-the-art CNNs performing the same task, they still remained comparable.

In Chapter 4 we studied the use of PCNNs for brain tumor segmentation and detection. This contribution aims to go further in reducing computational time and cost of medical image analysis methods based on neural networks. It marks the start of the transition to spiking neuron models to build brain tumor recognition systems. We thus first proposed a comparative study of several PCNN models to perform a binary segmentation on the BraTS dataset. Our experiments showed that optimizing the activity of PCNNs' neurons and provide a better segmentation accuracy could be done by tuning the models' parameters using a simple evolutionary algorithm. Then, we tackled the brain tumor detection task by proposing a new algorithm based on the FLSCM and the Selective Search algorithm. Our method relied on tuning an m-PCNN for MRI modalities fusion and use the FLSCM as a feature extraction method creating image signatures. By comparing local signatures created from Selective Search patches and the global signatures representing an entire MRI slice, our algorithm could successfully find whole tumor regions. This work addressed the limitations of past researches on PCNN object recognition that employed moving windows to find region of interests by reducing the amount of possible detection proposals with Selective Search. An optimization scheme was also proposed using the differential evolution algorithm to find the PCNN parameters that led to more accurate detection. We showed the importance of fusion parameters and discussed possible extensions to our work to provide multi-label segmentation and detection.

Finally, in Chapter 5 we proposed the last step of the transition between deep neural network and spiking bio-plausible networks. This work discusses the implementation of C-SNNs for brain tumor classification and aims to prove the efficiency of SNNs in solving complex vision tasks like the ones exposed throughout this thesis. We first proposed a data processing pipeline to let the computational workload of the network be as low as possible and to facilitate its training. Then we reviewed two brain tumor classification tasks with different levels of complexity to have a better proof of the performances of spike-based models in medical image analysis. A full description of the proposed network was given along with a bio-plausible supervised learning rule based on a reward / punishment system. Implemented using a redefined neural simulator, the C-SNN proved its classification abilities in both task and could easily be trained on CPU resources in less than an hour. A comparison to other state-of-the-art methods showed that a single trainable layer C-SNN could approach and even slightly outperform some deep and machine learning based solutions. However, although our model was significantly more computationally efficient than the CNNs exposed in the comparison, most of them obtained high accuracy scores. The work in this chapter thus succeeded in proving the opportunities offered by SNNs in computer vision and more importantly in the medical field as they do not only provide lighter and less expensive recognition systems, they also let medical image analysis step towards bio-plausible solutions that can bring the explainability required in clinical settings.

6.0.2 Future works

The work covered in this manuscript does not seek to outperform state-of-the-art models for brain tumor diagnosis but addresses real requirements needed in order to realistically use deep learning based solutions in the medical field. If we were thus successful in building brain tumor diagnosis systems while considering these requirements, some points still need to be addressed to find more efficient or more general solutions.

Firstly, we believe that the development of compressed CNNs could be further investigated in the embedded domain by studying the use of smaller cost-efficient hardware like Field Programmable Gate Arrays (FPGAs). Indeed, FPGAs can provide outstanding flexibility and cost-efficiency when building deep learning applications. However, implementing neural networks on FPGAs is not as simple as using common frameworks like Tensorflow or PyTorch. Although frameworks like Vitis AI appeared to address this issue by easing the deployment of deep learning inference on FPGAs, it does not rely on a hardware description language and consequently does not use their full potential. Some works could thus study this possibility to move the computation of brain tumor diagnosis systems to FPGAs. This

type of hardware could also not only provide the cost-efficiency needed for deep learning applications but also be used to train and deploy SNNs which can benefit from the low latency responses in these platforms.

Secondly, further investigations could be carried on the use of SNNs for medical image analysis. If our work proposed the study of PCNN and C-SNN models, we did not deeply explore the domain of spiking neurons. Although, we carried some experiments to build multi-layer SNNs like the one proposed by Diehl and Cook [64] to detect or segment tumors, they were mostly tedious to tune or analyse and did not appeared in this final work. There is thus room for investigations to build the tools necessary to have a better understanding of neural computation and its application to computer vision. In fact, most well-know neural simulators like Brian2, PyNN, ANNarchy or Nengo were not designed to build deep learning like applications and do not ease the training of SNNs for complex tasks.

Finally, the unavailability of high quality and large brain imaging data is forcing researchers in the domain to solve the same tasks or focus their investigations on specific types of tumors, like we did for the glioma. These datasets are also often pre-processed and provide the same MRI sequences what can lead to build models that are failing when scans with other acquisition types are added to the experiments. Hence, we believe that generative models could be investigated to construct larger MRI datasets while adding the variations needed for generalization. Models like Generative Adversarial Networks could for example be trained to enlarge small open-source dataset or create brand new cases of scans showing brain tumors of any type.

Publications

Journal Articles

1. Niepceron, B., A. Nait-Sidi-Moh and F. Grassia. “Moving Medical Image Analysis to GPU Embedded Systems: Application to Brain Tumor Segmentation.” Applied Artificial Intelligence 34 (2020): 866 - 879.

Conference Articles

1. Niepceron, B., A. Nait-Sidi-Moh and F. Grassia. “Study of Pulse-Coupled Neural Network for Glioma Segmentation.” In Proceedings, 26th International Symposium on Artificial Life and Robotics (2020): 110 - 115. Young Author Award.

Working Titles

1. Niepceron, B., A. Nait-Sidi-Moh and F. Grassia. ”Brain tumor detection using Selective Search and Pulse-Coupled Neural Network feature extraction”. Submitted and Accepted in International Conference on Informatics Revolution for Smarter Healthcare 2021.

2. Niepceron, B., A. Nait-Sidi-Moh and F. Grassia. “Study of Pulse-Coupled NeuralNetwork for Glioma Segmentation.” Submitted in Applied Artificial Intelligence.

