Réseaux de neurones impulsionnels appliqués à la vision par ordinateur

Veïs Oudjail

THÈSE

en vue de l'obtention du grade de

Docteur
délivré par l'université de Lille

Discipline : Informatique

Laboratoire CRIStAL

École Doctorale MADIS-631

Présentée et soutenue publiquement le 12 décembre 2022
par **Veïs OUDJAIL**

Réseaux de neurones impulsionnels appliqués à la vision par ordinateur

---

Thèse dirigée par : M. Jean MARTINET

Devant le jury formé de :

M.	Chaabane Djeraba	Professeur des universités	*Université de Lille*	Président
M.	Philippe Mulhem	Chargé de recherche HDR	*CR1 CNRS, Grenoble*	Rapporteur
M.	Renaud Péteri	Maître de conférences HDR	*La Rochelle Université*	Rapporteur
Mme.	Fatma Bouali	Professeure des universités	*Université de Lille*	Examinatrice
M.	Bernard Girau	Professeur des universités	*Université de Lorraine*	Examinateur
M.	Jean Martinet	Professeur des universités	*Université Côte d'Azur*	Directeur de thèse

Centre de Recherche en informatique
Signal et Automatique de Lille
(CRIStAL)
Université de Lille - Campus
scientifique - Bâtiment ESPRIT
Avenue Henri Poincaré
59655 Villeneuve d'Ascq

École doctorale MADIS-631
Cité Scientifique - Bât. P3
59655 Villeneuve d'Ascq

Remerciements

Ce manuscrit de thèse présente les réflexions et les résultats découlant des travaux effectués au cours des quatre années de doctorat. Ces travaux ont été le fruit d'une collaboration entre plusieurs personnes, impliquées directement ou non dans la réalisation des différentes parties. C'est pourquoi, avant de débuter ce manuscrit, je tiens à exprimer ma reconnaissance à toutes les personnes ayant participé de près ou de loin à ces travaux de thèse.

Tout d'abord, je tiens à remercier vivement M. Jean MARTINET, professeur des universités et directeur de ma thèse, pour son accueil, le temps passé ensemble et le partage de son expertise au quotidien. Grâce à sa confiance, j'ai pu m'accomplir dans mes missions. Il m'a accompagné durant ces 4 années, m'a challengé au travers de différents défis pour que je puisse m'améliorer à ses côtés. Il fut d'une aide précieuse dans les moments les plus délicats, même sur le plan personnel. Au-delà de ses compétences scientifiques, Jean m'a énormément apporté grâce à ses qualités humaines. Merci infiniment pour tout !

Merci aux rapporteurs Philippe MULHEM et Renaud PÉTERI pour leurs remarques et critiques intéressantes sur ce manuscrit. Merci également aux autres membres du jury, les examinateurs Fatma BOUALI et Bernard GIRAU pour leurs présences et pour leurs questions/commentaires constructifs.

Je tiens à exprimer ma gratitude envers toute l'équipe FOX pour leur accueil et leur esprit d'équipe. Je remercie tout particulièrement M. Pierre TIRILLY, M. Marius BILASCO et M. Chaabane DJERBA pour les moments agréables et les conversations enrichissantes que nous avons partagés. Merci de m'avoir laissé la porte ouverte ! Je souhaite également adresser une pensée à mes collègues avec qui j'ai travaillé quotidiennement : Benjamin ALLAERT pour m'avoir accueilli à mon arrivé et prodigué de précieux conseils, Delphine POUX et Romain BELMONTE qui m'ont accompagné durant cette aventure.

Je veux aussi remercier toutes les personnes qui m'ont conseillé et relu lors de la rédaction de ce document. Je souhaite tout particulièrement remercier ma sœur Somêya OUDJAIL pour son aide précieuse, notamment pour la correction minutieuse des nombreuses figures du chapitre 4, qui ont requis de sa part un investissement en temps considérable. Je suis tout autant reconnaissant envers Victoria BAMBA, qui s'est avérée être une alliée précieuse dans la lutte contre les nombreuses fautes pouvant parasiter un écrit. Je la remercie également pour son soutien moral indispensable dont j'ai pu bénéficier au cours de ces deux dernières années. Tes conseils/encouragements m'ont été précieux et m'ont permis de ne pas abandonner mes projets. Tu es une personne extraordinaire, je te dois énormément, tu as changé beaucoup de choses en moi. Grâce à toi, j'ai pu évoluer, merci

du fond du cœur d'avoir été à mes côtés, je ne t'oublierai jamais.

Je tiens également à remercier les personnes que j'ai rencontrées pendant mes études et avec qui j'ai développé une amitié sincère. Vous avez été une source d'enrichissement intellectuel et humain. Nous avons emprunté le même chemin, le temps de nos études, et avons progressé ensemble. Merci à mon premier binôme Kevin GAMELIN (je ne t'oublie pas malgré les années) pour ta patience et ton esprit d'équipe exceptionnel. Merci également à Marc BALOUP avec qui j'ai mené à bien de nombreux projets, j'ai été honoré de travailler avec une personne aussi intelligente que toi. Un grand merci à Eddy EL KHATIB, dont les réflexions m'ont poussé à me dépasser sur divers sujets. Tu m'as permis d'affûter mon esprit critique. Enfin, je ne peux pas oublier de remercier Alexandre TROUCHAUD, Pape BADIANE et Maxime VILLET pour tous les bons moments, les fous rires et le soutien indéfectible qu'ils m'ont apporté. Sans vous, cette expérience n'aurait pas été la même.

Je remercie enfin l'ensemble de ma famille et de mes amis pour leur soutien et l'intérêt qu'ils ont porté à mes travaux. Mes parents ont toujours été un soutien inconditionnel pour moi et m'ont encouragé à aller au-delà de mes limites dans mes études. Malgré les moments difficiles et les ressources limitées, ils ont su offrir un cadre familial propice à mon épanouissement et m'ont donné une éducation de qualité. Leur persévérance a été la clé de mon succès actuel, et je suis de plus en plus conscient de la valeur inestimable des efforts constants qu'ils ont déployés à mon égard. Évidemment, je remercie mes sœurs Imàne OUDJAIL et Somêya OUDJAIL pour leur bienveillance. Vous êtes des femmes incroyables et vous constituez des piliers essentiels de ma vie. De même, je souhaite exprimer ma profonde reconnaissance envers Samantha NGOUTANE, qui a été très présente pour notre fille, en particulier lors de la rédaction de ce manuscrit. Malgré les obstacles et les imprévus de la vie, tu as fait preuve d'une grande compréhension, de courage et d'une détermination sans faille pour compenser mes absences. Ta façon d'être une maman est une véritable fierté pour moi. Nous avons grandi ensemble et, pour être de meilleurs parents, nous devons constamment poursuivre notre apprentissage. Cela représente notre plus grand défi.

Je voudrais conclure ces remerciements avec quelques mots adressés à ma fille Yéléna. Pour une petite fille de huit ans, tu as développé des qualités et une maturité qui surpassent largement celles de ton père au même âge. Quand tu t'exprimes, je remarque l'intelligence dont tu fais preuve, même sur le plan émotionnel. Tu es mon moteur dans la vie et ma principale source de motivation. Tu as en toi toutes les capacités pour atteindre tes rêves, n'oublie pas que rien ne t'empêche de réaliser ce que tu désires vraiment. Tu me dis souvent que tu es impressionnée par mon savoir, mais sache que tu es capable de surpasser ton père dans chaque domaine si tu t'en donnes les moyens, ne doute jamais de ton potentiel. Peut-être un jour voudras-tu lire mes écrits, mais en attendant, j'ai hâte de découvrir les tiens. La dernière fois, tu m'as épaté en rédigeant ta propre réflexion sur cette célèbre citation "*Être ou ne pas être, telle est la question ?*". Tu as parfaitement fait le lien entre la nécessité de vivre l'instant présent et celle d'avoir des pensées tournées vers le futur. J'attends avec impatience la suite. Je t'aime, ma Yéyé.

- Veïs OUDJAIL

Résumé

Abstract : Artificial neural networks (ANN) have become a must-have technique in computer vision, a trend that started during the 2012 ImageNet challenge. However, this success comes with a non-negligible human cost for manual data labeling, very important in model learning, and a high energy cost caused by the need for large computational resources. Spiking Neural Networks (SNN) provide solutions to these problems. It is a particular class of ANNs, close to the biological model, in which neurons communicate asynchronously by representing information through spikes. The learning of SNNs can rely on an unsupervised rule : the STDP. It modulates the synaptic weights according to the local temporal correlations observed between the incoming and outgoing spikes. Different hardware architectures have been designed to exploit the properties of SNNs (asynchrony, sparse and local operation, etc.) in order to design low-power solutions, some of them dividing the cost by several orders of magnitude. SNNs are gaining popularity and there is growing interest in applying them to vision. Recent work shows that SNNs are maturing by being competitive with the state of the art on "simple" image datasets such as MNIST (handwritten numbers) but not on more complex datasets. However, SNNs can potentially stand out from ANNs in video processing. The first reason is that these models incorporate an additional temporal dimension. The second reason is that they lend themselves well to the use of event-driven cameras. They are bio-inspired sensors that perceive temporal contrasts in a scene, in other words, they are sensitive to motion. Each pixel can detect a light variation (positive or negative), which triggers an event. Coupling these cameras to neuromorphic chips allows the creation of totally asynchronous and massively parallelized vision systems. The objective of this thesis is to exploit the capabilities offered by SNNs in video processing. In order to explore the potential offered by SNNs, we are interested in motion analysis and more particularly in motion direction estimation. The goal is to develop a model capable of learning incrementally, without supervision and with few examples, to extract spatiotemporal features. We have therefore performed several studies examining the different points mentioned using synthetic event datasets. We show that the tuning of the SNN parameters is essential for the model to be able to extract useful features. We also show that the model is able to learn incrementally by presenting it with new classes without deteriorating the performance on the mastered classes. Finally, we discuss some limitations, especially on the weight learning, suggesting the possibility of more delay learning, which are still not very well exploited and which could mark a break with ANNs.

Keywords : *Video Analysis, Spiking Neural Network, Machin Learning, Bio-inspired approach.*

Résumé : Les réseaux de neurones artificiels (RNA) sont devenus des techniques incontournables en vision par ordinateur, cette tendance ayant débuté lors du challenge ImageNet de 2012. Cependant, ce succès s'accompagne d'un coût humain non-négligeable pour l'étiquetage manuel des données, très important dans l'apprentissage des modèles et d'un coût énergétique élevé causé par le besoin de ressources de calcul importantes. Les réseaux de neurones impulsionnels (Spiking Neural Network, SNN) apportent des solutions à ces problématiques. C'est une classe particulière des RNAs, proche du modèle biologique, dans lequel les neurones communiquent de manière asynchrone en représentant l'information via des impulsions (spikes). L'apprentissage des SNN peu reposer sur une règle non supervisée : la STDP. Elle module les poids synaptiques en fonction des corrélations temporelles locales constatées entre les impulsions entrantes et sortantes. Différentes architectures matérielles ont été conçues dans le but d'exploiter les propriétés des SNN (asynchronie, opération éparse et locale, etc.) afin de concevoir des solutions peu énergivores, certaines divisant le coût de plusieurs ordres de grandeur. Les SNNs gagnent en popularité et il y a un intérêt croissant à les appliquer à la vision. Des travaux récents montrent que les SNNs acquièrent en maturité en étant compétitifs par rapport à l'état de l'art sur des datasets d'images "simples" tels que MNIST (chiffres manuscrits) mais pas sur des datasets plus complexes. Cependant, les SNNs peuvent potentiellement se démarquer des RNAs dans le traitement de vidéos. La première raison est que ces modèles intègrent une dimension temporelle en plus. La deuxième raison est qu'ils se prêtent bien à l'utilisation des caméras événementielles. Ce sont des capteurs bio-inspirés qui perçoivent les contrastes temporels d'une scène, autrement dit, ils sont sensibles au mouvement. Chaque pixel peut détecter une variation lumineuse (positive ou négative), ce qui déclenche un événement. Coupler ces caméras aux puces neuromorphiques permet de créer des systèmes de vision totalement asynchrones et massivement parallélisés. L'objectif de cette thèse est d'exploiter les capacités offertes par les SNNs dans le traitement vidéo. Afin d'explorer le potentiel offert par les SNNs, nous nous sommes intéressés à l'analyse du mouvement et plus particulièrement à l'estimation de la direction du mouvement. Le but est de développer un modèle capable d'apprendre incrémentalement, sans supervision et avec peu d'exemples, à extraire des caractéristiques spatio-temporelles. Nous avons donc effectué plusieurs études examinant les différents points mentionnés à l'aide de jeux de données événementielles synthétiques. Nous montrons que le réglage des paramètres des SNNs est essentiel pour que le modèle soit capable d'extraire des caractéristiques utiles. Nous montrons aussi que le modèle est capable d'apprendre de manière incrémentale en lui présentant des classes inédites sans détérioration des performances sur les classes maîtrisées. Pour finir, nous évoquerons certaines limites, notamment sur l'apprentissage des poids en suggérant la possibilité d'apprendre plutôt les délais, encore peu exploités et qui pourrait marquer davantage la rupture face aux RNAs.

Mot clé : *Analyse vidéo, Réseaux de neurones impulsionnels, Apprentissage automatique, Approche bio-inspirée.*

Table des matières

Table des figures

Chapitre 1

Introduction

1.1 Contexte

La vision par ordinateur est devenue en quelques années un domaine stratégique, avec des applications potentielles qui peuvent révolutionner notre quotidien. On peut citer le guidage autonome d'un véhicule, le diagnostic des maladies à partir d'imagerie médicale, etc. Les humains sont capables de réaliser ces tâches en utilisant leur capacité d'analyse, de compréhension et d'apprentissage. De manière générale, les êtres vivants développent des aptitudes permettant de s'adapter aux différents dangers inhérents à l'environnement, au risque de disparaître au profit d'une autre espèce mieux armée. Ce constat favorise, avec le temps, l'émergence d'organismes possédant des systèmes cognitifs de plus en plus sophistiqués. Ces systèmes biologiques répondent aux contraintes imposées par l'environnement. Par exemple, la perception du monde qui nous entoure et notre habilité à agir en conséquence via des prédictions pertinentes sont des mécanismes hérités du processus évolutionniste auquel chaque être vivant est soumis (Poli 2010). L'idée d'une machine qui puisse imiter ces comportements remonte aux prémisses de l'informatique, avec un article visionnaire d'Alan Turing et Haugeland (Turing et Haugeland 1950), considéré par beaucoup comme le père de l'informatique théorique (Bowen 2016). Cette article pose les principes fondamentaux permettant de créer une Intelligence Artificielle (IA). L'un des postulats principaux de l'article énonce qu'il n'est pas possible d'écrire explicitement toutes les instructions d'un algorithme reproduisant une intelligence équivalente à celle d'une personne adulte. La raison est qu'un tel algorithme a une complexité en termes de nombre d'instructions non compressibles (complexité de Kolmogorov 1963) trop importante. Par contre, il est envisageable d'écrire un programme qui imite l'intelligence d'un bébé, avec un processus d'apprentissage qui lui permet d'augmenter sa complexité implicitement avec des données d'entrée. Il décrit ce processus en 3 étapes en utilisant une métaphore liée à l'éducation d'un enfant : l'état initial (la naissance), l'éducation avec une méthode de récompenses et de punitions afin d'inculquer les modèles souhaités dans l'esprit de la machine et la confrontation à d'autres expériences non éducatives. Plus de 72 ans après cette publication, les progrès de l'IA suivent, dans les grandes lignes, le chemin tracé par Turing. L'apprentissage automatique (ou *Machine Learning*, ML) est une discipline de l'IA qui formalise ces principes en proposant différentes méthodes qui offrent les résultats les plus spectaculaires sur différentes problématiques. Certaines de ces

méthodes sont inspirées du fonctionnement du cerveau humain. Le premier modèle de Réseaux de Neurones Artificiels (RNA) proposé est le *perceptron* composé d'une seule couche de neurones (Rosenblatt 1958). Depuis, ces méthodes se sont améliorées avec le développement de l'apprentissage profond (*deep learning*) (LeCun, Bengio et G. Hinton 2015) et sont capables de rivaliser avec les humains sur différentes tâches spécialisées. Les modèles ont évolué en introduisant des architectures de plus en plus efficaces comme : les réseaux de neurones convolutifs (CNN) qui sont plus adaptés pour traiter des images (LeCun, Bottou et al. 1998), les auto-encodeurs qui permettent de reconstruire les données d'entrée (modèles génératifs) en apprenant de manière non supervisée à trouver une compression basée sur des représentations discriminantes et de haut niveau (Jing et Tian 2020 ; Rumelhart, G. E. Hinton et Williams 1985), les réseaux générative adverses qui introduisent une architecture qui met en compétition un réseau générateur qui produit un échantillon et un réseau discriminateur qui essaie de détecter si un échantillon est réel ou bien s'il est le résultat du générateur via un processus d'apprentissage non supervisé suivant un scénario de théorie des jeux(Goodfellow et al. 2014), les réseaux *Transformers* qui ajoutent un mécanisme d'attention (Vaswani et al. 2017), etc. Ces évolutions ont permis de faire des avancées dans des tâches comme la reconnaissance de forme (Yu et al. 2022), le traitement du langage naturel (Brown et al. 2020) pour ne citer que celles-là. Il y a des tâches, inexistantes il y a peu car trop complexes, qui ont vu le jour récemment grâce aux possibilités offertes par l'amélioration des architectures (Alzubaidi et al. 2021). C'est le cas de la génération d'images/vidéos à partir d'une simple description textuelle (Ramesh et al. 2022 ; Singer et al. 2022). Pour que ces techniques apprennent à généraliser à de nouvelles données, des millions d'échantillons doivent être utilisés lors de phase d'apprentissage. Par exemple, il faut un entraînement intensif pour qu'un modèle puisse distinguer les images comprenant des chiens et des chats. Il faut donc collecter des milliers de photos de chats et de chiens de différentes races, dans des positions et des endroits divers. Un label ("chien" ou "chat" dans notre exemple) doit être associé à chaque image présentée au modèle afin d'orienter le processus d'apprentissage. Si le système ne prédit pas la bonne réponse, une correction est appliquée aux paramètres du réseau en fonction de l'erreur commise. Cette opération doit être répétée un grand nombre de fois.

1.2 Motivation

La majorité des méthodes d'apprentissage utilisées se base sur la descente de gradient (Rumelhart, G. E. Hinton et Williams 1986) , qui calcule l'erreur globale du système afin d'ajuster les paramètres du modèle. Deux facteurs majeurs ont permis le développement de ces modèles au début du XXI° siècle : la quantité massive de données disponibles aujourd'hui et la puissance de calcul qui n'a pas cessé d'augmenter grâce au progrès des composants matériels. Ces éléments ont permis de créer des réseaux composés de milliards de paramètres (GPT-3 (Brown et al. 2020) est un modèle qui possède 175 milliards de paramètres) pouvant résoudre des problèmes de plus en plus complexes. Cependant, un certain scepticisme concernant les tailles gigantesques des réseaux et leurs méthodes d'entraînement, commence à envahir la communauté, notamment sur l'orientation des recherches actuelles en matière d'apprentissage profond. Cette opinion est exprimée par des personnes comme Yann Lecun (LeCun 2022), récompensé par le prix Turing 2018 pour ses travaux sur ce sujet (Tappert 2019) avec une phrase qui résume cette pensée : "*Les*

adeptes de ce type de modèle (GPT-3) croient qu'en pré-traitant les données d'entrée (supervision, tokénisation, augmentation, etc.) puis en entraînant des modèles gigantesques pour faire des prédictions discrètes, l'IA émergera de tout cela d'une manière ou d'une autre. Cela peut être un composant d'un futur système intelligent, mais je pense qu'il manque des pièces essentielles". En effet, ces modèles souffrent d'un certain nombre de limites, surtout si on les compare au cerveau humain. La première limite identifiée est que l'entraînement de ces modèles nécessite d'avoir de grands jeux de données annotées manuellement pour la plupart. Des efforts considérables sont ainsi déployés pour créer et annoter ces jeux de données contenant parfois des dizaines de millions d'éléments possédant pour chacun des centaines de labels (Roh, Heo et Whang 2019). Certaines tâches ne possèdent pas de jeux de données répondant aux exigences de l'apprentissage profond, par manque de données ou en raison de leur coût (temps ou argent), la supervision pouvant demander de faire appel à des expertises rares. Des progrès dans la conception des RNAs ont permis de réduire la dépendance à cette supervision via des architectures comme les auto-encodeurs (Rumelhart, G. E. Hinton et Williams 1985) ou des méthodes d'entraînement adverse (Wang, Q. She et Ward 2021). Il en résulte des modèles dont l'apprentissage prend une forme intermédiaire entre un apprentissage supervisé et non supervisé, on parle d'apprentissage auto-supervisé (Jing et Tian 2020). Une autre limite est que ces modèles sont très peu évolutifs, ils résolvent des tâches spécifiques sur des données appartenant aux classes spécifiées lors de l'entraînement. L'ajout éventuel de nouvelles classes d'objets demande de refaire l'apprentissage sur l'ensemble des données, comprenant les anciennes classes et les nouvelles classes. Sans ce processus, si on soumet le modèle à une phase d'apprentissage ne contenant que les classes inédites, les capacités acquises lors du précédent apprentissage vont se détériorer. Ce problème porte le nom de *l'oubli catastrophique* (French 1999). Ceci est une conséquence de la méthode d'apprentissage qui a besoin de connaître la globalité du problème afin d'optimiser correctement l'ensemble des paramètres du réseau. Encore une fois, si on fait un parallèle avec le cerveau humain, celui-ci peut apprendre de nouveaux concepts avec très peu d'exemples, voire un échantillon unique (Yger, Stimberg et Brette 2015). Ce type d'apprentissage, qu'on nomme *fast learning* ou *one-shot learning* selon le nombre d'exemples présentés, est une habilité caractéristique du cerveau biologique. Nous n'avons pas besoin de voir des milliers de fois un chat pour pouvoir le reconnaître. Nous capitalisons sur notre compréhension du monde, avec un modèle mental acquis en apprenant les relations de corrélations et causalités entre les différents objets qui nous entourent. L'apprentissage chez l'humain se fait donc majoritairement de manière non supervisée (Storrs, Anderson et Fleming 2021), en renforçant les connaissances via des expériences positives ou négatives (apprentissage par renforcement) et en affinant des concepts via des comparaisons entre les prédictions et la réalité, avec une correction en cas d'erreur (apprentissage supervisé). Ce processus nous permet de développer la notion de *bon sens*. Cette notion est centrale dans notre capacité à contextualiser et à généraliser les informations perçues. Les modèles profonds peuvent être très puissants pour réaliser des tâches spécifiques, mais ne sont pas doués de bon sens (LeCun 2022). Pour compenser ce manque de bon sens, les modèles profonds utilisés pour réaliser des tâches complexes sont construits avec un nombre de couches gigantesque et sont entraînés avec de très grandes quantités de données. Reprenons l'exemple du modèle GPT-3 avec ses 175 milliards de paramètres et dont ses concepteurs ont pour objectif de développer un modèle capable de comprendre le langage humain. Lors de son entraîne-

ment, il a fallu déployer 1024 GPUs (140 teraFLOP/s par GPU en terme de puissance de calcul) pendant plus de 34 jours. Ce dispositif a engendré un coût énergétique estimé à 936 MWh, équivalent à la consommation moyenne de 188 foyers français pendant une année (Patterson et al. 2021). Ce type de solution n'est pas vraiment compatible avec le contexte actuel de réchauffement climatique et de coût de l'énergie. Au contraire, si on se réfère au cerveau humain, celui-ci ne consomme pas plus de 30 watts (Maass 2015). Au delà des considérations énergétiques, cette surenchère concernant la puissance de calcul montre une dépendance forte entre les performances des réseaux profonds et celles des architectures matérielles. Techniquement, les réseaux profonds utilisent massivement des calculs matriciels qui sont des opérations synchrones sur des structures denses. Les GPUs et CPUs sont des architectures matérielles de Von Neumann, parfaitement adaptées pour ce type calcul. En principe, pour ce type d'architecture matérielle, la puissance de calcul peut être améliorée en augmentant le nombre de composants et via des solutions permettant de les miniaturiser. Le problème est que nous sommes arrivés à des tailles de composants qui se rapprochent des limites physiques possibles (quelques atomes d'épaisseur), marquant la fin de la loi de Moore pour ces architectures matérielles (Waldrop 2016).

Finalement, malgré les performances spectaculaires des réseaux profonds pour différentes tâches, ces modèles souffrent de problématiques inhérentes à la méthode d'apprentissage employée. Cette méthode est coûteuse en données, en annotations, en calculs et en énergie. Les principes énoncées par Turing, sur une machine capable d'imiter les capacités cognitives d'un humain en élaborant un modèle qui reproduit l'intelligence d'un bébé et qui évolue au travers d'une éducation, ont été cependant respectés, en mettant certes un fort accent sur l'apprentissage supervisé.

1.3 Objectifs

En observant le fonctionnement du cerveau humain, une alternative dans la modélisation des réseaux a été proposée, considérée comme la troisième génération de RNAs (Maass 1997). Ce sont les réseaux de neurones impulsionnels (*Spiking Neural Network*, SNN) qui sont des modèles avec une inspiration biologique plus marquée, notamment dans la dynamique intrinsèque des neurones et dans la transmission d'informations entre neurones, permettant d'apporter de nouvelles pistes dans la résolution des problématiques liées à la consommation énergétique et à l'apprentissage non supervisé. La principale différence entre les RNAs classiques et les SNNs réside dans la modélisation du neurone. Les neurones impulsionnels ne sont pas modélisés comme de simples fonctions d'activation, mais comme un ensemble d'équations différentielles. Elles décrivent des évolutions, au cours du temps, de propriétés intrinsèques (comme le potentiel de membrane du neurone), afin de déclencher un potentiel d'action (*spike* sortant) selon la situation. De ce fait, l'information est encodée par une séquence d'impulsions qui se propagent dans le réseau. Cette façon de communiquer permet aux neurones de réagir dynamiquement de manière asynchrone et non plus en fonction d'une cadence définie pour tout le réseau. Le modèle permet l'exécution de processus d'apprentissage bio-plausibles, comme la règle "Spike-Timing Dependent Plasticity" (STDP), qui met à jour les poids synaptiques au cours d'une simulation en fonction des relations de cause à effet constatées entre les impulsions entrantes et sortantes. Le but de cette règle est le renforcement des connexions entrantes

qui sont la cause des impulsions sortantes. Cette méthode d'apprentissage est non supervisée et s'exécute localement sur chaque synapse. Les propriétés des SNNs permettent de concevoir des architectures matérielles neuromorphiques qui divisent le coût énergétique de leur exécution de plusieurs ordres de grandeur par rapport à un équivalent classique en proposant des composants représentant des neurones artificiels très basse consommation (11 à 30 pW par neurone et 1 fJ par spike, Danneville et al. 2019) et/ou des circuits massivement parallèles (Akopyan et al. 2015 ; Davies et al. 2018 ; S. B. Furber et al. 2012). Les principales raisons sont que les composants sont **asynchrones** avec une exécution **locale des opérations** contrairement à la descente de gradient et une communication effectués via des **impulsions ponctuelles**. La STDP offre des perspectives intéressantes sur le développement des différents types d'apprentissages suivants : non supervisé, par renforcement, rapide, supervisé et incrémental (Vigneron et Martinet 2020), ces aspects étant nécessaires dans l'élaboration d'un modèle possédant cette notion de bon sens. En vision par ordinateur, l'utilisation des RNAs est devenue la norme au détriment des descripteurs manuels couplés à des algorithmes de classification (SVM, arbre de décision, etc.), cette tendance ayant commencé après le challenge ImageNet de 2012 (Krizhevsky, Sutskever et G. E. Hinton 2012). Cependant, aucun système de vision par ordinateur n'a encore résolu une tâche en proposant une solution qui valide les exigences suivantes : peu énergivore, fonctionnant en temps réel, avec des capacités d'évolutions et possédant différents types d'invariances qui sont nécessaires pour que le système évolue dans un environnement non contrôlé (géométrie, bruit du capteur, changement de lumière, occultation, pose du capteur). Le seul système de vision connu répondant à tous ses critères est le cortex visuel. Concevoir un système se rapprochant du fonctionnement biologique pourrait permettre de se rapprocher des performances du cortex visuel. Il y a donc un intérêt croissant à appliquer les SNNs à la vision (Nunes et al. 2022). De nombreuses études montrent les capacités des SNNs à résoudre diverses tâches de vision, que ce soit dans la reconnaissance de forme (Falez et al. 2019b ; Masquelier et Thorpe 2007), de geste (Amir et al. 2017 ; Machado et al. 2018 ; Zhao et al. 2015), l'extraction du flux optique (Orchard et Etienne-Cummings 2014 ; Paredes-Vallés, Scheper et De Croon 2019), etc. Cependant, on constate que la majorité des études se concentrent sur la reconnaissance de chiffres manuscrits (MNIST) (Tavanaei, Ghodrati et al. 2019), qui est une tâche élémentaire en vision. Ces études permettent de comparer les SNNs et les RNAs classiques afin d'identifier les forces et les faiblesses de chacun des modèles. Cela permet d'améliorer les modèles SNNs, en capitalisant sur les progrès faits avec les RNAs classiques au niveau des architectures, des méthodes d'apprentissages, etc. Malgré la simplicité des données MNIST, cela permet d'analyser une capacité fondamentale qu'un SNN doit avoir, à savoir la détection et l'extraction de motifs, afin d'adresser d'autres problème en vision. Enfin, travailler sur MNIST ne nécessite pas de ressources en calcul importantes (Shahsavari et Boulet 2017), ce qui permet d'explorer en profondeur diverses problématiques, comme l'encodage des données, la mise en place de topologies profondes, etc. Enfin, les SNNs peuvent se démarquer des RNAs classiques sur le traitement de vidéos (Deng et al. 2020). La première raison est que ces modèles représentent l'information avec une dimension intrinsèquement temporelle. La deuxième raison est qu'il existe une catégorie de capteurs visuels, dit 'événementiels", qui partagent des propriétés intéressantes avec les SNNs. Ils s'inspirent du fonctionnement des rétines biologiques (Lichtsteiner, Posch et Delbruck 2008). Ces capteurs perçoivent les contrastes temporels d'une scène,

provoqués en très grande majorité par des mouvements. Chaque pixel peut détecter, de manière asynchrone et avec un taux d'échantillonnage très élevé, une variation lumineuse (positive ou négative) et déclencher un évènement. Ces caméras événementielles fournissent des données de manière asynchrone et massivement parallélisable. Associer ces capteurs avec les SNNs permet de créer un système de vision entièrement bio-inspiré de l'acquisition au traitement de données. De plus, des processeurs neuromorphiques sont compatibles avec ce type de capteur, en offrant des circuits permettant de connecter la sortie des caméras événementielles directement sur les différents composants permettant d'exécuter le SNN afin de ne créer aucun goulot d'étranglement dans la communication des données. Ainsi les propriétés d'asynchronie, d'opérations locales et parallèles dans un système basse consommation sont donc conservées et permettent le traitement de données en temps réel avec des méthodes d'apprentissages non supervisées.

1.4 Résumé des contributions

L'objectif de cette thèse est d'exploiter les capacités offertes par les SNNs dans le traitement vidéo. Le modèle doit être exécutable sur des architectures matérielles basse consommation en profitant au maximum des accélérations matérielles possibles. Le but est de développer un modèle capable d'apprendre sans supervision, incrémentalement et avec peu d'exemples, à extraire des caractéristiques spatio-temporelles à partir de données événementielles. Afin d'explorer le potentiel offert par les SNNs, nous nous sommes intéressés à une tâche inhérente à l'analyse du mouvement, à savoir, l'estimation de la direction du mouvement. Cela nous permet d'explorer les capacités de reconnaissance de motifs spatio-temporels après un entraînement non supervisé, l'apprentissage se faisant par le biais de la règle STDP. Afin de pouvoir développer à terme des architectures plus complexes, sensibles aux informations spatio-temporelles, nous avons caractérisé un modèle dont la structure est de taille petite et qui peut être vu comme un détecteur local sensible aux directions cardinales d'un mouvement après un apprentissage non supervisé. Ces détecteurs capitalisent sur les capacités de traitements supplémentaires de l'information temporelles qu'offrent les SNNs en comparaison des réseaux classiques. Ces travaux ont été valorisés par 2 publications dans des conférences nationales et internationales avec comité de relecture (Oudjail et Martinet 2018, 2019). Ces détecteurs ont donc une topologie mono-couche feedforward, composée d'un petit nombre de neurones de sortie. Ces neurones sont connectés à un champs récepteur par des synapses excitatrices dont la plasticité est modulée par une STDP simplifiée. Enfin, ces neurones de sortie sont connectés latéralement entre eux par des synapses d'inhibition afin d'instaurer une compétition neuronale sur les différents type de motifs à détecter. À l'aide de jeux de données synthétiques, nous avons étudié la capacité du modèle à analyser dynamiquement les directions d'un mouvement. La première étude nous a permis de mesurer différentes propriétés comme le nombre de neurones nécessaires pour que le modèle puisse différencier les directions, l'influence des paramètres des neurones sur la performance du modèle pour reconnaître les différentes classes de mouvement et une analyse sur les phases d'apprentissage afin de déterminer la durée optimale d'activation de la STDP ainsi que l'impact de l'ordre de présentation des différentes classes lors de l'entraînement. Une exploration de l'espace de valeurs des paramètres des neurones du réseau nous a permis de faire le lien entre le réglage de ces paramètres et la sélectivité des neurones sur des motifs spatio-

temporels. Les données d'entrée étant variables, un mécanisme de seuil permet d'adapter cette sélectivité pour chaque neurone et donc d'augmenter l'expressivité du réseau dans l'extraction de caractéristiques pertinentes liées au mouvement. Ce travail a fait l'objet d'une publication dans une conférence internationales (Oudjail et Martinet 2020). Les résultats ont montré que quelques millisecondes de stimulation non supervisée suffisent pour atteindre une situation stable, et la poursuite de la stimulation ne dégradent pas l'apprentissage. Ces propriétés nous ont permis d'orienter nos recherches sur les capacités d'apprentissage incrémental du modèle. Nous avons montré, lors d'une deuxième étude, que le modèle peut apprendre de manière incrémentale en lui présentant des classes inédites sans détérioration des performances sur les classes déjà apprises. Nous avons évalué la qualité de différentes stratégies de décodage des caractéristiques extraites par le réseau en s'appuyant sur des classifieur. Cette partie des travaux a été présentée lors du workshop ONSVP en marge de la conférence de robotique ICRA 2021 (Oudjail et Martinet 2021). Les détecteurs présentés permettent d'extraire des motifs spatio-temporels après un court apprentissage non supervisé, local et incrémental. Ces détecteurs capitalisent sur les avantages des SNNs, notamment dans le traitement de l'information temporelle, pour pouvoir reconnaître des caractéristiques liées au mouvement. Avec les RNAs classiques, ce type de détecteur ne pourrait pas exister en utilisant la même topologie. Pourtant, lors de l'apprentissage du modèle, les paramètres qui sont optimisés sont les poids synaptiques. Ces poids sont communs aux deux types de réseaux. Par contre, avec les SNNs, les poids peuvent caractériser des informations temporelles ou spatiales. La principale raison est que les neurones évoluent dans le temps en communiquant avec des impulsions qui représentent une information temporelle (le moment de l'impulsion) ainsi qu'une information spatiale détenue par le neurone émetteur de cette même impulsion. La sélectivité des neurones sur les motifs spatio-temporels est sélectionnée par la STDP dans notre modèle. Cette sélection est limitée par les paramètres des neurones. Une manière d'améliorer cette sélectivité est d'utiliser les délais synaptiques. Notre dernière étude porte donc sur l'utilisation des délais afin d'améliorer la sélectivité du réseau sur différents motifs spatio-temporels. Nous proposons une modélisation théorique sous la forme de plusieurs cas d'études dans lesquels nous déterminons des détecteurs élémentaires sensibles à des motifs temporels, composés d'un neurone, et qui utilisent des délais et des poids synaptiques et la fuite du neurone. Nous faisons le lien entre ces différents paramètres afin de proposer une règle d'apprentissage, basée sur les délais.

Cette thèse s'inscrit dans un projet pluridisciplinaire mené à l'USR IRCICA dont la thématique centrale est "Architectures bio-inspirées" pour le traitement de l'information. Il regroupe trois équipes des laboratoires CRIStAL et IEMN ayant des expertises dans la conception de nouveaux composants matériels, dans l'élaboration de simulations reproduisant les comportements les plus fins des modèles et sur l'analyse d'images et de vidéos. Cette collaboration permet une certaine synergie dans le développement de solutions innovantes.

1.5 Plan du manuscrit

Ce manuscrit rend compte de toutes les réflexions et travaux menés à bien durant cette thèse. Le sujet étant vaste, il se concentre sur les principaux point soulevés lors des études menées. Afin de bien comprendre les concepts manipulés et connaître l'état actuel

des connaissances sur les différents sujets évoqués, le CHAPITRE 2 fournit un résumé des approches classiques de vision pour ordinateur, le fonctionnement des SNNs, les caméras événementielles, les applications existantes des SNNs sur le traitement vidéo, etc. Une conclusion vient clôturer ce chapitre en exposant les avantages fournis par ce type de modèle et soulève les problématiques qu'il reste à résoudre.

Les chapitres suivants se concentrent sur les travaux menés. Le CHAPITRE 3 introduit les méthodes communes mises en place pour les différentes études en décrivant les détails techniques, en expliquant les choix effectués afin de mettre en place les différentes expérimentations.

Les CHAPITRES (4 et 5) décrivent les contributions principales de cette thèse.

Le CHAPITRE 4 présente les premières études menées dont le but est d'élaborer un modèle SNN de taille petite capable de distinguer le mouvement de différents motifs synthétiques avec un apprentissage court et non supervisé. Ces résultats ont été publiés lors de la conférence *VISAPP'2019* (Oudjail et Martinet 2019). Une analyse approfondie de ce modèle nous a permis de mettre en évidence les avantages ainsi que les limites que l'on peut rencontrer. L'une d'entre-elles est le réglage des paramètres. Aux vues de la taille du modèle, nous avons pu mettre en place une exploration des valeurs possibles de différents paramètres afin de comprendre l'impact qu'ils ont sur les performances du modèle. Ces travaux ont été présentés lors de la conférence *VISAPP'2020* (Oudjail et Martinet 2020).

Le CHAPITRE 5 présente une étude sur la capacité d'apprentissage incrémentale non supervisée du modèle. En effet, le réseau est entraîné sans supervision, sur des périodes courtes, en appliquant une règle exécutée localement et dynamiquement. Le fait d'avoir un système qui peut s'adapter à de nouvelles données sans déstabiliser ses connaissances est une propriété importante lorsqu'on traite des informations aussi riches que la vidéo (Oudjail et Martinet 2021).

Le CHAPITRE 6 présente une modélisation théorique sous forme d'une étude de cas qui permet d'identifier des détecteurs élémentaires sensibles au mouvement en utilisant les délais synaptiques. Cette étude permet d'identifier les liens entre les délais, les autres paramètres du réseau et leurs influences sur la sélectivité d'un neurone sur des motifs spatio-temporels. La deuxième partie de la modélisation nous permet de proposer une règle d'apprentissage non supervisé basée sur la modulation des délais synaptiques. Cette règle permet de faire converger le réseau vers un état où les neurones se comportent comme un des détecteurs identifiés dans la première partie de l'étude.

Finalement, le dernier CHAPITRE 7 clôture ce manuscrit, avec une discussion des résultats, une conclusion des travaux ainsi qu'une vision sur les SNNs et les perspectives qui découlent de ce travail de thèse.

Chapitre 2

État de l'art

2.1 Apprentissage automatique

L'apprentissage automatique (ou *machine learning*, en anglais) est un domaine de l'IA dans lequel on conçoit des algorithmes génériques qui peuvent s'adapter à de nombreux problèmes. Ces algorithmes permettent d'extraire de l'information pertinente de données ou de prédire un comportement à partir de l'observation d'un phénomène. Pour ce faire, ils passent par une phase d'apprentissage établie sur une base d'exemples, on peut identifier deux familles d'approches : l'apprentissage supervisé et l'apprentissage non-supervisé.

Concernant l'apprentissage supervisé, le but est de déterminer la sortie Y correspondant à une nouvelle entrée X, connaissant un ensemble d'observations $\{(X_1, Y_1), \ldots, (X_n, Y_n)\}$. Lorsque les Yi prennent des valeurs discrètes, il s'agit d'un problème de classification alors que des Y_i avec des valeurs réelles nous référent à un problème de régression.

La FIGURE 2.1, présente un problème de classification binaire où les données en entrée (X_1, X_2) sont les coordonnées et leur forme (rond pour $Y = 0$ ou étoile pour $Y = 1$) est la classe à laquelle ces données correspondent. La ligne rouge est nommée "*limite de décision*" et est le résultat qui découle de l'algorithme de classification. Le but de ces algorithmes est de trouver la limite de décision optimale.

De manière plus concrète, un exemple de problème de classification peut être la détection d'un chat ou d'un chien à partir d'une image. La base d'exemples est alors un ensemble d'images avec la classe chat ou chien précisée pour chacune d'entre-elles. Pour la régression, on peut imaginer un problème où l'on doit prédire le prix d'une maison selon sa superficie, son emplacement, sa conception, etc...

En apprentissage non-supervisé, en revanche, les données d'apprentissage ne sont pas étiquetées, et il s'agit alors de construire un modèle permettant de représenter au mieux les observations $\{X_1, \ldots, X_n\}$ de manière à la fois précise et compacte.