Seminars

1. **Colloque Droit & Médecine** - September 2021
   Artificial Intelligence in the medical field.

2. **EU Interreg AiBle** - July 2021
   AI & Exoskeleton Workshop Webinar.
   Move brain tumor diagnosis to cost-efficient systems.

3. **Journée de la SAGIP** - July 2021
   Developing brain tumor diagnosis applications using Artificial Neural Networks.

4. **6ème Journées Régionales des Doctorants de l'Automatique** - July 2019
   Brain tumor segmentation using convolutional neural networks.

5. **Journée des Doctorants du LTI** - June 2019
   Compressed convolutional neural network for brain tumor segmentation.
   Best poster award.

References

[1] G. Bi and M. Poo. Synaptic modifications in cultured hippocampal neurons: Dependence on spike timing, synaptic strength, and postsynaptic cell type. *The Journal of Neuroscience*, 18:10464 – 10472, 1998.

[2] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[3] Catherine D. Schuman, T. Potok, R. Patton, J. Birdwell, M. Dean, G. Rose, and J. Plank. A survey of neuromorphic computing and neural networks in hardware. *ArXiv*, abs/1705.06963, 2017.

[4] Y. LeCun, P. Haffner, L. Bottou, and Yoshua Bengio. Object recognition with gradient-based learning. In *Shape, Contour and Grouping in Computer Vision*, 1999.

[5] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[6] Donald Olding Hebb. *The organization of behavior: A neuropsychological theory*. Psychology Press, 2005.

[7] Frank F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65 6:386–408, 1958.

[8] Vijay Badrinarayanan, Alex Kendall, and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39:2481–2495, 2017.

[9] Shaoqing Ren, Kaiming He, Ross B. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39:1137–1149, 2015.

[10] Eunhee Kang, Junhong Min, and J. C. Ye. A deep convolutional neural network using directional wavelets for low-dose x-ray ct reconstruction. *Medical Physics*, 44:e360–e375, 2017.

[11] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014.

[12] Hao Dong, Guang Yang, Fangde Liu, Yuanhan Mo, and Yike Guo. Automatic brain tumor detection and segmentation using u-net based fully convolutional networks. *Medical Image Understanding and Analysis*, page 506–517, 2017.

[13] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications, 2017.

[14] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[15] J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, June 2009.

[16] N. Hawkes. Cancer survival data emphasise importance of early diagnosis. *British Medical Journal*, 364, 2019.

[17] Adriano Lucieri, Muhammad Naseer Bajwa, A. Dengel, and Sheraz Ahmed. Achievements and challenges in explaining deep learning based computer-aided diagnosis systems. *ArXiv*, abs/2011.13169, 2020.

[18] G. Litjens, Thijs Kooi, B. E. Bejnordi, A. Setio, F. Ciompi, M. Ghafoorian, J. V. D. Laak, B. Ginneken, and C. Sánchez. A survey on deep learning in medical image analysis. *Medical image analysis*, 42:60–88, 2017.

[19] F. Milletari, N. Navab, and Seyed-Ahmad Ahmadi. V-net: Fully convolutional neural networks for volumetric medical image segmentation. *2016 Fourth International Conference on 3D Vision (3DV)*, pages 565–571, 2016.

[20] Qing Li, Weidong Cai, Xiaogang Wang, Yun Zhou, D. Feng, and Mei Chen. Medical image classification with convolutional neural network. *2014 13th International Conference on Control Automation Robotics & Vision (ICARCV)*, pages 844–848, 2014.

[21] Md. Zahangir Alom, M. Hasan, C. Yakopcic, T. Taha, and V. Asari. Recurrent residual convolutional neural network based on u-net (r2u-net) for medical image segmentation. *ArXiv*, abs/1802.06955, 2018.

[22] Mohammad Havaei, Axel Davy, David Warde-Farley, Antoine Biard, Aaron Courville, Yoshua Bengio, Chris Pal, Pierre-Marc Jodoin, and Hugo Larochelle. Brain tumor segmentation with deep neural networks. *Medical Image Analysis*, 35:18–31, Jan 2017.

[23] Bumshik Lee, Nagaraj Yamanakkanavar, and J. Choi. Automatic segmentation of brain mri using a novel patch-wise u-net deep architecture. *PLoS ONE*, 15, 2020.

[24] B. H. Menze, A. Jakab, S. Bauer, J. Kalpathy-Cramer, K. Farahani, J. Kirby, Y. Burren, N. Porz, J. Slotboom, R. Wiest, L. Lanczi, E. Gerstner, M. Weber, T. Arbel, B. B. Avants, N. Ayache, P. Buendia, D. L. Collins, N. Cordier, J. J. Corso, A. Criminisi, T. Das, H. Delingette, C. Demiralp, C. R. Durst, M. Dojat, S. Doyle, J. Festa, F. Forbes, E. Geremia, B. Glocker, P. Golland, X. Guo, A. Hamamci, K. M. Iftekharuddin, R. Jena, N. M. John, E. Konukoglu, D. Lashkari, J. A. Mariz, R. Meier, S. Pereira, D. Precup, S. J. Price, T. R. Raviv, S. M. S. Reza, M. Ryan, D. Sarikaya, L. Schwartz, H. Shin, J. Shotton, C. A. Silva, N. Sousa, N. K. Subbanna, G. Szekely, T. J. Taylor, O. M. Thomas, N. J. Tustison, G. Unal, F. Vasseur, M. Wintermark, D. H. Ye, L. Zhao, B. Zhao, D. Zikic, M. Prastawa, M. Reyes, and K. Van Leemput. The multimodal brain tumor image segmentation benchmark (brats). *IEEE Transactions on Medical Imaging*, 34(10):1993–2024, Oct 2015.

[25] Li Liu, Wanli Ouyang, Xiaogang Wang, P. Fieguth, J. Chen, Xinwang Liu, and M. Pietikäinen. Deep learning for generic object detection: A survey. *International Journal of Computer Vision*, 128:261–318, 2019.

[26] D. Lu and Qihao Weng. A survey of image classification methods and techniques for improving classification performance. *International Journal of Remote Sensing*, 28:823 – 870, 2007.

[27] A. Garcia-Garcia, S. Orts, Sergiu Oprea, Victor Villena-Martinez, P. Martinez-Gonzalez, and J. G. Rodríguez. A survey on deep learning techniques for image and video semantic segmentation. *Appl. Soft Comput.*, 70:41–65, 2018.

[28] Neil C Thompson, Kristjan H. Greenewald, Keeheon Lee, and Gabriel F. Manso. The computational limits of deep learning. *ArXiv*, abs/2007.05558, 2020.

[29] Y. LeCun. The power and limits of deep learning. *Research-Technology Management*, 61:22 – 27, 2018.

[30] Emma Strubell, Ananya Ganesh, and A. McCallum. Energy and policy considerations for deep learning in nlp. In *ACL*, 2019.

[31] A. Arrieta, Natalia D'iaz-Rodr'iguez, J. Ser, Adrien Bennetot, S. Tabik, A. Barbado, Salvador Garc'ia, Sergio Gil-L'opez, D. Molina, Richard Benjamins, Raja Chatila, and Francisco Herrera. Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *ArXiv*, abs/1910.10045, 2020.

[32] Ravid Shwartz-Ziv and Naftali Tishby. Opening the black box of deep neural networks via information. *ArXiv*, abs/1703.00810, 2017.

[33] A. Tavanaei, M. Ghodrati, S. R. Kheradpisheh, T. Masquelier, and A. Maida. Deep learning in spiking neural networks. *Neural networks : the official journal of the International Neural Network Society*, 111:47–63, 2019.

[34] Filippo Giovanni Grassia. *Silicon neural networks : implementation of cortical cells to improve the artificial-biological hybrid technique*. Theses, Université Sciences et Technologies - Bordeaux I, January 2013.

[35] A. Georghiades, P. Belhumeur, and D. Kriegman. From few to many: Illumination cone models for face recognition under variable lighting and pose. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23:643–660, 2001.

[36] J. Leksut, Jiaping Zhao, and L. Itti. Learning visual variation for object recognition. *Image Vis. Comput.*, 98:103912, 2020.

[37] J. Wright, Allen Y. Yang, Arvind Ganesh, S. Sastry, and Y. Ma. Robust face recognition via sparse representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31:210–227, 2009.

[38] Ross B. Girshick, Jeff Donahue, Trevor Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 580–587, 2014.

[39] J. Uijlings, K. V. D. Sande, T. Gevers, and A. Smeulders. Selective search for object recognition. *International Journal of Computer Vision*, 104:154–171, 2013.

[40] M. S. Hoque and M. Fairhurst. Face recognition using the moving window classifier. In *BMVC*, 2000.

[41] Jianbo Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22:888–905, 2000.

[42] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24:603–619, 2002.

[43] P. Arbeláez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33:898–916, 2011.

[44] Pedro F. Felzenszwalb and D. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59:167–181, 2004.

[45] Ayodeji Olalekan Salau and Shruti Jain. Feature extraction: A survey of the types, techniques, applications. *2019 International Conference on Signal Processing and Communication (ICSC)*, pages 158–164, 2019.

[46] Leila Kabbai, M. Abdellaoui, and A. Douik. Image classification by combining local and global features. *The Visual Computer*, 35:679–693, 2018.

[47] Daniel Bashir, George D. Montañez, Sonia Sehra, Pedro Sandoval Segura, and Julius Lauw. An information-theoretic perspective on overfitting and underfitting. *ArXiv*, abs/2010.06076, 2020.

[48] Connor Shorten and T. Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6:1–48, 2019.

[49] Yuanqing Lin, Fengjun Lv, Shenghuo Zhu, Ming Yang, Timothée Cour, Kai Yu, L. Cao, and Thomas S. Huang. Large-scale image classification: Fast feature extraction and svm training. *CVPR 2011*, pages 1689–1696, 2011.

[50] K. Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36:193–202, 2004.

[51] Y. LeCun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, and L. Jackel. Handwritten digit recognition with a back-propagation network. In *NIPS*, 1989.

[52] S. Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *Int. J. Uncertain. Fuzziness Knowl. Based Syst.*, 6:107–116, 1998.

[53] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010.

[54] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network. *ArXiv*, abs/1505.00853, 2015.

[55] Djork-Arné Clevert, Thomas Unterthiner, and S. Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv: Learning*, 2016.

[56] D. Louis, H. Ohgaki, O. Wiestler, W. Cavenee, P. Burger, A. Jouvet, B. Scheithauer, and P. Kleihues. The 2007 who classification of tumours of the central nervous system. *Acta Neuropathologica*, 114:97 – 109, 2007.

[57] D. Louis. Molecular pathology of malignant gliomas. *Annual review of pathology*, 1:97–117, 2006.

[58] A. Olar and K. Aldape. Using the molecular classification of glioblastoma to inform personalized treatment. *The Journal of Pathology*, 232, 2014.

[59] E. Izhikevich. Which model to use for cortical spiking neurons? *IEEE Transactions on Neural Networks*, 15:1063–1070, 2004.

[60] A. Hodgkin and A. Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of Physiology*, 117, 1952.

[61] E. Izhikevich. Simple model of spiking neurons. *IEEE transactions on neural networks*, 14 6:1569–72, 2003.