Les réseaux de neurones artificiels sont une famille d'algorithmes supportant majoritairement l'apprentissage supervisé. On peut, tout de même, trouver des modèles qui sont de nature non-supervisée, c'est le cas des réseaux impulsionnels. De manière simplifié, on peut voir un RNA comme un modèle de type "boite noire", illustré par la FIGURE 2.2. Il réalise une association d'un espace d'entrée X (données pouvant représenter des images, du texte, dans l'encodage choisi) vers un espace de sortie Y. La "fonction" φ est définie

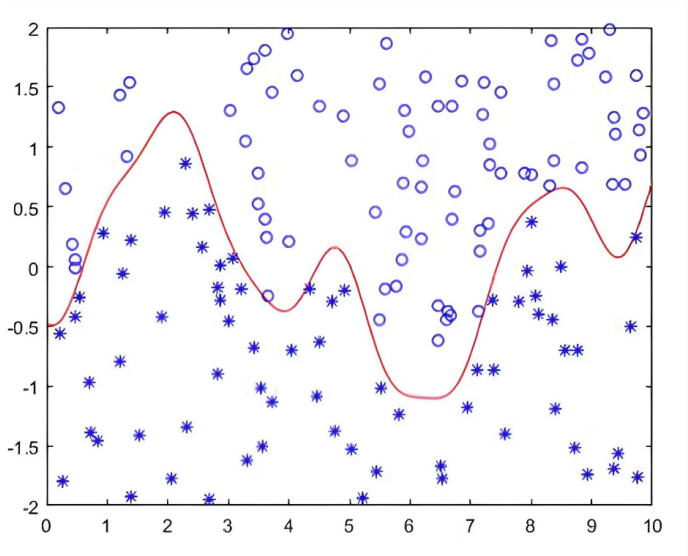


FIGURE 2.1 – Exemple de problème de classification à deux dimensions.

par les poids du réseau. Ceux-ci sont ajustés au problème par une règle d'apprentissage, à partir d'une base de n exemples (X^n, Y^n), Y^n étant la réponse attendue, on parle d'apprentissage supervisé si Y^n est connu. Lorsque le réseau passe d'un état où ses poids ne sont pas adaptés à un état où ils le sont pour la tâche demandée, on dit que le modèle a convergé.

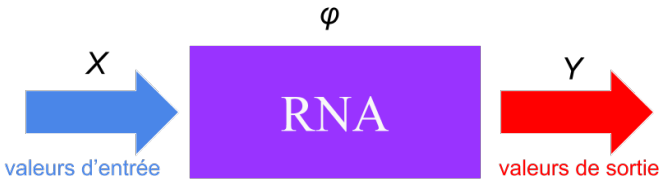


FIGURE 2.2 – Algorithme des réseaux de neurones, prennent un ensemble de valeurs X renvoie un ensemble de valeurs Y, les valeurs étant encodées d'une certaine manière. La " fonction " φ définit la manière dont les poids sont modifiés lors de la phase d'apprentissage.

2.2 Réseaux de neurones impulsionnels

2.2.1 Inspiration biologique

Le concept de réseaux de neurones vient de l'étude du cerveau humain et plus généralement de celui des animaux. Cette discipline porte le nom de neurosciences. En conséquence, introduire le modèle biologique permet de mieux comprendre la terminologie et les origines de certains mécanismes utilisés dans les réseaux de neurones artificiels. Si on schématise de manière globale, le cerveau est un organe qui reçoit des stimulations sensorielles et qui, via des mécanismes internes, produit différentes réactions plus ou moins complexes comme des raisonnements logiques, des émotions, etc. Dans le cerveau humain,

on compte plus de 10^{11} neurones et 10^4 synapses pour chacun d'entre eux en moyenne. Le cerveau humain est composé de cellules qui sont les unités élémentaires de calculs qui traitent l'information : les neurones. Les neurones sont connectés par des synapses qui permettent la circulation de l'information.

2.2.1.1 Neurones

Même s'il existe différents types de neurones avec des propriétés internes qui leur sont propres, ils partagent néanmoins une structure de base. On peut distinguer trois parties dans le neurone, qui sont fonctionnellement distinctes :

— *Les dendrites* peuvent être apparentées à un dispositif d'entrée car leur rôle est de recevoir les signaux en provenance d'autres neurones.

— *Le soma* (ou corps cellulaire) effectue ensuite un traitement non-linéaire des signaux d'entrée reçus.

— *L'axone* se charge d'émettre le signal de sortie vers d'autres neurones via des connexions synaptiques. Ceci est illustré par la FIGURE 2.3.

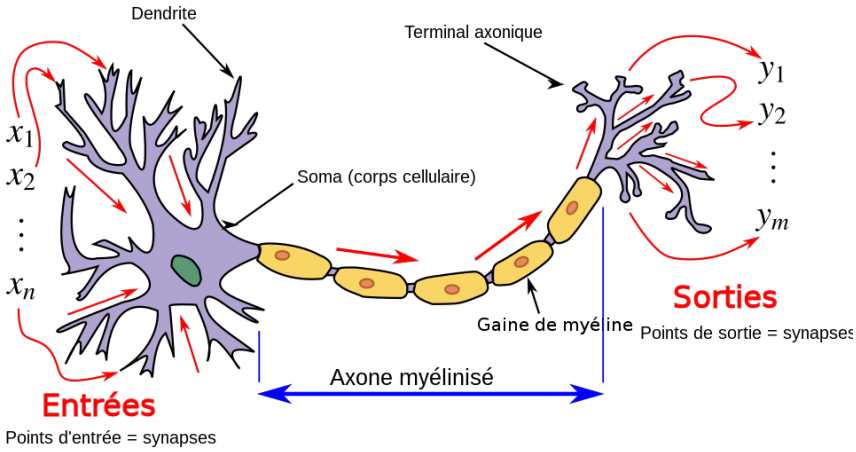


FIGURE 2.3 – Structure d'un neurone se trouvant dans le cerveau humain. Schéma simplifié d'un neurone, illustre les composants principaux. Image tirée de (D CAT LAZ WIKI s. d.)

2.2.1.2 Messages électriques dans les neurones

La différence de potentiel électrique entre l'intérieur de la cellule et son environnement externe s'appelle le potentiel membranaire. Il faut distinguer deux types de potentiels :

— Le *potentiel de repos* est la polarisation de la membrane du neurone à une valeur de -70 mV environ lorsqu'il n'y a pas de stimulation.

— Le *potentiel d'action* (PA) est une inversion brusque et temporaire de la polarisation de la membrane du neurone, du fait du dépassement du seuil.

Si le neurone est fréquemment stimulé sur une courte période, alors le potentiel de la membrane augmente jusqu'à éventuellement atteindre une valeur critique connue sous le nom de *seuil* d'activation. Le neurone émet alors un potentiel d'action qui peut être découpé en plusieurs phases, comme illustré à la FIGURE 2.4 :

— dépolarisation de la membrane,

— repolarisation,

— hyperpolarisation,

— retour au potentiel de repos.

Pendant 1-2 ms environ, le potentiel atteint environ 35 mV (soit une amplitude de 100 mV) comme cela est illustré dans la FIGURE 2.4.

Après un potentiel d'action (ou impulsion, ou spike), on observe une phase d'hyperpolarisation pendant laquelle la tension chute en dessous du potentiel de repos. C'est ce qu'on appelle la période de *réfraction* du neurone durant laquelle il ne peut pas émettre de spike. Cette période marque le pas temporel minimal entre deux spikes.

Après une stimulation qui n'a pas permis d'atteindre le seuil d'allumage, le potentiel du neurone diminue jusqu'à atteindre le potentiel de repos si celui-ci n'est de nouveau excité. Ce phénomène est appelé fuite du neurone.

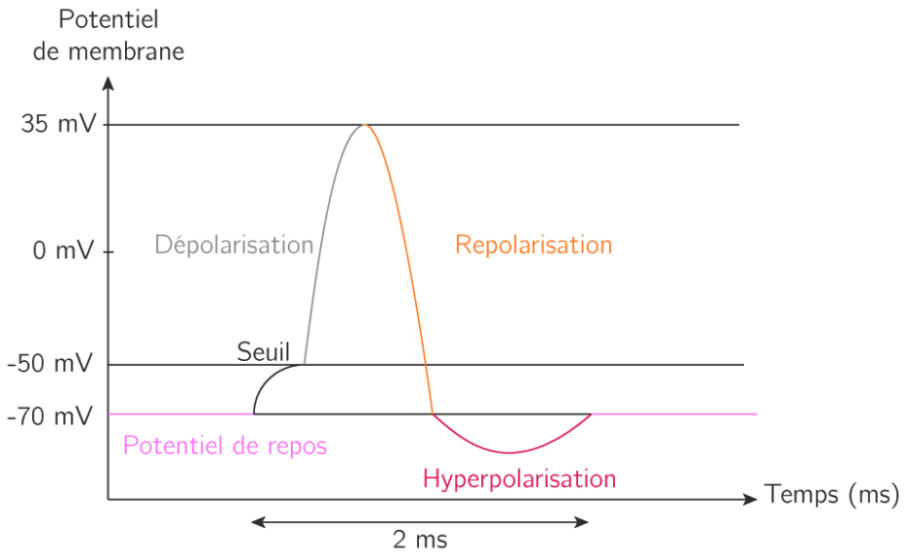


FIGURE 2.4 – Illustration du potentiel d'action. Image tirée de (KARTABLE s. d.).

2.2.1.3 Synapses et plasticité

La plupart des connexions synaptiques sont de nature chimique. En effet, lors du processus d'émission d'un signal électrique par un neurone présynaptique, la synapse reçoit cette impulsion et la convertit en signal chimique en entraînant la libération de molécules spécifiques appelées neurotransmetteurs. À leurs tours, celles-ci sont captées par des récepteurs de l'autre coté de la synapse. Ainsi, ils peuvent provoquer un afflux d'ions qui modifie le potentiel électrique de la membrane du neurone postsynaptique (voir FIGURE 2.5). Il faut noter qu'il existe d'autres synapses qui sont, quant à elles, de nature électrique, dans lesquelles des protéines membranaires spécialisées établissent une connexion électrique directe entre les deux neurones.

Enfin, les synapses peuvent être aussi divisées en deux groupes selon l'effet qu'elles produisent sur le neurone postsynaptique. Les synapses *excitatrices* ont pour effet d'augmenter le potentiel membranaire du neurone via un potentiel postsynaptique excitateur. Les synapses *inhibitrices* ont pour effet de diminuer le potentiel membranaire du neurone via le potentiel via un potentiel postsynaptique inhibiteur.

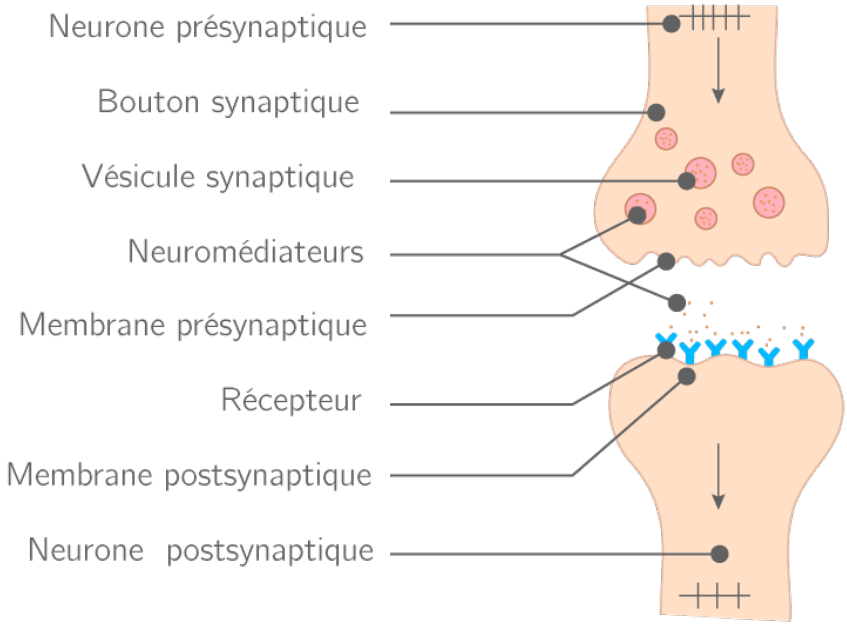


FIGURE 2.5 – Illustration d'une jonction synaptique entre le neurone présynaptique et le neurone postsynaptique. Image tirée de (KARTABLE s. d.).

Il est intéressant de noter que lorsque des impulsions sont émises vers plusieurs neurones cibles avec des tensions similaires, les potentiels de membrane des neurones postsynaptique n'évoluent pas de la même manière. Cette différence est causée par les synapses. En effet, elles transmettent le signal avec une efficacité variable, c'est une propriété qui leur est propre que l'on appelle plasticité synaptique.

Ce mécanisme est à la base du phénomène d'apprentissage, selon Donald Hebb : « *lorsque deux neurones ou systèmes sont fréquemment activés de manière simultanée, le poids de leurs connexions synaptiques serait renforcé de sorte que l'activité d'un de ces neurones ou systèmes suscite automatiquement l'activation de l'autre* » (D. HEBB 1949).

La règle d'apprentissage *Spike-timing-dependent plasticity* (STDP) est souvent considérée comme la mise en œuvre de la théorie de la plasticité synaptique de Hebb. Elle a été élaborée d'après des relevés d'étude électrophysiologique (Bi et Poo 2001). Son principe de base est de renforcer ou déprimer les synapses selon deux situations spécifiques (voir FIGURE 2.6) :

— Potentialisation à long terme (Long-term Potentiation, LTP) qui augmente les connexions des neurones présynaptiques ayant émis un spike peu de temps avant une décharge du neurone postsynaptique.

— Dépression à long terme (Long-term Depression, LTD) qui diminue les connexions des neurones
présynaptiques ayant émis un spike peu de temps après une décharge du neurone postsynaptique.

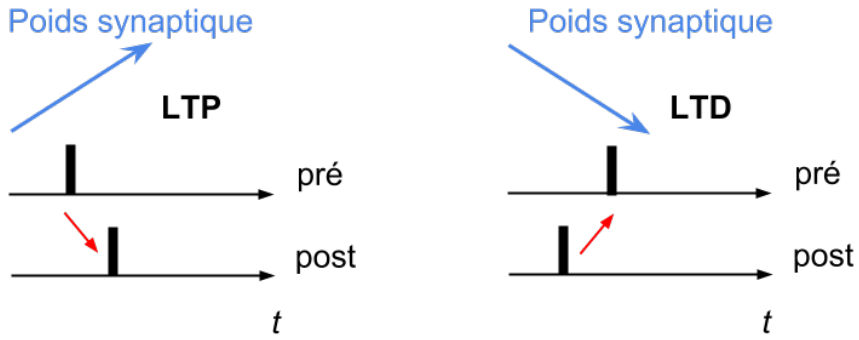


Figure 2.6 – À gauche : Potentialisation à long terme (Long-term Potentiation, LTP). À droite : Dépression à long terme (Long-term Depression, LTD). Pour les deux cas, on a l'évolution temporelle des impulsions (marqueur noir) du neurone pré- et post-synaptique.

Le moment où les impulsions sont produites va affecter la modulation des poids synaptiques. Plus le temps entre l'impulsion du neurone présynaptique et du neurone postsynaptique est court, plus l'impact de la LTD/LTP est fort. C'est ce qu'illustre la FIGURE 2.7.

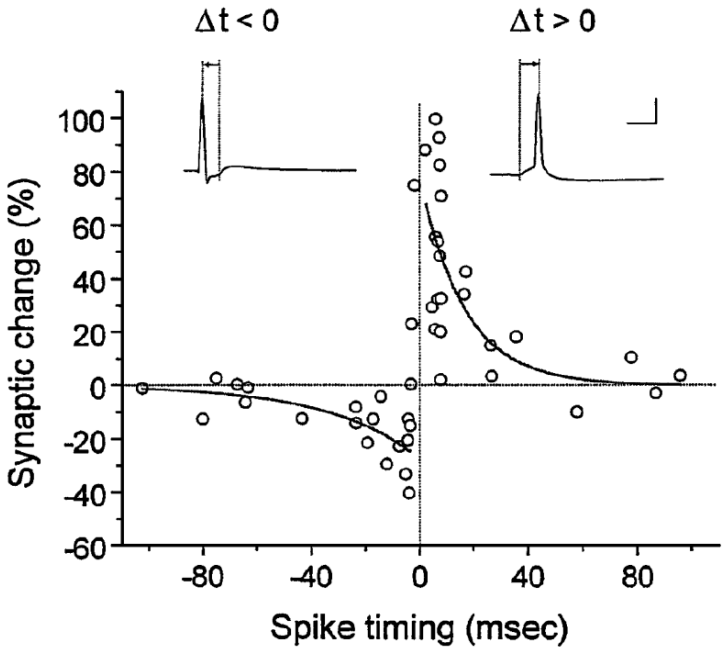


FIGURE 2.7 – Observation du comportement d'une synapse et de sa plasticité en fonction des impulsions émises par le neurone pré- et postsynaptique. Les relevés électrophysiologiques faits par (BI et POO 2001) montrent que la plasticité d'une synapse peut être décrite par une loi. La LTP se déclenche lorsqu'il y a une séquence $(Spike_{pre}, Spike_{post})$. Alors que la LTD se déclenche lorsqu'il y a une séquence $(Spike_{post}, Spike_{pre})$. L'impact de la LTD/LTP est régi par la distance temporelle qu'il y a entre l'impulsion présynaptique et l'impulsion postsynaptique.

2.2.2 Réseaux de neurones artificiels

Le modèle des *réseaux de neurones artificiels* (RNA) a été développé en observant le comportement du cerveau. On est dans une approche bio-inspirée. Les premiers neurones formels ont été proposés par les neurologues Warren McCulloch et Walter Pitts en 1943 (MCCULLOCH et PITTS 1943). Le modèle est composé de deux types d'éléments, les neurones et les synapses.

Le neurone formel peut être vu comme un automate doté d'une fonction de transfert régie par des règles lui permettant de renvoyer une valeur scalaire. Dans sa version la plus simple, un neurone formel calcule la somme pondérée des entrées reçues, puis applique à cette valeur une transformation appelée fonction d'activation, généralement non-linéaire. La valeur finale obtenue est la sortie du neurone. Une fonction d'activation type peut être par exemple une fonction dont la valeur de l'entrée est renvoyée si elle dépasse un seuil. La FIGURE 2.8 illustre un neurone formel.

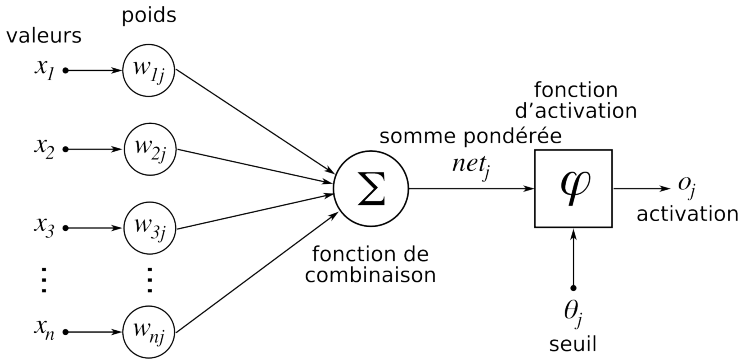


FIGURE 2.8 – Illustration d'un neurone formel. En se référant au neurone biologique de la FIGURE 2.3, on peut faire le parallèle entre les entrées $(x_1, \ldots, x_n)$ et les dendrites, la fonction de transfert et le soma ainsi que la sortie et l'axone. Les entrées sont pondérées par les poids synaptiques $(w_{1j}, \ldots, w_{nj})$. Image tirée de (CHRISLB s. d.).

Les poids synaptiques sont des valeurs qui pondèrent les synapses. Comme les connexions synaptiques sont orientées, on établit une distinction entre le neurone *présynaptique* et le neurone *postsynaptique*. Le poids est modélisé par un scalaire, souvent noté w.

2.2.3 Évolution historique des réseaux de neurones

2.2.3.1 Modèles des neurones artificiels

L'idée de s'inspirer des processus biologiques du cerveau pour créer un modèle informatique est apparue assez tôt dans l'histoire de l'informatique. D'après Wolfgang Maass (MAASS 1997), on peut identifier 3 générations de réseau de neurones. Cette classification se base en grande partie sur le type de modèle de neurones.

2.2.3.2 Première génération

L'un des premiers modèles de réseaux de neurones est le *perceptron* proposé par Frank Rosenblatt en 1958 (ROSENBLATT 1958). Dans sa version la plus simplifiée, le perceptron est mono-couche et n'a qu'une seule sortie à laquelle toutes les entrées sont connectées. Il s'agit d'un classifieur binaire, c'est-à-dire que les entrées ainsi que la sortie sont de type booléen.

Un perceptron à n entrées $(x_1, \ldots, x_n)$ et à une seule sortie o est défini par la donnée de n valeurs de poids (ou coefficients synaptiques) $(w_1, \ldots, w_n)$ et un biais (ou seuil) θ réel par :

$$o = f_\theta(\sum_{i=1}^{n} w_i x_i). \tag{2.1}$$

La sortie o résulte alors de l'application de la fonction suivante nommée Heaviside au potentiel postsynaptique
avec $z = \sum_{i=1}^{n} w_i x_i$:

$$f_\theta(z) = \begin{cases} 0, & \text{si } z \leq \theta \\ 1, & \text{sinon.} \end{cases} \tag{2.2}$$

Des architectures plus complexes peuvent utiliser ce type de neurone comme la version multi-couches du perceptron (*Multiple Layer Perceptron* en anglais, MLP).

Dans cette même période, les mathématiciens, appartenant au courant de pensée connexionniste, tentaient de démontrer que tout raisonnement pouvait se formaliser par des combinaisons complexes de règles logiques. Les neurones étant des unités élémentaires, les assembler dans un réseau contenant un grand nombre d'entre eux, provoquerait l'émergence de raisonnement complexe intelligent. Cette pensée se résume par : neurones $\longrightarrow$ logique $\longrightarrow$ raisonnement $\longrightarrow$ intelligence.

Cependant, Minsky et Papert (Minsky et Papert 1969) montrèrent des limitations théoriques du perceptron, notamment sur l'impossibilité de traiter des problèmes non-linéaires (par exemple, le cas de la fonction XOR). Ils généralisèrent implicitement ce constat à tous les modèles de RNAs. De ce fait, une partie de la communauté IA les délaissa au profit d'autres techniques.

2.2.3.3 Deuxième génération

La deuxième génération de RNAs est basée sur des unités neuronales qui utilisent une fonction d'activation réelle. Elle prend en paramètre une somme pondérée des entrées et génère un ensemble continu de valeurs de sortie possibles. Cette fonction doit avoir certaines propriétés, l'une d'entre elles est la dérivabilité. C'est une condition nécessaire pour pouvoir optimiser les poids en rétropropagant le gradient. L'objectif est de minimiser l'erreur entre la prédiction déterminée par le réseau et la valeur réellement attendue. Pour cela, la méthode détermine la correction en se basant sur la sortie et la dérivée de la fonction d'erreur. Cette correction est alors appliquée sur les poids synaptiques, ceux qui contribuent à engendrer une grande erreur se verront modifiés de manière plus importante que les poids qui ont engendré une petite erreur.

Les fonctions d'activation les plus communes sont la fonction ReLU (la plus populaire à l'heure actuelle), la fonction sigmoïde, ainsi que la fonction identité. La ReLU et la sigmoïde sont illustrées respectivement en FIGURE 2.9 et FIGURE 2.10.

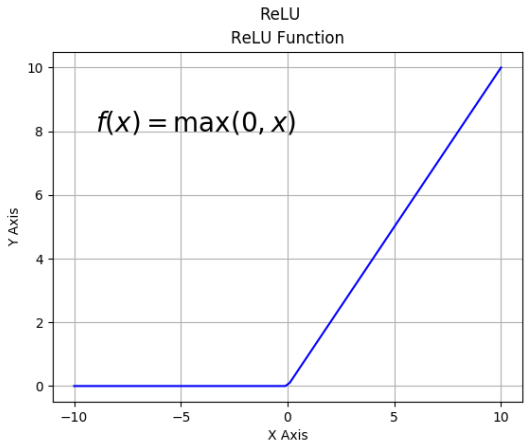


Figure 2.9 – Fonction ReLU.

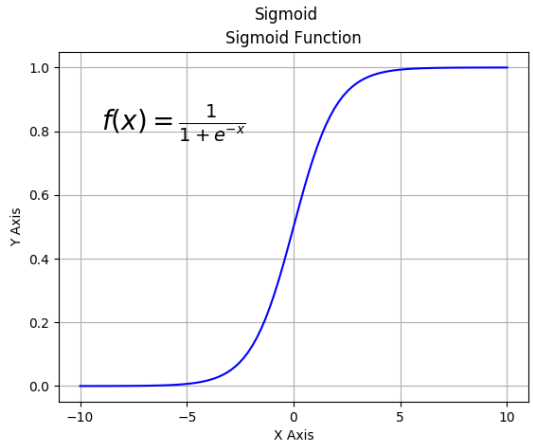


Figure 2.10 – Fonction sigmoïde.

Si nous nous intéressons à la fonction sigmoïde en particulier, comme dit précédemment, elle s'avère appropriée mathématiquement puisqu'elle est différentiable et sa dérivée

peut être facilement exprimée avec les termes de la fonction originale comme le montre l'ÉQUATION (2.3) suivante :

$$\frac{d}{dx}f(x) = \frac{e^{-x}}{(1+e^{-x})^2} = f(x)(1-f(x)). \tag{2.3}$$

Ces réseaux de deuxième génération sont également plus réalistes d'un point de vue biologique que la première génération, en ce sens que la sortie d'une unité peut être interprétée comme une représentation de la fréquence ou du taux d'émission d'impulsions à un instant t d'un neurone biologique (*rate coding*).

Les perceptrons et plus généralement les architectures *profondes* maintenant classiques et contribuant au succès du machine learning utilisent ce type de neurones. Ces approches ont notamment permis des progrès importants et rapides dans les domaines de l'analyse du signal sonore ou visuel et particulièrement sur la reconnaissance visuelle. On peut citer par exemple le challenge *ImageNet* de 2012 qui a propulsé ces solutions en mettant en avant des résultats exceptionnels (de 25% pour les meilleures solutions en 2011 à 16% pour les premières approches RNAs de 2012) (Krizhevsky, Sutskever et G. E. Hinton 2017). De plus, il a été montré, que les limitations théoriques sur les problèmes non-linéaires, pouvaient être palliées avec des architectures profondes utilisant la rétro-propagation du gradient comme règle d'apprentissage (Rumelhart, G. E. Hinton et Williams 1986). En effet, les couches cachées forment une représentation intermédiaire des données, celle-ci supprimant cette non-linéarité, lorsque le réseau a fini son apprentissage (voir FIGURE 2.11).

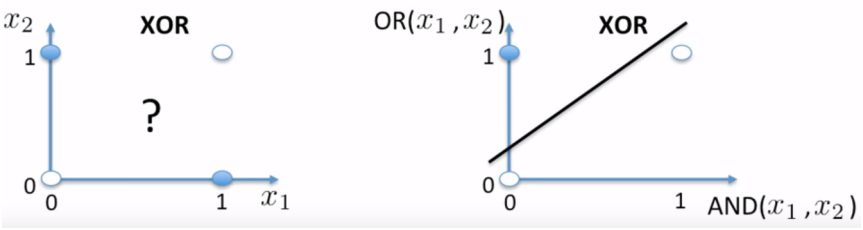


Figure 2.11 – À gauche, une première représentation du problème de classification de la fonction XOR, les points bleus symbolisant le résultat $Vrai$ et les points blancs $Faux$. On peut voir qu'aucune séparation linéaire n'est possible. À droite, le même problème présenté sous une forme différente qui permet d'avoir une solution. Un réseau *profond* est capable, lors du processus d'apprentissage, de générer des représentations intermédiaires transformant la nature du problème. Image tirée du cours de (Larochelle s. d.).

Pour autant, des limites dans ce modèle existent, la principale d'entre elles est le nombre d'exemples nécessaires pour l'apprentissage. Plus le problème est complexe et plus la quantité de données nécessaires est importante, si bien qu'il existe des datasets comme AffectNet contenant plus d'un million d'images dont plus de la moitié est annotée manuellement (Mollahosseini, Hasani et Mahoor 2017). Une autre limite est la ressource computationnelle. En se référant à des architectures profondes classiques comme *AlexNet*, composé de 62 378 344 de paramètres à optimiser lors d'un apprentissage. Il faut en moyenne deux à trois semaines avec 8 cartes graphiques (GPU) pour faire converger le réseau (Krizhevsky, Sutskever et G. E. Hinton 2012).

2.2.3.4 Troisième génération

D'un point de vue conceptuel, la troisième génération est la plus proche du modèle biologique, il s'agit des réseaux impulsionnels (*Spiking Neural Network*, abrégé *SNN*). Les neurones communiquent par des séquence d'impulsions (Ponulak et Kasinski 2011), ce qui induit une notion temporelle centrale dans ce modèle. L'état d'un neurone est décrit par son potentiel, qui est modélisé par une variable dynamique, et le neurone se comporte comme un composant qui augmente son potentiel lorsqu'il reçoit des impulsions en les accumulant au cours du temps et qui diminue si celui-ci n'est pas stimulé. De plus, lorsqu'une impulsions est émise, on prend en compte le concept de période réfractaire qui permet un réglage sur le pas de temps minimal entre deux spikes.

L'une des premières motivations ayant poussé les chercheurs à développer ce modèle est que l'on a observé que l'information pertinente résidait dans l'instant d'émission des spikes plutôt que dans la fréquence. C'est pourquoi, contrairement aux réseaux de neurones de seconde génération qui sont largement utilisés, les neurones à spike ne se déclenchent pas à chaque cycle de propagation, mais seulement lorsque leur niveau d'activation atteint une valeur seuil spécifique (Ponulak et Kasinski 2011). Par conséquent, le réseau est asynchrone et susceptible de bien gérer les données temporelles telles que la vidéo.

Ce modèle est donc plus général car il peut prendre en compte les deux encodages : fréquentiel et temporel. Le codage fréquentiel est déterminé par le nombre d'impulsions dans une période donnée d'un neurone. D'ailleurs, les valeurs numériques manipulées par les fonctions d'activations des neurones de seconde génération représentent l'information fréquentiel d'un neurone. Par contre, le codage temporel est déterminé par moment d'émission d'une impulsion. Ce type d'information ne peut pas être représenté par des neurones de seconde génération. Par conséquent, les SNNs sont des modèles plus généraux. Il est possible d'émuler les principaux modèles classiques (MLP, ...) avec des SNNs (Maass 1997 ; Tavanaei, Ghodrati et al. 2019).

Les SNNs peuvent être entraînés par différentes méthodes d'apprentissages. Il est possible de modéliser des synapses, qui mettent en œuvre un mécanisme d'apprentissage inspiré de la biologie : il repose sur la règle Spike-timing-dependent plasticity (STDP). C'est une règle qui met à jours les poids synaptiques en fonction des temps de spikes, et augmente le poids lorsqu'un neurone présynaptique envoie une impulsion un peu avant le neurone postsynaptique (de l'ordre de quelques millisecondes). Par conséquent, le processus d'apprentissage n'est intrinsèquement pas supervisé, et les SNNs peuvent être utilisés pour apprendre des motifs de manière non supervisée.

Malgré tout, plusieurs travaux essaient de reproduire les mécanismes faisant le succès des réseaux profonds. L'apprentissage supervisé par l'algorithme de rétropropagation (J. H. Lee, Delbruck et Pfeiffer 2016), les architectures profondes de types convolutifs (Cao, Chen et Khosla 2015 ; Tavanaei et Maida 2017) sont autant d'aspects qui suscitent un grand intérêt.

Les SNNs sont utilisés depuis longtemps dans la communauté neuroscientifique comme modèle fiable pour simuler avec précision la biologie et comprendre les mécanismes du cerveau (Paugam-Moisy et Bohte 2012).

En outre, les SNNs sont de plus en plus utilisés dans le traitement des données en raison de leur mise en œuvre sur du matériel à faible consommation d'énergie (circuits neuromorphique) (P. A. Merolla et al. 2014 ; Sourikopoulos et al. 2017).

Les SNNs ont été utilisés dans des tâches de vision, notamment, pour de la recon-

naissance d'objets. Les travaux de (Masquelier et Thorpe 2007) ont montré que la catégorisation d'images pouvait se faire de manière rapide via une solution qui couple un modèle SNN entraîné par une STDP et une architecture inspirée de HMAX, un extracteur de caractéristiques spatiales utilisé pour les images (inspiré du cortex visuel des chimpanzés).

2.2.4 Modèle des neurones impulsionnels

Dans les SNNs, l'état d'un neurone est décrit par son potentiel de membrane (sa tension). Elle est modélisée par une variable dynamique, et le neurone fonctionne comme un intégrateur fuyant, c'est-à-dire un composant qui augmente son potentiel lorsqu'il reçoit des impulsions en les accumulant au cours du temps et qui diminue si celui-ci n'est pas stimulé. De plus, lorsqu'une impulsion est émise, le neurone entre dans une période réfractaire où les stimulis entrants ne sont plus pris en compte. Cela permet de régler l'intervalle temporelle minimale entre deux spikes. Il existe une multitude de modèles reproduisant différents mécanismes avec un réalisme biologique plus ou moins grand. Afin de pouvoir modéliser la dynamique du neurone, les modèles définissent ou se fondent sur des équations différentielles qui ont pour but de décrire le comportement du potentiel membranaire. On peut citer le modèle de ***Hodgkin & Huxley (HH)*** (Hodgkin et Huxley 1952) qui décrit la dynamique temporelle du neurone par les variations de concentrations ioniques (K+, Ca2+) dans différents compartiments et reproduit les principaux "modes" de fonctionnement du neurone biologique. Dans le même style, le modèle ***Izhikevich*** (Izhikevich 2003) réduit la complexité mathématique du modèle de H&H tout en ayant le même niveau de réalisme. Le modèle ***Integrate & Fire (IF)*** (Lapicque 1907) se place à un niveau de détails moins précis que les autres neurones cités. En effet, celui-ci ne fait qu'intégrer les spikes en entrée, augmentant par conséquent son potentiel de membrane jusqu'à atteindre le seuil d'activation, déclenchant un spike. Une variante à ce modèle est le ***Leaky Integrate & Fire (LIF)*** qui inclut un mécanisme supplémentaire permettant de simuler la fuite du potentiel de membrane en faisant décroître la valeur de tension du neurone lorsque celui-ci n'est pas stimulé. Cette décroissance peut amener le neurone à atteindre une valeur de tension appelée le potentiel de repos, valeur à laquelle le neurone est dans un état nul. Ces deux derniers modèles ne nécessitent pas de calculs complexes afin d'être simulés. La FIGURE 2.12 compare les différents modèles suivant la complexité en calcul requis et la plausibilité du comportement par rapport aux neurones biologiques.

Les modélisations des neurones impulsionnels suivent 2 tendances : d'un côté, celles qui cherchent à reproduire au mieux les comportements des neurones biologiques, au prix d'une complexité accrue du modèle, et de l'autre, celles qui font une abstraction des mécaniques biologiques en ne conservant que les principes fondamentaux, sacrifiant au passage le réalisme des comportements simulés. La majorité des architectures SNNs employées en vision sont basées sur des modèles de neurones moins réalistes afin de simplifier leur développement en réduisant le nombre de paramètres à ajuster. Par ailleurs, les solutions proposées requièrent moins de ressources de calcul.

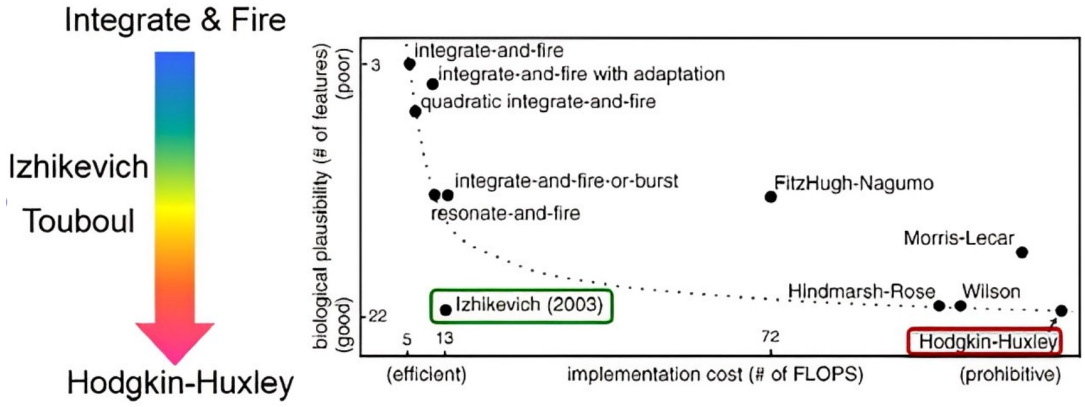


FIGURE 2.12 – Classement des différents modèles suivant le coût en calcul nécessaire afin d'être simulés et leur plausibilité par rapport à la biologie. La figure provient de www.izhikevich.com.

2.2.4.1 Modèle général

Un neurone impulsionnel peut être caractérisé par sa réaction à un courant d'entrée u. Ce courant peut être exprimé à partir des spikes d'entrée, sous une forme simple, comme suit :

$$u(t) = \sum_{s \in S} v_{exc_i} . f_{spike}(t - t_i) \tag{2.4}$$

avec S l'ensemble des spikes entrants, v_{exc_i} la tension du i$^{\text{ème}}$ spike, t_i le moment du i$^{\text{ème}}$ spike et f_{spike} la fonction représentant la forme des spikes (le noyau des spikes). La modélisation la plus simple et la plus courante d'un noyau s'appuie sur une distribution de Dirac, notée δ :

$$f_{spike} = \delta(x) = \begin{cases} 1 & \text{si } x = 0 \\ 0 & \text{sinon} \end{cases} \tag{2.5}$$

2.2.4.2 Modèle Integrate and Fire

C'est le modèle le plus simple présent dans l'état de l'art. Il introduit les concepts essentiels dans la description d'un neurone impulsionnels. Son fonctionnement repose sur l'intégration des impulsions entrantes afin de faire évoluer le potentiel de membrane (v). Si le potentiel v dépasse un certain seuil V_{thres}, le neurone émet un spike de sortie et revient à sa tension de repos, notée V_{rest}. Après avoir émis une impulsion, le neurone entre dans un état réfractaire pendant une certaine période dont la durée est paramétrée par la variable T_{refrac}. Pendant cette période, aucun stimuli entrant n'est intégré, le neurone se met en pause et ne fait plus évoluer sa tension de membrane. L'ÉQUATION (2.6) décrit ces comportements.

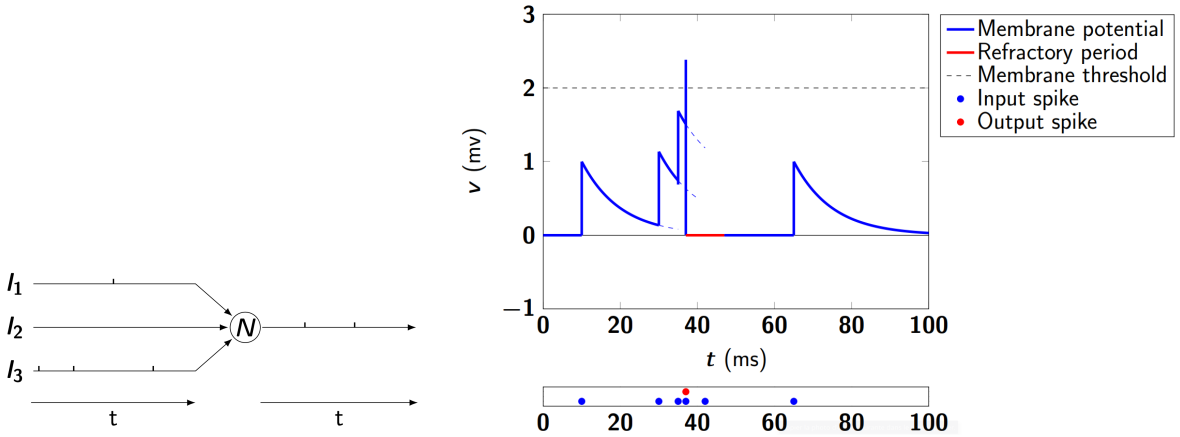
$$\frac{dv}{dt}(t) = u(t), v \leftarrow V_{reset} \text{ quand } v \geq V_{thres} \tag{2.6}$$

2.2.4.3 Modèle Leaky Integrate and Fire

Dans la littérature, les SNNs utilisés en vision sont généralement composés de neurones LIF. La différence entre la modélisation IF et LIF réside dans l'ajout d'un mécanisme de fuite influençant la tension membranaire au cours du temps. L'ÉQUATION (2.7) décrit l'évolution de la tension membranaire notée $v(t)$ au cours du temps $\frac{dv}{dt}(t)$.

$$\frac{dv}{dt}(t) = -\frac{1}{\tau_{leak}}v(t) + u(t), v \leftarrow V_{rest} \text{ quand } v \geq V_{thres}. \tag{2.7}$$

Le paramètre τ_{leak} symbolise le taux de fuite du neurone en ajoutant un coefficient dans les termes permettant une décroissance de la tension au cours du temps. La FIGURE 2.13b montre l'évolution du potentiel (panneau du haut), lorsqu'il y a des stimuli (panneau du bas, point bleu). La tension entrante est intégrée, ce qui à pour effet d'augmenter le potentiel de membrane. Par contre, lorsque le neurone n'est pas stimulé, le potentiel diminue. Quand la tension atteint la valeur du seuil d'activation, un spike est généré et il s'ensuit une période réfractaire dans laquelle le potentiel de membrane reprend une valeur de repos.



(a) Neurone impulsionnel. (b) Neurone LIF

FIGURE 2.13 – L'image (a) illustre le fonctionnement d'un neurone impulsionnel est illustré. L'image (b), présente l'évolution du potentiel de membrane d'un neurone LIF au cours d'une simulation donnée.

Avec ce modèle, les 3 hyper-paramètres permettent de régler le comportement du neurone. Bien les régler est important car ils influencent les performances du SNNs.

Le premier paramètre est le ***seuil d'activation*** d'un neurone V_{thres} qui détermine la tension minimale de déclenchement d'un potentiel d'action. Le seuil a un impact important sur le comportement du réseau, il est essentiel de bien le régler. En effet, s'il est trop bas, on observera une sur-activation des neurones, dans le cas contraire, s'il est trop haut, l'activité dans le réseau sera trop faible, rendant l'apprentissage difficile. Dans les deux cas, le processus d'apprentissage est impacté. Le réglage dépend de plusieurs facteurs : la topologie du réseau et le nombre de spikes généré par les données d'entrées. Ce facteur n'étant pas contrôlé, il peut conduire à une situation de sur-activation ou de sous-activation suivant la distribution des impulsions représentant les données d'entrées. Pour pouvoir limiter l'impact des données d'entrée, un mécanisme ***homéostatique*** appliqué

au seuil peut être ajouté. Cela permet d'adapter le seuil à une valeur correspondant à l'historique d'activité des stimuli d'entrée. En biologie, on retrouve des neurones ayant ce type de seuils adaptatifs (W. Zhang et Linden 2003).