[62] A. Borst and F. Theunissen. Information theory and neural coding. *Nature Neuroscience*, 2:947–957, 1999.

[63] D. Hubel and T. Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of Physiology*, 160, 1962.

[64] Peter U. Diehl and Matthew Cook. Unsupervised learning of digit recognition using spike-timing-dependent plasticity. *Frontiers in Computational Neuroscience*, 9, 2015.

[65] S. Thorpe, A. Delorme, and R. V. Rullen. Spike-based strategies for rapid processing. *Neural networks : the official journal of the International Neural Network Society*, 14 6-7:715–25, 2001.

[66] S. Thorpe and J. Gautrais. Rank order coding. 1998.

[67] Sen Song, K. D. Miller, and L. Abbott. Competitive hebbian learning through spike-timing-dependent synaptic plasticity. *Nature Neuroscience*, 3:919–926, 2000.

[68] A. Watt and N. S. Desai. Homeostatic plasticity and stdp: Keeping a neuron's cool in a fluctuating world. *Frontiers in Synaptic Neuroscience*, 2, 2010.

[69] Wenrui Zhang and P. Li. Information-theoretic intrinsic plasticity for online unsupervised learning in spiking neural networks. *Frontiers in Neuroscience*, 13, 2019.

[70] M. Goodale and A. Milner. Separate visual pathways for perception and action. *Trends in Neurosciences*, 15:20–25, 1992.

[71] D. Pollen and S. Ronner. Phase relationships between adjacent simple cells in the visual cortex. *Science*, 212 4501:1409–11, 1981.

[72] J. Daugman. Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters. *Journal of the Optical Society of America. A, Optics and image science*, 2 7:1160–9, 1985.

[73] Reinhard Eckhorn, HJ Reitbock, M Arndt, and P Dicke. A neural network for feature linking via synchronous activity: results from cat visual cortex and from simulations. 1989.

[74] G Kuntimad and Heggere S Ranganath. Perfect image segmentation using pulse coupled neural networks. *IEEE transactions on neural networks*, 10(3):591–598, 1999.

[75] Xiaodong Gu, Yuanyuan Wang, and Liming Zhang. Object detection using unit-linking pcnn image icons. In Jun Wang, Zhang Yi, Jacek M. Zurada, Bao-Liang Lu, and Hujun Yin, editors, *Advances in Neural Networks - ISNN 2006*, pages 616–622, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[76] Mario I. Chacon M., Alejandro Zimmerman S., and Pablo Rivas P. Image processing applications with a pcnn. In Derong Liu, Shumin Fei, Zengguang Hou, Huaguang Zhang, and Changyin Sun, editors, *Advances in Neural Networks – ISNN 2007*, pages 884–893, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[77] N. Kasabov, K. Dhoble, Nuttapod Nuntalid, and G. Indiveri. Dynamic evolving spiking neural networks for on-line spatio- and spectro-temporal pattern recognition. *Neural networks : the official journal of the International Neural Network Society*, 41:188–201, 2013.

[78] F. Grassia, L. Buhry, T. Levi, J. Tomas, A. Destexhe, and S. Saïghi. Tunable neuromimetic integrated system for emulating cortical neuron models. *Frontiers in Neuroscience*, 5, 2011.

[79] Bodo Rueckauer, Iulia-Alexandra Lungu, Yuhuang Hu, Michael Pfeiffer, and Shih-Chii Liu. Conversion of continuous-valued deep networks to efficient event-driven networks for image classification. *Frontiers in Neuroscience*, 11, 2017.

[80] Jianhao Ding, Zhaofei Yu, Yonghong Tian, and Tiejun Huang. Optimal ann-snn conversion for fast and accurate inference in deep spiking neural networks. In *IJCAI*, 2021.

[81] S. Bohté, J. Kok, and H. L. Poutré. Spikeprop: backpropagation for networks of spiking neurons. In *ESANN*, 2000.

[82] B. Meftah, O. Lézoray, S. Chaturvedi, A. Khurshid, and A. Benyettou. Image processing with spiking neuron networks. In *Artificial Intelligence, Evolutionary Computing and Metaheuristics*, 2013.

[83] Q. Wu, T. McGinnity, L. Maguire, A. Belatreche, and B. Glackin. Edge detection based on spiking neural network model. In *ICIC*, 2007.

[84] Q. Wu, T. McGinnity, L. Maguire, German D. Valderrama-Gonzalez, and P. Dempster. Colour image segmentation based on a spiking neural network model inspired by the visual system. In *ICIC*, 2010.

[85] Zewen Li, Wenjie Yang, Shouheng Peng, and Fan Liu. A survey of convolutional neural networks: Analysis, applications, and prospects. *CoRR*, abs/2004.02806, 2020.

[86] Jiantao Qiu, Jie Wang, Song Yao, Kaiyuan Guo, Boxun Li, Erjin Zhou, Jincheng Yu, Tianqi Tang, Ningyi Xu, Sen Song, Yu Wang, and Huazhong Yang. Going deeper with embedded fpga platform for convolutional neural network. In *Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, FPGA '16, pages 26–35, New York, NY, USA, 2016. ACM.

[87] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015.

[88] L. Minh Dang, Kyungbok Min, Dongil Han, Md Jalil Piran, and Hyeonjoon Moon. A survey on internet of things and cloud computing for healthcare, June 2019.

[89] Yuchi Tian, Kexin Pei, Suman Jana, and Baishakhi Ray. Deeptest: Automated testing of deep-neural-network-driven autonomous cars. *Proceedings of the 40th International Conference on Software Engineering - ICSE '18*, 2018.

[90] S. Alyamkin, M. Ardi, Achille Brighton, A. Berg, Yiran Chen, Hsin-Pai Cheng, Bo Chen, Zichen Fan, Chen Feng, Bo Fu, Kent W. Gauen, Jongkook Go, A. Goncharenko, Xuyang Guo, Hong Hanh Nguyen, Andrew G. Howard, Yuanjun Huang, Donghyun Kang, Jaeyoung Kim, A. Kondratyev, Seungjae Lee, Suwoong Lee, Junhyeok Lee, Zhiyu Liang, Xin Liu, Juzheng Liu, Zi mei Li, Yang Lu, Yung-Hsiang Lu, Deeptanshu Malik, Eunbyung Park, Denis Repin, Tao Sheng, Liang Shen, Fei Sun, D. Svitov, G. K. Thiruvathukal, Baiwu Zhang, Jingchi Zhang, Xiaopeng Zhang, and Shaojie Zhuo. 2018 low-power image recognition challenge. *ArXiv*, abs/1810.01732, 2018.

[91] Roger Pujol, Hamid Tabani, Leonidas Kosmidis, Enrico Mezzetti, Jaume Abella, and Francisco J. Cazorla. Generating and exploiting deep learning variants to increase heterogeneous resource utilization in the nvidia xavier. In Sophie Quinton, editor, *31st Euromicro Conference on Real-Time Systems (ECRTS 2019)*, volume 133 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 23:1–23:23, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[92] Sparsh Mittal and Jeffrey S. Vetter. A survey of methods for analyzing and improving gpu energy efficiency. *ACM Comput. Surv.*, 47(2):19:1–19:23, August 2014.

[93] N. Tijtgat, W. V. Ranst, B. Volckaert, T. Goedemé, and F. D. Turck. Embedded real-time object detection for a uav warning system. In *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, pages 2110–2118, Oct 2017.

[94] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning internal representations by error propagation. In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1: Foundations*, pages 318–362. MIT Press, Cambridge, MA, 1986.

[95] Y. LeCun, L. Bottou, G. Orr, and K. Muller. Efficient backprop. In G. Orr and Muller K., editors, *Neural Networks: Tricks of the trade*. Springer, 1998.

[96] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 448–456. JMLR.org, 2015.

[97] Sergey Ioffe. Batch renormalization: Towards reducing minibatch dependence in batch-normalized models. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 1945–1953. Curran Associates, Inc., 2017.

[98] Chao Peng, Tete Xiao, Zeming Li, Yuning Jiang, Xiangyu Zhang, Kai Jia, Gang Yu, and Jian Sun. Megdet: A large mini-batch object detector. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun 2018.

[99] Yuxin Wu and Kaiming He. Group normalization. *Lecture Notes in Computer Science*, page 3–19, 2018.

[100] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization, 2016.

[101] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016.

[102] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, June 2016.

[103] Guangyong Chen, Pengfei Chen, Yujun Shi, Chang-Yu Hsieh, Benben Liao, and Shengyu Zhang. Rethinking the usage of batch normalization and dropout in the training of deep neural networks, 2019.

[104] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[105] Laurent Sifre and Stéphane Mallat. Rigid-motion scattering for texture classification. *CoRR*, abs/1403.1687, 2014.

[106] François Chollet. Xception: Deep learning with depthwise separable convolutions. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1800–1807, 2017.

[107] X. Zhang, X. Zhou, M. Lin, and J. Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6848–6856, June 2018.

[108] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, June 2018.

[109] Ram Krishna Pandey, Aswin Vasan, and A G Ramakrishnan. "segmentation of liver lesions with reduced complexity deep models", 2018.

[110] J. A. Parker, R. V. Kenyon, and D. E. Troxel. Comparison of interpolating methods for image resampling. *IEEE Transactions on Medical Imaging*, 2(1):31–39, March 1983.

[111] E. Park, J. Ahn, and S. Yoo. Weighted-entropy-based quantization for deep neural networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7197–7205, July 2017.

[112] Szymon Migacz. Gpu technology conference. In *8-bit Inference with TensorRT*, 5 2017.

[113] N. J. Tustison, B. B. Avants, P. A. Cook, Y. Zheng, A. Egan, P. A. Yushkevich, and J. C. Gee. N4itk: Improved n3 bias correction. *IEEE Transactions on Medical Imaging*, 29(6):1310–1320, June 2010.

[114] Bradley C. Lowekamp, David T. Chen, Luis Ibáñez, and Daniel J. Blezek. The design of simpleitk. In *Front. Neuroinform.*, 2013.

[115] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[116] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Yee Whye Teh and Mike Titterington, editors,

Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR.

[117] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[118] B. Niepceron, A. Nait-Sidi-Moh, and F. Grassia. Moving medical image analysis to gpu embedded systems: Application to brain tumor segmentation. *Applied Artificial Intelligence*, 34:866 – 879, 2020.

[119] Xiaomei Zhao, Yihong Wu, Guidong Song, Zhenye Li, Yazhuo Zhang, and Yong Fan. A deep learning model integrating fcnns and crfs for brain tumor segmentation. *Medical Image Analysis*, 43:98 – 111, 2018.

[120] S. Pereira, A. Pinto, V. Alves, and C. A. Silva. Brain tumor segmentation using convolutional neural networks in mri images. *IEEE Transactions on Medical Imaging*, 35(5):1240–1251, May 2016.

[121] Geoffrey E. Hinton, Oriol Vinyals, and J. Dean. Distilling the knowledge in a neural network. *ArXiv*, abs/1503.02531, 2015.