Le deuxième paramètre est la ***fuite*** du neurone τ_{leak} qui représente le taux de décroissance dans le temps du potentiel de membrane en l'absence de stimulation entrante. Une analogie pertinente serait de se représenter une fuite d'eau d'une bouteille d'eau. Dans cette image, τ_{leak} est un coefficient qui serait inversement proportionnel à la taille du trou dans la bouteille percée – voir FIGURE 2.14. Le réglage de ce paramètre pose lui aussi une réflexion. En effet, si le taux est trop petit, la tension diminuera trop vite, empêchant ainsi les neurones de réagir à des stimulations espacées dans le temps. Dans le cas contraire, le déclenchement d'un potentiel d'action pourra être causé par des stimuli trop éloignés dans le temps et qui ne décrivent pas le même type de données (par exemple, deux classes différentes dans un problème de classification). Comme pour le seuil d'activation, ce paramètre permet donc de contrôler l'intensité de l'activité en sortie du neurone. C'est pourquoi, lors du réglage du seuil, il faut également considérer l'impact du taux de fuite du neurone : si celui-ci fait décroître trop rapidement le potentiel (diminution de l'activité), alors un seuil plus bas sera susceptible de palier la baisse d'activité.

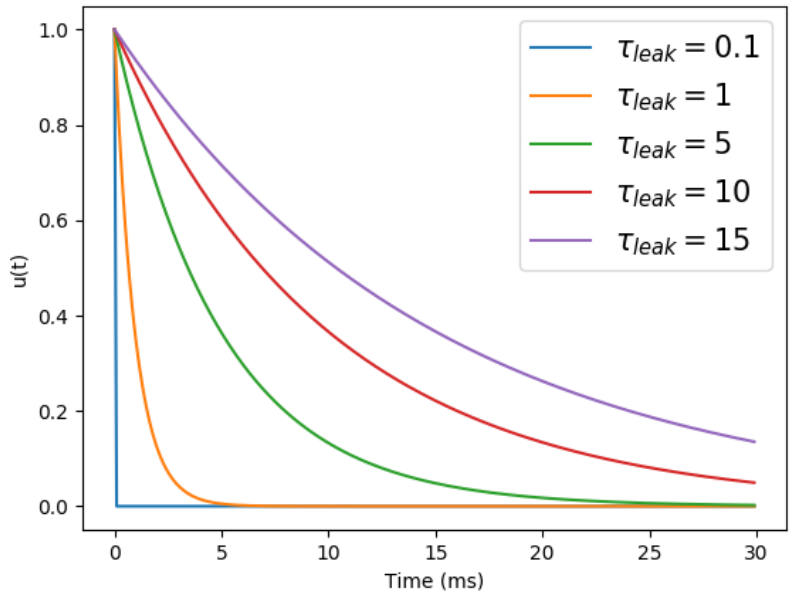


Figure 2.14 – Fuite du neurone avec différentes valeurs τ_{leak}.

Le troisième paramètre est la ***période de réfraction*** d'un neurone T_{refrac}. C'est une période pendant laquelle aucune impulsion ne peut être émise par le neurone. Cette période intervient juste après un spike. Le réglage de ce paramètre permet de contrôler la cadence maximale d'émission de spikes pour un neurone. Ce mécanisme permet au neurone de ne pas réagir à tous les stimuli d'entrée en appliquant un filtre temporel.

2.2.5 Synapses

2.2.5.1 Modulation des poids synaptiques

Afin que le système puisse apprendre des motifs utiles dans le but de résoudre des tâches précises, il est nécessaire que le modèle change d'état. Dans la SECTION 2.2.1 et la SECTION 2.2.3, nous avons montré que pour qu'un réseau puisse apprendre, il faut que les relations entre les différents neurones évoluent par le biais d'une règle d'apprentissage. Ces relations sont décrites par des synapses qui comportent des paramètres permettant cette modulation. L'un de ces paramètres est le poids synaptique w qui module la tension des spikes transmis, faisant varier l'influence qu'à un neurone présynaptique sur un neurone postsynaptique. Si ce facteur de modulation est faible, avec un poids proche de zéro, le neurone postsynaptique intégrera les spikes entrants avec des tensions très faibles qui auront un impact léger voir inexistant sur le potentiel de membrane. Au contraire, un poids élevé permettra à la synapse de délivrer des spikes dont les tensions sont grandes, affectant de manière significative l'état du neurone post. Les poids définissent des structures dans le réseau qui sont spécialisées sur des caractéristiques liées aux données traitées par le SNN. L'entraînement permet de déterminer les bonnes caractéristiques servant à résoudre la tâche demandée via une règle d'apprentissage.

2.2.5.2 Modulation des délais synaptiques

Un autre moyen de pouvoir moduler la connexion entre deux neurones est d'utiliser le délai synaptique d. Il permet de faire varier le délai de transmission d'une impulsion émis par un neurone présynaptique au neurone postsynaptique $t_{post} = t_{pre} + d$. Ce mécanisme est propre au modèle SNN et nécessite des topologies récurrentes si on veut mettre en place un équivalent pour les RNAs classiques. Il permet de créer des structures sensibles à des motifs temporels. En effet, avec des délais bien choisis, des groupes de neurones peuvent activer un neurone de sortie, s'il y a eu une séquence d'impulsions dont les activations se produisent à des moments précis. Pour cela, les délais introduisent une latence dans la transmission de certaines impulsions afin que le neurone cible reçoive la totalité des spikes au même moment. Il faut donc que les neurones présynaptiques s'activent en respectant une certaine chronologie avec un timing précis. Ces structures peuvent être complexes, on parle de groupe de neurones polychrones (Guise, Knott et Benuskova 2015). Dans les systèmes visuels biologiques, on retrouve ce type de structure, notamment pour la détection de mouvements. Certains modèles mathématiques se basent sur le principe d'activation synchrone d'un ensemble de neurones comme les détecteurs Hassenstein-Reichardt, inspirés par l'étude de la réponse optomotrice du coléoptère Clorophanus. Ce modèle calcule la corrélation entre le signal d'un photorécepteur et le signal retardé d'un photorécepteur voisin. En choisissant les bons délais synaptiques, on se retrouve avec un détecteur sensible à la présence d'un mouvement dans une direction et une vitesse préférée (voir FIGURE 2.15).

Ces détecteurs peuvent être utilisés pour être sensible à des motifs spatio-temporels. La solution de (Orchard et Etienne-Cummings 2014) utilise des délais pré-définis afin de construire un ensemble de détecteurs sensibles à une vitesse et une direction d'un mouvement dans le but d'extraire du flux optique (mouvement apparent des objets, surfaces et contours d'une scène visuelle).

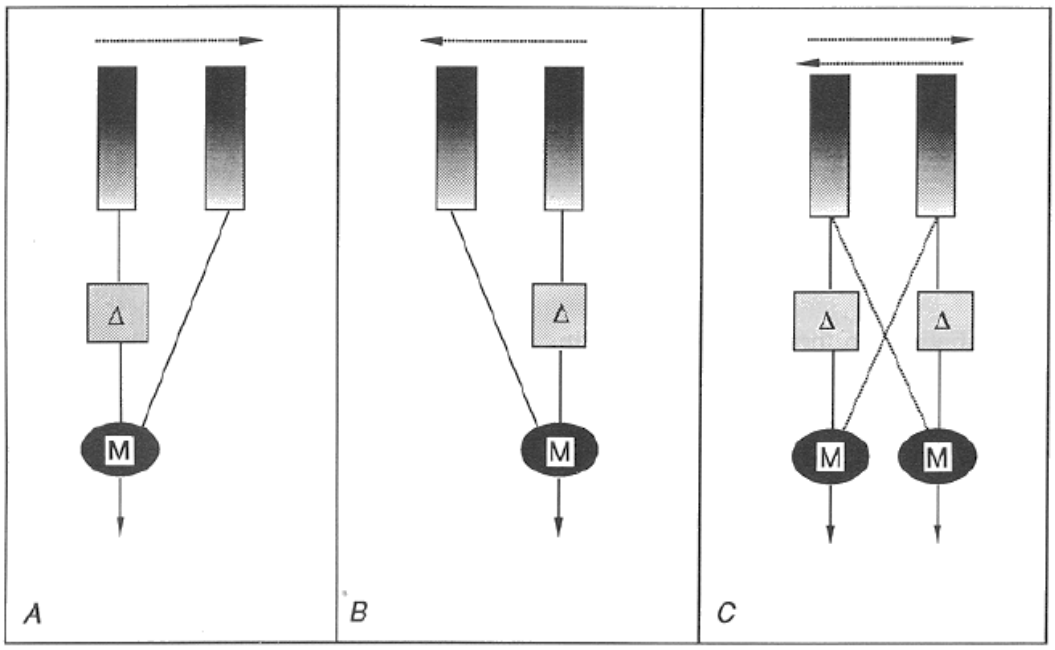


FIGURE 2.15 – Détecteurs de Reichardt illustrés dans l'article (REICHARDT 1961). Dans le panneau A, afin d'être sensible à un mouvement vers la droite avec une vitesse Δ, un délai ayant un retard égal au temps de déplacement du mouvement Δ est appliqué à la synapse de gauche du détecteur. Par conséquent, lorsque le photorécepteur de droite s'active, les deux impulsions arrivent au même moment, ce qui déclenche le neurone de sortie. Le panel B permet de visualiser le cas d'un mouvement dans la direction opposée (vers la gauche), avec un délai qui est appliqué cette fois ci à la synapse de droite du détecteur. Le panel C décrit un détecteur à deux sorties, sensible à deux types de direction.

Les délais peuvent être aussi utilisés afin d'augmenter la sélectivité des neurones sur des motifs spatio-temporels. La méthode *Delay selection* duplique les synapses entre un neurone présynaptique et une neurone postsynaptique avec des délais de transmission différents. Lorsqu'un neurone présynaptique se déclenche, le spike est propagé dans les différentes synapses dupliquée. Le neurone postsynaptique reçoit ce stimuli plusieurs fois à des temporalités distinctes. Lors de l'entraînement, la modulation des poids par une régle d'apprentissage va sélectionner les synapses ayant les délais les plus adaptés (PAREDES-VALLÉS, SCHEPER et DE CROON 2019).

Enfin l'apprentissage des délais peut se faire directement avec des règles d'apprentissage comme le *Delay Shift* qui va chercher à réduire le temps entre les spikes reçus et émis lorsqu'il y a un lien causal et augmenter le temps dans le cas contraire (EURICH et al. 1999).

Il faut toutefois noter que l'apprentissage conjoint des poids synaptiques et des délais est un problème de calcul NP-complet (PAUGAM-MOISY et BOHTE 2012).

2.2.6 Règles d'apprentissages

En se référant aux observations biologiques, sur la plasticité des synapses et les mécanismes régissant ce phénomène, Donald Hebb à élaborer une théorie énonçant les principes

fondamentaux permettant d'expliquer le phénomène d'apprentissage dans un système biologique. La règle d'apprentissage STDP est une méthode inspirée du fonctionnement du cerveau. Son principe global consiste à augmenter les connexions (LTP) des neurones afférents impliqués dans la décharge du neurone postsynaptique et à diminuer les connexions dans le cas contraire (LTP) (voir SECTION 2.2.1). La règle agit comme un détecteur de coïncidences et de corrélations dans le domaine temporel. En terme computationnel, cette règle peut être modélisée de manière locale dans l'espace et le temps. Cela permet de créer des architectures matérielles dédiées, exploitant ces propriétés afin de fournir des solutions peu coûteuses en énergie. De plus, l'apprentissage ne se base que sur les impulsions permettant un entraînement sans supervision.

Cette règle possède plusieurs variantes développées. Si on se réfère à la modélisation de (Masquelier et Thorpe 2007), nommé Classical-STDP (C-STDP), proche des observations biologiques, celle-ci peut s'exprimer de la manière suivante :

$$\begin{aligned} &\Delta t = t_{post} - t_{pre} \\ &\Delta w = stdp(\Delta t) = \begin{cases} \Delta w_+ \times \exp \frac{\Delta t}{T_{LTP}} & \text{si } t_{pre} \leq t_{post} \\ \Delta w_- \times \exp \frac{\Delta t}{T_{LTD}} & \text{si } t_{pre} > t_{post} \end{cases} \end{aligned} \tag{2.8}$$

Avec :

- t_{pre} et t_{post} : moment d'émission d'un spike pré- et postsynaptique ;
- Δw : variation apportée à la connexion synaptique entre le neurone postsynaptique et le neurone présynaptique ;
- Δw_+ et Δw_- : amplitude de la modification appliquée suivant une LTP ou une LTD ;
- T_{LTP} et T_{LTD} : taille de la fenêtre temporelle LTP et LTD ;

Avec ce type de modélisation, l'amplitude de la variation au sein de la LTD et de la LTP est régie par la différence de temps entre le moment où l'impulsion présynaptique et l'impulsion postsynaptique se produisent. Cette amplitude décroît de façon exponentielle par rapport à cette différence. Il existe des variantes à la règle STDP qui proposent de calculer cette amplitude autrement. L'Anti-STDP inverse les amplitudes LTP/LTD afin de correspondre à certaines observations biologiques. Le but est d'équilibrer les poids en contrebalançant certains biais qu'une STDP peut introduire, comme des corrélations excédentaires (Rumsey et Abbott 2004).

La STDP additive est une simplification de la règle C-STDP présentée puisque l'amplitude à une valeur fixée pour la LTP et pour la LTD (Bichler et al. 2012). La STDP multiplicative ajoute un terme dans ces calculs qui prend en compte la valeur du poids courant. L'impact de ce terme est réglé par un facteur. Le but est de contrôler les effets de saturation (poids proches des intervalles w_{min} et w_{max}) (Falez et al. 2019a). La Probabilistic-STDP (P-STDP) (Tavanaei, Masquelier et Maida 2016) calcul l'amplitude appliquée lors d'une LTP en se basant sur le poids synaptique courant et attribut une valeur fixée par un paramètre lors d'une LTD. Une conséquence de cette règle est que les poids sont bornés. La valeur maximale étant égale au logarithme de la probabilité qu'a le neurone présynaptique de se déclencher avant le neurone postsynaptique pendant la durée T_{LTP}. Cette règle permet d'avoir de bonne performance avec les neurones d'Izhekevich.

La Stable-STDP introduit par (Paredes-Vallés, Scheper et De Croon 2019) est une règle qui calcul l'amplitude en fonction de l'activité des neurones présynaptiques au cours du temps. Elle intègre un indicateur d'excitabilité dynamique qui diminue et réduit l'amplitude lorsqu'un neurone postsynaptique est lies à des neurones présynaptiques très actifs, et inversement, augmente lorsqu'ils se déclenchent peu. Ce processus particulier permet l'autorégulation des poids synaptiques sans mettre en œuvre de limites explicites, le tout en prenant en compte les variabilités des données d'entrées.

Si nous revenons à la modélisation décrite par l'ÉQUATION (2.8). Elle ne permet pas de prendre en compte tous les spikes émis, mais seulement un couple composé des derniers spikes produit respectivement par le neurone présynaptique et le neurone postsynaptique. Lorsque la fréquence d'émission des spikes est élevée, la STDP se déclenche sur des fenêtres temporelles très proches. Une conséquence est qu'il devient complexe de choisir une paire puisque suivant les fenêtres temporelles, une paire de spikes peut être considéré chronologiquement en $t_{pre} < t_{post}$ dans une fenêtre et inversement si on prend une autre fenêtre. Afin d'avoir des associations moins arbitraires, on peut considérer un triplet composé de deux spikes pré- et un spike post (ou inversement). La règle STDP-Triplet implémente ce principe en gardant un historique des traces synaptiques (Pfister et Gerstner 2006). D'autres règles comme la Mirror-STDP déplace la fenêtre temporelle de la LTP de telle sorte à prendre en compte les neurones corrélés lors de leur déclenchement au détriment du lien causal qui définit la relation (Burbank 2015).

2.2.7 Topologies des réseaux

D'un point de vue structurel, un réseau de neurones peut être perçu comme un graphe orienté pondéré où les neurones sont les sommets et les synapses sont les arêtes. En étudiant le cerveau humain, on retrouve différentes organisations dans la façon dont sont connectés les neurones. Par exemple, dans le cortex visuel, les connexions sont globalement construites hiérarchiquement. L'information capturée par la rétine va se propager en avant vers les différentes régions du cortex qui vont les traités en extrayant des caractéristiques de plus en plus complexes (neurones de la couche V1, connectés aux neurones de V2, eux-mêmes connectés à V3, etc.). Bien sûr, cette hiérarchie n'est pas totalement respectée puisqu'on observe des connexions allant des régions les plus profondes aux régions les plus proches de la rétine, voire des connexions internes entre les neurones d'une même région.

On peut catégoriser ces différentes topologies en deux grands groupes. Les topologies *feedforward*, où les connexions sont organisées en couches, sans cycle et les topologies *reccurent* qui admettent des cycles. De plus, on retrouve différentes architectures possibles lorsqu'on connecte un neurone aux autres unités.

Dans les RNAs, de manière générale, la topologie la plus utilisée et la plus maîtrisée reste celle où on organise hiérarchiquement les neurones en différentes couches. Le but est d'avoir des couches qui vont se spécialiser en extrayant des caractéristiques de plus en plus complexes. Dans cette topologie, il n'y a aucune boucle dans le réseau. L'information circule toujours vers l'avant, de l'entrée à la sortie, jamais vers l'arrière. Ces réseaux sont construits comme une succession de couches de neurones. On fait généralement la distinction entre couche d'entrée qui correspond à la première, couche de sortie qui correspond à la dernière et couches cachées qui correspondent à toutes celles qui se trouvent entre la couche d'entrée et celle de sortie. Lorsqu'il y a plus de deux couches (en excluant la

couche d'entrée) dans le réseau, on le définit comme *profond*. C'est ce type de topologie qui est utilisé dans l'*apprentissage profond* (*deep learning* en anglais). En augmentant le nombre de couches, on augmente l'expressivité du modèle dans sa capacité à avoir des représentations internes pertinentes. La mise en place de modèle multicouche SNN est plus complexe en comparaison aux réseaux classiques. L'une des principales difficultés est de pouvoir propager les impulsions au travers de toutes les couches, en maintenant une activité suffisante pour les neurones des couches profondes (Falez et al. 2019a). Enfin, il existe plusieurs stratégies permettant de connecter ces différentes couches. Les deux plus connues sont les couches pleinement connectées et les couches convolutives.

2.2.8 Architectures des réseaux

2.2.8.1 Dense

Cette stratégie est l'une des premières qui a vu le jour, elle consiste à connecter un neurone à tous les neurones de la couche suivante (voire à la totalité du réseau). Cela permet au neurone de traiter l'information globale présentée. Cependant, plus un réseau est grand en taille, plus le nombre de connexions augmente exponentiellement. Cela limite la taille des réseaux et nécessite un coût en calcul important. De plus, les règles d'apprentissage permettant de régler les poids deviennent inefficaces à cause du trop grand espace de valeurs possibles. Il faut une grande quantité de données d'entrée pour espérer pouvoir explorer efficacement cet espace.

2.2.8.2 Convolutionnelles

Il y a cependant plusieurs problèmes qui se posent lorsqu'on applique des connexions denses entre les neurones. Le nombre de synapses augmente de manière exponentielle en premier lieu, compliquant l'apprentissage. De plus, lorsqu'on traite des images, on souhaite préserver les propriétés de spécialisations des neurones, indépendamment d'une localisation spatiale particulière. Lorsqu'un neurone apprend à reconnaître un certain motif, cette propriété doit respecter une invariance en translation. Les couches convolutives répondent à ces deux problématiques. On applique un ensemble de filtres effectuant un pavage (chevauché ou non selon les paramètres), dont le but est d'extraire un ensemble d'attributs de l'image (comme des couleurs ou les oreilles d'un chat), selon la dimension (ou champ récepteur) de ses filtres. La fonction de convolution 2D peut être définie de la façon suivante :

$$C(i,j) = \sum_m \sum_n I(m,n)F(i-m, j-n). \tag{2.9}$$

Avec $I(m,n)$ le signal d'entrée et $F(i-m, j-n)$ le filtre de convolution. Dans la FIGURE 2.16, une schématisation possible de l'opération est proposée.

Dans le cadre d'un réseau de neurones convolutionnels, les coefficients d'un filtre sont vus comme les paramètres à apprendre. Les poids sont donc partagés et calculés sur toute l'image, permettant de limiter le nombre de connexions avec une invariance en translation.

En vision artificielle, la plupart des réseaux sont conçus avec ce type d'architecture. Les réseaux de neurones impulsionnels convolutifs (Convolutional Spiking Neural Network, CSNN) sont utilisées pour résoudre différentes tâches : reconnaissance de motifs (Falez

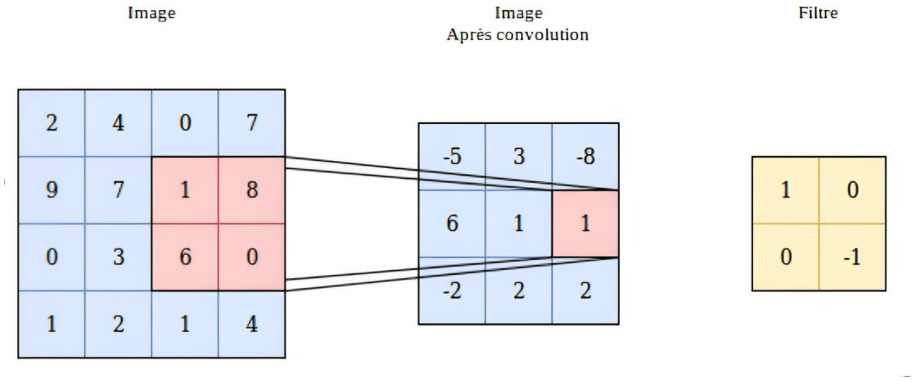


FIGURE 2.16 – Convolution 2D classique avec un filtre 2×2.

et al. 2019a ; MASQUELIER et THORPE 2007 ; TAVANAEI et MAIDA 2017), extraction du flux optique (PAREDES-VALLÉS, SCHEPER et DE CROON 2019), détection d'objets (S. KIM et al. 2019), etc.

2.2.9 Codage de l'information

La différence majeure entre les RNAs classiques et les SNNs réside dans la communication des informations entre les neurones du réseau qui s'effectue par des séquences d'impulsions. Ce mécanisme est inspiré de l'étude des systèmes biologiques où l'information circule via des impulsions discrètes et non de manière continue (KHERADPISHEH et al. 2018). Cela implique un changement de paradigme par rapport aux RNAs classiques, dans la manière d'encoder et d'analyser l'information. Cette approche offre plus de liberté dans la représentation des informations traitées et permet de choisir avec finesse des encodages maximisant certaines propriétés. Des facteurs comme la densité d'impulsions nécessaires pour décrire les données permettent de choisir entre des représentations qui encodent l'information avec une faible redondance dans les données transmises au détriment de la robustesse concernant les modifications liées au bruit inhérent à tout système. Par conséquent, l'information peut être codée par des caractéristiques liées à des notions fréquentielles (nombre d'impulsions) ou à des notions temporelles (moment où une impulsion est émise). Suivant la représentation choisie, nous pouvons adapter les différents mécanismes d'un SNN pour améliorer les performances, notamment sur la consommation énergétique du réseau, sur ses performances dans le traitement des taches et sur sa vitesse d'exécution. On répertorie l'utilisation de plusieurs types d'encodage, avec pour chacun des avantages différents. Les débats autour de cette question restent ouverts. Si on se réfère à la biologie, la question sur le type d'encodage n'est pas tranchée (BRETTE 2015) et suggère que plusieurs représentations cohabitent dans différentes régions du cerveau. Par exemple, des études suggèrent que dans certaines zones du cerveau, ce n'est pas le nombre d'impulsions qui régit la sémantique de l'information mais plutôt le moment où les impulsions sont émises. En effet, la rapidité de traitement dans ces zones ne correspond pas à une représentation fréquentielle de l'information puisque ce type d'encodage nécessite une fenêtre temporelle trop grande, incompatible avec la durée de réponse de ces régions du cerveau (KRUNGLEVICIUS 2016).

2.2.9.1 Codage fréquentiel

On retrouve ce type de comportement dans des zones régissant le fonctionnement des muscles ou des régions du cortex visuel. Les fonctions d'activation propres aux réseaux de neurones classiques et les valeurs qu'elles retournent peuvent être vu comme une représentation des fréquences d'émissions d'un neurone biologique sur une fenêtre de temps donné. Lorsqu'on utilise les SNNs, la plupart des données d'entrée sont composées de nombres scalaires et nécessitent donc une fonction de conversion afin de les transformer sous la forme d'une séquence d'impulsions. En pratique, cette fonction suit une loi de Poisson qui permet de générer une séquence d'impulsions qui se produisent à des moments aléatoires sur une fenêtre de temps donné et dont la distribution respecte la fréquence paramétrée. Cela permet d'avoir une représentation robuste au bruit puisque l'information est encodée avec un grand nombre d'impulsions. Par contre, pour pouvoir représenter précisément une valeur, l'intégration des impulsions entrantes doit être faite sur une fenêtre de temps suffisamment grande ou avec des fréquences moyennes élevée. Ces conditions ont des conséquences sur la vitesse de traitement du réseau en introduisant une certaine latence. Dans le cortex visuel, certaines observations montrent qu'une telle latence n'est pas compatible avec les mesures biologiques faites.

2.2.9.2 Codage temporel

À l'opposé du codage fréquentiel, le codage temporel représente les données en utilisant l'information temporelle d'un spike, en se focalisant sur le moment où celui-ci est émis par un neurone. De ce fait, un seul spike peut suffire pour encoder une information. Cette forme d'encodage est très économe en énergie puisqu'un seul spike est nécessaire pour transmettre l'information. De plus, les problèmes de latence soulevés par un codage fréquentiel sont réduits. En revanche, elle est plus sujette aux erreurs qu'une stratégie basée sur la fréquence, car de légères variations entraînent une modification complète du contenu de l'information. Il existe plusieurs méthodes permettant d'implémenter ce principe. Le codage par latence ("latency coding") est une représentation où les valeurs les plus élevées sont encodées par les spikes les plus précoces tandis que les derniers spikes représentent les valeurs les plus faibles. Autrement dit, la latence par rapport à une référence temporelle localisant le début des stimuli présentés est inversement proportionnelle à la valeur encodée. Une autre stratégie, nommée le codage par rang ("rank order coding") se focalise sur l'ordre d'arrivée des spikes plutôt que sur les moments exacts où ils sont émis. Ainsi, le premier neurone ayant produit le spike le plus tôt représentera la valeur la plus élevée.

2.3 Données événementielles

Les capteurs d'images conventionnels échantillonnent la scène visuelle à des périodes temporelles fixes. Tous les pixels acquièrent la lumière de manière synchrone en intégrant les photons sur une période de temps fixe. Lors de l'observation d'une scène dynamique, cette fréquence d'image, quelle que soit sa valeur, ne permettra pas d'acquérir fidèlement les dynamiques temporelles de la scène car il n'existe aucune relation entre la période d'acquisition d'une image et l'évolution dans le temps de la scène. Il en résulte un sur-échantillonnage et un sous-échantillonnage simultanés de différentes parties de la scène,

un flou de mouvement pour les objets se déplaçant et une redondance des données pour les arrière-plans statiques (Farabet et al. 2012). Or, pour de nombreuses applications, notamment celles qui impliquent une analyse du mouvement, seuls les changements locaux au niveau du pixel et leurs synchronisations sont vraiment pertinents. Ces changements locaux représentent une petite fraction de toutes les données transmises par un capteur conventionnel. Lorsqu'on analyse du mouvement sur des séquences vidéo standard, des opérations de filtrage sont nécessaires afin d'extraire ses caractéristiques (variation des pixels dans le temps) induisant un coût en calcul supplémentaire. Lorsqu'il n'y a pas ou peu de mouvement/variation, l'encodage/décodage d'un pixel qui ne varie pas semble inutile car il ne fournit pas d'informations supplémentaires. D'ailleurs, l'élimination de ces informations redondantes est à la base même de tout algorithme de compression vidéo.

De ce fait, un nouveau type de capteur à vu le jour, répondant aux critiques faites plus haut sur les caméras classiques. Ce sont les caméras événementielles qui ont la particularité d'être des capteurs asynchrones. Cela représente un changement de paradigme dans la façon dont l'information visuelle est acquise (Gallego et al. 2020). En effet, elles échantillonnent la lumière en fonction de la dynamique de la scène, plutôt qu'en fonction d'une horloge globale qui n'a aucun rapport avec la scène vue. Ces capteurs imitent la rétine biologique et plus particulièrement la voie magnocellulaire qui fonctionne de manière asynchrone en étant sensible aux variations rapides de luminosités. Ces caméras envoient donc des événements asynchrones à chaque fois qu'un pixel détecte un changement de luminosité, éliminant ainsi la transmission de données redondantes. La FIGURE 2.17 illustre les différences induites par la méthode d'acquisition des caméras standards et événementielles.

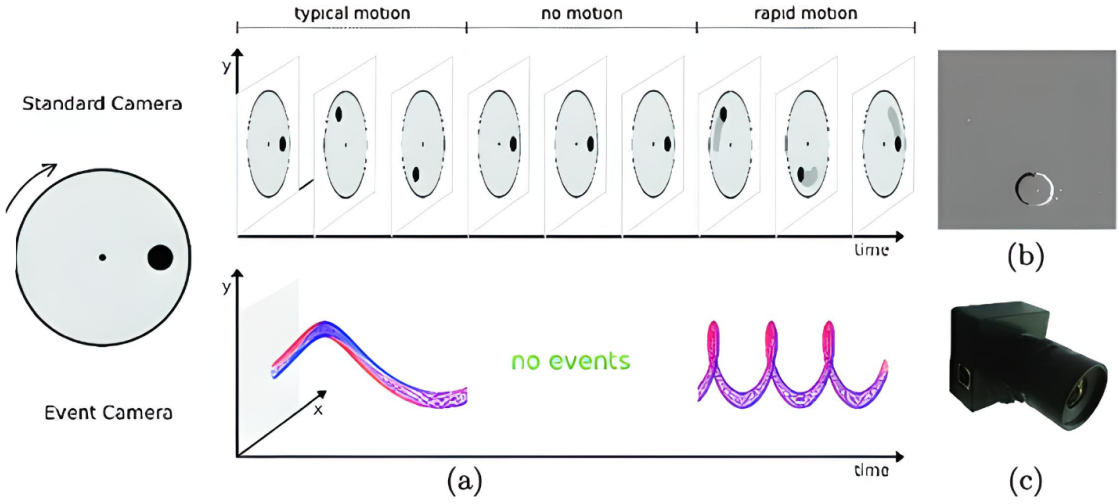


Figure 2.17 – Différences entre des caméras standards (frame-based, graphique du haut) et événementielles (event-based, graphique du bas) (H. Kim, Leutenegger et Davison 2016)

Ainsi, la sortie d'une caméra événementielle est une séquence d'évènements, chaque événement représentant un changement de luminosité (intensité logarithmique) d'une ampleur prédéfinie sur un pixel à un moment donné. Techniquement, chaque pixel mémorise l'intensité logarithmique $\log(I_{x,y,t})$ à chaque fois qu'il envoie un événement. Dès qu'il y a

un changement de variation lumineuse d'une ampleur suffisante (dépasse un seuil défini I_{thres}) par rapport à cette valeur mémorisée, un événement est généré. Cet événement ($< x, y, t, p >$), peut être décrit par l'emplacement spatial x, y, le temps t et la polarité p (c'est-à-dire l'augmentation ("ON") ou la diminution ("OFF") de la luminosité). L'ÉQUATION (2.10) définit un évènement :

$$\begin{aligned} \Delta I_{x,y,t} &= \log(I_{x,y,t}) - \log(I_{x,y,t-1}) \\ (x, y, t, p) &= \begin{cases} (x, y, t, \text{ON}) & \text{si } \Delta I_{x,y,t} \geq I_{thres} \\ (x, y, t, \text{OFF}) & \text{si } \Delta I_{x,y,t} \leq -I_{thres} \end{cases} \end{aligned} \tag{2.10}$$

Les événements sont transmis de la matrice de pixels à la périphérie, puis hors de la caméra à l'aide d'un bus de sortie numérique partagé, généralement en utilisant le format `Address-event representation` (AER) (Boahen 2004). Avec ce type de fonctionnement, les caméras événementielles ont des avantages sur les taux de lecture très élevés allant de 2MHz (Lichtsteiner, Posch et Delbruck 2008) à 1200MHz (Suh et al. 2020), selon l'implémentation matérielle, en consommant peu d'énergie (24mW).

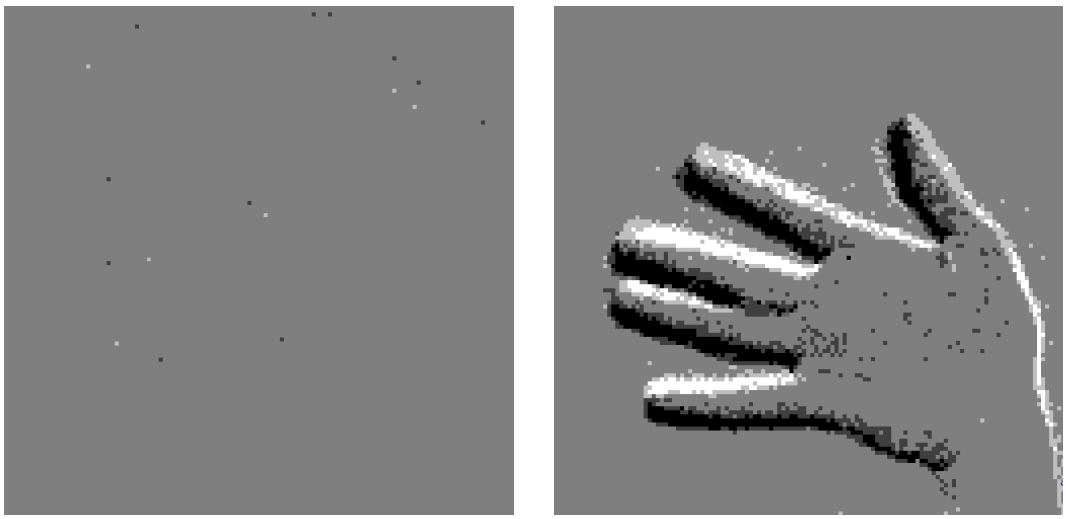
Avec cette représentation, il est très facile de pouvoir associer un évènement à un spike. Concrètement, on peut associer deux neurones à chaque pixel, chacun s'occupant d'une polarité, il ne reste plus qu'à parcourir l'ensemble de la file d'évènements afin d'activer au bon moment le neurone concerné.

À l'heure actuelle, il existe plusieurs caméras événementielles sur le marché. L'un des premiers modèles implémentant les principes cités plus haut est la caméra Dynamic Vision Sensor (DVS) fabriqué par l'entreprise iniVation (Lichtsteiner, Posch et Delbruck 2008). Sa conception repose sur une rétine en silicium, où les photorécepteurs sont exposés à la lumière en continu. Ce dispositif est couplé de manière capacitive à un circuit de lecture qui est réinitialisé chaque fois que le pixel a été échantillonné (Delbruck et Mead 1991). Avec ce type de conception, la caméra est sensible aux variations temporelles. Bien qu'un grand nombre d'applications puissent être résolues en ne traitant que les événements (c'est-à-dire les changements de luminosité) comme la reconnaissance de gestes (Amir et al. 2017), l'analyse du mouvement (Bichler et al. 2012 ; Paredes-Vallés, Scheper et De Croon 2019) ; il est devenu évident que certaines d'entre elles nécessitent également une sortie représentant le contraste spatial d'une scène (c'est-à-dire une luminosité "absolue" ou en d'autres termes, des images issues d'un capteur classique). En effet, lorsqu'on acquiert une scène où il n'y a pas de mouvement, théoriquement, il n'y a pas d'information qui peut être capturée par ce type de caméra (voir les images de la FIGURE 2.18).

Pour remédier à cette lacune, plusieurs caméras ont été développées pour fournir simultanément des informations dynamiques et statiques. Le capteur Asynchronous Time Based Image Sensor (ATIS), conçu par le fabriquant Prophesee (Posch, Matolin et Wohlgenannt 2010), possède des pixels qui contiennent un sous-pixel DVS qui déclenche un autre sous-pixel pour capturer l'intensité absolue. D'autres caméras événementielles sont en cours de développement, comme les capteurs DVS-Gen (par Samsung), qui ont une résolution spatiale plus élevée mais sacrifient la résolution temporelle. On peut noter que depuis 2017, un réel marché à vu le jour, avec un catalogue de solutions proposées en constante évolution alimenté par un nombre de fabriquant qui ne cesse d'augmenter. Cette tendance montre un intérêt croissant pour ce type de technologies articulé par une demande dans différents secteurs comme la robotique.

Cependant, il faut tout de même souligner les difficultés auxquelles on doit faire face en utilisant ces capteurs. En effet, le fonctionnement des caméras événementielles étant fondamentalement différent des caméras classiques, notamment dû à cette notion d'asynchronicité liée aux mesures des variations lumineuses, il est donc nécessaire de créer de nouvelles méthodes capables de traiter les données générées en tirant parti des avantages proposés par ce type de technologie. Un nouveau champs de recherche a donc vu le jour sur la conception d'algorithme répondant à des problématiques classiques rencontrée en vision comme la reconnaissance de forme, de profondeur, etc. [1]

Il est tout de même possible d'utiliser les algorithmes classiques en vision en adaptant les données événementielles. Une pratique courante consiste à diviser les événements en tranches de temps fixes (20 ms, par exemple) (Zhao et al. 2015) et à les accumuler pour former une pseudo-image. En effet, chaque événement est associé à une position spatiale qui est utilisée pour éclairer le pixel correspondant dans l'image. Un code couleur est défini pour distinguer le type de variation associé. C'est cette méthode qui est utilisée pour visualiser les événements. Les images de la FIGURE 2.18 sont obtenues avec ce type de procédé. Récemment, des méthodes plus complexes ont vu le jour et permettent de reconstruire l'information de luminance à partir d'événements. On obtient des images proches de ce qui est obtenu via des caméras traditionnelles. Ce type d'algorithme nécessite plus de ressources en calcul, du fait de l'utilisation d'auto-encodeurs et/ou de réseaux profonds (Gehrig et al. 2020 ; Rebecq et al. 2019a ; Scheerlinck et al. 2020).



(a) Aucun mouvement (b) Mouvement d'une main vers le bas

Figure 2.18 – Visualisation obtenue avec la librairie `jAER`. Les pixels blanc et noir représentent les variations lumineuses (resp. ON et OFF). Dans l'image (a), la scène filmée ne contient aucun mouvement. On peut toutefois distinguer quelques événements que l'on qualifie de bruit. Dans l'image (b), on distingue les contours d'une main, bougeant du haut vers le bas. Si on connaît le contraste entre la main et l'arrière-plan, grâce à la distribution spatiale des polarités des événements, on peut inférer une information sur la direction du mouvement.

1. Pour aller plus loin, un document listant un grand nombre d'algorithmes développés pour des données événementielles est disponible – URL : https://github.com/uzh-rpg/event-based_vision_resources/blob/master/README.md.

Les avantages des capteurs événementiels sont fortement réduits si le flux d'événements est converti en frames synchrones. Ce type de méthode est encore assez courant, puisqu'il permet de faciliter le traitement des données à l'aide d'algorithmes conventionnels. De plus, le traitement des images se fait principalement avec des CPUs et des GPUs qui ne sont pas conçus pour traiter des données éparses sous forme de flux d'évènements asynchrones. Ils sont plutôt élaborés pour manipuler des structures de données denses et synchrones. Une des conséquences est que pour maintenir un débit de traitement régulier et rapide, ces architectures doivent avoir le plus d'instructions/données possibles présentes dans leur pipeline, quitte à exécuter des calculs redondants sur des données immuables (Chetlur et al. 2014).

Pour pouvoir exploiter au maximum le potentiel de ces caméras dans le traitement des données au niveau algorithmique et matériel, il est nécessaire d'utiliser une méthode qui partage les mêmes propriétés. Les SNNs font partie des modèles qui ont des propriétés similaires puisqu'ils sont parallèles, temporels et communiquent de manière éparse. De plus, une nouvelle génération de processeurs neuromorphiques est apparue (S. Furber 2016). Ils permettent de traiter nativement les flux événementiels. Ces systèmes *many-core* (processeurs multicœurs conçus pour un niveau élevé de traitement parallèle) instancient de grandes populations de neurones impulsionnels sur un matériel massivement parallèle et de faible puissance. Cela permet de conserver tous les avantages des capteurs événementiels, un SNN fonctionnant sur une puce neuromorphique peut réagir en quelques dizaines de millisecondes sur des données événementielles. Les échelles de temps entre les événements et les réponses des neurones impulsionnels sont les mêmes. Ainsi un mouvement rapide peut être identifié et traité. Avec ce type d'architecture, il est possible d'adresser des tâches en temps réel traitant de données représentant des scènes possédant des dynamiques temporelles très rapides (Amir et al. 2017).

Afin de pouvoir entraîner des modèles sur ce type de données, des jeux de données ont été développés. On peut notamment citer Poker-DVS, MNIST-DVS (Serrano-Gotarredona et Linares-Barranco 2015), NMNIST (Orchard, Jayawant et al. 2015) et CIFAR-DVS (Li et al. 2017) qui sont des versions événementielles de jeux de données historiques dans la communauté vision. Ces jeux de données ont été obtenus en utilisant un capteur DVS. Pour acquérir les données événementielles, le capteur ou l'image projetée par l'écran oscille légèrement au cours du temps afin que l'on puisse distinguer la texture cible. Le TABLEAU 2.1 liste les principaux jeux de données événementielles disponibles permettant d'entraîner des modèles sur des tâches de reconnaissance, d'objet, d'action, etc.

Nom	Nombre de classes	Description
N-MNIST	10	Reconnaissance de chiffres manuscrits
MNIST-DVS	10	Reconnaissance de chiffres manuscrits
DvsGesture	11	Reconnaissance des gestes
GESTURE-DVS	3	Reconnaissance des gestes de la main
AER-POSTURE	3	Reconnaissance des postures humaines
N-Caltech101	101	Reconnaissance d'objets
CIFAR10-DVS	10	Reconnaissance d'objets
Cards	4	Reconnaissance d'objets
Poker-DVS	4	Reconnaissance d'objets
N-CARS	2	Détection de voitures
DDD17	N/A	Estimation de l'égo-motion de véhicules
Freeway-DVS	Nombre de véhicules	Comptage de véhicules

TABLEAU 2.1 – Les différents jeux de données événementielles.