[122] C. Neftel, J. Laffy, Mariella G Filbin, T. Hara, M. Shore, G. Rahme, Alyssa R. Richman, Dana Silverbush, McKenzie L. Shaw, Christine M. Hebert, J. DeWitt, S. Gritsch, Elizabeth M. Perez, L. Castro, X. Lan, Nicholas Druck, C. Rodman, Danielle Dionne, Alexander B Kaplan, Mia S Bertalan, J. Small, K. Pelton, Sarah Becker, D. Bonal, Q. Nguyen, Rachel L. Servis, Jeremy M. Fung, R. Mylvaganam, Lisa Mayr, Johannes Gojo, C. Haberler, R. Geyeregger, T. Czech, I. Slavc, B. Nahed, W. Curry, B. Carter, H. Wakimoto, P. Brastianos, T. Batchelor, A. Stemmer-Rachamimov, M. Martinez-Lage, M. Frosch, I. Stamenkovic, Nicolò Riggi, Esther Rheinbay, M. Monje, O. Rozenblatt-Rosen, D. Cahill, Anoop P. Patel, T. Hunter, I. Verma, K. Ligon, David N. Louis, A. Regev, B. Bernstein, I. Tirosh, and M. Suvà. An integrative model of cellular states, plasticity, and genetics for glioblastoma. *Cell*, 178:835–849.e21, 2019.

[123] Sharan Kumar and Dattatreya P. Mankame. Optimization driven deep convolution neural network for brain tumor classification. *Biocybernetics and Biomedical Engineering*, 40(3):1190 – 1204, 2020.

[124] M. Bada and M. Barjaktarovic. Classification of brain tumors from mri images using a convolutional neural network. *Applied Sciences*, 10:1999, 2020.

[125] N. Arunkumar, M. A. Mohammed, S. A. Mostafa, D. Ibrahim, J. Rodrigues, and V. Albuquerque. Fully automatic model-based segmentation and classification approach for mri brain tumor using artificial neural networks. *Concurrency and Computation: Practice and Experience*, 32, 2020.

[126] Xiao-Dong Gu, Shi-De Guo, and Dao-Heng Yu. A new approach for automated image segmentation based on unit-linking pcnn. In *Proceedings. International Conference on Machine Learning and Cybernetics*, volume 1, pages 175–178. IEEE, 2002.

[127] Kun Zhan, Jinhui Shi, Qiaoqiao Li, Jicai Teng, and Mingying Wang. Image segmentation using fast linking scm. In *2015 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2015.

[128] V. Bhavana and H. Krishnappa. Multi-modality medical image fusion using discrete wavelet transform. *Procedia Computer Science*, 70:625–631, 2015.

[129] A Anoop Suraj, Mathew Francis, TS Kavya, and TM Nirmal. Discrete wavelet transform based image fusion and de-noising in fpga. *Journal of Electrical Systems and Information Technology*, 1(1):72–81, 2014.

[130] Zhaobin Wang and Yide Ma. Medical image fusion using m-pcnn. *Information Fusion*, 9(2):176–185, 2008.

[131] Yaqian Zhao, Qinping Zhao, and Aimin Hao. Multimodal medical image fusion using improved multi-channel pcnn. *Bio-medical materials and engineering*, 23:S221–S228, 11 2013.

[132] J. W. McClurkin, J. A. Zarbock, and L. Optican. Temporal codes for colors, patterns, and memories. 1994.

[133] J.L. Johnson. Time signatures of images. In *Proceedings of 1994 IEEE International Conference on Neural Networks (ICNN'94)*, volume 2, pages 1279–1284 vol.2, 1994.

[134] Xiaodong Gu, Y. Wang, and L. Zhang. Object detection using unit-linking pcnn image icons. In *ISNN*, 2006.

[135] Xiaodong Gu. Feature extraction using unit-linking pulse coupled neural network and its applications. *Neural Processing Letters*, 27:25–41, 2007.

[136] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. *CoRR*, abs/1703.06870, 2017.

[137] Yongqiang Cao, Y. Chen, and D. Khosla. Spiking deep convolutional neural networks for energy-efficient object recognition. *International Journal of Computer Vision*, 113:54–66, 2014.

[138] Saeed Reza Kheradpisheh, Mohammad Ganjtabesh, Simon J. Thorpe, and Timothée Masquelier. Stdp-based spiking deep neural networks for object recognition. *CoRR*, abs/1611.01421, 2016.

[139] Jun Cheng, W. Huang, Shuangliang Cao, Ru Yang, Wei Yang, Z. Yun, Zhijian Wang, and Qianjin Feng. Enhanced performance of brain tumor classification via tumor region augmentation and partition. *PLoS ONE*, 10, 2015.

[140] D. Shattuck, S. Sandor-Leahy, K. Schaper, D. Rottenberg, and R. Leahy. Magnetic resonance image tissue classification using a partial volume model. *NeuroImage*, 13:856–876, 2001.

[141] Pietro Perona and Jitendra Malik. Scale-space and edge detection using anisotropic diffusion. *IEEE Trans. Pattern Anal. Mach. Intell.*, 12:629–639, 1990.

[142] G. Gerig, O. Kübler, R. Kikinis, and F. Jolesz. Nonlinear anisotropic filtering of mri data. *IEEE transactions on medical imaging*, 11 2:221–32, 1992.

[143] K. Krissian and S. Aja-Fernández. Noise-driven anisotropic diffusion filtering of mri. *IEEE Transactions on Image Processing*, 18:2265–2274, 2009.

[144] Caio Palma, F. Cappabianco, J. S. Ide, and P. Miranda. Anisotropic diffusion filtering operation and limitations - magnetic resonance imaging evaluation. *IFAC Proceedings Volumes*, 47:3887–3892, 2014.

[145] M. Riesenhuber and T. Poggio. Hierarchical models of object recognition in cortex. *Nature Neuroscience*, 2:1019–1025, 1999.

[146] Nicolas Frémaux and W. Gerstner. Neuromodulated spike-timing-dependent plasticity, and theory of three-factor learning rules. *Frontiers in Neural Circuits*, 9, 2016.