2.4 Conclusion

La vision par ordinateur est un domaine très actif, en raison de son large éventail d'applications. L'avènement des méthodes d'apprentissage profond a eu un impact significatif sur ce domaine, en améliorant considérablement les performances des méthodes d'analyse et de traitement de données artificielles sur des tâches complexes. Malgré leurs performances élevées sur des tâches complexes, les réseaux neuronaux profonds présentent l'inconvénient d'être très énergivores, limitant ainsi leur utilisation dans certains contextes. Les réseaux de neurones impulsionnels (SNN) offrent une solution prometteuse pour contourner ce problème, car ils permettent des architectures hautement économes en énergie, en particulier lorsque l'activité à l'intérieur du réseau est faible. Cependant, l'utilisation de tels réseaux impose certaines contraintes matérielles, telles que la localité du calcul et de la mémoire. Les SNNs disposent de plusieurs méthodes d'apprentissage, la STDP étant une règle qui permet de répondre aux exigences des architectures neuromorphiques, tout en proposant une solution qui optimise le modèle sans recourir à des données annotées. Cependant, ces règles sont encore à un stade précoce de développement et ne permettent pas encore de traiter de manière satisfaisante des tâches complexes. Toutefois, les SNNs, qualifiés comme la troisième génération de réseaux de neurones (Maass 1997), ouvrent de nouvelles perspectives en matière de traitement de l'information et s'inscrivent dans la continuité des RNAs. Le fonctionnement global des RNAs est très comparable à celui des SNNs, à l'exception que ces derniers requièrent une dimension temporelle pour encoder l'information. Cette différence notable permet aux neurones du modèle de caractériser des motifs spatio-temporels. De ce fait, les SNNs sont de bons candidats pour le traitement de signaux visuels. Ces signaux visuels peuvent être acquis par des caméras événementielles, dont la conception s'inspire de la rétine biologique. Ce type de caméra fonctionne de manière asynchrone en délivrant de façon parcimonieuse des données correspondant aux différentes variations lumineuses générées dans une scène. Dans la majorité des cas, les variations lumineuses sont causées par du mouvement, c'est pourquoi, on associe souvent ces caméras à des capteurs sensibles au mouvement. En

somme, les caméras événementielles reposent sur une technologie qui permet d'exploiter au maximum les avantages liés aux SNNs et à leur accélération matérielle. L'association de ces caméras aux puces neuromorphiques permet donc la création de systèmes de vision massivement parallèles fonctionnant en temps réel avec un faible coût énergétique. Ce manuscrit vise à explorer les capacités des SNNs dans le traitement vidéo, en capitalisant un maximum sur les avantages mentionnés tels que l'apprentissage local non supervisé via la règle STPD, l'encodage parcimonieux des données événementielles et la caractérisation de motifs spatio-temporels via le paramétrage des neurones du modèle.

La suite de ce manuscrit se concentre sur des tâches de classification de mouvement. Ce type de tâche permet d'étudier les propriétés des SNNs sur des données qui peuvent être facilement représentées par des événements. La première contribution se concentre sur les différents points évoqués en proposant un modèle capable de reconnaître les mouvements d'un motif se déplaçant dans les 4 directions cardinales sur un jeu de données synthétiques. Dans le CHAPITRE 4, différents aspects sont étudiés, comme la pertinence des caractéristiques extraites après un entraînement STDP, les conditions de convergence du réseau vers un état qui permet de résoudre la tâche présentée ainsi que l'impact du réglage des hyper-paramètres du modèle sur la sélectivité des neurones pour des motifs spatio-temporels. Suite aux conclusions faites lors de cette première étude, le CHAPITRE 5 se concentre sur la capacité du modèle à apprendre de manière incrémentale. Enfin, le CHAPITRE 6 présente une étude de cas sur l'utilisation des délais synaptiques lors de l'apprentissage afin de contrôler plus finement la sélectivité des neurones sur des motifs spatio-temporels.

Chapitre 3

Méthodologie : outils et modèle SNN

Ce chapitre a pour but d'introduire les détails techniques ainsi que les méthodes communes utilisées dans les études présentées dans le CHAPITRE 4 et le CHAPITRE 5.

3.1 Outils utilisés

Afin de pouvoir évaluer l'expérience, nous avons utilisé la librairie `brian2` (langage `python`). Celle-ci est employée pour simuler des réseaux impulsionnels dont le fonctionnement peut être complexe. Elle donne la possibilité de décrire les différents composants du réseau via des équations différentielles. Elle embarque plusieurs solveurs permettant la résolution de ces dernières. Bien sûr, il existe d'autres simulateurs, on peut distinguer deux catégories : les simulateurs *clock-driven* et *event-driven*. Pour `brian2`, il fonctionne en mode *clock-driven*. Cela signifie que le temps de simulation est divisé en une grille temporelle dont l'espacement est quantifié par une valeur minimale $(0, dt, 2 \times dt, 3 \times dt, \dots)$. À chaque pas de temps dt, les équations différentielles spécifiant les modèles sont d'abord intégrées en $t + dt$. L'avantage principal d'une telle méthode est que l'on peut simuler n'importe quel modèle, par contre, le coût en terme de calcul est plus important que pour un simulateur *event-driven* (moins flexible mais plus rapide).

3.2 Modèle SNN

3.2.1 Topologie du réseau

Notre réseau est un SNN de type feedforward composé d'une seule couche totalement connectée. Elle reçoit des données événementielles de manière continue via des neurones d'entrée représentant les pixels de la fenêtre (voir FIGURE 3.1). Parmi les modèles de neurones existants, nous avons choisi l'un des plus utilisés, à savoir le modèle LIF (plus de détails dans la SECTION 2.2.4). Concernant l'apprentissage, nous avons appliqué une règle STDP simplifiée dont l'ÉQUATION (3.1) est décrite plus haut (voir SECTION 3.2.4).

Par ailleurs, notre réseau dispose de plusieurs mécanismes décrites :

— *compétition neuronale* (voir SECTION 3.2.3), qui empêchent toutes les unités de sortie d'apprendre le même motif : les neurones ont une capacité d'inhibition latérale leurs permettant d'empêcher leurs voisins d'émettre des impulsions lorsqu'ils s'activent ;

— *période réfractaire*, permet d'éviter qu'une seule unité de sortie ne se déclenche en continu sans donner la moindre chance aux neurones voisins de pouvoir se déclencher ;

— *délai synaptique*, empêche que tous les spikes émis par un neurone ne parviennent en même temps aux neurones suivants. Ceci ajoute une diversité dans le traitement temporel de l'information par le modèle. De plus, il permet de se rapprocher des réalités biologiques observée. À noter que cela résout un détail technique liés à l'utilisation du simulateur. [1]

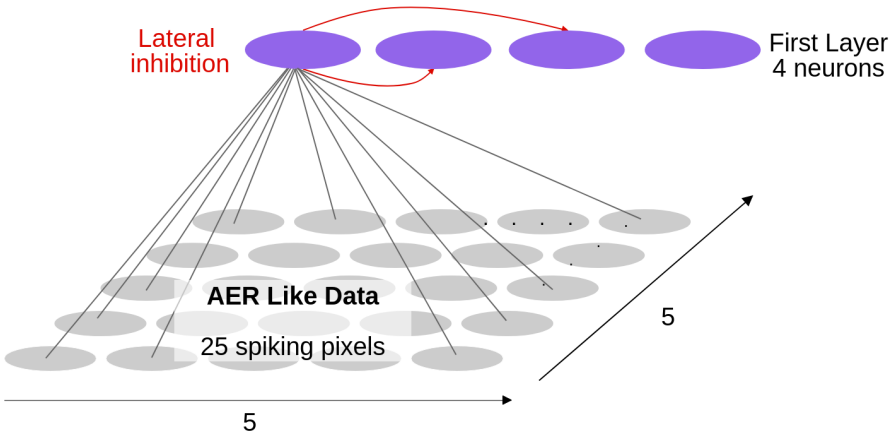


FIGURE 3.1 – Topologie utilisée pour cette étude.

1. Pour le dernier point, il faut savoir que lors d'une simulation, si on prend en compte les points suivants, à savoir : le type de données, la configuration du réseau, ainsi que la librairie utilisée – alors les impulsions émises en simultanées (causé par la nature synthétique des données) ne vont pas être pris en compte à l'instant t (causé par le fonctionnement de `brian2`), mais à $t+dt$ (voir SECTION 3.1), avec une propagation de cette synchronicité, non souhaitée, surtout lorsqu'on utilise l'inhibition latérale qui, dans ce contexte, ne permet pas une spécialisation des neurones. De ce fait, un délai synaptique est attribué à chaque connexion, celui-ci étant tiré de manière aléatoire avec une valeur ce situant de 1 ms à 3 ms.

3.2.2 Modèle du neurone

Le modèle que nous avons utilisé dans nos travaux est le *Leaky-Integrate Fire* (*LIF*). Les détails concernant le fonctionnement du neurone sont décrits dans la SECTION 2.2.4. Le choix d'un tel modèle est motivé par son coût en calcul faible et son expressivité permettant de modéliser et d'ajouter un ensemble de mécanisme intéressant, propre aux SNNs, afin d'avoir plusieurs facteurs intéressants pouvant résoudre certaine problématique rencontrer lorsqu'on essaye de résoudre une tâche précise.

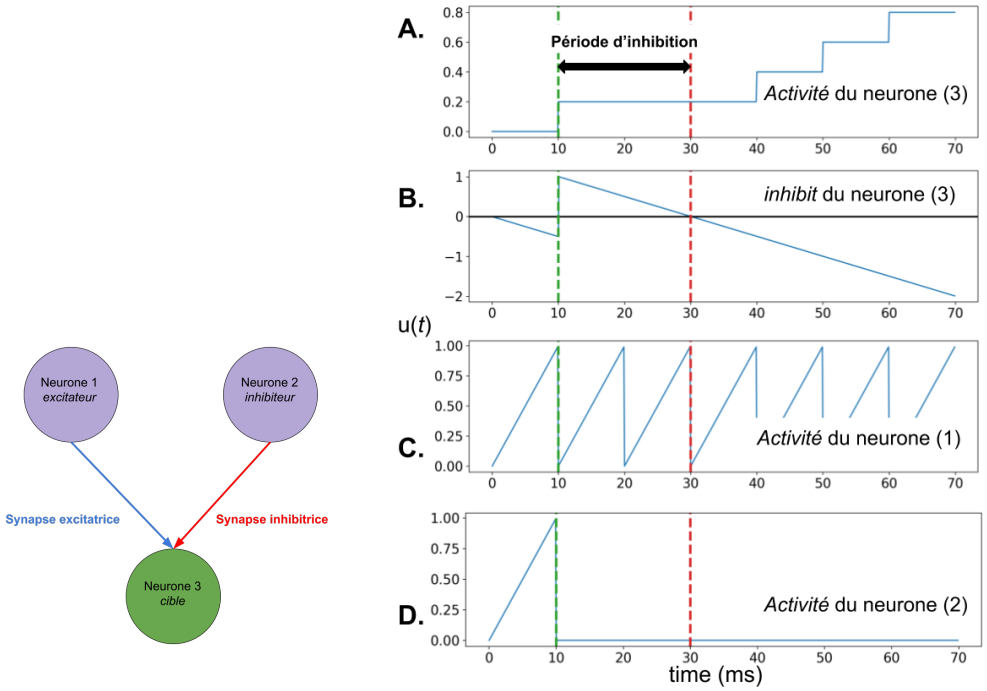
3.2.3 Inhibition latérale

Nous allons décrire la solution qui nous permet de mettre en place le mécanisme d'inhibition latéral du modèle, en illustrant les effets et les conséquences que cela implique lors d'une simulation.

Dans notre implémentation, chaque neurone dispose d'une variable *inhibit*, son évolution étant décrite par l'équation ($\frac{dinhibit}{dt} = -1 \,/\, T_{inhibit}$), $T_{inhibit}$ correspondant à un paramètre constant qui représente la période d'inhibition du neurone cible. Finalement, on peut la réécrire sous la forme d'une équation de droite (descendante) $-a*t$ dont le coefficient directeur est $a = 1/T_{inhibit}$ (voir FIGURE 3.2b, panel **B**). Si la variable $inhibit > 0$, cela veut dire qu'on est en période d'inhibition, on filtre donc les impulsions sortantes du neurone pour qu'il n'ait aucun impact sur le réseau ; dans le cas contraire où $inhibit \leq 0$, on laisse le neurone envoyer des spikes. Donc, pour qu'un neurone puisse déclencher une inhibition sur ses voisins, il lui suffit de mettre à 1 les variables *inhibit* des neurones cibles. On délègue ce mécanisme à des synapses particulières (en rouge sur la FIGURE 3.1), c'est elles qui déclenchent l'inhibition d'un neurone post-synaptique lorsqu'elles reçoivent une impulsions du neurone présynaptique. Dans la FIGURE 3.2, on décrit un exemple illustrant deux neurones (1 et 2) émettant un spike au même moment sur un troisième ne faisant qu'intégrer les entrées. Le neurone (1) envoie un simple signal de sortie tous les 10 ms (panel **C**), tandis que l'autre (2, panel **D**) déclenche une inhibition ponctuelle à 10 ms pour une durée de 20 ms sur le neurone (3). Le phénomène observé est que l'inhibition ne se déclenche pas exactement à 10 ms mais juste un peu après, laissant ainsi passer le spike du neurone (1). Une situation équivalente ce produit à la fin de la période d'inhibition, celle-ci se terminant à 30 ms, au même moment qu'une des impulsions du neurone (1). On observe alors que la période est encore active et que le spike n'est pas pris en compte. Ce petit exemple illustre les conséquences d'une telle implémention avec `brian2` sur des évènements synchrones.

3.2.4 STDP

Notre modèle utilise une règle STDP additive simplifiée lors de son apprentissage. Cette règle est tirée des travaux de Bichler et al. 2012. Nous avons appliqué une condition afin que les poids w soit bornée par l'intervalle $[w_{min}, w_{max}]$. La FIGURE 3.3 et l'équation l'ÉQUATION (3.1) présente l'implémention de cette règle.



(a) Topologie (b) Activité des neurones

FIGURE 3.2 – **Gauche** : Le neurone (1) *excitateur* émet des impulsions toutes les 10 ms. Le neurone (2) *inhibiteur* émet une impulsion à 10 ms. Le neurone (3) *cible* est connecté aux neurones présynaptiques (1) et (2). **Droite** : Illustration d'une inhibition. Le panel **A.** représente le neurone (3), celui-ci ne faisant qu'intégrer les impulsions reçu en entrées. Le panel **B.** montre l'évolution de la variable d'inhibition du neurone (3). Le panel **C.** montre le potentiel membranaire du neurone (1), celui ci excitant le neurone (3). Enfin le panel **D.** montre le potentiel membranaire du neurone (2) qui a pour rôle d'inhiber le neurone (3) lorsqu'une impulsion est émise.

$$
\begin{aligned}
&\text{clip}(a, min, max) = \begin{cases} min & \text{si } a < min \\ max & \text{si } a > max \\ a & \text{sinon} \end{cases} \\
&\Delta t = t_{pre} - t_{post} \\
&\Delta w = \text{stdp}(\Delta t) = \begin{cases} \Delta w_+ & \text{si } \Delta t < T_{LTP} \\ \Delta w_- & \text{sinon} \end{cases} \\
&w = \text{clip}(w + \Delta w, w_{min}, w_{max})
\end{aligned}
\tag{3.1}
$$

Les variables t_{pre} et t_{post} représentent respectivement le moment où les neurones pré- et post-synaptique ont émis un spike. Les périodes en bleues correspondent à une dépression et les périodes en rouges à une potentialisation. La fenêtre temporelle T_{LTP} permet de régler la période dans laquelle on active une LTP. Il est impératif de bien la paramétrer, en fonction de la durée du motif que l'on souhaite apprendre. Si T_{LTP} est trop petite, les premières synapses à ce renforcer risquent de désapprendre à l'étape suivante. Dans l'ÉQUATION (3.1), on remarque que la valeur du pas pour la LTP (w_+) est plus forte (deux fois dans ce cas-ci) que pour la LTD (w_-). Ce choix est motivé par le fait qu'il

y aura plus d'évènements LTD que LTP au cours de l'apprentissage. Ainsi, lorsqu'une potentialisation se produit, on lui donne plus d'impact. La nature de la règle présentée par l'ÉQUATION (3.1) est additive ou soustractive, ce qui sous-entend qu'un événement se produisant à un certain moment dans l'un des deux cas (LTD ou LTP), aura le même impact qu'un autre se produisant à un moment différent dans cette même période.

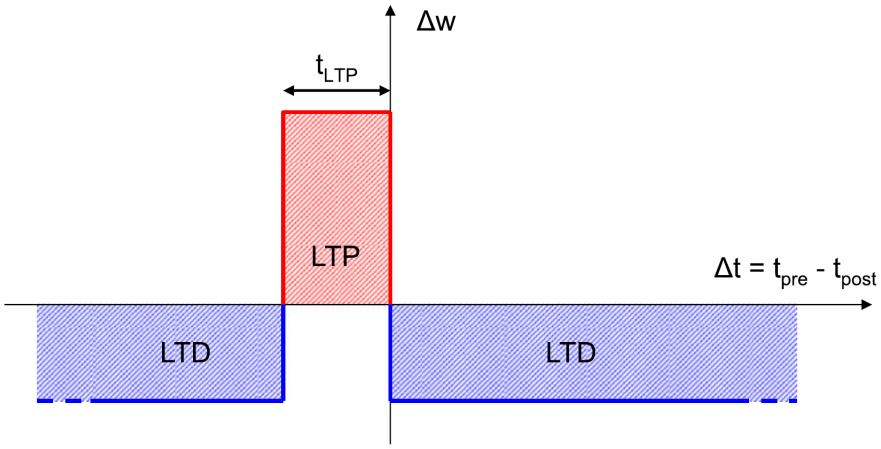


FIGURE 3.3 – Les synapses subissent une LTP quand $\Delta t < T_{LTP}$ et une LTD dans les autres cas. La valeur Δw correspond au changement imposé au poids synaptique à un instant donné. Image tirée de BICHLER et al. 2012.

3.2.5 Paramétrage du modèle

Le réseau décrit dans la SECTION 3.2.1 est composé de neurones LIF, de synapses implémentant une STDP simplifiée, avec une topologie feed-forward mono-couche du réseau. Malgré ces choix de conception qui se base sur des modèles simples de l'état de l'art (voir les SECTIONS 2.2.4 2.2.5, nous pouvons lister un ensemble de paramètres permettant de régler le modèle.

Le TABLEAU 3.1 décrit les paramètres globaux du modèle.

Paramètre	**Description**
N_{output}	Nombre de neurones de sortie.
N_{inhib}	Nombre de voisins à inhiber par neurone.
T_{LTP}	Durée de la période d'action d'une LTP (milliseconde).

TABLEAU 3.1 – Paramètres liés à l'architecture du réseau.

Le TABLEAU 3.2 décrit les paramètres appliqués à l'ensemble des neurones du réseaux. Le réglage des paramètres liées aux neurones du réseau a des conséquences importantes sur les performances du modèle. La sélectivité du réseau sur des motifs spatio-temporels est fortement liés à ces paramètres.

Paramètre	**Unité**	**Description**
V_{thres}	millivolt	Seuil d'activation des neurones.
T_{refrac}	milliseconde	Durée de la période réfractaire des neurones.
$T_{inhibit}$	milliseconde	Durée de la période d'inhibition.
τ_{leak}	1	Taux de fuite des neurones.

TABLEAU 3.2 – Paramètres liés aux neurones du réseau. Lorsque la valeur de l'unité est égal à 1, cela signifie que cette valeur est simplement décrit par un scalaire.

Le TABLEAU 3.3 décrit les paramètres appliqués à l'ensemble des synapses du réseaux. La qualité de l'apprentissage dépend de ces paramètres. La plupart des paramètres présents sur le tableau décrivent le comportement de la STPD.

Paramètre	**Unité**	**Description**
w_{min}	millivolt	Poids minimal.
w_{max}	millivolt	Poids maximal.
d_{min}	milliseconde	Délai minimal.
d_{max}	milliseconde	Délai maximal.
w	millivolt	Poids synaptique.
w_+	millivolt	LTP.
w_-	millivolt	LTD.
d	milliseconde	Délai synaptique.

TABLEAU 3.3 – Paramètres liés aux synapses du réseau.

Chapitre 4

Analyse non supervisée du mouvement

4.1 Motivations et objectifs de l'étude

Dans cette section, nous étudions la capacité d'un SNN de petite taille à reconnaître les directions d'un mouvement après un apprentissage non supervisé sur des données événementielles. Le but est de proposer une solution qui tire profit des avantages offerts par les SNNs, tels que l'apprentissage non supervisé et la prise en compte de la dimension temporelle dans le fonctionnement du modèle. Nous avons opté pour une méthode d'apprentissage basée sur la règle STDP, laquelle permet de modifier les poids avec des opérations locales et sans supervision. Afin de comprendre au mieux les différents facteurs influençant les performances du modèle, la tâche à résoudre est réduite à un problème de reconnaissance des 4 directions cardinales d'un mouvement engendré par un motif simple. Cette tâche est adaptée à l'échelle du réseau examiné. Dans ce cadre expérimental, il est possible de mener une étude minutieuse de la dynamique du réseau. Ces réseaux de petites tailles peuvent être considérés comme des détecteurs extrayant des caractéristiques spatio-temporelles de bas niveau. D'ailleurs, à plus long terme, nous pourrions agréger ces détecteurs élémentaires au sein d'architectures hiérarchiques multi-couches dans le but résoudre des tâches nécessitant l'analyse de caractéristiques spatio-temporelles plus complexes.

Dans la SECTION 2.2, des approches combinant des SNNs sur des données événementielles sont présentées. Deux d'entre elles (Bichler et al. 2012 ; Paredes-Vallés, Scheper et De Croon 2019) sont proches de nos objectifs, mais se différencient de nos travaux en prenant en compte la distinction faite entre les deux polarités d'évènements (ON et OFF) fournis par le capteur. En effet, en connaissant la polarité des évènements, le modèle peut utiliser cette information lors de son apprentissage pour déduire la direction d'un mouvement. En prenant en compte ce type d'information et en fonction des conditions d'acquisition de la scène traitée, le modèle peut donc réduire son analyse du mouvement à un problème de reconnaissance statique des formes. Pour mieux comprendre cette idée et la problématique qu'elle soulève, prenons l'exemple de la balle en mouvement utilisée lors des expériences de Bichler et al. 2012. Cette balle est de couleur claire et évolue en se déplaçant vers la droite sur un fond qui est plus sombre. Dans ces conditions, le bord droit de la balle va générer un ensemble d'évènements ON (ces pixels deviennent plus clairs) alors que le bord opposé à gauche va générer simultanément un ensemble

d'événements OFF (ces pixels deviennent plus sombres). Ainsi, l'information dynamique est codée de manière statique dans l'image intégrée des évènements : la position spatiale et la polarité des événements deviennent des caractéristiques suffisantes pour la reconnaissance d'une direction. Cependant, cette caractérisation du mouvement faite par le modèle ne permet pas de s'adapter à toutes les situations. Par exemple, si l'objet est partiellement visible à cause de sa taille ou de sa position par rapport au capteur, les événements ON et OFF seront trop éloignés dans l'espace et/ou dans le temps (c.à.d. qu'ils ne se produisent pas simultanément) rendant presque impossible une telle approche statique pour inférer la bonne direction du mouvement. Ce point est la raison pour laquelle nous faisons délibérément abstraction des polarités et considérons un unique type d'événement représentant les variations aussi bien positives que négatives. Le but est de forcer le réseau à extraire des caractéristiques spatio-temporelles en capitalisant sur les notions temporelles induites par le comportement des neurones et leurs manières de communiquer.
Notre étude se décompose en trois parties :

1. La première partie (SECTION 4.2) décrit précisément l'architecture du modèle avec les paramètres utilisés (SECTION 4.2.1), les détails des données d'entrées (SECTION 4.2.2) et les métriques permettant d'évaluer les performances du modèle (SECTION 4.2.3).
2. La deuxième partie (SECTION 4.3) étudie la capacité du modèle à analyser le mouvement après un apprentissage non supervisé. Nous cherchons à répondre à plusieurs questions, principalement :
   — Est-il possible de résoudre cette tâche dans le contexte exposé plus haut (STDP, analyse dynamique du mouvement) sur un ensemble de formes différentes ? (SECTION 4.3.4).
   — La méthode d'entraînement a-t-elle une influence sur les performances de la STDP (durée d'activation de la STDP, ordre de présentation des différentes classes) ? (SECTION 4.3.7)
   — Quelle est la taille adéquate du réseau pour résoudre ce problème ? (SECTION 4.3.7)

   Nous avons émis une liste d'hypothèses à confronter afin de répondre aux questions posées (SECTION 4.3.1). Au travers de ces hypothèses, plusieurs facteurs sont étudiés via un protocole expérimental comprenant l'évaluation de différentes métriques. La performance du modèle pour distinguer les classes de mouvement est mesurée à partir de l'activité des neurones en sortie. Nous avons conçu une fonction de score qui se base sur un encodage spécifique de cette activité et qui permet d'évaluer la capacité du modèle à fournir des réponses distinguables pour chaque mouvement présenté (SECTION 4.2.3). L'évolution des poids synaptiques lors des différentes phases d'apprentissage permet d'étudier la stabilité du modèle. Le calcul de l'écart-type global des poids synaptiques est un indicateur qui permet d'évaluer cet aspect. Lorsque la distribution des valeurs de poids synaptiques montre un écart-type élevé après l'entraînement, cela suggère que la règle d'apprentissage a conduit le modèle à converger vers un état spécifique souhaité où les neurones se sont spécialisés. L'écart-type des poids évolue progressivement en augmentant et converge vers un plateau au cours du temps, signifiant que le réseau à atteint un état spécifique et qu'il n'évolue plus ou très peu. En revanche, lorsque des distributions présentent un écart-type

faible, cela suggère que les synapses ont des valeurs proches, indiquant qu'il n'y a pas (ou pas suffisamment) de neurones qui se sont spécialisés sur des caractéristiques spécifiques. En observant l'évolution de l'écart-type au cours du temps, il apparaît que les valeurs oscillent dans le temps, révélant que le réseau alterne entre différents états et n'arrive pas à converger (SECTION 4.3.3). Les résultats obtenus mettent en évidence les capacités du modèle à reconnaître les diverses classes de mouvement présentées (SECTION 4.3.4), même après un apprentissage court (SECTION 4.3.7) et avec peu de neurones (SECTION 4.3.6). La difficulté principale soulevée dans cette partie porte sur le réglage des paramètres du modèle (seuil d'activation du neurone, fuite du neurone, etc.).

3. En se basant sur les conclusions tirées dans les sections précédentes, la dernière partie de ce chapitre se concentre sur le problème de réglage des paramètres en proposant une exploration des différentes valeurs possibles (SECTION 4.4). Cette exploration permet de faire un lien entre les données d'entrée présentées et certains paramètres du modèle, afin d'étudier l'impact du seuil d'activation du neurone (SECTION 4.4.2), de la fuite du potentiel du neurone (SECTION 4.4.3) et du nombre de pixels d'entrée activés par un motif (SECTION 4.4.4). Enfin, les différents facteurs sont explorés par paires pour identifier leurs influences combinées. La principale conclusion révèle que le seuil d'activation a un effet significatif sur la capacité du modèle à généraliser la reconnaissance du mouvement pour différentes entrées. Des solutions sont apportées dans la discussion, parmi lesquelles figure l'ajout d'un mécanisme de seuil adaptatif permettant au modèle de pouvoir traiter une plus grande variété de données d'entrée.

4.2 Choix des méthodes

4.2.1 Modèle SNN

L'architecture du modèle utilisé est détaillée dans la SECTION 3.2. Les paramètres liés (voir SECTION 3.2.5) aux neurones du réseau sont configurés avec les valeurs indiquées dans le TABLEAU 4.1. Ces valeurs ont été choisies de manière empirique en s'inspirant de certaines références (Bichler et al. 2012). Les valeurs des paramètres décrivant l'architecture globale sont spécifiées dans le TABLEAU 4.3.

Pour chaque synapse, les bornes des poids (notés w_{min} et w_{max}) sont initialisées aléatoirement, puis les valeurs des poids (w), les coefficients de potentialisation (w_+) et de dépression (w_-) sont déterminés de manière à appartenir à l'intervalle $[w_{min}\,;w_{max}]$. Les valeurs de tous ces paramètres, propres à chaque synapse, sont initialisées aléatoirement selon une loi Normale en début de simulation. Les valeurs correspondantes sont répertoriées dans le TABLEAU 4.2.

Paramètre	**Valeur**	**Description**
V_{thres}	20 mV	Seuil d'activation des neurones.
T_{refrac}	10 ms	Durée de la période réfractaire des neurones.
$T_{inhibit}$	1.5 ms	Durée de la période d'inhibition.
τ_{leak}	5	Taux de fuite des neurones.

TABLEAU 4.1 – Réglage des paramètres liés aux neurones du réseau décrit dans la SECTION 3.2.2.

Paramètre	**Moyenne**	**Écart-type**	**Description**
w_{min}	0.02 mV	0.002 mV	Poids minimal
w_{max}	6.00 mV	2.00 mV	Poids maximal
w	50%	30%	Poids synaptique
w_+	10%	5%	LTP
w_-	5%	2.5%	LTD

TABLEAU 4.2 – Valeurs utilisées pour l'initialisation des paramètres synaptiques selon une loi Normale. Notez que les poids sont exprimés en mV, étant donné que la tension est délivrée par les synapses. De même, lorsqu'il est fait mention de pourcentages, ceux-ci sont calculés par rapport aux valeurs de l'intervalle $[w_{min}\,;w_{max}]$.

Pour rappel, les paramètres définissant l'architecture globale du réseau sont décrits dans le TABLEAU 5.2, tandis que les réglages sont présentés dans le TABLEAU 4.3 :

Paramètre	**Valeur**	**Description**
N_{output}	10	Nombre de neurones de sortie.
N_{inhib}	$N_{output} - 1$	Nombre de voisins à inhiber par neurone.
T_{LTP}	2 ms	Durée de la période à laquelle l'apprentissage (LTP) fonctionne.

TABLEAU 4.3 – Paramètres liés à l'architecture du réseau.

4.2.2 Données événementielles synthétiques

Comme nous avons pu le voir dans la SECTION 2.3, les données événementielles et les réseaux impulsionnels partagent des propriétés qui permettent de les associés facilement. Dans le format standard AER (voir SECTION 2.3), les variations de luminosité positives et négatives des pixels sont encodées en évènements ON et OFF. Dans notre expérimentation, nous avons opté pour une représentation simplifiée ne considérant plus qu'un unique type d'événement. Comme expliqué précédemment, l'objectif est de contraindre le modèle à apprendre à classifier le mouvement de manière dynamique. Pour mener à bien cette validation expérimentale, nous avons développé un générateur de données synthétiques capable de produire des séquences d'évènements. Ces séquences sont de courte durée et représentent différents mouvements de formes se déplaçant dans champs récepteur réduit (fenêtre d'entrée). Le générateur permet de créer ces séquences d'évènements avec divers réglages possibles :

1. la résolution de la fenêtre ;
2. la forme du motif ;
3. la trajectoire du motif est représentée par une séquence de directions élémentaires, chacune étant décrite par :
   - la direction, celle-ci pouvant prendre l'une des quatre valeurs suivantes : NORD, SUD, EST, OUEST ;
   - la longueur de la séquence ;
   - la vitesse de déplacement du motif.
4. la géométrie du plan sur lequel le motif se déplace (plate ou torique), dans le cas d'une géométrie torique, le motif réapparaît immédiatement sur le bord opposé en sortant du champ récepteur.

Les réglages choisis sont les suivants : la fenêtre à une résolution 5×5 (1), avec différentes formes de motifs en mouvement (2) (la FIGURE 4.1 illustre certaines de ces formes), avec une vitesse de déplacement fixée à 480 pixel/sec (3) et une géométrie torique appliquée au plan sur lequel les motifs évoluent (4).

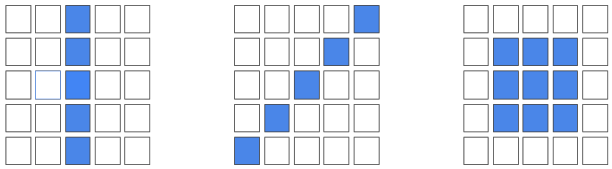


FIGURE 4.1 – Illustration de 3 formes différentes utilisées dans la validation expérimentale.

4.2.3 Évaluation du score de performance

Afin de pouvoir mesurer la performance du modèle dans la reconnaissance des directions d'un mouvement, il est nécessaire d'interpréter sa sortie.

Pour ce faire, nous avons alterné entre des phases d'apprentissages, correspondant aux périodes où la STDP est activée et des phases de test dans lesquelles la plasticité synaptique du réseau est désactivée (voir FIGURE 4.2). La performance du système est évaluée à la fin de chaque période d'apprentissage, lors d'une phase de test.

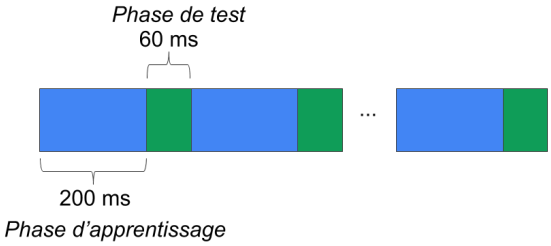


FIGURE 4.2 – Les phases d'apprentissage ont une durée de 200 ms, elles correspondent à la présentation d'une classe de mouvement (= direction). Les phases de test durent 60 ms et correspondent à la présentation de la séquence NORD-SUD-EST-OUEST (NSEO).

Pendant cette phase de test, l'activité des neurones de sortie est convertie sous la forme de code binaire pour chaque classe de mouvement. L'objectif est de déterminer si l'activité des neurones de sortie permet de distinguer une classe présentée. Le cas idéal serait qu'un neurone se spécialise pour une classe de mouvement particulière, sans que les autres neurones ne se déclenchent. Si d'autres neurones sont également activés, une méthode alternative consiste à identifier un groupe de neurones qui s'activent différemment pour chaque classe de mouvement. La FIGURE 4.3 illustre une manière d'encoder l'activité des neurones de sortie. Chaque code binaire associé à une classe indique l'activation d'un neurone de sortie. Le score correspond au nombre de codes binaires différents.

Lors d'une phase de test, N_c codes sont calculés, N_c étant le nombre de classes ($N_c = 4$). Pour chaque classe, un code B_c est attribué (indexé par c représentant la classe). Le code est composé de bits b_o^c qui sont associés à chaque neurone de sortie (indexé par o représentant un neurone de sortie), N_o étant le nombre de neurone de sortie. Un bit indique si un neurone a été activé lors de la présentation d'une séquence S_c. Une séquence S_c est décrite par les moments d'activation des neurones de sorties pour une classe de mouvement particulière $(t_0^c, t_1^c, ..., t_{N_o^c}^c)$. Cette séquence est représentée par un n-uplet de taille N_o^c qui correspond au nombre d'activations pour un neurone de sortie o concernant

FIGURE 4.3 – Exemple de code obtenu lors d'une phases de test. Le réseau possède une couche de sortie composée de 4 neurones de sortie. Dans cet exemple, les 4 codes sont distincts, ce qui signifie que le réseau fournit une réponse différente selon la classe.

une classe de mouvement c. L'ÉQUATION (4.1) définit la transformation d'une séquence d'activation pour une classe de mouvement S_c en un bit décrivant s'il y a eu une activité d'un neurone de sortie :

$$\begin{aligned} b_o^c : \mathbb{R}_+^{N_o^c} &\to [0,1] \\ \emptyset &\mapsto 0 \\ (t_0^c, t_1^c, ..., t_{N_o^c}^c) &\mapsto 1 \end{aligned} \tag{4.1}$$

L'ÉQUATION (4.2) définit la construction de chaque code binaire pour l'ensemble des classes en appliquant la transformation à tous les neurones de sortie de la N_o :

$$\begin{aligned} &B_c : [0,1]_+^{N_o} \\ &B_c = (b_0^c, b_1^c, ..., b_{N_o}^c) \end{aligned} \tag{4.2}$$

Avec les différents codes de sortie, on peut déduire un score $perf$, en construisant l'ensemble des codes distincts B. On compte le nombre d'éléments de cet ensemble Card(B) et on divise cette valeur par le nombre de classes présentées N_c. La valeur du score est comprise entre 0 et 1. L'ÉQUATION (4.3) permet de formuler le calcul du score :

$$\begin{aligned} \mathbf{0} &= (0, 0, \ldots, 0) \\ B &= \{B_1\} \cap \{B_2\} \cap ... \cap \{B_{N_c}\} \setminus \{\mathbf{0}\} \\ perf &= \text{Card}(B) \times \frac{1}{N_c} \end{aligned} \tag{4.3}$$

où $\mathbf{0}$ représente le n-uplet de taille N_o contenant uniquement des zéros répété N_o fois.

Il convient de souligner une précision importante : le code $\mathbf{0}$ n'est pas pris en compte pour déterminer le score de performance du réseau, car il correspond à une situation où aucun neurone du réseau n'a été activé pendant la présentation d'une classe.

4.2.4 Évaluation du score de stabilité

L'objectif de l'étape d'entraînement non supervisé est de parvenir à un état où le réseau est capable de générer des sorties distinctes pour chaque classe. De ce fait, certains neurones de la couche de sortie doivent se spécialiser dans la détection d'une direction particulière. Pour un neurone de sortie, une situation de spécialisation optimale est atteinte lorsque certains poids synaptiques se stabilisent autour de valeurs proches de 0 ou 1, indiquant que ce neurone est soit sensible, soit insensible à des neurones d'entrées spécifiques. Au cours de l'apprentissage, les poids synaptiques subissent des modifications. Pour évaluer dans quelle mesure le réseau converge vers un état spécifique, comme celui permettant la reconnaissance des classes de mouvement, il est possible de calculer l'écart-type sur la distribution des poids synaptiques après une phase d'apprentissage. Si la valeur de l'écart-type est élevée, cela signifie que le réseau est dans un état spécifique. Pendant la phase d'apprentissage, le calcul de cet indicateur permet de déterminer si le réseau a convergé vers un état spécifique avec une distribution qui reste stable au cours du temps ou si au contraire, il n'a pas convergé. Dans ce dernier cas, la courbe d'évolution des écarts-types oscille, ce qui se caractérise par des transitions répétées entre différentes configurations, sans que le réseau parvienne à en maintenir une de manière stable. La score *stab* permet de mesurer la stabilité du réseau en évaluant le degré de convergence du réseau à partir des poids du réseau W. L'ÉQUATION (4.4) décrit le calcul du score :

$$stab = \text{stdev}(W) \tag{4.4}$$

4.3 Reconnaissance des directions d'un mouvement

Cette section vise à évaluer la capacité du modèle à reconnaître les différentes directions de mouvement d'une forme, en utilisant la métrique présentée dans la SECTION 4.2.3. Différents facteurs seront analysés lors des validations expérimentales, tels que la capacité du modèle à s'adapter à des formes variables en entrée, le nombre de neurones de sorties nécessaires permettant de classifier les différents mouvements, ainsi que les conditions d'apprentissage favorisant la convergence du réseau vers un état souhaité. Le but de cette étude est de caractériser un réseau qui doit être en mesure d'effectuer une classification précise après un apprentissage relativement court, tout en étant composé d'un nombre limité d'éléments, et en étant sensible aux caractéristiques spatio-temporelles qui décrivent le mouvement d'une forme.

4.3.1 Formulation d'hypothèses

Afin d'évaluer et explorer divers aspects de notre modèle, nous avons formulé plusieurs hypothèses.

— La première hypothèse notée (**H0**) stipule que lorsqu'il y a apprentissage, l'écart-type des poids synaptiques augmente et atteint un seuil spécifique, indiquant une distribution où les poids ont des valeurs proches de 0 ou de 1. Il convient de préciser que l'état initial des poids avant l'apprentissage suit une distribution gaussienne centrée sur une valeur moyenne avec une faible variance.

— La deuxième hypothèse notée (**H1**) stipule que pour tous les motifs, le réseau est capable d'apprendre les directions. Cependant, nous allons apporter une nuance en la subdivisant en deux sous-hypothèses : L'hypothèse (**H1-A**) reprend l'énoncé de base de (**H1**) tandis que l'hypothèse (**H1-B**) ajoute la condition que le réseau peut converger vers l'état souhaité pour tous les motifs *avec un unique jeu de paramètres*.

— La troisième hypothèse notée (**H2**) stipule que l'ordre de présentation des directions a une influence sur l'apprentissage.

— La quatrième hypothèse notée (**H3**) stipule qu'en augmentant le nombre de neurones de sortie, on améliore l'efficacité du réseau dans l'extraction de caractéristiques pertinentes.

— La cinquième hypothèse notée (**H4**) stipule que la longueur des phases d'apprentissage influence la stabilité du réseau.

— La sixième hypothèse notée (**H5**) stipule que la longueur des phases de test n'influence pas la mesure au delà d'un temps minimal.

Nous allons donc confronter ces hypothèses avec les différentes indicateurs détaillées dans les SECTIONS 4.2.3 et 4.2.4.

4.3.2 Protocole expérimental

Pour cette première étude, nous avons sélectionné 8 types de formes géométriques distinctes : deux lignes verticales de 5 et 3 pixels, deux lignes diagonales 5 et 3 pixels, deux motifs carrés de tailles 3×3 et 2×2 pixels, un point (un unique pixel) ainsi qu'un angle de 7 pixels. Ces formes sont illustrées par la FIGURE 4.4.

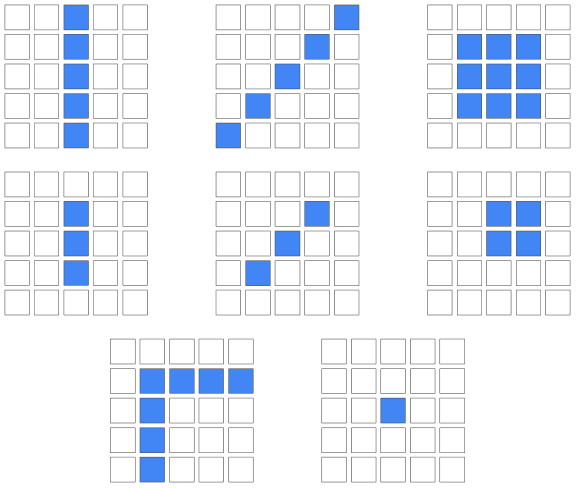


FIGURE 4.4 – Illustration des 8 formes différentes utilisées dans la validation expérimentale.

Nous examinons également l'impact possible qu'entraîne le choix d'une stratégie de présentation des directions dans une séquence de stimulation. En effet, des séquences ordonnées différemment sont susceptibles d'influencer la façon dont les poids synaptiques sont mis à jour. Pour un motif donné, le stimulus est présenté au réseau selon les 3 stratégies suivantes :

1. séquence ordonnée, par exemple NNNSSSOOOEEE (où N=NORD, S=SUD, E=EST et O=OUEST) ;
2. séquence de quadruplés où les classes apparaissent dans un ordre aléatoire, par exemple NSOE-ESON-ONES-... ceci garantissant une distribution uniforme des classes ;
3. séquence aléatoire, par exemple NSSESON...

Ainsi, nous avons stimulé le réseau en utilisant ces 3 stratégies. Pour chaque exécution, une séquence de 40 mouvements est présentée au réseau, la durée de chaque entrée de la séquence étant de 200 ms, c'est-à-dire 8 s au total.

Enfin, nous avons testé différentes tailles pour la couche de sortie : $2, 3, 4, 5, 6$. Nous soulignons que :

— 4 neurones de sortie correspond à la taille naturelle pour quatre classes distinguables ;

— 2 neurones de sortie la taille naturelle pour deux classes (la ligne diagonale) ;

— 6 neurones de sortie, situation qui augmente les chances d'obtenir un neurone de sortie spécialisé pour chaque classe.

En résumé, le protocole comporte 3 variables : le *motif*, la *stratégie* de présentation et le *nombre* de neurones de sortie (voir la FIGURE 4.5). Pour chacune de ces configurations, nous avons effectué 10 exécutions pour obtenir des indicateurs statistiques fiables, totalisant ainsi 8 formes×3 stratégies de présentation×5 dimensions pour la couche de sortie× 10 runs = 1050 runs. Nous avons observé l'activité des neurones de sortie ainsi que l'évolution des poids synaptiques au cours d'un apprentissage. La qualité des caractéristiques extraites par le réseau est évaluée à l'aide du score $perf$, qui se base sur l'activité des neurones de sortie pendant une phase de test. La stabilité du réseau et son niveau de convergence vers un état spécifique lors d'une phase d'apprentissage est mesuré via la métrique *stab* qui se fonde sur l'évolution des poids synaptiques du réseau.

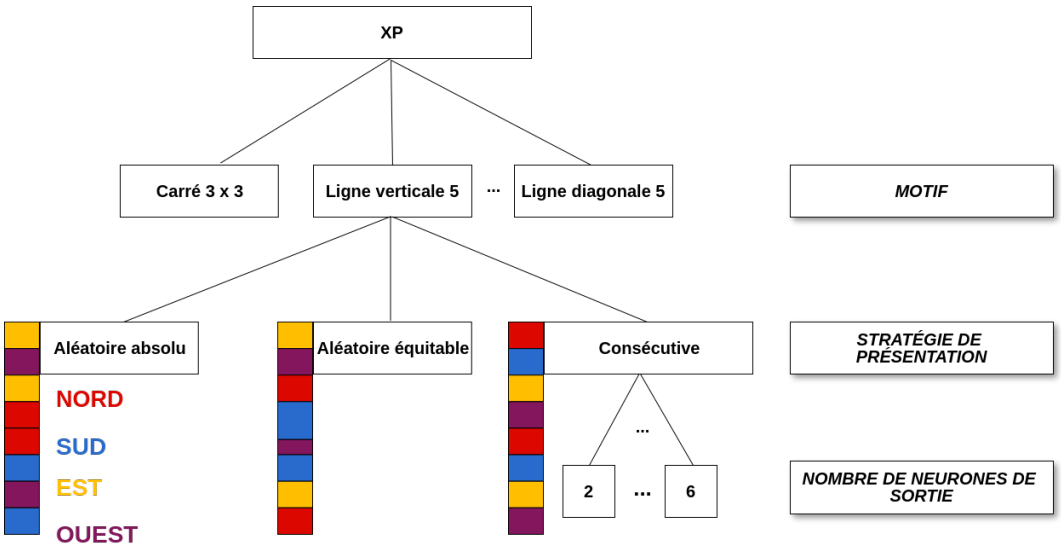


FIGURE 4.5 – Diagramme représentant les différentes facteurs variables du protocole.

4.3.3 Spécialisation du réseau

Nous souhaitons vérifier la première hypothèse notée **H0** en observant l'évolution des poids synaptiques pour chaque simulation, auxquels on donne initialement des valeurs aléatoires.

La FIGURE 4.6 montre une illustration de la spécialisation des poids synaptiques pour un réseau comprennent 6 neurones de sortie. Chaque cellule de ce panneau de 6×25 représente le poids synaptique normalisé de 0 à 1. Les colonnes représentent les 25 neurones d'entrée et les lignes correspondent aux 6 neurones de sortie. Le panneau de gauche montre les poids après une initialisation aléatoire, avant l'entraînement, et le panneau de droite montre les mêmes poids après 8 s de stimulation lorsque celui converge vers la reconnaissance d'un mouvement.

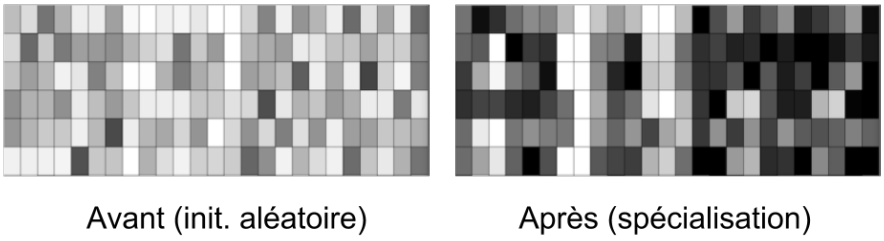


FIGURE 4.6 – Évolution des poids synaptiques : les cellules sombres et claires indiquent respectivement des valeurs faibles et fortes des poids synaptiques connectés au neurones de sortie – GAUCHE : avant l'entraînement (initialisation aléatoire). DROITE : après l'entraînement.

Afin de suivre l'évolution du système, nous avons tracé la valeur du score *stab*, basée sur l'écart type des poids synaptiques. Cela nous permet de visualiser l'évolution de l'état du réseau au cours d'une phase d'apprentissage (FIGURE 4.7). Ces résultats nous permettent de vérifier l'hypothèse **H0** qui stipule que les poids du réseau devraient, au cours de la simulation, s'écarter de leurs valeurs initiales en se spécialisant : soit en tendant vers 0 – ce qui indique que le neurone est devenu insensible au stimulus, soit vers 1 – ce qui indique que le neurone est devenu très sensible au stimulus. La figure montre une forte augmentation au début de la simulation, indiquant la spécialisation des neurones, puis un plateau indiquant un état stable du réseau.

Dans la FIGURE 4.7, nous observons qu'un certain nombre de poids ont changé vers une valeur inférieure ou supérieure entre l'initialisation aléatoire et après une phase d'apprentissage, ce qui indique bien une spécialisation des connexions synaptiques.

Comme indiqué précédemment, en raison des initialisations aléatoires, nous avons effectué un certain nombre de simulations pour chaque situation afin d'évaluer la stabilité des résultats. Nous avons entraîné avec succès plusieurs configurations de réseaux qui se sont spécialisés et qui ont atteint un état stable. Avec ces résultats, l'hypothèse notée **H0** est confirmée. Cependant, dans de rares cas, il arrive que le modèle n'atteigne pas un état stable, ce qui se manifeste par une évolution du score *stab* qui présente un profil tel que celui illustré dans la FIGURE 4.8, avec des poids qui oscillent sans se stabiliser.

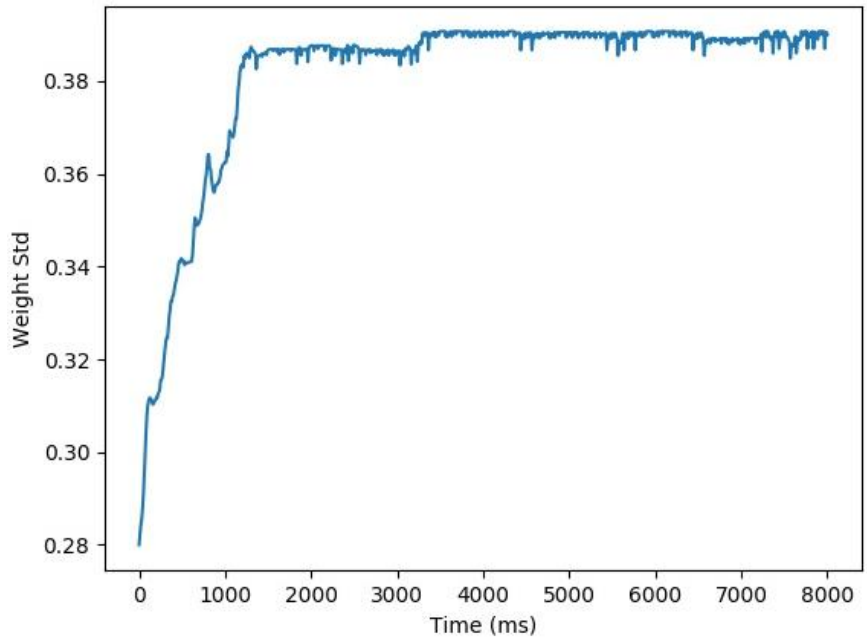


FIGURE 4.7 – Évolution des poids synaptiques lors d'une simulation lorsque le réseau se spécialise.

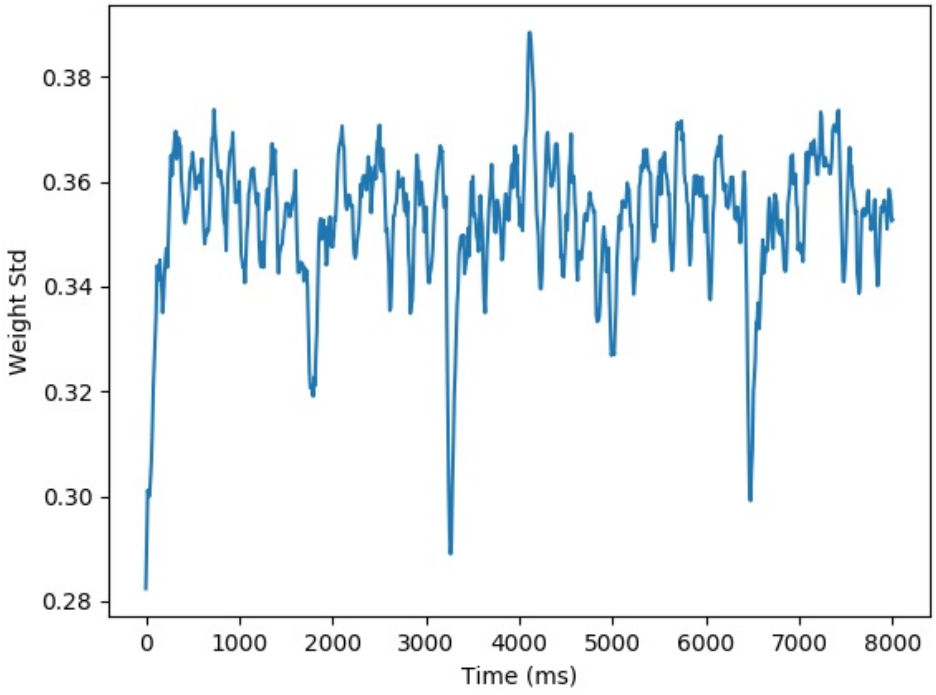


FIGURE 4.8 – Évolution des poids synaptiques lors d'une simulation dans un cas de non-convergence.

Une piste plausible pour expliquer cette situation réside dans la quantité de neurones d'entrée actifs. En effet, on retrouve ce type de résultats lorsque le nombre de pixels qui composent la forme présentée est trop grand ou trop petit par rapport au paramétrage du réseau de neurone. Dans de tels cas, la tension d'entrée globale attendue est soit trop élevée ou soit trop faible. Il est donc probable que le réseau ne puisse atteindre un état stable que si le volume d'impulsions entrantes se situe dans un certain intervalle défini par les paramètres tels que le seuil d'activation du neurone ou sa fuite. Ces paramètres permettent de définir l'ensemble des motifs spatio-temporels auxquels le réseau est sensible. Il est donc nécessaire d'avoir une bonne connaissance du problème, en particulier sur la nature des données d'entrée, ainsi qu'une bonne compréhension de l'influence des autres mécanismes du réseau (topologie, inhibition latérale, STDP, etc.). Par conséquent, une phase d'exploration de l'espace des paramètres est donc généralement nécessaire.

4.3.4 Impact des différentes formes

L'hypothèse (**H1-A**) postule que pour tous les motifs proposés, le réseau est capable de converger lors de son apprentissage. Une autre hypothèse (**H1-B**), qui est une variante de la première, ajoute que cette convergence peut être atteinte pour tous les motifs *avec un unique jeu de paramètres*.

L'analyse des scores générés lors des simulations (voir la SECTION 4.2.3 pour plus de détails) en faisant varier la variable $MOTIF$ du protocole expérimental révèle un effet significatif sur le score, comme le montre la FIGURE 4.9.

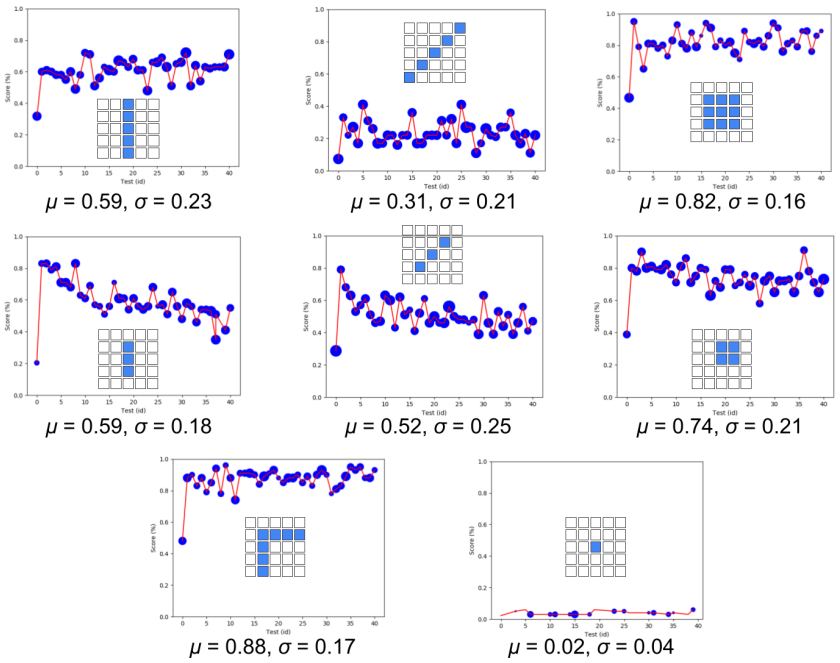
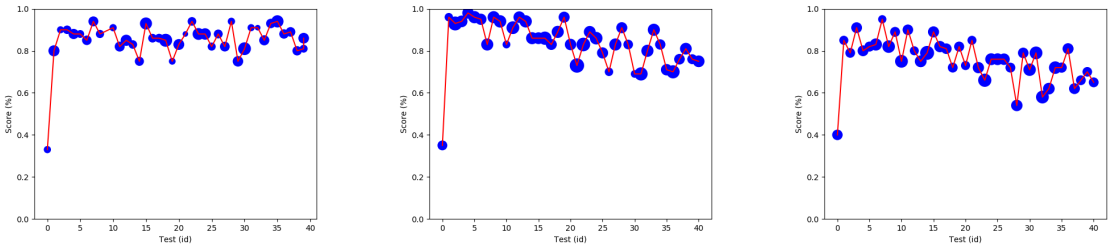


FIGURE 4.9 – Courbes des scores en fonction des formes, les mesures sont effectuées à chaque phase de test (40 au total), la moyenne sur tous les runs (hauteur des marqueurs bleus) ainsi que l'écart-type (diamètre des marqueurs bleus) sont affichés.

Ces résultats suggèrent l'existence de 3 types de situations distinctes :

1. Des courbes qui convergent vers un score stable (> 0.8) pour les 3 formes suivantes : le carré 2×2, le carré 3×3 et l'angle ;
2. Des courbes qui sont initialement croissantes après la première phase d'apprentissage jusqu'à atteindre une certaine valeur puis décroissent jusqu'à la fin, pour les 2 formes suivantes : ligne diagonale 3 pixels et verticale 3 pixels.
3. Des courbes qui convergent vers un score stable faible (< 0.6) pour les 3 formes suivantes : ligne diagonale 5 pixels, ligne verticale 5 pixels et le point.

Pour le cas (2), un changement de valeur du paramètre du seuil d'activation des neurones $V_{thres} = 15$ mV, permet d'obtenir de meilleurs scores, la FIGURE 4.10 illustre ce résultat. Pour le cas (3), en observant la courbe correspondant au motif représentant le point, on constate que celle-ci fluctue légèrement autour de 0. En s'intéressant à l'activité des neurones de sortie, on remarque qu'ils n'émettent pratiquement aucune impulsion. Étant donné qu'il n'y a qu'un pixel en déplacement, cela n'entraîne pas une stimulation élevée. En diminuant le seuil d'activation V_{thres}, on augmente le score en moyenne de $+0,5$. Si on augmente le taux de fuite τ_{leak} à 20, l'effet sur le score est encore plus significatif puisqu'il atteint cette fois-ci une valeur moyenne de $(+0.8)$.



$\mu = 0.86(+0.34)$, $\sigma = 0.17$ $\mu = 0.84(+0.25)$, $\sigma = 0.15$ $\mu = 0.82(+0.8)$, $\sigma = 0.16$

FIGURE 4.10 – Score en fonction du nouveau seuil $V_{thres} = 15$ mV. GAUCHE : Diagonale de taille 3 - CENTRE : Ligne verticale de taille 3 - DROITE : Point.

D'après les résultats obtenus, le réseau n'arrive pas à apprendre le motif correspondant à la diagonale de 5 pixels. En poussant les investigations, nous avons identifié que la principale cause expliquant ces résultats est le problème d'ouverture. Ce phénomène, souvent rencontré en vision dans la perception du mouvement, entraîne une mauvaise interprétation de la fonction de score sur les performances du réseau. Lorsque certaines conditions sont réunies, des mouvements différents peuvent être perçus comme identique. Le problème d'ouverture est illustré pour la ligne diagonale de 5 pixels dans la FIGURE 4.11. On constate que les déplacements dans les directions NORD/OUEST sont confondues. La même remarque peut être fait pour les directions SUD/EST. Il est impossible de déterminer la bonne direction tant que les extrémités de la ligne ne sont pas visibles dans la fenêtre.

Pour la ligne verticale 5 pixels, nous avons détecté 2 classes qui ne sont pas discernables, il s'agit des classes NORD/SUD. Cette ambiguïté est provoquée par le déplacement d'une ligne verticale dans une direction verticale en sachant que la hauteur de la fenêtre est égale à la longueur de la ligne et que le plan sur lequel se déplace cette forme est torique. Les résultats montrent que le réseau réagit de manière identique pour ces deux classes. En effet, lors du déplacement de la ligne dans les directions NORD/SUD, on observe les

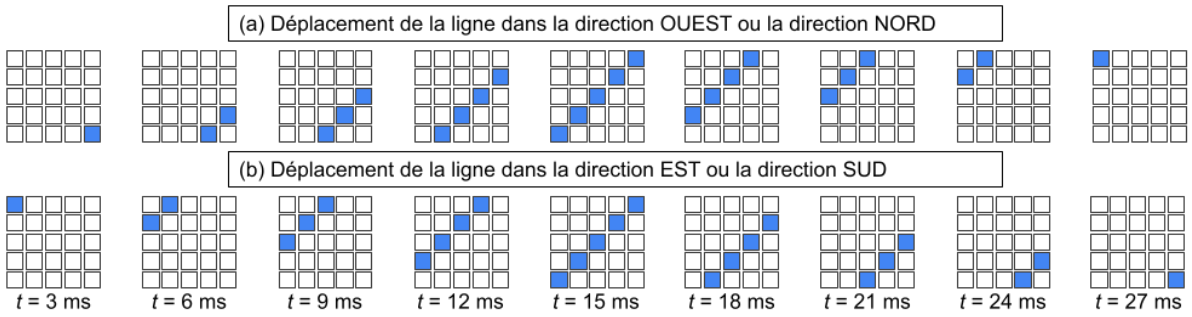
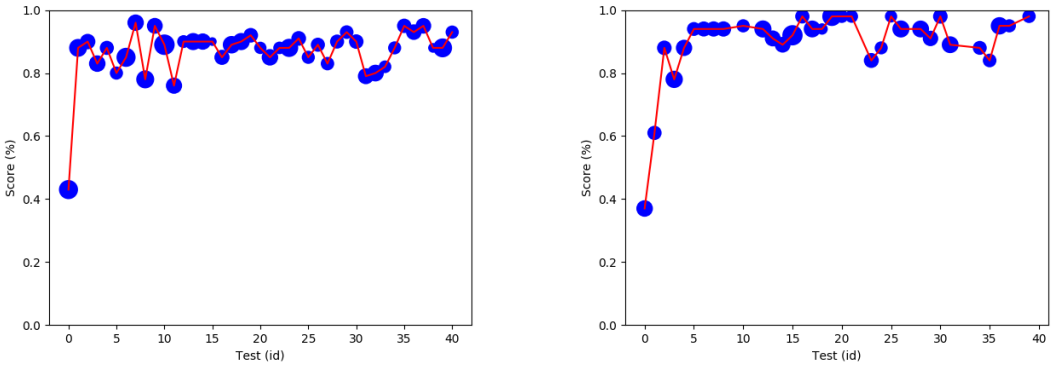


FIGURE 4.11 – Déplacement de la ligne 5 pixels illustrant le problème d'ouverture.

5 mêmes pixels actifs (colonne du milieu, aucun mouvement discernable) pendant toute la séquence.

Si nous réduisons, pour chacun de ces cas, le nombre de classes à déterminer pour le calcul du score de performance en fusionnant les directions NORD/OUEST et SUD/EST pour la diagonale 5 pixels et NORD/SUD pour la ligne verticale 5 pixels, nous obtenons des résultats plus représentatifs des performances du modèle (voir la FIGURE 4.12).



$\mu = 0.89$ (+0.37), $\sigma = 0.17$ $\mu = 0.91$ (+0.62), $\sigma = 0.21$

FIGURE 4.12 – Score en fonction du nouveau nombre de classes à déterminer. GAUCHE : 2 classes à déterminer - DROITE : 3 classes à déterminer.

Après avoir pris en compte les observations mentionnées précédemment, nous pouvons valider l'hypothèse (**H1-A**) qui stipule que le réseau est capable d'apprendre à reconnaître les différentes classes de mouvement pour tous les motifs. Toutefois, à ce stade, nous ne pouvons ni confirmer l'hypothèse (**H1-B**) ni la rejeter et ce malgré nos tentatives pour modifier les paramètres du réseau afin d'améliorer le score de performance du SNN sur les motifs ayant un faible taux de reconnaissance. Cette incertitude est due en grande partie à l'immensité de l'espace des paramètres du modèle, impliquant qu'il est possible qu'il existe un réglage particulier pouvant être compatible avec tous les motifs présentés, mais que ce réglage peut être difficile à trouver.

Afin d'avoir un indicateur visuel permettant de savoir si un réseau s'est stabilisé ou non, nous avons affiché dans la FIGURE 4.13 l'activité de sortie du réseau. Ce graphique nous permet de savoir quels sont les neurones qui réagissent à une classe donnée, dans notre cas : à l'une des 4 directions. Dans ce graphe : l'axe des ordonnées correspond aux indices des neurones de sortie et l'axe des abscisses correspond aux temps de simulation en ms. Les marqueurs noirs indiquent les impulsions du neurone de sortie correspondant. Les patches de couleur affichés en arrière-plan sont associés à la classe d'entrée. Lorsque le réseau est entraîné, nous ciblons une situation où certains neurones (ou certaines combinaisons de neurones) réagissent à une classe particulière et où la classe peut-être déterminée de manière non ambiguë selon le sous-ensemble de neurones de sortie qui réagissent.

À titre d'exemple, dans la FIGURE 4.13, chaque ligne correspond à une configuration composée d'un motif, d'un ordre de présentation associé à un réseau doté d'un nombre de neurones de sortie compris entre 2 et 6. Pour chacune de ces configurations, on affiche le graphe d'activité de sortie du réseau (à gauche) ainsi que l'évolution de l'écart-type de ses poids synaptiques pendant la simulation (à droite). La durée de la simulation est toujours de 8 secondes pour toutes les configurations.

La première configuration implique la diagonale de 5 pixels, un ordre de présentation aléatoire, ainsi qu'un réseau de 6 neurones de sortie. Après 1000 ms, nous voyons que le neurone d'index 3 se spécialise pour les classes SUD et EST (bleu et jaune) et le neurone d'index 2 devient sélectif pour les classes NORD et OUEST (rouge et violet). La deuxième configuration implique un motif diagonal de 5 pixels, une stratégie aléatoire équitable (quadruplet), ainsi qu'un réseau composé de 2 neurones de sortie – le plot montre une spécialisation des deux neurones. Le neurone 1 est sensible aux directions SUD/EST et le neurone 2 aux directions NORD/OUEST. Nous sommes dans un cas où chaque neurone de sortie s'est spécialisé pour une direction (nous rappelons que la direction est double en raison du problème d'ouverture). Enfin, la troisième configuration implique la ligne verticale de 5 pixels, un ordre de séquence aléatoire, ainsi qu'un réseau composé de 3 neurones de sortie. La figure montre que pour cette configuration, 3 classes d'entrée OUEST, EST et NORD/SUD sont reconnues par des neurones de sortie avec les indices 3, 2 et 1 respectivement.

Pour chacune de ces configurations, nous avons constaté que le réseau convergeait, comme en témoignent les courbes qui décrivent l'évolution de l'écart-type des poids synaptiques. En somme, nous avons observé une forte augmentation de cette mesure au début de la simulation, suivie d'un plateau indiquant un état stable du réseau.

Les résultats obtenus montrent qu'après un court apprentissage, le réseau est généralement capable de reconnaître les directions présentées. Cette observation donne des indications préliminaires sur la validité de l'hypothèse (**H4**). Pour rappel, l'hypothèse (**H4**) stipule que la durée des phases d'entraînement influence l'apprentissage du réseau.

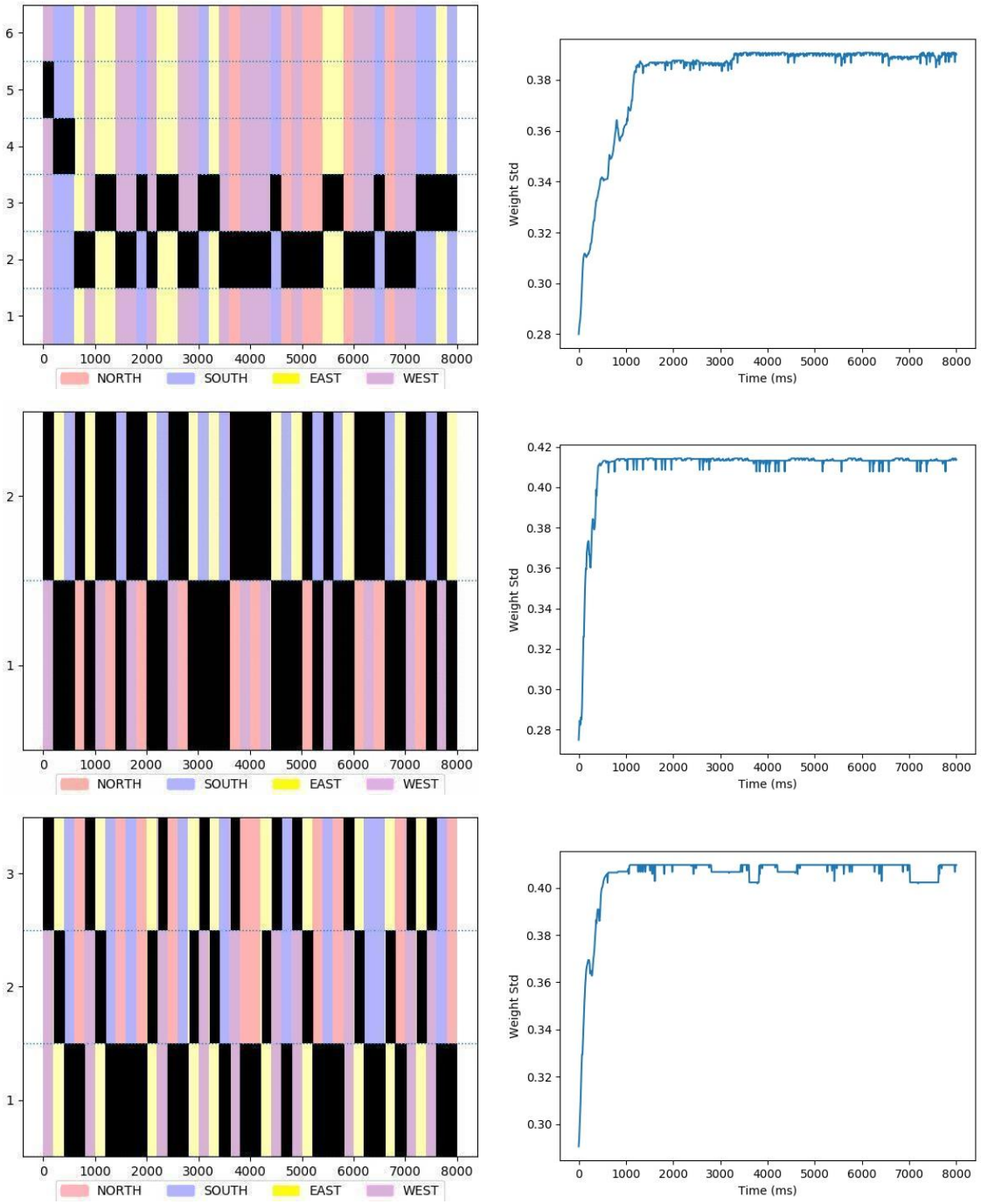


FIGURE 4.13 – **Colonne de gauche :** Graphique d'activité de sortie du réseau. **Colonne de droite :** Évolution de l'écart-type des poids synaptiques pendant la simulation. **En haut :** Ligne diagonale 5 pixels, ordre de présentation aléatoire, avec 6 neurones de sortie pour le réseau. **Au centre :** Ligne diagonale 5 pixels, ordre de présentation aléatoire équitable, avec 2 neurones de sortie pour le réseau. **En bas :** Ligne verticale 3 pixels, ordre de présentation aléatoire, avec 3 neurones de sortie pour le réseau.

4.3.5 Impact des stratégies de présentation des classes

L'un des facteurs variables du protocole expérimental est le choix de la stratégie de présentation des motifs ($PRES$), avec trois modes possibles : consécutive (CONS), aléatoire absolu (ABS) et aléatoire équitable (EQ) (voir SECTION 4.3.2). L'hypothèse principale (**H2**) stipule que l'ordre de présentation peut influencer le processus d'apprentissage du réseau et, en conséquence, ses performances. Pour valider ou invalider cette hypothèse, nous comparons les scores en fonction de chaque stratégie de présentation, afin d'évaluer s'il existe une différence significative entre les différentes présentations possibles.

Les résultats d'expérimentation sont illustrés par la FIGURE 4.14 et le TABLEAU 4.4. En faisant une ANOVA à un facteur variable, celle-ci ne montre aucun effet significatif de $PRES$ sur le score ($p = 0.9730$, $p > 0.05$). Des comparaisons individuelles montrent également qu'il n'y a aucune différence significative entre $CONS$ et EQ, entre $CONS$ et ABS et entre EQ et ABS qui ont toutes la *p-value* $p = 0.899$, $p > 0.05$. Donc le choix de la stratégie de présentation n'influence pas l'efficacité du réseau. L'hypothèse (**H2**) est donc rejetée.

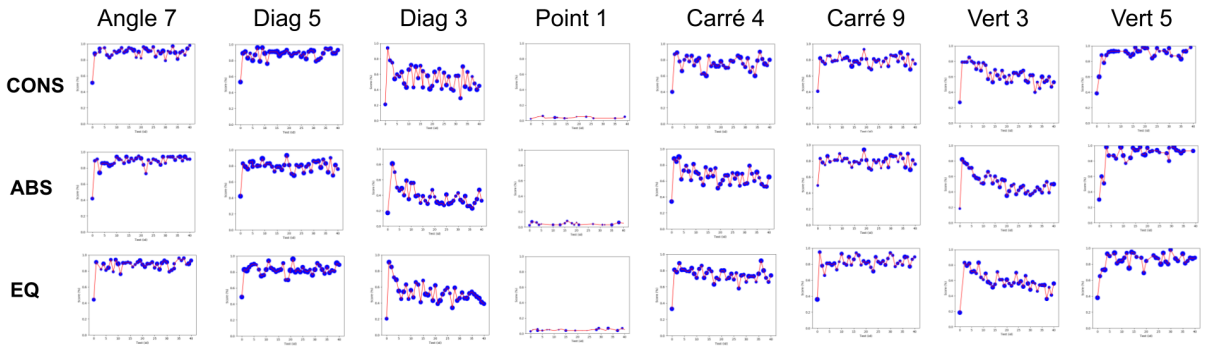


FIGURE 4.14 – Score calculé pour chaque motif (colonne) et pour chaque stratégie de présentation (ligne).

Stratégie	μ	σ	Angle 7	Diag 5	Diag 3	Point 1	Carré 4	Carré 9	Vert 3	Vert 5
CONS	0.69	0.30	0.89	0.90	0.55	0.02	0.76	0.83	0.63	0.92
ABS	0.64	0.29	0.88	0.89	0.47	0.02	0.66	0.83	0.51	0.91
EQ	0.65	0.29	0.88	0.88	0.52	0.02	0.74	0.82	0.51	0.87

TABLEAU 4.4 – Moyenne des courbes de score de la FIGURE 4.14.

4.3.6 Impact du nombre de neurones de sortie

Le nombre de neurones de sortie est la dernière variable $NEURONE$ du protocole expérimental, sa valeur est comprise dans l'intervalle $[2\,;6]$. Il est souvent admis intuitivement que les problèmes complexes nécessitent des réseaux de neurones ayant un nombre important de paramètres entraînables. Par conséquent, afin d'augmenter le nombre de paramètres à optimiser, il est courant d'augmenter le nombre de neurones dans le réseau.

Dans notre cas, l'hypothèse (**H3**) s'inspire de cette idée, en supposant que lorsqu'on augmente le nombre de neurones de sortie et donc indirectement le nombre de paramètres entraînables, on augmente son pouvoir expressif et son efficacité. Afin de confirmer ou rejeter l'hypothèse, nous comparons le score pour chaque motif en fonction des différents nombres de neurones de sortie.

Les résultats sont mis en évidence par la FIGURE 4.15 et le TABLEAU 4.5. En appliquant à nouveau une ANOVA à un facteur variable, celle-ci révèle qu'il y a un effet significatif de la variable $NEURONE$ sur le score ($p = 0.0018$, $p < 0.05$).

En examinant les données de la FIGURE 4.15, nous pouvons observer visuellement que le score est influencé par le nombre de neurones de sortie pour la plupart des motifs (sauf pour la ligne diagonale 5 pixels et le point, les raisons étant expliquées dans la SECTION 4.3.4). Une interprétation possible de ce résultat serait que la probabilité que l'un des neurones de sortie se spécialise augmente avec le nombre de neurones de sortie.

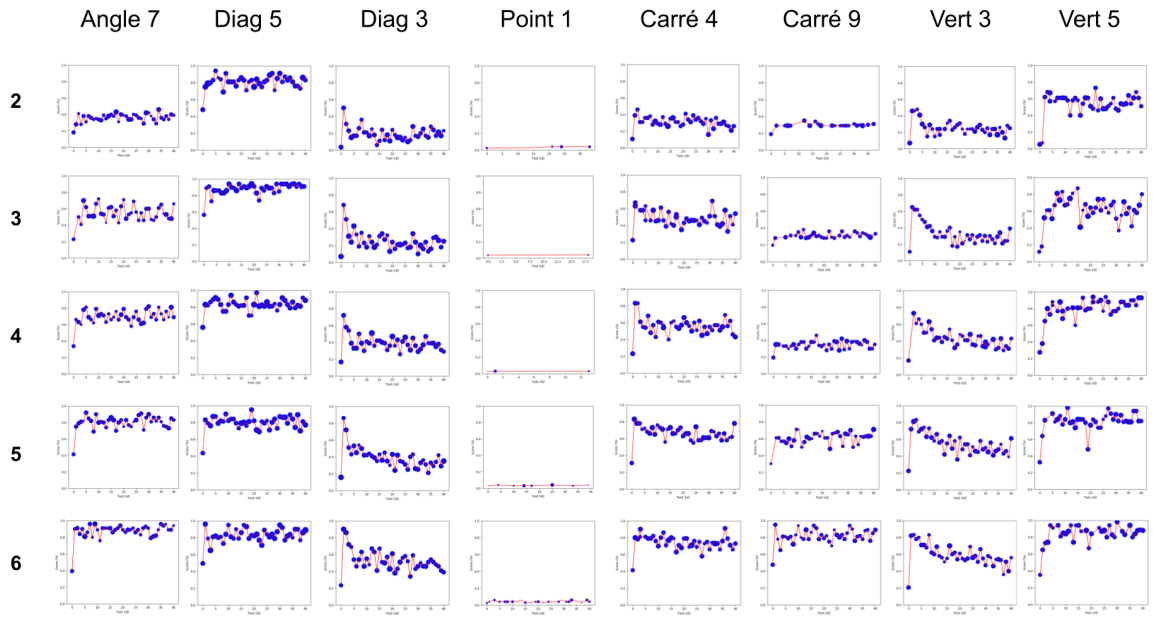
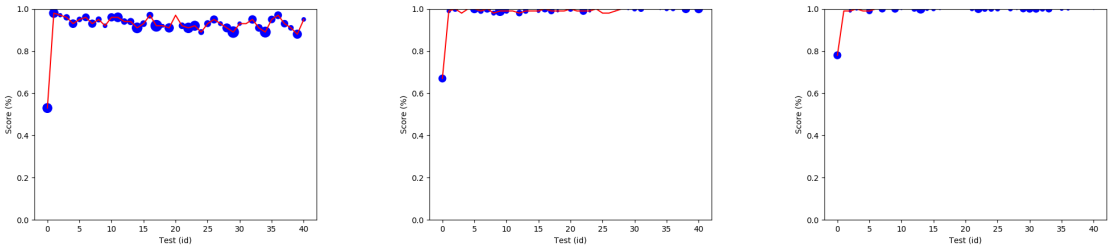


FIGURE 4.15 – Score calculé pour chaque motif (colonne) et pour chaque nombre de neurones de sortie composant le réseau (ligne).

Stratégie	μ	σ	Angle 7	Diag 5	Diag 3	Point 1	Carré 4	Carré 9	Vert 3	Vert 5
2	0.37	0.21	0.36	0.89	0.19	0.00	0.31	0.26	0.20	0.54
3	0.44	0.28	0.54	0.91	0.25	0.00	0.48	0.29	0.30	0.63
4	0.55	0.29	0.7	0.90	0.39	0.01	0.56	0.39	0.39	0.81
5	0.59	0.31	0.81	0.89	0.39	0.01	0.66	0.60	0.45	0.82
6	0.60	0.35	0.88	0.88	0.52	0.02	0.74	0.79	0.51	0.87

TABLEAU 4.5 – Moyenne des courbes de score de la FIGURE 4.15.

La FIGURE 4.16, nous permet d'évaluer si la tendance observée pour le nombre de neurones de sortie n'est pas spécifique à l'intervalle $[2, 6]$. Nous avons donc effectué des tests avec des valeurs plus élevées que celles initialement prévues dans le protocole, en réalisant 100 exécutions pour garantir la fiabilité des résultats. Nous avons observé que pour 20 ou 50 neurones de sortie, le résultat était également proche de 1. Dans l'ensemble, ces résultats viennent conforter l'hypothèse (**H3**).



$\mu = 0.93$, $\sigma = 0.02$
10 neurones de sortie

$\mu = 1.0$, $\sigma = 0.02$
20 neurones de sortie

$\mu = 1.0$, $\sigma = 0.01$
50 neurones de sortie

FIGURE 4.16 – Score calculé à partir du motif ligne verticale 5 pixels, moyenné pour 100 exécutions.

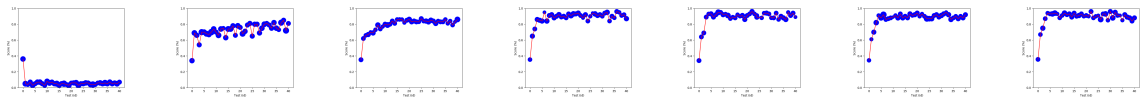
4.3.7 Longueur des phases d'apprentissage et de test

Faire varier la taille des phases d'apprentissage (t_{train}) ainsi que celle des tests (t_{test}) constitue des facteurs pertinents à étudier. Connaître l'impact de la durée d'apprentissage sur les performances du réseau nous permet d'identifier le temps minimal nécessaire à la STDP pour stabiliser le réseau. L'impact de la durée de test nous renseigne sur le temps nécessaire pour que le réseau produise une réponse. Ainsi, en faisant varier la longueur de ces phases, on peut déterminer l'impact induit par ce facteur sur l'efficacité du réseau à reconnaître les différentes classes. La première hypothèse (**H4-A**) stipule qu'en dessous d'une certaine longueur de fenêtre d'apprentissage, le réseau ne parvient pas à atteindre un état stable. La deuxième hypothèse (**H4-B**) stipule qu'au-dessus d'une certaine longueur, t_{train} n'a plus d'impact sur le réseau. Pour les phases de test, l'hypothèse (**H5-A**) stipule qu'en dessous d'une certaine taille de fenêtre de test, on ne peut pas effectuer de mesure puisque le temps est trop court pour que les neurones réagissent. L'hypothèse (**H5-B**) stipule quant à elle qu'une période trop longue n'apporte pas plus d'information du fait que les neurones spécialisés auront déjà réagi pour la classe en question.