[147] E. Izhikevich. Solving the distal reward problem through linkage of stdp and dopamine signaling. *BMC Neuroscience*, 8:S15 – S15, 2007.

[148] Milad Mozafari, M. Ganjtabesh, A. Nowzari-Dalini, and T. Masquelier. Spyketorch: Efficient simulation of convolutional spiking neural networks with at most one spike per neuron. *Frontiers in Neuroscience*, 13, 2019.

[149] Subhashis Banerjee, S. Mitra, F. Masulli, and S. Rovetta. Deep radiomics for brain tumor detection and classification from multi-sequence mri. *ArXiv*, abs/1903.09240, 2019.

[150] F. P. Polly, S. K. Shil, M. A. Hossain, A. Ayman, and Y. M. Jang. Detection and classification of hgg and lgg brain tumor using machine learning. *2018 International Conference on Information Networking (ICOIN)*, pages 813–817, 2018.

[151] Hiba Mzoughi, Ines Njeh, A. Wali, M. Slima, A. Benhamida, C. Mhiri, and Kharedine Ben Mahfoudhe. Deep multi-scale 3d convolutional neural network (cnn) for mri gliomas brain tumor classification. *Journal of Digital Imaging*, pages 1–13, 2020.

[152] Ge Cui, J. Jeong, Bob Press, Y. Lei, H. Shu, Tian xing Liu, W. Curran, H. Mao, and Xiaofeng Yang. Machine-learning-based classification of lower-grade gliomas and high-grade gliomas using radiomic features in multi-parametric mri. *arXiv: Medical Physics*, 2019.

[153] F. Díaz-Pernas, M. Martínez-Zarzuela, M. Antón-Rodríguez, and D. González-Ortega. A deep learning approach for brain tumor classification and segmentation using a multiscale convolutional neural network. *Healthcare*, 9, 2021.

[154] Amin Kabir Anaraki, M. Ayati, and Foad Kazemi. Magnetic resonance imaging-based brain tumor grades classification and grading via convolutional neural networks and genetic algorithms. *Biocybernetics and Biomedical Engineering*, 39:63–74, 2019.

[155] Nyoman Abiwinanda, Muhammad Hanif, S. T. Hesaputra, A. Handayani, and T. Mengko. Brain tumor classification using convolutional neural network. 2019.

[156] A. Pashaei, H. Sajedi, and N. Jazayeri. Brain tumor classification via convolutional neural network and extreme learning machines. *2018 8th International Conference on Computer and Knowledge Engineering (ICCKE)*, pages 314–319, 2018.

Appendices

Appendix A

Implementation and reproduction

The entirety of experiments discussed in this work were designed using the Python programming language and can all be found in distinct github repositories to use for reproduction or extension. To ensure that the code can be reused, we also provide a Dockerfile for each implementation. This document gives instructions on how to retrieve and run these implementation.

A.1 BraTS 2015 data pipeline

The pipeline used in Chapter 3 can be found in the following repository : https://github.com/bniepce/mmiages-data-pipeline. Once the dataset downloaded and the setup steps followed, the script can be launched with the following command line :

```
$ python make_training_data.py
```

The program asks for path variables to locate the BraTS training, validation and test datasets, and save the tfrecord files containing the processed data. The pipeline was designed as a Python generator and can thus be extended by simply providing new image processing functions under `./pipeline/processing.py`. These functions take a tuple containing the images and the ground truth as input and return the same tuple containing the transformed data. Minor modifications can be done to support new version of the BraTS dataset. Indeed, the get_scan function in `./pipeline/processing.py` supports the listing of mha files but can easily be modified to list the nii files of the newer versions of the dataset.

Other types of data augmentations can also be used by providing new functions under `./pipeline/augmentations.py` and add them to the aug_operations list variable in the same file. Note that the augmentations functions in this script use the tensorlayer library, new augmentation functions will thus need to have the same input and output shapes.

A.2 Compressed U-Net

The implementation of our compressed U-Net is available at : https://github.com/bniepce/mmiages-neural-network. Launching the training of the network can be done by simply running the bash file under `./scripts/train_default.sh` or by calling the `./train.py` file with the required flags as follows :

```
python train.py \
    --data_path "./data/" \
    --log_dir "log/" \
    --optimizer "Adam" \
    --learning_rate 0.0001 \
    --batch_size 64 \
    --epochs 10 \
    --job_name training_unet
```

When the training is completed the model is saved an H5 file and can be reuse for transfer learning or inference. Running inference can be done two ways, namely native and tensorrt. The scripts to run both methods can be found under `./experiments/inference/`. The `./predict.py` file and the `./scripts/predict_default.sh` script can also be used to run a tensorrt inference given the right data and model file path. The scripts then performs the segmentation of the given data and displays each segmentation map found over the MRI slices in a matplotlib grid. A video illustration of this process working on the JAX embedded platform can be found at : https://www.youtube.com/watch?v=33nlnjTrmSc.

A.3 PCNN Models for segmentation

The implementation of the PCNN models used in Chapter 4 for brain tumor segmentation can be found at : https://github.com/bniepce/pcnn-brain-tumor-segmentation. Three scripts are available to reproduce our experiments :

1. `./run_all.py` : Evaluate the segmentation on a list of scans contained in an h5 file. This script takes a `dataset_path` and a `param_file` flag to provide the path to the h5

file containing the dataset and the json file containing the parameters of the PCNNs. By default, it evaluate all models presented in the chapter but this can be changed by removing or adding new models to the `models` list found in the script.

2. `./run_single.py` : Unlike the previous script, this one only evaluate one MRI case for segmentation. It takes the same flags with an additional one called `image_indice` to choose which case will be segmented in the dataset. By default, this evaluation is done using the Unit-linking PCNN but can be changed by redefining the `model` object.