En faisant varier t_{train} pour les phases d'apprentissage (voir la FIGURE 4.17), nous constatons qu'à partir de 10 ms, les allures des courbes sont similaires et cela jusqu'à 300 ms. Cependant, lorsque $t_{train} = 6$ ms, on observe des fluctuations plus importantes par rapport aux autres valeurs (> 10 ms). En revanche, lorsque $t_{train} = 3$ ms, les scores sont presque nuls. Ce résultat peut s'expliquer par le fait que la vitesse de déplacement des motifs est de 3 pixels/ms, ce qui ne permet pas de distinguer clairement les mouvements pendant une phase d'apprentissage aussi courte. De plus, une période de pause sans stimulation intervient entre la phase d'apprentissage et la phase de test, ce qui réduit encore plus l'activité des neurones de sortie, la durée étant calibrée par rapport au temps de

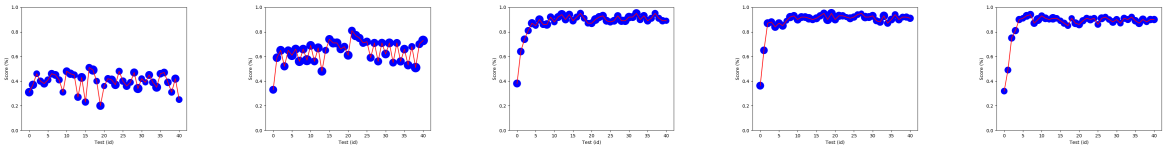
fuite des neurones. Ces observations confirment les deux hypothèses (**H4-A**) et (**H4-B**) relatives à l'influence de la durée des phases d'apprentissage.

On peut faire le même constat pour les phases de test en s'appuyant sur les résultats illustrés par la FIGURE 4.18, confirmant ainsi les hypothèses (**H5-A**) et (**H5-B**). Ici, il faut ajouter quelques précisions : pour (**H5-A**), l'explication des résultats faibles pour $t_{test} = 3$ ms est liée à la vitesse du motif qui n'est pas assez rapide pour causer un mouvement dans une fenêtre de temps aussi courte, et par conséquent le réseau ne subit qu'une excitation partielle. Pour (**H5-B**), les résultats montrent qu'à partir d'une certaine longueur de la phase de test, les scores n'évoluent plus selon ce facteur. Ces résultats sont intéressants car ils montrent une stabilité du réseau dans le déclenchement des neurones de sortie pour une classe donnée. En augmentant la taille de la phase de test, on pourrait s'attendre à une diminution du score car celui-ci est calculé à partir de codes binaires de sortie (voir SECTION 4.2.3) qui représentent une projection dans le temps des impulsions des neurones de sortie pendant la présentation des différentes classes de mouvement. Plus la durée de la fenêtre est grande, plus il y a de chances d'avoir des bits à 1 dans un code (sur une période de 200 ms, un neurone a plus de chances d'émettre une impulsion que sur une fenêtre de taille 10 ms). Cela signifie que ce sont toujours les mêmes neurones qui se déclenchent pour une direction donnée.



$\mu = 0.1$	$\mu = 0.64$	$\mu = 0.8$	$\mu = 0.89$	$\mu = 0.89$	$\mu = 0.88$	$\mu = 0.89$
$\sigma = 0.34$	$\sigma = 0.34$	$\sigma = 0.27$	$\sigma = 0.27$	$\sigma = 0.28$	$\sigma = 0.24$	$\sigma = 0.27$
3 ms	6 ms	10 ms	50 ms	150 ms	200 ms	300 ms

FIGURE 4.17 – Score calculé sur différentes longueurs de phases d'apprentissage (motif ligne verticale 5 pixels, moyenné pour 100 exécutions).



$\mu = 0.39$	$\mu = 0.64$	$\mu = 0.91$	$\mu = 0.93$	$\mu = 0.9$
$\sigma = 0.28$	$\sigma = 0.34$	$\sigma = 0.28$	$\sigma = 0.27$	$\sigma = 0.25$
3 ms	6 ms	10 ms	15 ms	50 ms

FIGURE 4.18 – Score calculé sur différentes longueurs de phases de test (motif ligne verticale 5 pixels, moyenné pour 100 exécutions).

4.3.8 Estimation du coût énergétique du réseau

Nous nous sommes concentrer uniquement sur la phase d'inférence du modèle pour estimer son coût énergétique, en ignorant les opérations effectuées pendant l'apprentissage. Pour cela, le protocole à consisté à compter le nombre d'impulsions générées par le réseau, après l'avoir entraîné sur un motif de l'étude avec un jeu de paramètres appropriés pour reconnaître les différentes classes de mouvement. Le réseau en question est composé de 10 neurones de sortie. Pour calculer la consommation énergétique du réseau, le nombre d'impulsions générées est multiplié par le coût énergétique d'une impulsion sur du matériel dédié aux SNNs. Selon des travaux de la littérature, certains circuits ont une consommation énergétique en veille de 11pW (picowatt) avec un coût unitaire de 2 fJ par une impulsion (Sourikopoulos et al. 2017). En comptant le nombre de spikes générés par le réseau pour les 8 formes, sur des séquences de mouvement de 2 secondes, la moyenne s'élève à 9.3 spikes (avec un écart-type de 7.1), soit environ 1 spike toutes les 200 ms. En appliquant ces chiffres à la consommation énergétique du circuit mentionné, on obtient des ordres de puissance inférieurs au picowatt.

4.3.9 Synthèse des résultats et discussion

Lorsqu'un réseau est entrainé, nous constatons bien que ses poids synaptiques suivent une distribution particulière avec un écart type plus élevé qu'à l'initialisation, validant ainsi la première hypothèse (**H0**). Pour ce qui est de la présentation des motifs, malgré le fait que l'on n'a pas trouvé un unique jeu de paramètres commun pour tous les motifs, on voit que le réseau reconnaît les différents mouvements pour chaque motif. Avec le premier jeu de paramètres, le réseau est capable de reconnaître différents motifs, à l'exception de la diagonale 3 pixels, de la ligne verticale de taille 3 et du point. En ajustant le seuil d'activation pour chacun de ces motifs, nous avons pu rectifier les performances du réseau pour ces entrées. À ce stade, nous ne pouvons ni valider l'hypothèse (**H1**) dans sa globalité, ni la rejeter, mais nous pouvons confirmer l'hypothèse alternative (**H1-A**) qui stipule que le réseau peut reconnaître les classes de mouvement sur l'ensemble de ces motifs. Par contre, nous ne pouvons pas valider la deuxième partie de l'hypothèse (**H1-B**) qui impose un jeu de paramètres unique pour tous les motifs. Concernant la stratégie de présentation, nous avons montré qu'elle n'a pas d'influence sur le score, rejetant ainsi l'hypothèse (**H2**). En revanche, les mesures ont démontré que le nombre de neurones sur la couche de sortie a un impact sur les performances du réseau, validant ainsi l'hypothèse (**H3**).

Concernant la longueur des phases d'apprentissage (**H4**), nous avons observé que si elle est trop courte (**H4-A**), le réseau est incapable de distinguer les différentes directions et ne peut donc pas apprendre de manière efficace. À l'inverse, à partir d'une certaine durée, le fait d'augmenter la taille n'a plus aucune influence sur le score (**H4-B**), validant ainsi (**H4**). Pour la taille des phases de test (**H5**), si celle-ci est trop courte, les neurones n'ont pas le temps d'émettre suffisamment d'impulsions (**H5-A**), tandis que si elle est trop longue, cela n'a pas d'effet significatif sur la performance du réseau (**H5-B**). Ces observations montrent une certaine stabilité dans la mesure du résultat de l'inférence, ce qui confirme l'hypothèse (**H5**).

Nos principales constatations sont les suivantes : nous avons entraîné avec succès des SNN de petites tailles pour la reconnaissance de la direction du mouvement de motifs simples. Quelques millisecondes de stimulation non supervisée suffisent pour atteindre une situation stable, et la poursuite de la stimulation ne dégrade pas l'apprentissage.

Ce travail nous a également permis de constater que le paramétrage du modèle n'est pas trivial, et il serait utile de proposer des mécanismes permettant de régler automatiquement certains de ses aspects. Compte tenu des modalités du protocole expérimental, il est tout à fait envisageable de faire une exploration exhaustive sur l'espace des paramètres, dans le but de parvenir à une meilleure compréhension du modèle et d'établir un lien théorique entre les paramètres du réseau et les conditions de convergence. Cette approche pourrait permettre de déterminer des règles automatiques, pour améliorer le modèle existant, afin qu'il soit capable de généraliser la détection sur des motifs arbitraires contenant un nombre variable de pixels et apparaissant sur des fenêtres de tailles diverses. Nous traiterons de cet aspect dans la prochaine SECTION (4.4).

4.4 Exploration des paramètres du modèle étudié

Dans cette section, nous souhaitons à présent étudier l'impact du réglage des paramètres du réseau afin de mettre en évidence le comportement le modèle, ainsi que l'impact que ce réglage peut avoir sur l'entraînement et sur la résolution de la tâche ciblée.

4.4.1 Protocole expérimental

En conservant le même protocole expérimental que dans la SECTION 4.2 précédente, nous introduisons de nouvelles formes qui permettent de tester plusieurs réglages différents du modèle. Nous explorons les 2 paramètres suivants du réseau : le seuil d'activation des neurones et la fuite des neurones. Nous évaluons également l'impact du nombre de pixels activés en entrée (N) en utilisant 24 motifs (voir FIGURE 4.19) où N varie de 1 à 24.

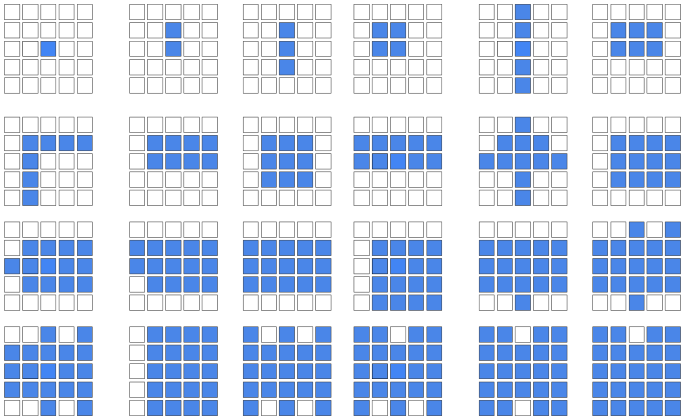


FIGURE 4.19 – Illustration des 24 motifs différentes utilisées dans notre protocole expérimental.

En suivant le même protocole expérimental exposé dans le SECTION 4.3.2, une valeur de score est obtenue lors de chaque phase de test. Afin d'avoir une estimation fiable du score et réduire l'impact de l'initialisation aléatoire des poids avant l'entraînement pouvant favoriser un apprentissage, nous effectuons 10 exécutions pour chaque configuration de paramètres (seuil, fuite du neurone, etc.). Il y a 40 phases de test par exécution ; pour chaque phase de test, on fait la moyenne des scores de toutes les exécutions. Il convient de préciser que le calcul du score de performance $perf$ diffère légèrement de celui présenté dans l'étude précédente (SECTION 4.3). Dans cette dernière, le score était calculé en fonction du nombre d'éléments $\mathrm{Card}(B)$ constituant l'ensemble des codes binaires distincts B associés à chaque classe à déterminer dont la construction est détaillée dans la SECTION 4.2.3 avec les ÉQUATIONS (4.2) (4.1) (4.3). Cette quantité était ensuite divisée par le nombre de classes à distinguer N_c afin d'obtenir une valeur de score normalisée appartenant à l'intervalle $[0\,;1]$. Cette normalisation était justifiée par le fait que certaines formes géométriques, choisies lors des différentes validations expérimentales (SECTION 4.3.2), impliquaient des déplacements induisant un nombre de classes de mouvement discernable réduit (voir SECTION 4.3.4).

— Les formes diagonales, dont la longueur était égale à la taille de la fenêtre, étaient sujettes au problème d'ouverture qui réduisait à 2 le nombre de classes discernables ;

— La géométrie torique sur laquelle ces formes se déplaçaient ne permettait pas de distinguer les directions d'un même axe lorsque la taille de ces formes sur le long de cet axe s'étendait à toute la fenêtre, réduisant à 3 le nombre de classes.

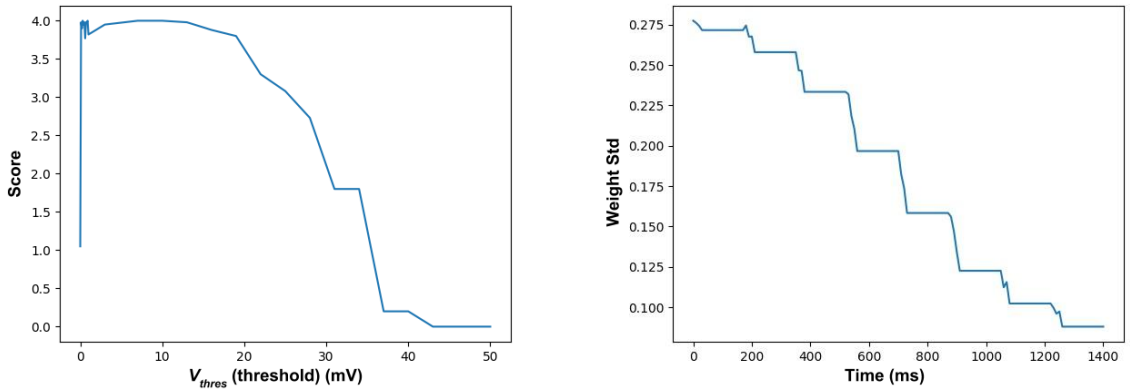
Dans cette étude, ces cas ont été supprimés en retirant les formes diagonales et en désactivant la géométrie torique de la fenêtre (ce qui signifie que le motif se déplace sans se répéter). Les motifs utilisés ont tous des déplacements identifiables dans les 4 directions ($N_c = 4$). Par conséquent, il n'est plus nécessaire de normaliser les scores selon le nombre de classes attendues, puisque que celui-ci ne varie plus en fonction de la forme présentée. Ainsi, le score est maintenant calculé de la manière suivante :

$$perf = \mathrm{Card}(B) \tag{4.5}$$

Le score de performance représente maintenant le nombre de classes distinguables. Lorsqu'un paramètre du réseau ne varie pas lors d'une expérimentation, sa valeur est indiquée dans les TABLEAUX 4.1 et 4.2.

4.4.2 Impact du seuil d'activation des neurones

Afin d'évaluer l'impact du seuil d'activation des neurones V_{thres}, nous présentons la valeur du score obtenu lorsque l'on fait varier $V_{thres} \in [0\,;50]$ mV (FIGURE 4.20a). Cet intervalle a été choisi empiriquement pour couvrir une plage de valeurs pertinentes en nous basant sur les différents réglages testés dans le SECTION 4.3.



(a) Score de performance (b) Écart-type global

FIGURE 4.20 – Dans la figure (a), la courbe symbolisant l'évolution du score lorsque le paramètre V_{thres} varie. Dans la figure (b), la courbe symbolisant l'évolution de l'écart-type global des poids synaptiques au cours d'une simulation (cas non idéal). Les plateaux correspondent aux différentes phases de test, où la règle d'apprentissage STDP est désactivée.

Lorsque le seuil d'activation a une valeur nulle, le score de performance est égal à 1. Ce score indique indirectement que tous les neurones de sortie émettent un spike systématiquement lorsqu'ils reçoivent un spike pré-synaptique en entrée, quelle que soit la valeur de leurs poids synaptiques (strictement positifs). Dans ce cas, les neurones sont sensibles à tous les stimuli d'entrée et ne peuvent pas distinguer les classes, d'où la valeur du score à 1 (rappelons que le score indique le nombre de classes qui peuvent être distinguées). Comme il n'y a aucune corrélation entre les spikes de sortie et les spikes entrants, selon la règle d'apprentissage STDP, cette configuration entraîne une série de LTD qui réduit les connexions synaptiques à un tel point que la plupart des valeurs s'approchent de w_{min}.

Afin d'appuyer nos propos, nous avons observé l'évolution des poids synaptiques d'une telle configuration sur un réseau constitué de 10 neurones de sortie. La FIGURE 4.21 illustre les poids synaptiques finaux obtenus après un entraînement.

FIGURE 4.21 – Illustration des poids synaptiques du réseau lorsque $V_{thres} = 0$ mV (cas non idéal).

Chaque matrice représente les poids synaptiques normalisés entre l'un des 10 neurones de sortie et les 25 neurones d'entrée. Cette représentation préserve la disposition spatiale des neurones d'entrée. Les zones claires (respectivement obscures) indiquent les poids des connexions sensibles (respectivement insensibles) des neurones de sortie. On peut observer que la plupart des poids synaptiques ont convergé vers 0, indiquant des neurones de sortie principalement insensibles aux entrées. Pour chaque neurone de sortie, on se retrouve généralement avec une unique synapse dont le poids a une valeur élevée, conséquence de la compétition instaurée par le mécanisme d'inhibition latérale. Ce type de distribution ne permet pas aux neurones de caractériser un motif spatio-temporel.

En outre, en examinant la courbe représentant l'évolution des écarts-types des poids synaptiques au cours de la simulation (FIGURE 4.20b, page 69), on constate que cette dernière est décroissante. Cela constitue une preuve supplémentaire que les paramètres sont mal réglés.

Pour les valeurs suivantes de V_{thres} comprises entre 0,1 et 0,9 mV, on atteint le score de 4, ce qui est un cas idéal.

Cependant, pour des seuils d'activation ayant des valeurs plus élevées, les potentiels de membrane des neurones de sortie ont du mal à atteindre cette tension, voire ne la dépassent pas, même dans les configurations les plus favorables (poids synaptiques élevés). Cela entraîne une diminution, voire une absence d'activité des neurones de sortie dans le réseau, et donc un score faible.

4.4.3 Impact du taux de fuite des neurones

La valeur τ_{leak} permet d'augmenter la sensibilité du neurone aux informations temporelles des stimulus entrants. Si un neurone est réglé avec un τ_{leak} élevé, les poids synaptiques connectés à ce neurone caractériseront des motifs spatio-temporels pouvant s'étaler sur une plus longue durée dans le temps. Cependant, cela peut rendre la distinction entre les caractéristiques spatiales plus difficile, car le neurone sera davantage orienté vers le traitement des informations temporelles. Afin d'évaluer l'impact de la fuite des neurones, nous montrons dans la FIGURE 4.22 les valeurs de score en faisant varier $\tau_{leak} \in [0\,;30]$. Lorsque τ_{leak} est petit, cela signifie que le potentiel de membrane décroît rapidement. Dans ce cas, le score obtenu est faible puisque la tension du neurone décline rapidement, ce qui a pour conséquence d'empêcher d'atteindre le seuil d'activation du neurone.

Toutefois, lorsque la valeur de fuite devient plus élevée, à partir de 9 ms selon les résultats obtenus, ce paramètre n'a plus d'impact sur le score, qui se stabilise autour de la valeur optimale de 4. Dans ce cas, un neurone LIF doté d'une valeur de τ_{leak} élevée devient une approximation du neurone IF sans fuite. Le fait d'augmenter la valeur τ_{leak} au-delà d'un certain seuil sans qu'il n'y ait d'incidence sur le score, quelle que soit la forme spatiale présentée, peut s'expliquer par la nature des données d'entrée utilisées. En effet, les séquences vidéo traitées par le réseau possèdent 2 particularités : une forme se déplace *seule* lors d'un passage en ayant une vitesse *constante*. Il n'est donc pas possible de détecter des corrélations temporelles induites par le passage précédent d'une autre forme. De plus, la tâche proposée dans cette étude ne prend pas en considération la distinction entre différentes vitesses.

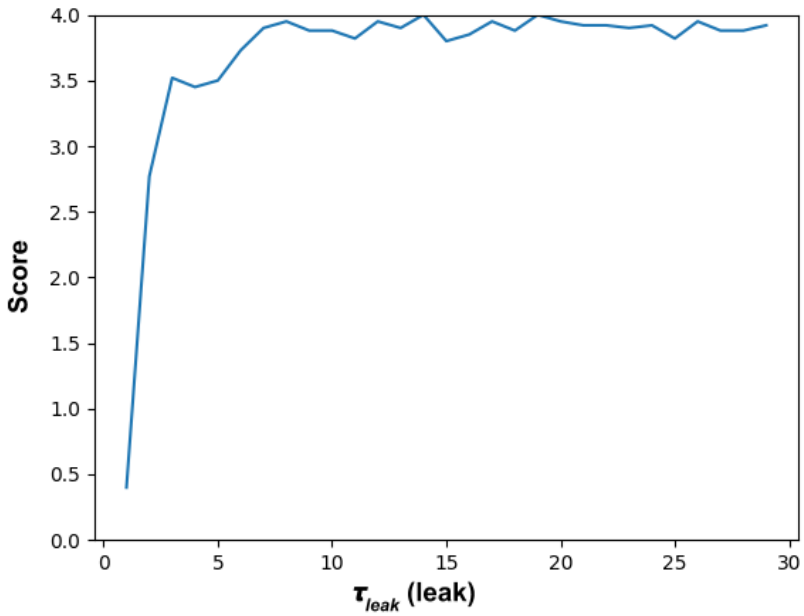


FIGURE 4.22 – Évolution du score lorsque le paramètre τ_{leak} varie.

4.4.4 Impact du nombre d'impulsions en entrée

Afin d'évaluer l'impact du nombre de pixels actifs d'entrée (N), nous utilisons les motifs présentés dans la FIGURE 4.19 (page 67), où le nombre de pixels d'entrée varie de 1 à 24. La FIGURE 4.23 montre les valeurs de score correspondantes avec un seuil pour les neurones $V_{thres} = 20$ mV.

Le nombre de pixels actifs d'entrée (N) a une influence sur le score, notamment dans les situations où sa valeur appartient à l'intervalle $[1\,;3]$, le score augmente linéairement dans l'intervalle $[0.5\,;3]$. Dans les cas où N est faible, l'entrée ne produit pas assez de spikes pour activer les neurones de sortie. Étonnamment, les scores montrent que le réseau produit des sorties distinguables même pour des motifs comportant un nombre élevé de pixels actifs, ne permettant pas de mettre en lumière le phénomène de sur-activation des neurones. Ce résultat est cohérent avec les observations faites sur l'impact du seuil (SECTION 4.4.2), les situations amenant à une sur-activation n'influencent pas le score. Une explication sur ce point est développée lors de la discussion.

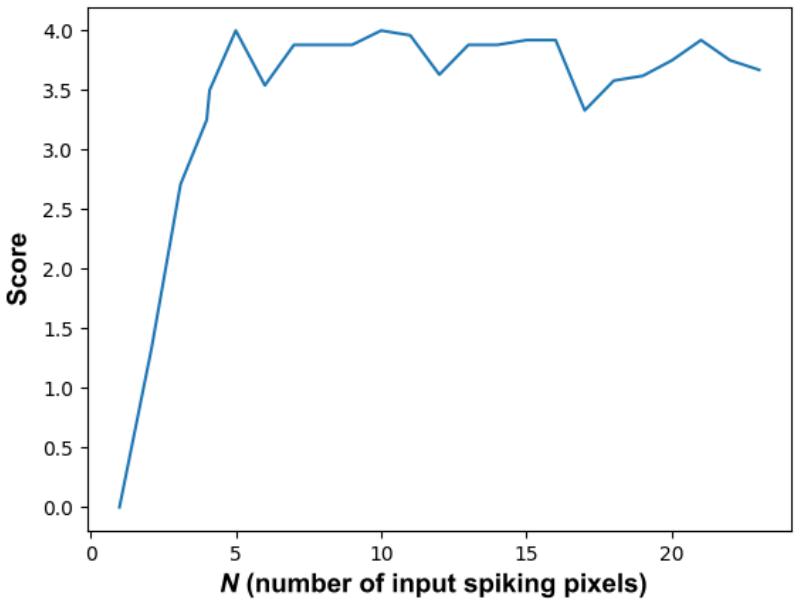


FIGURE 4.23 – Évolution du score lorsque le nombre de pixels actifs N varie.

4.4.5 Impact couplé

Afin d'évaluer l'impact conjoint du seuil (V_{thres}) et de la fuite du neurone (τ_{leak}), nous montrons dans la FIGURE 4.24 la valeur du score lorsque l'on fait varier $V_{thres} \in [1mV, 35mV]$ et $\tau_{leak} \in [0, 15]$.

On peut faire la même observation que lorsque τ_{leak} est faible, mais cela peut être compensé en ajustant le seuil à des valeurs plus faibles. Pour étayer cette observation, en choisissant un seuil $V_{thres} \in [20mV, 35mV]$, nous constatons que l'augmentation du coefficient de fuite entraîne une amélioration des scores.

Ce comportement s'explique par le fait que les deux paramètres influencent le déclenchement d'un neurone. La fuite contrôle un aspect de l'évolution du potentiel de membrane, tandis que le seuil paramètre la valeur que la tension de membrane doit atteindre pour déclencher un spike en sortie.

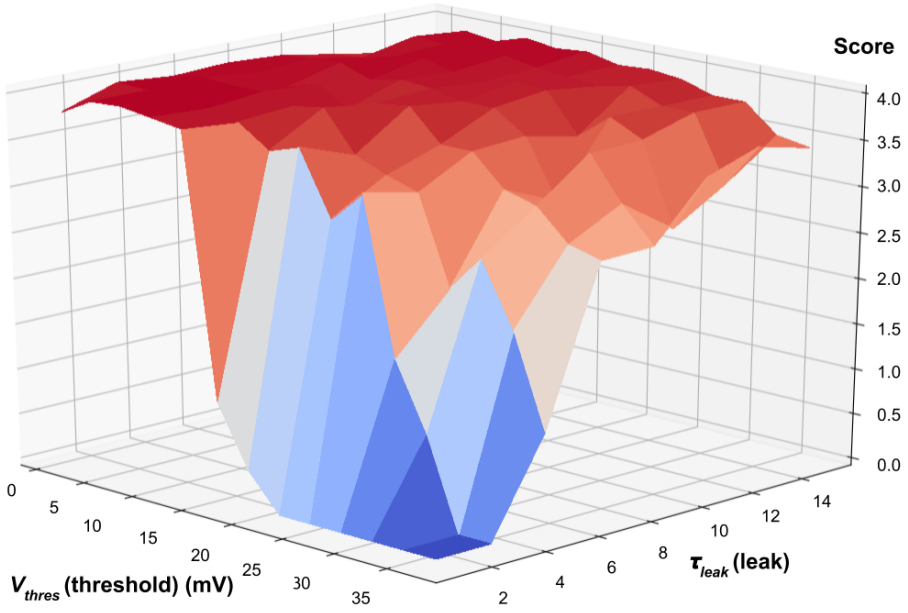


FIGURE 4.24 – Évolution du score lorsque le seuil V_{thres} et la fuite τ_{leak} varient.

Dans la même idée, nous avons regardé conjointement l'impact du nombre actif de pixels d'entrée (N) et du seuil (V_{thres}) en faisant varier $N \in [1\,;24]$ pixels et $V_{thres} \in [1\,;50]$ mV (FIGURE 4.25).

En observant le graphique, on constate que les configurations de paramètres donnant de faibles scores sont celles où V_{thres} est élevé et N est petit. En effet, les motifs dont le nombre de pixels est inférieur à un seuil (4 dans notre cas) sont très sensibles au réglage du seuil. En moyenne, le meilleur seuil est faible, à 10 mV. Enfin, il manque un dernier couple de paramètres à évaluer conjointement, à savoir, le nombre actif d'entrée et la fuite ($N \times \tau_{leak}$). Les intervalles d'exploration sont $N \in [1\,;24]$ et $\tau_{leak} \in [1\,;15]$ (FIGURE 4.26). Nous avons réduit l'intervalle τ_{leak}, en diminuant la borne supérieure à 15 ms. Nous nous sommes basés sur l'étude 4.4.3 qui montre qu'à partir d'une valeur de $\tau_{leak} = 9$ ms, le score n'est plus influencé. Avec un raisonnement empirique, nous avons mesuré le score en testant différents couples de valeurs (V_{thres}, τ_{leak}) afin de déterminer une borne intéressante.

Il y a une influence claire de N et de τ_{leak} sur le score de sortie, même si l'impact est moins important que le seuil V_{thres}. La plupart des valeurs de score sont proches de 4, sauf lorsque $\tau_{leak} < 4$ ou $N < 2$. Nous remarquons que lorsque τ_{leak} et N sont tous deux petits (c'est-à-dire une fuite rapide et une faible activité en entrée), le score devient nul.

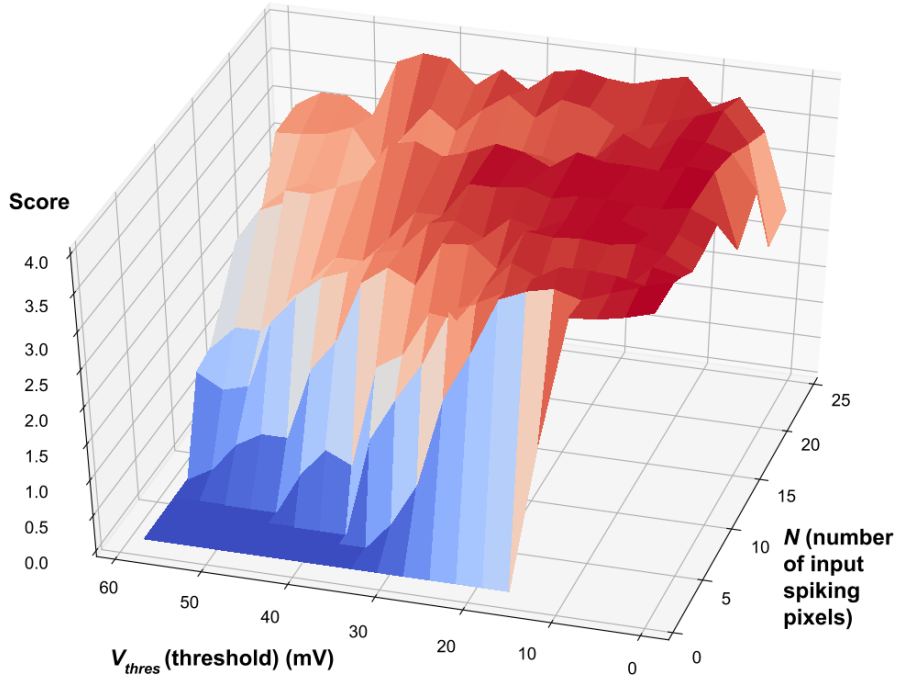


FIGURE 4.25 – Évolution du score lorsque le nombre de pixels actifs N et le seuil V_{thres} varient.

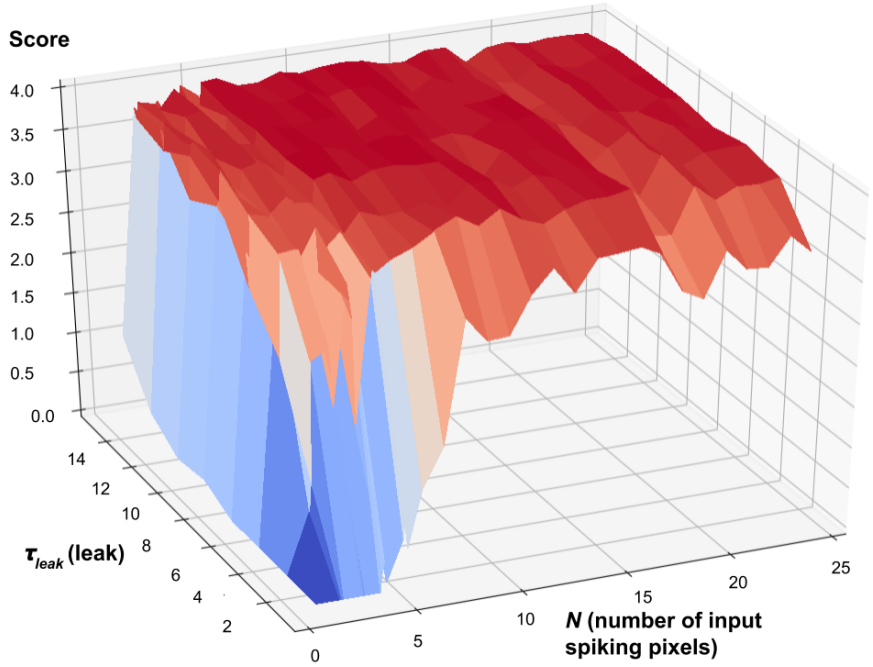


FIGURE 4.26 – Évolution du score lorsque le nombre de pixels actifs N et la fuite τ_{leak} varient.

4.4.6 Discussion

Cette étude portant sur l'exploration des paramètres vise à éclaircir les effets différents des paramètres sur la qualité des caractéristiques extraites par le réseau dans un contexte d'analyse non supervisée du mouvement. Suivant la nature des données, le modèle n'a pas les mêmes performances avec des jeux de paramètres différents.

Bien que des approches utilisant des algorithmes évolutionnistes pour trouver un réglage optimal puissent être envisagées, il convient de noter que leur utilisation systématique peut être coûteuse en termes de ressources de calcul (complexité exponentielle), car ces algorithmes nécessitent souvent de multiplier les phases d'apprentissage. En outre, ils ne garantissent pas nécessairement une convergence satisfaisante permettant une généralisation du modèle.

Afin d'étendre au mieux les capacités du modèle à résoudre la tâche cible, des mécanismes d'homéostasies peuvent être intégrés au réseau pour régler dynamiquement les paramètres. On peut notamment citer le seuil d'activation du neurone dont le réglage est très corrélé au type de données soumis en entrée et plus particulièrement aux nombres d'impulsions nécessaires pour décrire sa représentation. Dans des applications réelles, la distribution des impulsions constituant les données à traiter est non contrôlée et souvent imprévisible, il est donc essentiel de pouvoir contrebalancer et réduire l'impact de cette contrainte. En se rapportant aux découvertes faites lors des études sur le fonctionnement du cerveau, des mécanismes de seuil adaptatif ont été mis en évidence et peuvent régler les problèmes de sûr ou sous-activation des neurones.

Dans cette étude, les résultats montrent un phénomène de sous activation des neurones du réseau lorsque les séquences d'entrées sont décrites par un petit nombre d'impulsions et que le seuil d'activation des neurones est trop élevé. Cependant, le phénomène de sur-activation des neurones n'a pas été mis en évidence lors des différentes validations expérimentales. L'absence de sur-activation des neurones de sortie peut être justifiée par plusieurs facteurs. Tout d'abord, le mécanisme d'inhibition latérale du modèle favorise la réponse d'un seul neurone pour un motif spatio-temporel. La deuxième raison porte quant à elle sur la fonction de score utilisée lors des mesures de performance du réseau. En effet, ce score de performance est calculé en comptant les différents codes binaire de sortie basés sur l'activité en sortie du réseau (SECTION 4.2.3). L'une des critiques que l'on peut émettre à l'encontre de cet encodage est qu'il ne prend pas en compte le moment où l'impulsion s'est produite, on perd donc une partie de l'information. Il en résulte des situations où différentes séquences d'impulsions peuvent avoir le même code, même si elles ont des ordres chronologiques différents.

Chapitre 5

Apprentissage incrémental de motifs spatio-temporels

5.1 Motivations et objectifs de l'étude

Dans ce chapitre, nous allons étudier la capacité du modèle à apprendre de manière incrémentale (ou continue). Le but de l'apprentissage incrémental est que le modèle s'adapte aux nouvelles données sans oublier ses connaissances existantes. Dans l'étude précédente, nous avons identifié certaines propriétés remarquables permettant de mettre en place ce type d'entraînement. En effet, dans le cas des classes de mouvements étudiées dans le CHAPITRE 4, quelques présentations suffisaient afin que la STDP face converger le modèle vers un état où le réseau est capable de reconnaître la classe en question (SECTION 4.3.7). De plus, nous avons remarqué que l'ordre de présentation des classes n'influençait pas les performances du modèle (SECTION 4.3.5).

Lorsqu'on analyse les propriétés de la règle d'apprentissage STDP, certaines peuvent expliquer ces résultats. En premier lieu, la mise à jour des poids par la règle STDP ne requiert pas la connaissance de la vérité terrain. Elle se base plutôt sur l'observation des réactions des différents neurones et agit comme un détecteur de corrélations et de coïncidences dans le domaine temporel. Elle prend donc en compte le modèle de représentation du système et la nature des données afin de faire évoluer le réseau. En second lieu, cette règle s'applique localement sur un système dynamique évoluant avec le temps permettant de ne pas modifier tous les paramètres du modèle mais seulement ce qui son pertinent. Étonnamment, le problème d'apprentissage incrémental a rarement été abordé avec des réseaux de neurone impulsionnels et pourtant c'est un aspect central dans l'apprentissage dont font preuve les êtres vivants. Des travaux comme ceux de Thangarasa (Vithursan Thangarasa 2020) abordent le problème de l'apprentissage incrémental avec des réseaux de neurones classiques sur des tâches de reconnaissance d'image statique en appliquant leur modèle à des datasets tel que MNIST et CIFAR-10. Ils soulignent les difficultés du problème en évoquant notamment la question de l'oubli catastrophique. Ils proposent une solution alternant des phases d'apprentissage rapide et lente. Cossu et ses collègues (Cossu, Carta et Davide 2020) ont également abordé le problème de l'apprentissage continu, en utilisant des réseaux de neurones récurrents. G. Hocquet 2020 ont introduit un réseau de neurones inversible One-versus-All, capable d'apprendre

une classe à la fois par batches. Ororbia 2020 a introduit un SNN dans lequel les neurones prédisent l'activité des autres dans un schéma d'apprentissage continu. Allred et Roy 2020 ont exploité la localité de la règle STDP pour contrôler les mises à jour des poids dans les zones cibles, tout en préservant les informations précédemment apprises. Ils ont également appliqué leur modèle à MNIST. Parmi toutes les approches citées ici, seules les deux dernières utilisent les SNN et la règle STDP pour l'apprentissage continu. Cependant, aucun travail répertorié n'étudie la possibilité d'utiliser des SNN sur de la vidéo.

L'objectif de ce chapitre est donc d'étudier cette piste en analysant différentes propriétés d'un réseau feed-forward simple sur la capacité à apprendre la direction du mouvement. Nous avons pris en compte certaines critiques émises lors de l'étude précédente afin de limiter les biais lors de nos analyses des capacités d'apprentissage incrémental du modèle. Nous avons changé les données d'entrées par des séquences représentant des motifs plus variables dans la forme et la vitesse pendant leurs déplacements, limitant ainsi les effets de synchronicité entre les pixels et se rapprochant des données réels. Pour ce faire, nous avons élaboré un dataset événementiel composé de courtes séquences d'une main qui se déplace dans l'une des 4 directions cardinales. L'acquisition des vidéos s'est faite à l'aide d'une caméra standard. Nous avons ensuite transformé ces vidéos en évènements de manière logiciel (SECTION 5.2.3). De même, nous avons ajouté un mécanisme d'homéostasie pour les seuils d'activation des neurones afin qu'ils puissent s'adapter aux différentes données. Enfin, la méthode d'évaluation du score de performance du modèle (SECTION 4.2.3) a changé. En effet, la manière d'évaluer le score dans l'étude précédente ne prenait pas en compte les aspects temporels ou fréquentiels des réponses faites par le réseau, limitant l'interprétation sur la qualité des caractéristiques extraites par le modèle. Afin de remédier à ce problème, nous avons utilisé un classifieur qui analyse les caractéristiques extraites par le réseau. En pratique, ces algorithmes traitent des données d'entrée sous forme de vecteurs, les réponses du réseau ont donc dû être décodées avec des méthodes transformant les impulsions en scalaires. Pour éviter d'introduire des biais dans l'évaluation des performances du réseau, plusieurs méthodes de décodage (SECTION 5.2.4) ainsi que 2 classifieurs (SECTION 5.2.5) ont été utilisés.

Notre étude se décompose ensuite en deux parties afin d'évaluer la capacité du modèle à apprendre de manière non supervisée et incrémentalement les différentes classes : La première partie permet d'évaluer la capacité du modèle à extraire des caractéristiques pertinentes après un apprentissage non supervisé (SECTION 5.3.1). La deuxième partie permet d'évaluer la capacité d'apprentissage incrémental du modèle en introduisant les classes du problème séquentiellement. Les scores de performance du réseau sont ensuite analysés à différentes étapes de l'entraînement en vérifiant que les anciennes classes apprises le reste malgré l'introduction d'une nouvelle classe (SECTION 5.3.2).

Dans cette étude, les résultats montrent que le modèle possède des capacités d'apprentissage incrémental et permettent d'ouvrir de nouvelles perspectives intéressantes dans l'utilisation des SNNs en vision par ordinateur.

5.2 Méthodes

5.2.1 Vue d'ensemble

La FIGURE 5.1 montre une vue d'ensemble des blocs constituant la pipeline permettant de faire les différentes validations expérimentales.

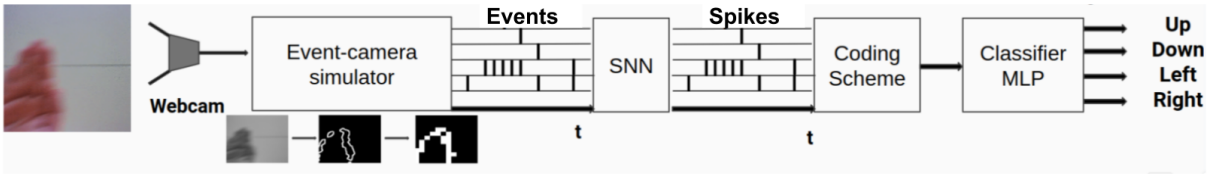


FIGURE 5.1 – Vue d'ensemble du cadre expérimental.

On peut identifier 4 blocs importants : le premier est le simulateur de vidéos événementielles permettant de construire le dataset à partir de vidéo RGB (voir la SECTION 5.2.3). Le deuxième bloc correspond au modèle SNN entraîné par la STDP en introduisant les classes progressivement et en sauvegardant l'état du réseau à chaque nouvel entraînement. Le but est d'extraire des caractéristiques spatio-temporelles pertinentes (voir la SECTION 5.2.2). La troisième partie permet d'encoder les spikes des neurones de sortie du modèle SNN dans un encodage adapté au classifieur. Dans ces travaux, 3 schémas d'encodage sont proposés afin d'évaluer lequel est le plus pertinent pour cette tâche de classification (voir la SECTION 5.2.4). Enfin, le dernier bloc permet de classifier les données extraites afin de mesurer la qualité de leur représentation : une précision de classification élevée indique une bonne qualité de la représentation (voir la SECTION 5.2.5).

5.2.2 Modèle SNN

L'architecture du modèle utilisé est décrite en détails dans la SECTION 3.2. Les paramètres (voir SECTION 3.2.5) des neurones du réseau sont configurés avec des valeurs identiques à celles de l'étude précédente (SECTION 4.2.1). Elles sont indiquées dans le TABLEAU 4.1, idem pour les paramètres des synapses. Cependant, nous avons ajouté un mécanisme adaptatif pour les seuils d'activations des neurones du modèles. Cet ajout permet de répondre aux problématiques liés à la sous/sur activation des neurones.

$$\frac{dv}{dt}(t) = -\frac{1}{\tau_{leak}}v(t) + u(t), v \leftarrow V_{reset} \text{ quand } v \geq V_{thres}^{homeo}. \tag{5.1}$$

L'ÉQUATION (5.1) décrit le comportement d'un neurone LIF (SECTION 2.2.4 pour les détails) avec l'ajout d'un seuil d'activation du neurone adaptatif. L'évolution du seuil d'activation adaptatif V_{homeo} est décrite par l'ÉQUATION (5.2) :

$$\begin{aligned} \frac{dV_{homeo}}{dt}(t) &= -\frac{1}{\tau_{homeo}} + \alpha_{homeo}, \\ V_{homeo} \leftarrow V_{homeo}^{max} &\text{ quand } V_{homeo} \geq V_{homeo}^{max}, \\ V_{homeo} \leftarrow V_{homeo}^{min} &\text{ quand } V_{homeo} \leq V_{homeo}^{min}, \\ V_{thres}^{homeo} &= V_{thres} + V_{homeo} \end{aligned} \tag{5.2}$$

où τ_{homeo} représente le paramètre de fuite linéaire du seuil et α_{homeo} la magnitude dans l'augmentation du seuil lorsque le neurone s'est déclenché. L'évolution de la valeur du seuil adaptatif est bornée par l'intervalle $[V_{homeo}^{min}; V_{homeo}^{max}]$. Le TABLEAU 5.1 indique les valeurs des différents paramètres permettant le réglage du mécanisme homéostasique présenté :

Paramètre	**Valeur**	**Description**
V_{homeo}^{min}	-5 mV	Borne inférieure de la variation du seuil adaptatif.
V_{homeo}^{max}	10 mV	Borne supérieure de la variation du seuil adaptatif.
τ_{homeo}	10 ms	Temps de fuite du seuil adaptatif.
α_{homeo}	1 mV	Pas d'augmentation du seuil adaptatif.

TABLEAU 5.1 – Paramètres liés au mécanisme adaptatif du seuil d'activation des neurones du réseau.

Pour rappel, les valeurs des paramètres définissant l'architecture globale du réseau sont décrites dans le TABLEAU 5.2 :

Paramètre	**Valeur**	**Description**
N_{output}	10	Nombre de neurones de sortie..
N_{inhib}	N_{output}	Nombre de voisins à inhiber par neurone.
T_{LTP}	2 ms	Durée de la période à laquelle l'apprentissage (LTP) fonctionne.

TABLEAU 5.2 – Paramètres liés à l'architecture du réseau.

5.2.3 Conception d'un dataset (HandCam)

Nous avons conçu un dataset événementiel regroupant plusieurs courtes séquences d'une main qui se déplace dans l'une des 4 directions cardinales. Celui-ci est disponible publiquement en ligne [1]. Afin de générer les données, nous avons utilisé une caméra standard (webcam) et un simulateur d'événements (description détaillée plus bas), comme le montre la FIGURE 5.1 (page 79). Nous avons acquis 400 séquences vidéos, soit 100 séquences pour chacune des 4 classes de mouvement.

Dans la suite, nous allons faire référence aux différentes classes de mouvement par des labels numérotés (TABLEAU 5.3).

Nom de la direction du mouvement	Label
Haut	0
Bas	1
Gauche	2
Droite	3

TABLEAU 5.3 – Tableau associant le nom des classes de mouvement et leurs labels associés.

5.2.3.1 Transformation des images RGB vers des évènements

Les images brutes extraites des vidéos nécessitent un pré-traitement afin d'être exploitées par le réseau. Ils existent des solutions permettant de transformer une vidéo classique en données événementielles, avec des résultats très réalistes (Gehrig et al. 2020 ; Rebecq et al. 2019a ; Scheerlinck et al. 2020). Cependant, ces méthodes nécessitent des ressources en calculs importants (utilisation de réseaux profonds) afin d'obtenir ces résultats. Dans notre cas, nous cherchons à maîtriser le type d'entrée fourni au réseau. C'est pourquoi nous avons conçu une méthode permettant de transformer les images matricielles en événements via différentes opérations de filtrage local dont le paramétrage est entièrement contrôlé. Nous avons en outre la possibilité de proposer un système qui peut s'exécuter en temps réel en étant compatible avec la plupart des équipements (tels que les webcams, les caméras de smartphone, etc.).

Les séquences d'images sont transformées en séquences d'événements en les redimensionnant vers un plus petit champ récepteur de 12×12, en convertissant d'abord les images en niveau de gris, puis en calculant un contraste temporel – après avoir appliqué un filtre passe-bas (flou gaussien) pour réduire le bruit. Le contraste temporel est calculé selon la différence entre l'image actuelle et l'image précédente (l'absence de mouvement n'entraîne aucun événement). La FIGURE 5.2 montre les différentes étapes du pré-traitement et le TABLEAU 5.4 (page 83) indique les valeurs des constantes utilisées lors des différentes transformations que nous allons détailler.

1. URL : https ://gitlab.univ-lille.fr/fox/handcam

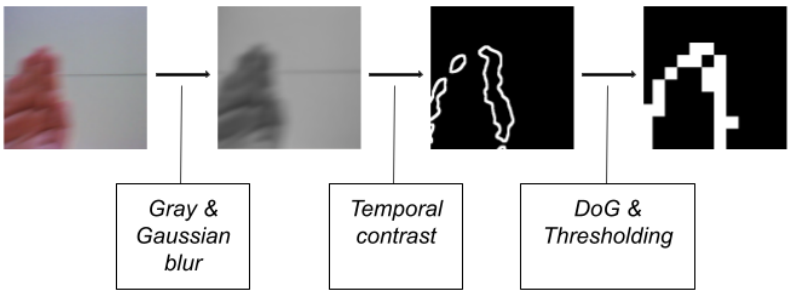


FIGURE 5.2 – Les différentes étapes du pré-traitement dans le simulateur de capteur événementiel.

Nous commençons donc par appliquer une première une transformation à l'image I afin d'obtenir l'image en niveau de gris $Grey$ décrit par l'ÉQUATION 5.3 :

$$Grey(x,y) = 0.299 \times I_R(x,y) + 0.587 \times I_G(x,y) + 0.114 \times I_B(x,y) \tag{5.3}$$

où $\times : \mathbb{R}^{N\times M} \to \mathbb{R} \to \mathbb{R}^{N\times M}$ est l'opérateur permettant d'appliquer une multiplication d'un scalaire sur tous les élément d'une matrice. Les 3 canaux de l'image (I_R, I_G, I_B) sont réduit en un seul.

Afin de réduire le bruit généré lors du calcul du contraste temporel, nous appliquons un flou Gaussien (GaB) afin d'obtenir l'image GaB qui est décrit par l'ÉQUATION (5.4) :

$$GaB(x,y) = Grey(x,y) * ((1/GaB_{size}) \times G_{GaB_{size},GaB_{center}}(x,y)) \tag{5.4}$$

où $*$ est l'opérateur de convolution et $G_{S,\sigma}$ est un noyau Gaussien normalisé de taille S et d'échelle θ défini par l'ÉQUATION (5.5) :

$$G_{S,\theta}(u,v) = \frac{g_\theta((u,v)}{\sum_{i=-\mu}^{\mu}\sum_{j=-\mu}^{\mu} g_\theta((i,j)}, u,v \in [-\mu,\mu], \mu = \frac{S}{2} \tag{5.5}$$

avec g_θ la fonction Gaussienne 2D centrée sur θ. Les paramètres du filtre sont sa taille GaB_{size} et la valeur centrale du noyaux Gaussien GaB_{center}.

Le contraste temporel est calculé en faisant la différence entre l'image actuelle et l'image précédente. Lorsqu'il n'y a pas de mouvement, l'image est nulle. Les valeurs négatives et positives des pixels représentant respectivement les polarités ON / OFF sont fusionnées. La sortie de l'image $Delta$ à la position (x,y) résultat de l'image actuelle GaB_t et de l'image précédente GaB_{t-1} est définie par l'ÉQUATION (5.6) :

$$Delta(x,y) = h_{Delta_{thres}}(abs(GaB_t(x,y) - GaB_{t-1}(x,y))) \tag{5.6}$$

où abs permet de fusionner les deux polarités, h_{thres} est la fonction de seuillage utilisée pour créer une image binaire. Elle est définie dans l'ÉQUATION (5.7) :

$$h_{thres}(x,y) = \begin{cases} 0 & \text{si } 0 < thres \\ 1 & \text{sinon} \end{cases} \tag{5.7}$$

avec $thres$ la valeur de seuil. Le paramètre de la binarisation est le seuil $Delta_{thres}$.

Les bords sont extraits de l'image à l'aide d'un filtre à différence de Gaussienne (DoG). La sortie du filtre DoG à la position (x, y) de l'image $Delta$ est définie par l'ÉQUATION (5.8) :

$$DoG(x, y) = Delta(x, y) * (G_{DoG_{size}, DoG_{center}} - G_{DoG_{size}, DoG_{surround}}) \tag{5.8}$$

avec $G_{S,\theta}$ un filtre Gaussien 2D centrée sur θ de taille S. Le filtre, de taille DoG_{size}, utilisé pour faire la différence Gaussienne se calcule en faisant la soustraction d'un filtre dont l'application donne une image floutée paramétrée par DoG_{center} avec un autre filtre permettant d'obtenir une version de l'image moins floutée paramétrée $DoG_{surround}$.

Une fois l'image DoG obtenue, les valeurs positives et négatives génèrent les évènements, pour obtenir une image binaire $Event$ définie par l'ÉQUATION (5.9) :

$$Event(x, y) = h_0(\text{abs}(DoG(x, y))) \tag{5.9}$$

Enfin, nous transformons l'image $Event$ dont la structure est une matrice en un ensemble d'événements $Events$ symbolisés par des triplets $W \times H \times \mathbb{R}_+$. Les premiers éléments du triplet correspondent aux coordonnées spatiales respectivement sur l'axe horizontal x et vertical y appartenant aux intervalles $W = [0; width[$ et $H = [0; height[$, dont les bornes supérieures $width : \mathbb{N}_+$ et $height : \mathbb{N}_+$ représentent la largeur et la hauteur de l'image. Le dernier élément du triplet correspond au temps courant t d'une image dans la séquence vidéo, elle appartient à l'intervalle $T = [0; t_{max}[$, avec t_{max} la durée de la vidéo. L'ensemble $Events$ est défini par l'ÉQUATION (5.10) :

$$Events = \{(x, y, t) \text{ si } Event_t(x, y) > 0, x \in W, y \in H \text{ et } t \in T\} \tag{5.10}$$

avec $Event_t(x, y)$ la valeur d'un pixel à une position spatiale et à un instant donné. Si la valeur est égale à 1, cela signifie la présence d'un événement.

Tous les paramètres utilisés lors des différentes transformations des images RGB en événements ont leurs valeurs décrites dans le TABLEAU 5.4.

Paramètres			
GaB_{size}	21	DoG_{size}	7
GaB_{center}	21	DoG_{center}	4
$Delta_{thres}$	50	$DoG_{surround}$	1
$Delta_{thres}$	50		

TABLEAU 5.4 – Valeur des différents paramètres utilisées pour la transformation d'une séquence d'image RGB en une séquence d'évènements.

5.2.3.2 Augmentation de données

Comme toutes les directions de mouvement sont symétriques, nous avons augmenté l'ensemble de données en effectuant des rotations des séquences de sortie de $\frac{\pi}{2}$, π et $\frac{3\pi}{2}$. Par conséquent, le jeu de données final contient $4 \times 100 = 1600$ séquences.

De plus, l'augmentation des données élimine tout biais lié à l'acquisition vidéo, puisque chaque séquence appartient maintenant aux quatre classes.

5.2.4 Stratégie de décodage

Notre objectif avec ce modèle SNN est de classifier les différentes séquences représentant des déplacements de la main en extrayant des caractéristiques pertinentes et distinguables (séquences de spikes) selon les classes présentées. Lorsqu'une séquence est présentée au réseau, celui-ci produit des spikes en sortie. Il est alors nécessaire d'être capable d'analyser cette sortie. Dans la littérature, il existe plusieurs schémas de codage qui peuvent être utilisés pour représenter les spikes, tels que mentionnés dans la SECTION 2.2.9. Nous avons choisi d'explorer 3 schémas de codage différents : un schéma de codage fréquentiel et deux schémas de codage temporel. Pour le codage fréquentiel (*spike count*), le principe est simple : il consiste à compter les spikes de chaque neurone de sortie. Pour le premier codage temporel (*latency*), le code est donné par le *timestamp* du premier spike de chaque neurone de sortie pendant la durée de la présentation de la séquence. Si un neurone n'a pas émis de spike pendant la durée de la séquence, le code correspondra à un *timestamp* supérieur à cette durée. Enfin, pour le second codage temporel (*rank order*), le code défini un rang pour chaque neurone de sortie, selon l'ordre d'arrivée de chaque spike. Le neurone qui émet le spike le plus précoce sera classé en premier. S'il y a une égalité temporelle entre deux neurones, les neurones de rangs identiques obtiendront le même code. Comme pour le codage *latency*, si un neurone n'a pas produit de spike pendant la durée de la séquence, il se verra attribuer une valeur particulière, dans notre cas 0. Il est possible que plusieurs neurones émettent leurs premiers spikes en même temps. Dans la FIGURE 5.3, ce cas se présente pour les neurones $n2$ et $n9$, qui viennent en première place ex-aequo. Lorsque nous mettons en œuvre ce codage, le classement du prochain neurone à émettre un spike ne sera pas influencé ($n5$ a le rang 3 et non le rang 2).

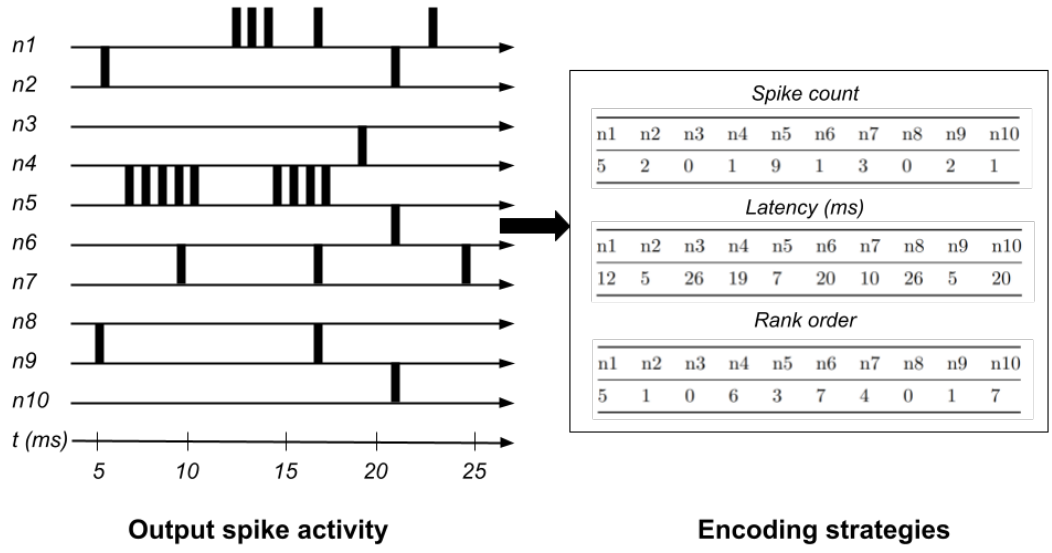


FIGURE 5.3 – Différentes stratégies de codage. À gauche : exemple d'activité des neurones de sortie. À droite : les codes correspondants.

5.2.5 Évaluation du score

Une manière d'évaluer la qualité des caractéristiques extraites par notre modèle SNN consiste à mesurer les performances obtenues en utilisant un algorithme de classification avec les données produites par le réseau. De ce fait, en quantifiant la performance des prédictions inférées par un classifieur sur la base des caractéristiques extraites, nous avons une mesure indirecte de la pertinence de ces caractéristiques. Afin de limiter l'influence du choix du classifieur sur l'indicateur de performance, celui-ci doit être un modèle simple, construit pour qu'il soit peu *expressif* et donc qu'il comporte peu de paramètres à optimiser. L'objectif est que le score reflète davantage le niveau de *distinguabilité* des données générées par le SNN. Dans le but de satisfaire ces exigences, deux algorithmes ont été choisi afin d'analyser les sorties du réseau impulsionnel. Le premier de ces algorithmes est le `k Plus Proches Voisins` ou `k Nearest Neighbors (kNN)`, considéré comme l'un des algorithmes les plus simples à mettre en œuvre en matière d'apprentissage automatique. En effet, le principe du kNN est basé sur une idée intuitive : prédire la classe d'un échantillon en se basant sur les classes des k-échantillons les plus proches dans l'espace des caractéristiques. C'est une méthode non-paramétrique qui ne nécessite pas d'étape d'apprentissage ou de modélisation complexe. Ainsi, pour obtenir une efficacité optimale du kNN, il est important que la distribution des données d'entrée soit uniforme avec des classes qui soient clairement séparées par des frontières distinctes. Les performances de classification du kNN peuvent alors être utilisées comme une mesure indirecte de la qualité des caractéristiques extraites par le SNN.

Le choix du deuxième algorithme s'est porté sur le `Multilayer Perceptron` (MLP), en complément du kNN. En effet, le MLP est considéré comme l'un des meilleurs algorithmes de classification avec des performances élevées, ce qui en fait un choix judicieux pour compléter les prédictions faites par le kNN.

Toutefois, il convient de limiter la complexité des représentations internes que le MLP pourrait développer dans ses couches cachées lors de l'apprentissage, car cela pourrait influencer de manière biaisée les résultats sur la qualité des données d'entrée fournies par le SNN. Il ne serait plus possible d'identifier clairement la contribution des caractéristiques extraites par le SNN dans le score de classification du MLP. Ainsi, pour limiter les représentations internes du MLP et éviter les biais dans les résultats mesurant la qualité des données d'entrée fournies par le SNN, l'architecture du classifieur est constituée d'une unique couche cachée de 1000 neurones. Nous avons donc à notre disposition deux méthodes distinctes pour l'analyse de la qualité des caractéristiques, l'une étant paramétrique (MLP) et l'autre non-paramétrique (kNN). Il convient de préciser que pour minimiser l'impact de la distribution des données sur l'ensemble d'apprentissage et de test, ainsi que le facteur aléatoire inhérent à l'initialisation des classifieurs, nous avons mis en place une validation croisée `K-Fold` avec $k = 5$, et nous avons répété 10 fois le processus de classification afin d'avoir des résultats qui minimisent ces biais. Pour finir, les paramètres utilisés pour le `kNN` sont les suivants : $k = 10$ pour le nombre de voisins et la formule Euclidienne pour calculer les distances entre les données. Concernant le MLP, le learning rate est fixé à $\eta = 0.001$.

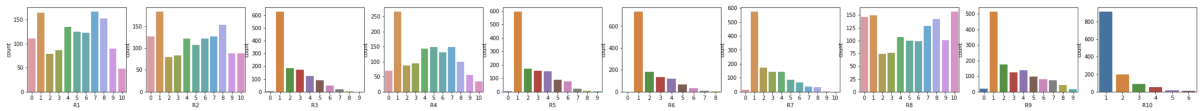
5.3 Résultats

5.3.1 Reconnaissance des mouvements de la main

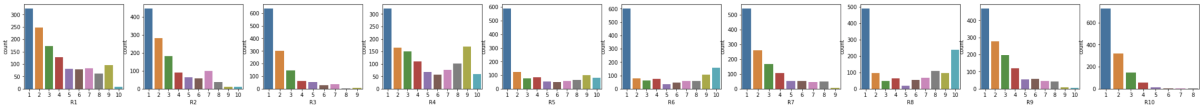
Nous présentons ici des résultats expérimentaux démontrant la capacité du réseau à apprendre à reconnaître les directions de mouvement de manière non supervisée. Toutes les séquences événementielles décrites dans la SECTION 5.2.3 sont utilisées pour nos validations expérimentales.

Dans un premier temps, nous mesurons les performance d'un réseau initialisé de manière aléatoire (BT : Before Training). Ce résultat indiquant la qualité des caractéristiques extraites est ensuite comparé avec le même réseau, mais cette fois-ci entraîné sur les 4 classes (AT4 : After Training with 4 classes) Cela nous permet d'évaluer la différence entre les deux états du réseau (BT vs AT4) afin d'estimer s'il y a eu un apprentissage de la tâche donnée.

Nous avons entraîné le réseau SNN en présentant aléatoirement (distribution uniforme) des séquences d'entrée. On observe dans la FIGURE 5.4 le comportement des neurones de sortie lorsqu'on leur montre les séquences d'entrée. Nous observons une nette différence entre les distributions des spikes en sortie, décodées à l'aide de la stratégie "rank order", avant et après l'entraînement STDP. En particulier, après l'entraînement, ces distributions sont en général plus uniformes et plates. Cette observation suggère que chaque neurone génère moins de spikes et que l'activité en sortie de chacun est plus spécifique, indiquant que nous pouvons interpréter ces séquences d'impulsions comme n'étant pas simplement le résultat d'un phénomène aléatoire (c'est-à-dire que les neurones ont tendance à ne pas produire la même sortie à chaque fois).



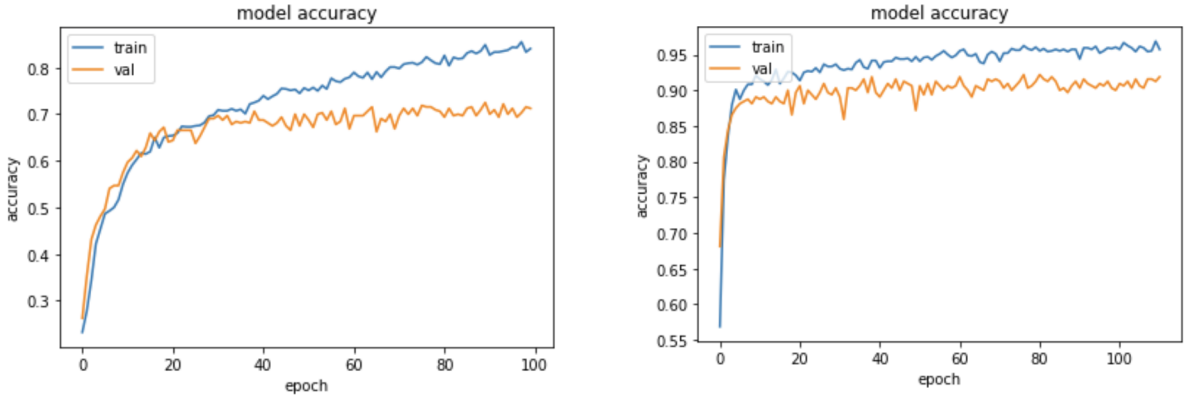
(a) BT : Before Training



(b) AT4 : After Training with 4 classes

FIGURE 5.4 – Illustration de l'activité de la couche de sortie en comptant pour chacun des neurones les valeurs obtenues par le codage "*rank order*". Pour ce faire, nous avons présenté les 1600 séquences, avant l'entraînement (a) et après l'entraînement sur les quatre classes (b). Autrement dit, les histogrammes correspondent à la distribution des codes de sorties pour chaque neurone enregistré sur l'ensemble du dataset. On observe que les distributions ont beaucoup changé après l'entraînement, par rapport à celles d'avant. Notez que pour l'état BT du réseau, presque tous les neurones ont des neurones qui n'ont pas émis de spike lors de la simulation (code 0), ce qui ne se produit pas pour AT4.

En examinant la FIGURE 5.5 illustrant l'efficacité du classifieur MLP, on constate que les caractéristiques d'AT4 fournissent de meilleurs résultats que celles de BT. Ces caractéristiques sont représentées par des vecteurs de taille 30 provenant de la fusion des 3 stratégies de décodage. Enfin, on note que le classifieur a tendance à converger plus rapidement lorsqu'on l'applique sur les données AT4.



(a) BT (b) AT4

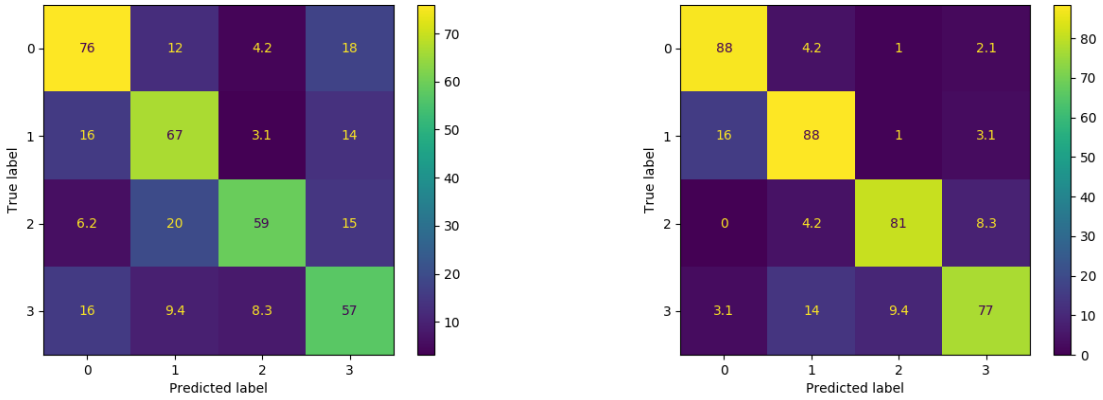
FIGURE 5.5 – La précision du classifieur (MLP) est tracée en fonction du jeu de données d'apprentissage et de validation basée sur les caractéristiques provenant des réseaux BT et AT4. Les résultats montrent que les performances obtenues avec les caractéristiques d'AT4 surpassent celles acquises avec les caractéristiques de BT.

Afin de voir si la tendance est respectée entre BT et AT4, nous pouvons analyser les résultats obtenus avec le kNN, le MLP et la fusion moyenne des deux scores (TABLEAU 5.5). Le pourcentage de bonnes prédictions est significativement supérieur entre BT et AT4 ($p = 0.00001$, $p < 0.01$), apportant un crédit supplémentaire aux observations précédemment effectuées. Par ailleurs, la FIGURE 5.6 montre une matrice de confusion pour une classification faites par le kNN. On voit que les résultats sont meilleurs pour chacune des classes du problème.

	kNN		MLP		Fusion moyenne	
accuracy	μ	σ	μ	σ	μ	σ
BT	0.64	0.013	0.73	0.012	0.69	0.013
AT4	0.86	0.012	0.92	0.014	0.89	0.013

TABLEAU 5.5 – Scores F1 pour les différents classifieurs. Les variables μ et σ, sont respectivement, la moyenne et l'écart-type du score F1 sur 10 exécutions.

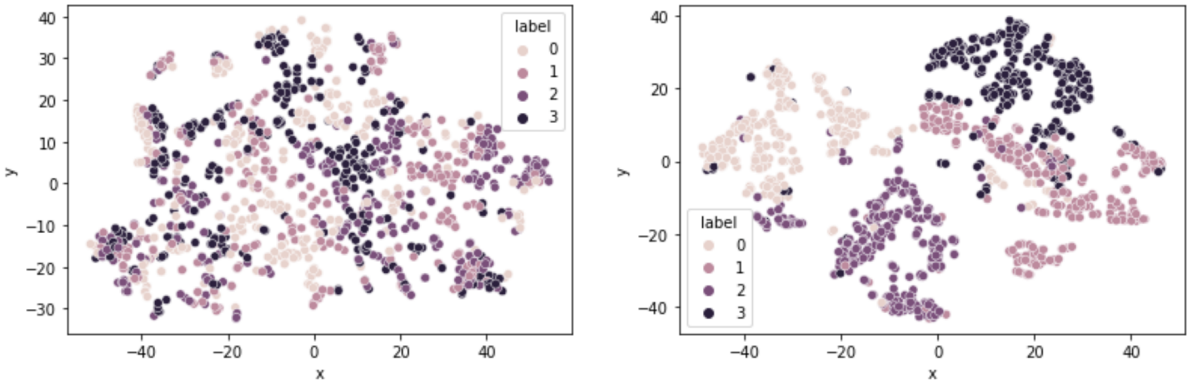
Pour aller plus loin, nous pouvons visualiser le degré de séparabilité des caractéristiques extraites en utilisant un outil de projection 2D des données : le *t-Distributed Stochastic Neighbour Embedding* : `t-SNE`. C'est une technique qui a pour but de proposer une visualisation des données de grande dimension en donnant à chaque point N-dimensionnel, un emplacement dans une carte en deux ou trois dimensions qui respecte localement les structures. Ici nous avons choisi de travailler avec les données représentées par la fusion des 3 encodages décrits par des vecteurs de dimension 30.



(a) BT (b) AT4

FIGURE 5.6 – Matrice de confusion en % obtenue en comptabilisant et en classant les différentes prédictions calculées via le kNN par rapport à la vérité terrain. Cela permet de voir les erreurs classe par classe.

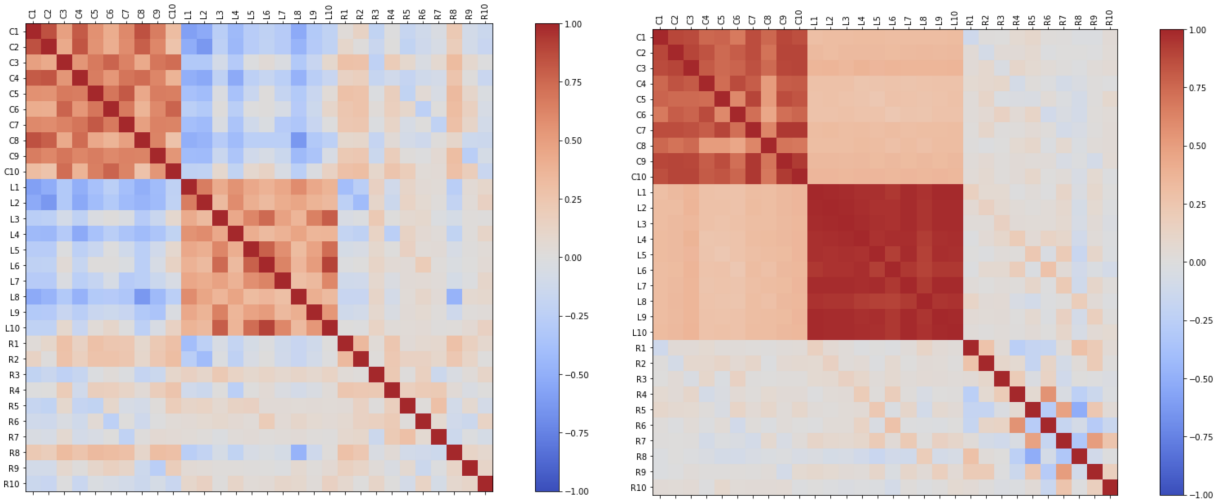
Les résultats, présentés dans la FIGURE 5.7, montrent que les données issues du réseau AT4 forment des clusters organisés en fonction de leur appartenance à une classe de mouvement. De plus, le nombre de clusters identifiés correspond au nombre de classes à distinguer. Si bien qu'on peut facilement les séparer via une simple régression linéaire ou un kNN dont la décision se base sur un petit nombre de voisins k. A contrario, les données résultant du réseau BT suivent une distribution beaucoup plus désorganisée et aucune structure évidente n'apparaît dans les données. Cela explique la performance moindre des classifieurs avec ce jeu de données.



(a) BT (b) AT4

FIGURE 5.7 – Visualisation t-SNE de l'ensemble des données BT (a) et AT4 (b). Cet affichage permet de faire une projection des vecteurs de caractéristiques dont la dimension est de 10 dans un plan en 2 dimensions. Cela permet d'avoir une idée des distances entre chaque point.

Les matrices de corrélation appliquées à chaque composante individuelle des différents codes neuronaux sont présentées dans la FIGURE 5.8. Une forte corrélation peut être observée pour les vecteurs utilisant les schémas de décodage "*spike count*" et "*latency*", tandis que les codes issus du "*rank order*" sont moins corrélés.



(a) BT (b) AT4

FIGURE 5.8 – Matrices de corrélation pour les 3 schémas de codage sur les caractéristiques issues de BT (a) et AT4 (b) avec C = *spike count*, L = *latency* et R = *rank order*. De fortes corrélations sont à souligner entre les caractéristiques issues des schémas de codage "*spike count*" et "*latency*" (pour les deux ensembles de données BT et AT4). Ceci indique de grandes similarités dans les codes des 10 neurones de sortie, ce qui rend plus difficile la distinction entre les classes. Par contre, les caractéristiques issues du codage "*rank order*" sont faiblement corrélées. Soulignons que, pour la matrice AT4, on distingue une corrélation négative, ce qui témoigne que les neurones de sortie sont devenus sélectifs.

5.3.2 Apprentissage incrémental : adaptation du réseau lors de la variation du nombre de classes

Nous allons maintenant nous intéresser à l'apprentissage incrémental qui est le cœur de la contribution de cette section. Nous menons des expérimentations permettant de démontrer que notre modèle est capable d'apprendre de nouvelles classes en maintenant la STDP lorsqu'on lui présente des données inédites. La raison principale pour laquelle nous pensons que cela est possible est que la STDP est une méthode d'apprentissage locale. De cette manière, la mise à jour des poids peut être ciblée sur une partie spécifique du réseau sans modifier le reste. Pour rappel, la STDP détecte les corrélations temporelles entre les spikes et renforce ainsi les connexions entre les neurones dont les événements se succèdent. Le mécanisme de compétition entre les neurones accélère le phénomène de spécialisation de chaque neurone. Après avoir entraîné le réseau sur une classe, nous pouvons identifier les sous-structures du réseau qui réagissent lors d'une stimulation induite par les données de cette classe. Lorsqu'une nouvelle classe est présentée, de nouvelles corrélations seront détectées par de nouvelles sous-structures et seront agrégées aux anciennes afin d'avoir une représentation discriminante pour la nouvelle classe. Nous changeons donc de protocole expérimental, puisque cette fois-ci le nombre de classes varie lors de l'entraînement. Concrètement, dans un premier temps, les poids du réseau sont initialisés de manière aléatoire. Puis des étapes d'apprentissages séquentielles sont appliquées pour chaque classe individuellement, avec un test à la fin de chacune de ces étapes. Pendant les phases d'entraînement, une séquence de 20 échantillons est présentée au réseau. Après une période d'entraînement, les poids synaptiques sont figés. On soumet alors au réseau un ensemble de données de test contenant les 4 classes afin de mesurer les performances.

Les spikes générés lors des simulations sont enregistrés pour créer des vecteurs de caractéristiques utilisés par un classifieur afin d'évaluer la qualité de ces derniers. Les poids sont alors modifiés de nouveau par la STDP pendant une nouvelle étape d'apprentissage. Nous répétons ce processus jusqu'à ce que le réseau ait appris l'ensemble des classes. Au total, 5 enregistrements sont collectés, correspondant aux différents états du réseau. Ces enregistrements incluent l'état initial aléatoire avant l'apprentissage (BT), ainsi que l'état du réseau après chaque étape d'entraînement (AT1, AT2, AT3 et AT4), lorsque chaque classe est présentée séparément. En raison de l'initialisation aléatoire, ce protocole est répété 10 fois avec différentes valeurs de germe pour le générateur de nombres aléatoires, dans le but de réduire la variabilité des résultats obtenus. Les vecteurs de caractéristiques extraites par le SNN sont classifiés via le MLP déjà utilisé précédemment.

Les résultats nous permettent d'évaluer plusieurs facteurs propres au protocole expérimental. Le premier facteur porte sur les différents schémas de codage. Le but est d'isoler et de quantifier l'impact des schémas sur les performances de classification du réseau. Il s'agit de déterminer si une représentation est plus appropriée dans un contexte d'apprentissage incrémental en vérifiant si elle répond à plusieurs critères essentiels. En effet, les caractéristiques principales permettant de distinguer les classes déjà apprises doivent être conservées par la représentation, même après l'évolution de l'activité en sortie du réseau suite à un entraînement sur de nouvelles classes. Cependant, il est nécessaire que la représentation soit expressive pour permettre l'intégration de caractéristiques liées à de nouvelles classes apprises. Il est ainsi possible de déterminer s'il existe un encodage plus pertinent qu'un autre grâce à cette analyse. Le deuxième facteur lié au protocole

expérimental est centré sur les performances du modèle au cours de ces différentes étapes d'apprentissage.

Concernant les différents codages, une différence notable est constatée quant à l'efficacité du classifieur pour prédire la bonne classe. En effet, l'utilisation de caractéristiques codées avec une stratégie fréquentielle (C) est moins efficace que l'utilisation de caractéristiques codées avec une stratégie temporelle (L et R). Les résultats présentés dans la FIGURE 5.9 ainsi que dans le TABLEAU 5.6 des scores de performances indiquent que le codage temporel offre de meilleurs résultats. De plus, il est à noter qu'il existe une légère différence entre les deux schémas temporels. Les observations montrent que le "*rank order*" est plus efficace que le codage "*latency*". Ceci est illustré par la FIGURE 5.9. En utilisant la représentation "*rank order*", on observe que les performances augmentent lorsqu'on introduit une nouvelle classe sans qu'il y ait de diminution pour les classes précédemment apprises. C'est une tendance qu'on observe aussi pour les stratégies de décodage "*latency*" et "*fusion*", avec un cas particulier concernant AT3. En effet, on observe une légère baisse entre l'introduction de la 2eme classe et de la 3ème classe. Si on se réfère aux classes en question, on présente d'abord des mouvements avec des directions verticales puis des mouvements ayant des directions horizontales. La stimulation de nouvelles régions dans le champs récepteur du réseaux va activer la STDP sur des synapses ayant des poids très peu modifiés. Pendant cette période, l'état du réseau est moins stable.

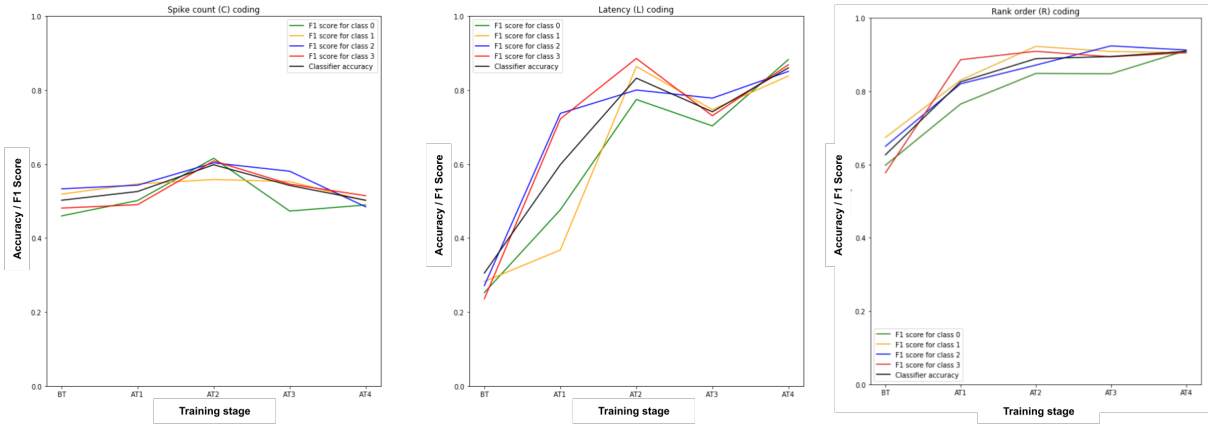


FIGURE 5.9 – Scores F1 pour différentes stratégies de décodage.

	BT		AT1		AT2		AT3		AT4	
accuracy	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
C : "*spike count*"	0.51	0.013	0.55	0.014	0.59	0.014	0.57	0.012	0.51	0.012
L : "*latency*"	0.32	0.019	0.61	0.013	0.82	0.013	0.77	0.012	0.88	0.013
R : "*rank order*"	0.62	0.012	0.82	0.013	0.89	0.013	0.90	0.012	0.93	0.012
F : "*fusion*"	0.73	0.012	0.90	0.013	0.94	0.012	0.93	0.012	0.92	0.014

TABLEAU 5.6 – Scores F1 pour les différents schémas de codage. Les variables μ et σ, sont respectivement, la moyenne et l'écart-type du score F1 sur 10 exécutions.

Afin d'évaluer la capacité d'apprentissage incrémental du modèle, le classifieur permettant d'estimer la qualité des vecteurs de caractéristiques extraites par le SNN utilise des vecteurs fusionnant les 3 stratégies de codage de l'information (stratégie F : "*fusion*"). En se penchant sur les résultats illustrés dans la FIGURE 5.10, nous pouvons voir qu'en introduisant de nouvelles classes, le réseau augmente son efficacité globale sans altérer les performances sur les classes précédemment apprises. Le graphique de gauche affiche le score F1 de chaque classe ainsi que la précision globale du classifieur pour un problème à deux classes (une classe contre toutes les autres, où les 3 autres classes sont artificiellement regroupées en une seule). Le graphique du milieu montre les résultats d'un problème à 3 classes (deux vs le reste). Enfin, le graphique de droite montre les résultats de la tâche de classification complète avec les 4 classes. Globalement, les résultats indiquent que les performances du réseau augmentent lorsqu'une nouvelle classe est introduite. L'hypothèse d'apprentissage incrémental pour ce type d'architecture appliquée à ce problème de vision est ainsi validée.

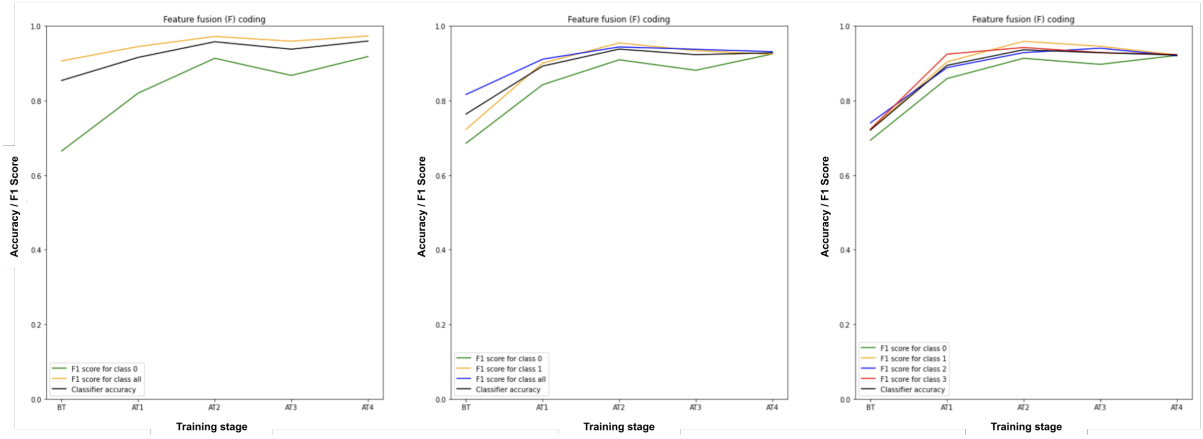


FIGURE 5.10 – Scores F1 et précision globale du classificateur pour la fusion de caractéristiques.