3. `./run_optimization.py` : This script performs the discussed optimization scheme based on Differential Evolution. By default, it is ran on an PCNN models and only needs the `dataset_path` flag.

The repository also provides default parameters for each PCNN in the `params.json` file.

A.4 PCNN Models for detection

The experiments carried for our detection algorithm presented in Chapter 4 is available at : https://github.com/bniepce/pcnn-brain-tumor-detection . It uses the PCNN models found in the PCNN tumor segmentation repository mentioned above. Unlike the previous implementation, this one was done in a jupyter notebook and can be used by executing the cells in the `tumor_detection.ipynb` file.

A.5 C-SNN development framework

The modified SpykeTorch framework as well as the code running the experiments carried in Chapter 5 are available at : https://github.com/bniepce/csnn-brain-tumor-classification

By default the experiments are ran on the available device found on the host computer. Hence if a GPU with CUDA is present, the networks will be computed using the GPU support, otherwise, the CPU will be used. Two scripts are provided to run each experiment in this repository, namely `ct1.sh` and `ct2.sh`. The user can also use the code present in the repository as a framework to build C-SNNs with the SpykeTorch simulator. To do so, the folder structure is as follows :

1. **dataset** : Provides the classes to load and encode data.

2. **estimator** : The SupervisedEstimator allows to automatically run the training of a C-SNN in a supervised way.

3. **models** : They are predefined C-SNNs from other works also available on the SpykeTorch repository to test the framework.

4. **topology** : Holds all the classes for network definition such as convolutional layers or pooling layers.

5. **utils** : Provides the SpykeTorch utils.

Appendix B

Covid-19 Task Force

The appearance and increasing number of Covid-19 infections during the year 2020 enlisted many researchers in the fight against the virus. The crisis had major negative effects on hospitals as they were quickly overwhelmed by an important amount of patients to treat. During the early stages of the pandemic, an increasing amount of daily positive cases was recorded in northern France. Regional hospitals were thus seeking for research and engineering support to take action against the consequences of the pandemic. To contain the upcoming infection wave, the University Hospital of Amiens-Picardie called up for the creation of a Task Force composed of researchers from local laboratories.

In the context of this research project initiated by the MIS laboratory in Amiens, this Task Force aimed to decrease the amount of people waiting at the emergency service and anticipate the number of hospitalizations due to Covid-19. Indeed, during the crisis, the average waiting time for hospitalization was of about 1h30 because of the increasing number of patients going to the hospital with or without Covid-19 symptoms. Ideally, the emergency staff should have taken care of incoming patients in no longer than 15 minutes.

Our contribution was thus sought to develop several models for predicting these hospitalizations on the basic of retrospective data. These models had to be used when a patient entered the emergency services and indicate, with the best possible reliability, the need for a hospital bed according to a predicted severity index. Two different classification tasks were then designed. The first one was binary, and aimed at deciding if a patient should be kept in the hospital rooms or not. The other one aimed to choose the destination of the patient to provide a more accurate decision making.

Accounting for the emergency to use such a tool, the Task Force was set up for 6 days

at the end of which the predictive models had to be integrated as a software used by the hospital's emergency staff. The data was composed of numerical and categorical variables summing up the patients state at the caring moment like blood pressure, body temperature or their level of pain. The variables were recorded between 2015 and 2019 for roughly 200,000 patients. To solve the binary classification problem and complete the work already done by the research team that called up the Task Force, we proposed the use of XGBoost, a gradient boosting algorithm and optimized it using the Differential Evolution algorithm. The fitness function of the algorithm was chosen to decrease the amount of false positives as needed by the hospital.

At the end of this Task Force, all the models proposed by the teams were merged into one and successfully integrated to the emergency software of Amiens' hospital.

Résumé

Dans le domaine clinique, les réseaux de neurones artificiels sont utilisés pour résoudre des tâches visuelles telles que la détection et la segmentation de tissus malsains. Ayant démontré leurs performances exceptionnelles, ils sont néanmoins contraints par leur besoin important en ressources de calcul qui empêche leur déploiement à grande échelle. L'optimisation ou le remplacement de ces méthodes par des solutions moins dépendantes de la disponibilité de ressources de calcul élevées est donc cruciale. L'objectif de ce travail est de proposer de nouvelles méthodes pour concevoir des systèmes d'analyse d'images médicales, en particulier pour le diagnostic de gliomes. Ce manuscrit couvre trois contributions qui marquent une transition entre les méthodes d'apprentissage profond et le calcul neuronal par le biais de la compression de réseaux de neurones et de l'implémentation des réseaux de neurones à impulsions afin de construire des applications de diagnostic de tumeurs cérébrales efficaces et rapides.

Mots clés

Réseaux de neurones artificiels, vision artificielle, analyse de tumeurs cérébrales, imagerie médicale, neurosciences computationnelles, réseaux de neurones impulsionnels.

Abstract

In the clinical field, Artificial Neural Networks (ANNs) are being used to solve visual tasks such as the detection and segmentation of unhealthy tissues. While they proved outstanding performances, they are constrained by their important need in computational resources, which prevents their large scale deployment. The optimization or replacement of these methods by solutions that are less dependent on the availability of high computational resources is thus crucial. The objective of this work is to propose new ways to design medical image analysis systems, specifically for glioma tumors diagnosis. This manuscript covers three contributions that mark a transition between deep learning methods and spike-based models by the mean of neural network compression and the use of neural computation in order to build efficient and fast brain tumor diagnosis applications.

Keywords

Artificial neural networks, computer vision, brain tumor analysis, medical imaging, computational neurosciences, spiking neural networks.