5.4 Discussion

Cette étude montre que la règle STDP peut être utilisée dans le cadre d'un apprentissage incrémental sur des SNNs. Nous avons abordé un problème de classification de mouvements en élaborant un modèle inspiré des cellules ganglionnaires, présentent dans les rétines des mammifères et qui sont sensibles à différentes directions lorsqu'il y a du mouvement. Dans le CHAPITRE précédent (4), nous avons démontré la capacité d'un SNN feed-forward mono-couche à extraire des caractéristiques pertinentes pour la classification des données de mouvement, de manière non supervisée. Nous avons montré que le réseau peut apprendre et séparer efficacement 4 classes d'entrée.

Dans ce chapitre, nous avons étudié la capacité d'adaptation de ce SNN à un nombre de classes qui varie dans le temps. Nous avons entraîné le modèle, en introduisant séquentiellement les classes lors de différentes phases d'apprentissage. Chaque classe est présentée séparément lors d'une étape d'apprentissage du modèle. Nous avons pu évaluer les capacités du réseau à conserver les caractéristiques acquises lors des précédents entraî-

nements, tout en continuant à apprendre à reconnaître de nouvelles classes. Les résultats montrent que le réseau est capable d'apprendre à distinguer correctement un nombre variable de classes. En outre, nous avons comparé 3 stratégies de décodage, décrivant les caractéristiques extraites par le SNN. Les résultats indiquent que le décodage "*rank order*" permet d'atteindre des performances supérieures aux autres stratégies "*latency*" et "*spike count*". Lors de nos expérimentations, nous avons remarqué que les résultats se détérioraient lors de l'introduction de la troisième classe. Cet effet était causé par l'ordre de présentation des différentes classes lors des étapes d'apprentissage du modèle. Les classes de mouvements verticaux sont présentées avant les classes de mouvements horizontaux. Une conséquence est que de nouvelles régions du réseau sont stimulées, induisant un état moins stable lors des premières étapes de modulation de la STDP. De plus, le problème de classification présenté comporte 4 classes que nous pouvons catégoriser dans 2 groupes en se basant sur des similitudes partagées par certaines classes (mouvement horizontal et vertical). Apprendre à distinguer l'un des groupes permet indirectement de développer des compétences sur la reconnaissance de l'autre groupe, sachant que ces deux groupes représentent toutes les classes du problème. Nous avons observé que le décodage "*rank order*" est la seule stratégie qui permet de conserver les performances du réseau sur les anciennes classes apprises, même lors de l'introduction de la troisième classe (AT3). Ce résultat représente un cas idéal d'apprentissage incrémental sur un problème de classification. En analysant l'activité de sortie du réseau, la distribution des impulsions change suffisamment entre AT2 et AT3 pour les classes déjà maîtrisées. Cette variation est plus impactante avec un décodage "*latency*" ou "*spike count*". Par contre le décodage "*rank order*" est plus robuste pour ce type de variation.

Une amélioration importante de ce travail consistera à classer les données d'entrée directement avec le SNN, sans avoir recours à un autre classificateur de réseau neuronal. Plusieurs possibilités sont explorées, l'une d'entre elles étant le développement d'une deuxième couche SNN dont l'apprentissage utilise une variante supervisée de la règle STDP (Mozafari et al. 2018).

Les travaux futurs seront de combiner ses différentes approches sur des jeux de données événementiels réels, en concevant des architectures plus complexes. Le modèle présenté jusqu'à maintenant peut être vu comme un détecteur local élémentaire, sensible aux motifs spatio-temporels avec des capacités d'apprentissage rapide, incrémental et non supervisé. La combinaison de ces détecteurs soulève de nouvelles questions sur la topologie du réseau (récurrents, multi-couches, connexions convolutives, etc.) et peut être envisagée comme une piste intéressante.

Ces travaux devront explorer pleinement les capacités des SNNs à traiter des informations temporelles afin de rivaliser avec les réseaux de neurones profonds, bien meilleurs dans les tâches de reconnaissance de motifs spatiaux. Le traitement de données comme la vidéo peut donc être à l'avantage des SNNs. L'analyse du mouvement sur des données réelles et non contrôlées pose plusieurs difficultés. Il faut être capable de pouvoir discerner la direction du mouvement d'un objet malgré le fait qu'il ait une vitesse plus ou moins élevée selon la situation. Afin de remédier à ce problème, une piste serait de s'inspirer des détecteurs de Reichardt. Ce modèle permet de discerner la direction d'un mouvement en utilisant des délais synaptiques. Pendant l'apprentissage, l'idée serait de modifier les délais et non les poids synaptiques, permettant d'avoir une plus grande sélectivité dans le nombre de motifs spatio-temporels caractérisables par le réseau.

Chapitre 6

Apprentissage des délais synaptiques

6.1 Motivations

Les réseaux de neurones artificiels (RNA) utilisent une méthode d'apprentissage qui implique la modification itérative des poids synaptiques pour amener le réseau dans un état qui lui permet de résoudre une tâche donnée. Les poids synaptiques permettent de moduler l'impact des entrées d'un neurone en le rendant sélectif à certains stimuli. Les neurones dans les RNA classiques n'évoluent pas dans le temps, rendant leur sélectivité purement spatiale. En revanche, les neurones impulsionnels possèdent une dimension temporelle, ce qui leur permet d'être sensibles à l'information temporelle. Le seuil d'activation et la fuite du neurone sont les deux paramètres qui régulent la sensibilité des neurones à un ensemble de motifs spatio-temporels. Par ailleurs, les SNNs disposent d'une autre propriété paramétrique qui permet l'évolution des connexions dans le domaine temporel uniquement, à savoir les délais de transmission des impulsions. Les délais synaptiques sont des propriétés qui permettent d'ajouter un degré de liberté supplémentaire dans la caractérisation des motifs spatio-temporels. En choisissant soigneusement les délais, on peut se retrouver avec des détecteurs sensibles à des caractéristiques liées au mouvement. Plusieurs travaux utilisent ce principe, et deux méthodes sont généralement employées pour mettre en place des délais synaptiques. La première méthode consiste à fixer les délais synaptiques manuellement. Le modèle que nous avons proposé utilise cette méthode puisque les délais ont été fixés par des valeurs aléatoires appartenant à un petit intervalle. Des travaux proposent de multiplier les synapses afin de pouvoir appliquer une gamme de délais prédéfinis (Orchard et Etienne-Cummings 2014 ; Paredes-Vallés, Scheper et De Croon 2019). Pour finir, il existe des modèles mathématiques qui s'inspirent des systèmes visuels biologiques, notamment pour la perception de mouvement et qui peuvent facilement être implémentés avec des SNNs. C'est le cas des détecteurs de Reichardt (SECTION 2.2.5.2), inspirés par l'étude de la réponse optomotrice du coléoptère Clorophanus (Reichardt 1961). Ce modèle calcule la corrélation entre le signal d'un photorécepteur et le signal retardé d'un photorécepteur voisin. En choisissant des délais synaptiques adéquats, un neurone se comporte comme un détecteur sensible à un mouvement du signal dans une direction et une vitesse préférées.

La deuxième méthode consiste à apprendre les délais synaptiques (*delay shift*) du réseau. La règle d'apprentissage réduit le délai de transmission entre deux neurones lors-

qu'une corrélation est détectée (Eurich et al. 1999 ; Nadafian et Ganjtabesh 2020). Les neurones ayant déchargé juste avant le spike du neurone post-synaptique verront alors leurs délais augmenter afin de s'approcher d'une synchronie et ceux ayant déchargé peu de temps après verront leurs délais diminuer. Cette règle est similaire à une règle anti-STDP (la magnitude a une valeur inverse lors d'une LTP/LTD) et suit les principes d'apprentissage hébbien. La FIGURE 6.1 permet de visualiser les modulations appliquées aux délais synaptiques par la règle *delay shift*.

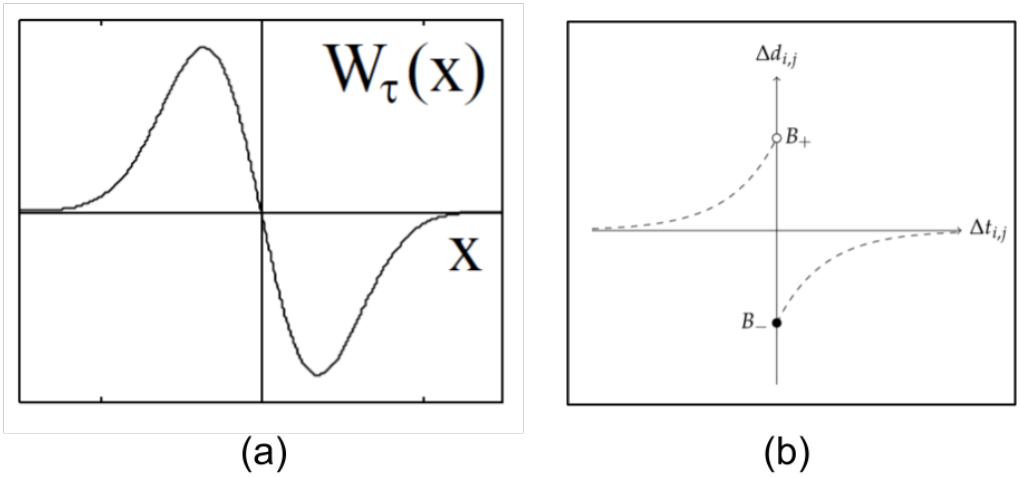


Figure 6.1 – Règle d'apprentissage permettant de moduler les délais. (a) Eurich et al. 1999. (b) Nadafian et Ganjtabesh 2020.

6.2 Objectifs de l'étude de cas

Ce chapitre présente un contenu différent par rapport aux deux autres études précédentes. En effet, nous proposons d'analyser le potentiel des délais synaptiques au travers d'une modélisation qui est présentée sous forme d'une étude de cas. Cette étude s'articule autour d'exemples simples (pixel en mouvement sur une fenêtre 1D) afin d'isoler différents concepts clés permettant de concevoir un modèle performant sur des caractéristiques spatio-temporelles. Nous avons décomposé notre étude en deux grandes parties. La première partie porte sur la modélisation d'un détecteur élémentaire sensible au mouvement. Pour établir les conditions de sélectivité d'un neurone sur des motifs spatio-temporels par rapport aux délais, nous confrontons ce modèle à différents types de stimuli provoqués par du mouvement. Nous faisons évoluer le modèle de manière itérative en ajoutant progressivement de nouveaux mécanismes (SECTION 6.3). Dans la deuxième partie, nous présentons une règle d'apprentissage non supervisé basée sur les délais en étudiant la convergence du modèle vers un détecteur identifié dans la première partie (SECTION 6.4).

La modélisation des détecteurs se base, initialement, sur le modèle de Reichardt. Ce type de détecteur est capable de se spécialiser dans la reconnaissance d'un motif spatio-temporel précis, défini par une vitesse et une direction. Pour modéliser ce détecteur, nous avons utilisé un neurone à impulsions et des connexions synaptiques dont les délais de

transmission sont paramétrables (SECTION 6.3.1). Lors de la modélisation des détecteurs élémentaires, nous avons, progressivement, complexifié le modèle, en ajoutant un mécanisme de fuite. Nous avons modélisé deux versions de la fuite du potentiel d'un neurone :

— La première version du mécanisme de fuite (binaire) est caractérisée par une durée définie pendant laquelle le potentiel de membrane du neurone demeure inchangé. Une fois cette durée écoulée, la tension du neurone est immédiatement remise à sa valeur de repos. En utilisant ce comportement, nous pouvons étudier et identifier plus facilement les motifs spatio-temporels auxquels le détecteur peut être sensible en contrôlant un seul paramètre. Cette approche nous permet de mieux comprendre l'impact de la fuite et du délai sur la sélectivité d'un motif, en isolant les autres paramètres du modèle, tels que le poids synaptique ou le seuil d'activation. Cela nous permet d'établir des contraintes qui encadrent le réglage des détecteurs sensibles à une variété de motifs spatio-temporels (SECTION 6.3.2) ;

— La deuxième version du mécanisme de fuite (linéaire) se caractérise par une décroissance linéaire du potentiel de membrane du neurone. Cette approche nous permet d'inclure l'influence des poids synaptiques et du seuil d'activation du neurone dans la sélectivité du détecteur pour les motifs spatio-temporels. Nous pouvons ainsi affiner les contraintes établies à partir de la modélisation de la première version du détecteur avec fuite binaire (SECTION 6.3.3).

En reprenant l'expression mathématique d'un neurone LIF, rappelée formellement par l'ÉQUATION (6.1) :

$$\frac{dv}{dt}(t) = -\frac{1}{\tau_{leak}}v(t) + u(t), v \leftarrow V_{rest} \text{ quand } v \geq V_{thres}. \tag{6.1}$$

où le potentiel de membrane v évolue au cours du temps en intégrant le courant d'entrée u. Lorsque la tension atteint le seuil d'activation V_{thres} le neurone émet un spike, puis la tension revient à la valeur de repos V_{rest}. La fuite est modélisée par une décroissance exponentielle et est réglée par le paramètre τ_{leak}.

Pour la première version du mécanisme de fuite, qui implique une décroissance binaire du potentiel de membrane, le comportement du neurone est formalisé par l'ÉQUATION (6.2) suivante :

$$\frac{dv}{dt}(t) = \begin{cases} u(t) & \text{si } t - t_{last} < \tau_{leak} \\ -v(t) - V_{rest} & \text{sinon} \end{cases} \tag{6.2}$$

où t_{last} est le moment du dernier spike entrant.

La seconde version du mécanisme de fuite, qui se caractérise par une décroissance linéaire du potentiel de membrane, est décrite par l'ÉQUATION (6.3) suivante :

$$\frac{dv}{dt}(t) = u(t) - \frac{1}{\tau_{leak}}, v \leftarrow V_{rest} \text{ quand } v \geq V_{thres}. \tag{6.3}$$

La FIGURE 6.2 illustre l'évolution du potentiel de membrane d'un neurone en réponse à des impulsions entrantes, en se basant sur la modélisation de trois types de fuites : fuite exponentielle, fuite binaire et fuite linéaire.

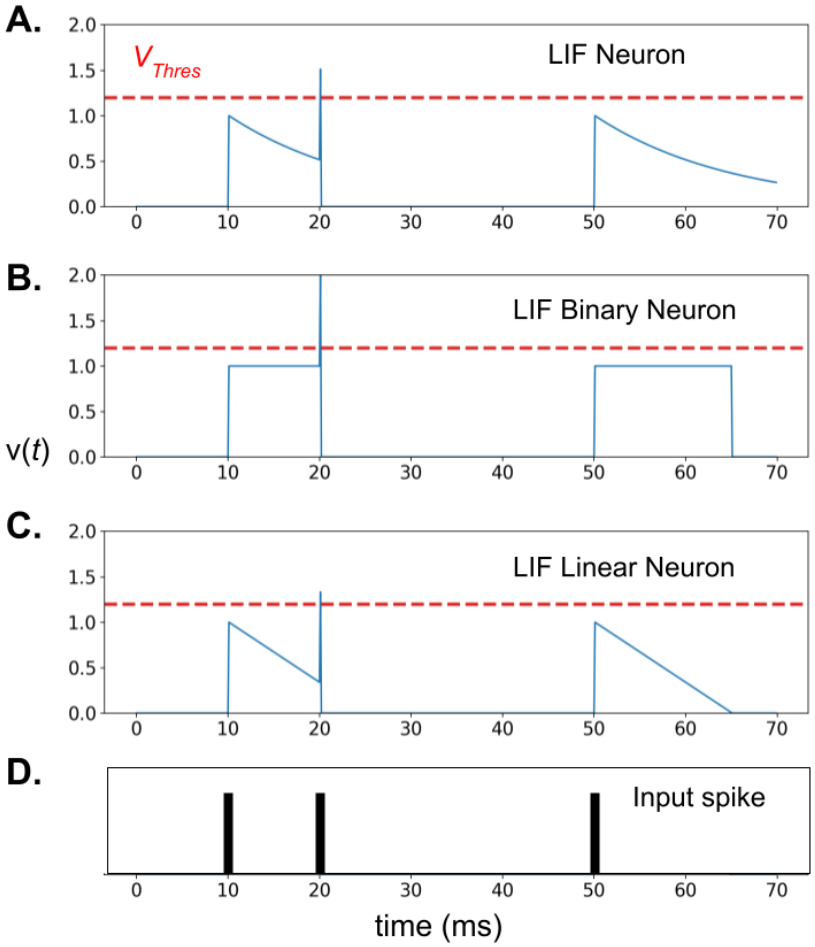


FIGURE 6.2 – Représentation de l'évolution du potentiel de membrane de différents modèles de neurones LIF en fonction de la modélisation du mécanisme de fuite. Les panels **A.**, **B.** et **C.** illustrent respectivement l'évolution du potentiel de membrane avec une décroissance exponentielle, binaire et linéaire. Le panel **D.** montre l'activité en entrée qui stimule les différents neurones LIF.

En identifiant différents types de détecteurs élémentaires sensibles au mouvement, nous pouvons établir un lien entre le type de sélectivité souhaité pour un ensemble de motifs spatio-temporels et les différents paramètres qui régulent ces détecteurs. Cette approche nous permet d'élaborer une règle d'apprentissage qui prend en considération tous ces éléments (SECTION 6.3).

6.3 Modélisation de détecteurs élémentaires

Dans cette section, nous allons modéliser le comportement d'un neurone impulsionnel similaire à un détecteur de Reichardt, capable de détecter un mouvement spécifique défini par une direction et une vitesse données. Pour faciliter les raisonnements, le neurone n_1 est connecté à une fenêtre 1D de 2 pixels et possède deux poids synaptiques (w_1, w_2) et deux délais synaptiques (d_1, d_2). Dans le scénario illustré par la FIGURE 6.3, un seul pixel se déplace soit vers le haut, soit vers le bas. Nous allons introduire progressivement les mécanismes d'un neurone LIF, en simplifiant certains d'entre eux pour établir un lien entre les différents paramètres qui définissent la sélectivité d'un neurone pour des motifs spatio-temporels. Les paramètres qui influencent la sélectivité d'un neurone sont le seuil d'activation V_{thres}, la fuite du neurone τ_{leak}, les poids synaptiques (w_1, w_2) et les délais synaptiques (d_1, d_2). Ces liens peuvent éventuellement nous permettre de définir une ou plusieurs règles d'apprentissage qui optimisent conjointement ou non tous les paramètres afin d'avoir un contrôle précis sur la sélectivité d'un neurone. Toutefois, il est important de souligner que l'optimisation simultanée de tous ces paramètres est un problème NP-difficile, ce qui en fait un objectif ultime difficile à atteindre (Paugam-Moisy et Bohte 2012).

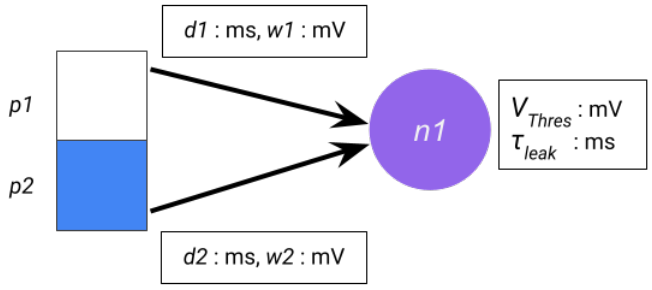


Figure 6.3 – Topologie retenue dans notre modélisation. Une fenêtre 1D composée de 2 pixels connectés à un neurone n_1.

6.3.1 Détecteurs stricts

Dans cette situation, où les détecteurs sont stricts, les délais sont bien réglés, les impulsions arrivent exactement au même moment. Le TABLEAU 6.1 donne les valeurs des différents paramètres pour cette configuration. En se référant à la FIGURE 6.4, on voit que la valeur du seuil d'activation V_{thres} oblige une synchronisation parfaite des spikes en entrée pour que le neurone n_1 se déclenche. Afin de décrire au mieux les situations, l'EQUATION (6.4) définit certaines variables du problème :

$$\begin{aligned} \Delta D &= d_2 - d_1 \\ t_{mvt} &= t_2 - t_1 \\ lat &= t_{mvt} - \Delta D \end{aligned} \tag{6.4}$$

Avec ΔD la différence entre les délais des synapses des neurones $n1$ et $n2$. La variable ΔD permet de connaître la sensibilité du neurone sur la direction du mouvement en

connaissant le signe de sa valeur, la vitesse de déplacement étant déduite avec sa valeur. La valeur de lat permet de mesurer la différence entre le temps du mouvement du pixel et le temps du mouvement attendu par le détecteur. Enfin t_{mvt} permet de connaître la vitesse de déplacement du pixel. Si le détecteur est strict sur le mouvement auquel il est sensible alors $lat = 0$.

Pour obtenir cette sélectivité spécifique, il est impératif de répondre aux contraintes définies par l'EQUATION (6.5) :

$$\begin{aligned} \tau_{leak} &< t_\epsilon \\ V_{thres} &> w_1 \\ V_{thres} &> w_2 \\ V_{thres} &\leq w_1 + w_2 \end{aligned} \tag{6.5}$$

avec t_ϵ étant le pas de temps minimal d'un mouvement. Pour que le neurone soit sensible au mouvement défini par ΔD, il est crucial que sa fuite de potentiel soit plus rapide que la vitesse maximale possible d'un mouvement. Dans le cas contraire, il y a un risque que le neurone détecte un mouvement trop rapide alors que sa tension n'a pas encore atteint son potentiel de repos, ce qui provoquerait un déclenchement non lié au mouvement qu'il est censé détecter. Par ailleurs, il est essentiel que les poids synaptiques w_1 et w_2 soient inférieurs au seuil V_{thres} pour éviter qu'un seul poids ne déclenche le neurone. Par contre, il est important que la somme de ces poids puisse suffire à déclencher le détecteur.

Paramètres	$d1$	$d2$	$w1$	$w2$	V_{thres}	τ_{leak}
Valeurs	2 ms	0 ms	1 mV	1 mV	2 mV	0 ms

TABLEAU 6.1 – Valeurs des paramètres utilisées lors du premier cas illustré par la FIGURE 6.4.

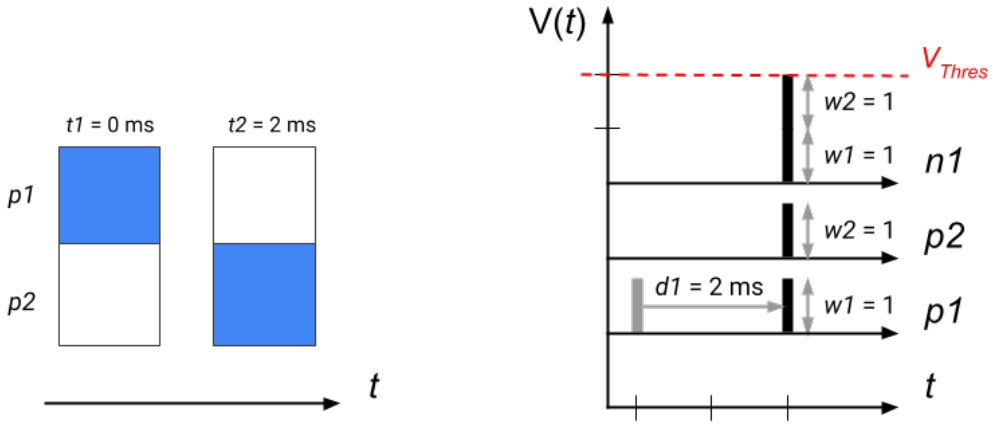


FIGURE 6.4 – Le détecteur est strict, il faut que le mouvement corresponde au réglage du détecteur pour qu'il soit détecté.

6.3.2 Détecteurs variables : fuite binaire

Dans cette configuration, le neurone $n1$ possède une fuite binaire, ce qui signifie que son potentiel reste constant pendant toute la durée de la fuite. À la fin de cette durée, le potentiel du neurone revient à sa valeur de repos. Ce type de modélisation nous permet de déterminer l'impact maximal de la fuite sur la sélectivité du neurone pour les motifs spatio-temporels, par rapport aux autres paramètres. Dans cette situation, les paramètres ont les mêmes valeurs que précédemment (voir TABLEAU 6.1), à l'exception de la fuite $\tau_{leak} = 0.5$. La FIGURE 6.5 illustre deux situations différentes avec le même détecteur. Dans la première situation, on présente au détecteur une séquence où un pixel se déplace dans la mauvaise direction. Le neurone ne se déclenche pas. Dans la deuxième situation, on présente au détecteur une séquence où un pixel se déplace dans la bonne direction, mais à une vitesse inférieure $lat = 0.3$. Malgré cela, le neurone se déclenche, car le réglage de la fuite permet d'ajouter une certaine tolérance sur la vitesse préférée par le détecteur (indiquée par ΔD). Cette tolérance est donc relative aux délais synaptiques.

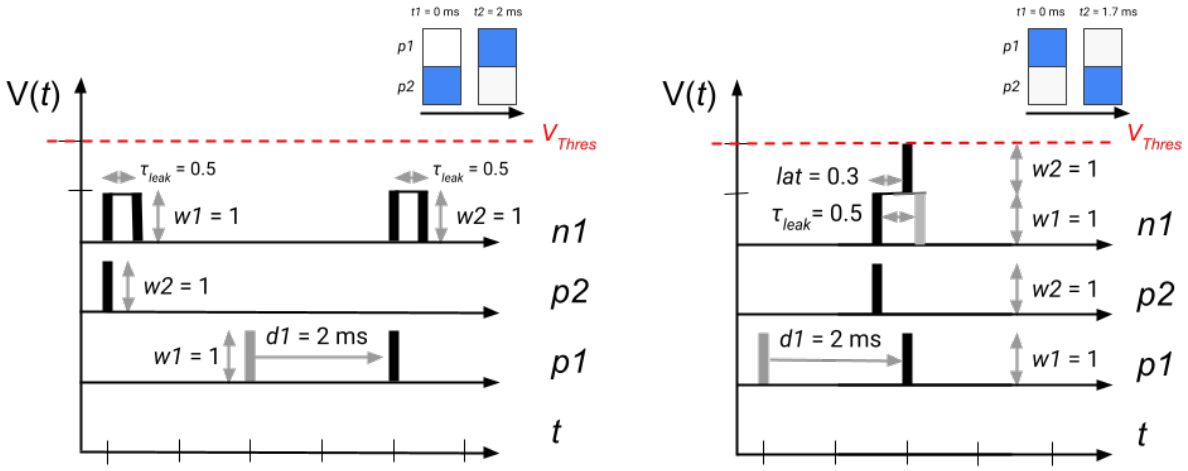


FIGURE 6.5 – Exemple de détecteur non strict. À gauche : Un mouvement est présenté dans une direction opposée aux réglages du détecteur. À droite : Un mouvement est présenté avec une vitesse inférieure à la vitesse préférée du détecteur, mais cela suffit à déclencher le détecteur.

Avec la première situation, nous pouvons constater qu'il est nécessaire de limiter l'intervalle de la fuite pour éviter que le détecteur ne puisse plus distinguer les directions. Pour cela, il est impératif que la valeur de la fuite respecte la contrainte suivante décrite par l'ÉQUATION (6.6) :

$$\tau_{leak} < t_{mvt} + |\Delta D| \tag{6.6}$$

6.3.3 Détecteurs variables : fuite linéaire

Nous allons maintenant remplacer le mécanisme de fuite précédent par un mécanisme de fuite linéaire, pour se rapprocher du comportement d'un neurone LIF. Ce nouveau

modèle nous permet de mesurer l'impact des poids synaptiques, car la fuite évolue en continu, et des poids variables peuvent rendre le détecteur sélectif sur la vitesse du mouvement pendant la fuite du neurone. Dans cette configuration, nous pouvons donc lier les trois paramètres du modèle : les délais synaptiques, la fuite du neurone et les poids synaptiques. La FIGURE 6.6 illustre une situation dans laquelle on présente au détecteur une séquence avec un pixel se déplaçant dans la direction préférée du détecteur, mais avec un temps de déplacement légèrement plus court ($lat = 0.5$ ms).

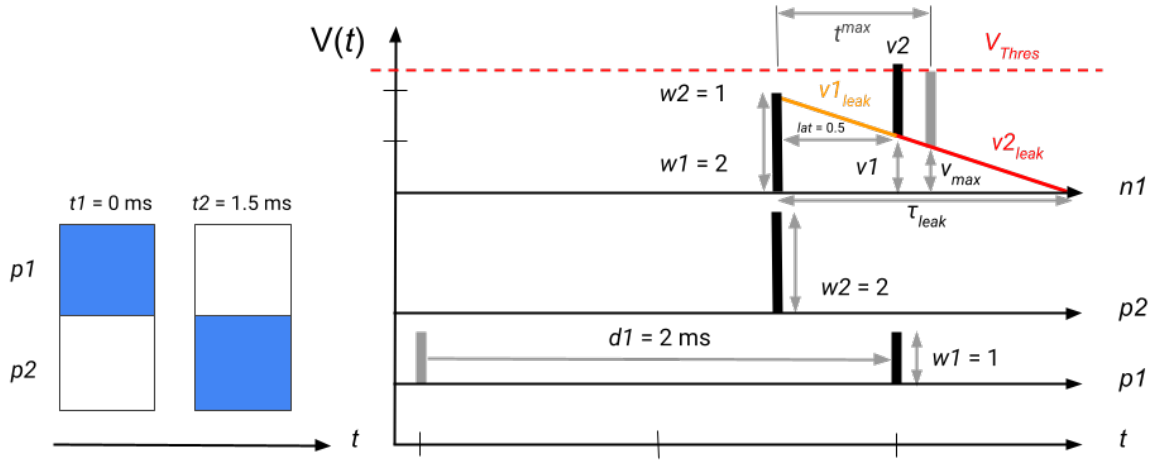


FIGURE 6.6 – Évolution du potentiel de membrane du détecteur avec une fuite linéaire. En adoptant un point de vue géométrique, nous pouvons établir un lien entre les différents paramètres en utilisant des outils tels que le théorème de Thalès.

Nous avons repris les contraintes sur les poids synaptiques décrites dans l'ÉQUATION (6.5). Ce sont les seules contraintes utiles pour définir ce détecteur, elles sont rappelées dans l'ÉQUATION (6.7) :

$$\begin{aligned} V_{thres} &\leq w_1 + w_2 \\ V_{thres} &> w_1 \\ V_{thres} &> w_2 \end{aligned} \tag{6.7}$$

Afin de simplifier les formules, certaines variables sont définies dans l'ÉQUATION (6.8) :

$$\begin{aligned} v_2 &= v_1 + w_2 \\ v_{tot} &= w_1 + w_2 \\ v_{max} &= v_{tot} - V_{thres} \end{aligned} \tag{6.8}$$

Avec la valeur v_2 qui représente la tension du potentiel de membrane du neurone déclenchée par le deuxième pixel. La valeur v_{tot} correspond à la tension totale possible résultant de la combinaison des poids w_1 et w_2. Enfin, la valeur v_{max} représente la tension supplémentaire possible par rapport au seuil d'activation du neurone V_{thres}.

En se basant sur la modélisation linéaire du mécanisme de fuite du potentiel illustrée dans la FIGURE 6.6, qui permet de simplifier les calculs par rapport au comportement

réel d'un neurone LIF, nous pouvons établir des liens entre les différents paramètres pour déterminer les variables de cette modélisation, en utilisant le théorème de Thalès. Plus précisément, l'ÉQUATION (6.9) nous permet de calculer la valeur de $v1$, qui correspond à la tension présente juste avant l'intégration du deuxième spike :

$$\begin{aligned} \frac{v_1}{w_1} &= \frac{v_2^{leak}}{v_2^{leak} - v_1^{leak}} = \frac{\tau_{leak} - lat}{\tau_{leak}} \quad \Leftrightarrow \\ \frac{v_1}{w_1} &= \frac{\tau_{leak} - lat}{\tau_{leak}} \quad \Leftrightarrow \\ v_1 &= \frac{\tau_{leak} - lat}{\tau_{leak}} \times w_1 \end{aligned} \tag{6.9}$$

Nous cherchons à déterminer une contrainte, de manière similaire au cas de la fuite binaire, qui permet de délimiter les paramètres du détecteur pour maintenir sa sélectivité à une direction et une vitesse plus ou moins variable. Dans notre cas, l'objectif est de trouver la valeur de t_{max}, qui représente la durée maximale pendant laquelle le neurone peut être déclenché à nouveau après avoir été stimulé. Pour calculer cette valeur, nous pouvons utiliser une fois de plus le théorème de Thalès. L'ÉQUATION (6.10) fournit une explication détaillée du raisonnement à suivre :

$$\begin{aligned} \frac{v_{max}}{w_1} &= \frac{t_{max}}{\tau_{leak}} \quad \Leftrightarrow \\ t_{max} &= \frac{v_{max}}{w_1} \times \tau_{leak} \end{aligned} \tag{6.10}$$

Il suffit de reprendre la contrainte $\tau_{leak} < t_{mvt} + |\Delta d|$ déterminée lorsque la fuite était binaire et de remplacer le terme τ_{leak} par t_{max}. L'ÉQUATION (6.11) décrit les différentes étapes permettant de définir cette contrainte :

$$\begin{aligned} t_{max} &< t_{mvt} + |\Delta D| \quad \Leftrightarrow \\ \frac{v_{max}}{w_1} \times \tau_{leak} &< t_{mvt} + |\Delta D| \quad \Leftrightarrow \\ \tau_{leak} &< \frac{(t_{mvt} + |\Delta D|) \times w_1}{v_{max}} \end{aligned} \tag{6.11}$$

La définition de cette contrainte établit un lien entre la vitesse du mouvement et les différents paramètres du détecteur tels que les délais, la fuite et les poids. Les différentes modélisations indiquent que la sélectivité du neurone est principalement définie par les délais qui permettent de régler la direction ainsi que la vitesse globale du mouvement qui doit être privilégiée. Une fois ce réglage effectué, la fuite du neurone ajoute une certaine tolérance à la vitesse du mouvement, et les poids modulent cette tolérance. Pour un motif temporel, la sélectivité du neurone à un motif temporel est déterminée, dans l'ordre, par : les délais $\Delta D \rightarrow$ la fuite du neurone $\tau_{leak} \rightarrow$ les poids w.

Le comportement des neurones LIF diffère légèrement car la fuite du potentiel suit une loi non linéaire décrite par une exponentielle. Néanmoins, notre contrainte permet de borner grossièrement les paramètres du LIF en prenant en compte le comportement non linéaire de la fuite du neurone, exprimé par ϵ_{leak} dans l'ÉQUATION (6.12).

$$\tau_{leak} + \epsilon_{leak} < \frac{(t_{mvt} + |\Delta D|) \times w_1}{v_{max}} \tag{6.12}$$

La comportement d'une fuite linéaire pouvant être plus lente $-\epsilon_{leak}$ ou plus rapide $+\epsilon_{leak}$ qu'une fuite exponentielle, le signe ϵ_{leak} varie.

Nous avons pu identifier des détecteurs élémentaires qui permettent d'augmenter ou diminuer la sélectivité selon certaines situations. Afin de généraliser le travail, il faudrait augmenter la taille de la fenêtre en vue d'ajouter la possibilité de détecter des motifs spatiaux. En effet, dans les situations traitées, seul un pixel se déplaçait, alors que dans les cas réels, c'est un ensemble de pixels qui sont en mouvement.

6.4 Règle d'apprentissage

Avec les détecteurs mis en évidence dans le SECTION 6.3, nous pouvons concevoir une règle d'apprentissage qui permet de générer, dans un premier temps, les détecteurs stricts suivant les corrélations établies avec les données en entrée fournies. En s'inspirant de la règle *delay shift* illustrée par la FIGURE 6.1, nous avons élaboré notre version définie par l'ÉQUATION (6.13) :

$$\begin{aligned} &\Delta t = t_{pre} - t_{post} \\ &\Delta d = \text{stdp-delay}(\Delta t) = \begin{cases} \Delta d_+ \times \exp \frac{\Delta t}{T_{CL}} & \text{si } \Delta t > T_{CL} \\ \Delta d_- \times \exp \frac{\Delta t}{T_{CL}} & \text{si } \Delta t < -T_{CL} \\ -\frac{\Delta d_-}{T_{CL}} \times \Delta t & \text{si } -T_{CL} < \Delta t < 0 \\ -\frac{\Delta d_+}{T_{CL}} \times \Delta t & \text{sinon} \end{cases} \end{aligned} \tag{6.13}$$

avec T_{CL} qui permet de contrôler la durée pendant laquelle la règle détecte une corrélation significative (LTP ou LTD), elle applique alors une modulation du délai avec une magnitude qui est linéairement proportionnelle à Δt. La FIGURE 6.7 illustre la forme de la règle.

L'objectif de cette expérience est d'analyser le mécanisme de la règle `stdp-delay` sur une configuration donnée. Nous allons utiliser un motif qui se déplace dans la direction du bas à une vitesse de 2 ms/pixel ΔD^{target}. Le TABLEAU 6.2 résume les valeurs des paramètres de la règle :

Paramètres	d_+	d_-	T_{CL}
Valeurs (ms)	0.1	0.1	0.5

TABLEAU 6.2 – Valeurs des paramètres utilisées pour la règle.

Nous allons tracer l'exécution de la règle `stdp-delay` dans cette situation afin de vérifier si les délais convergent vers un détecteur strict correspondant au mouvement cible lorsque celui-ci est sensible à une vitesse plus lente (+0.3) initialement. Le TABLEAU 6.3

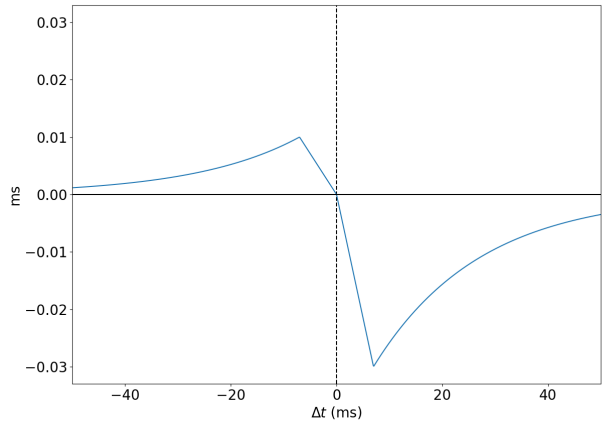


FIGURE 6.7 – Proposition d'une STDP modulant les délais synaptiques pour obtenir des détecteurs strictes.

détaille la trace de l'exécution de la règle `stdp-delay`, avec i le numéro de l'étape, t le temps courant et P le type de modification appliquée par la règle (LTD ou LTP). Nous allons présenter la séquence d'entrée au détecteur autant de fois que nécessaire pour que la règle fasse converger le détecteur vers un état stable, avec 10 ms entre chaque présentation :

i	t	P	ΔD	d_1	d_2	t^1_{pre}	t^2_{pre}	t_{post}	Δd_1	Δd_2
1	2.3	LTD	2.3	2.3	0	2.3	2.0	2.3	0.0	0.1
2	12.3	LTD	2.2	2.3	0.1	12.3	12.1	12.3	0.0	0.1
3	22.3	LTD	2.1	2.3	0.2	22.3	22.2	22.3	0.0	0.1
3	32.3	N/A	2.0	2.3	0.3	32.3	32.3	32.3	0.0	0.0

TABLEAU 6.3 – Trace de l'exécution de la règle `stdp-delay`.

La règle fait bien converger les délais vers un état qui permet d'obtenir un détecteur strict (SECTION 6.3.1) sensible au motif présenté. Il faut noter que cette règle permet d'obtenir le bon ΔD mais avec des délais $d1 = 2.3$ ms et $d2 = 0.3$ ms. Cependant, il existe une meilleure distribution, à savoir $d1 = 2.0$ ms et $d2 = 0.0$ ms qui permet de réduire les valeurs des délais du réseau et donc d'accélérer le traitement de l'information. Il faut donc ajouter une étape de régularisation des délais afin de les réduire afin de limiter la latence induite par les délais au sein du réseau.

À terme, le but est de pouvoir intégrer l'influence de la fuite du neurone et des poids synaptiques suivant certaines conditions afin de pouvoir converger vers une règle qui permet de spécialiser les neurones à des motifs spatio-temporels déterminés par les données d'entrée.

6.5 Discussion

Dans cette étude, nous avons proposé plusieurs modèles de détecteurs élémentaires sensibles au mouvement et avons déterminé les paramètres qui permettent de moduler leur sélectivité sur des motifs temporels simples. Nous avons constaté que les délais sont les principaux déterminants de la caractérisation d'un motif temporel précis (direction précise, vitesse précise), tandis que la fuite du neurone introduit une variabilité sur la vitesse du motif. Cette variabilité peut être modulée par les poids synaptiques, qui peuvent également être utilisés pour désactiver le détecteur. Ce constat s'explique par le fait que la diminution du potentiel de membrane d'un neurone est régulée par le paramètre de fuite, tandis que l'augmentation de ce potentiel en réponse aux signaux d'entrée est déterminée par les poids synaptiques.

La prochaine étape dans la modélisation des détecteurs élémentaires consiste à intégrer la dimension spatiale pour compléter le lien entre la sélectivité d'un motif spatio-temporel et les différents paramètres du détecteur. L'utilisation des poids synaptiques semble être la piste à privilégier.

De plus, nous avons proposé une règle d'apprentissage similaire au *Delay Shift* pour conduire un réseau neuronal vers un ensemble précis de détecteurs stricts. L'étape ultérieure consistera à prendre en compte les autres paramètres étudiés afin d'améliorer cette méthode d'apprentissage, de manière à favoriser l'émergence de nouveaux types de détecteurs. Dans les études de cas présentées, nous avons également souligné l'importance de régler le paramètre de fuite afin de préserver la sélectivité du détecteur pour les motifs temporels et éviter une trop grande permissivité. Ces résultats constituent une première avancée dans la compréhension de la modulation de la sélectivité des neurones sur des motifs temporels et pourraient avoir des implications importantes pour le développement de de systèmes de reconnaissance de mouvement et de vision artificielle.

Chapitre 7

Conclusion

La problématique générale abordée dans ce manuscrit est l'utilisation des RNAs bio-inspirés (les SNNs) en vision par ordinateur. Ce travail nous a permis d'explorer les avantages que peuvent offrir les SNNs par rapport aux approches basées sur des réseaux profonds de deuxième génération. La thèse défendue est motivée par le constat que, bien que ces méthodes soient très performantes, elles requièrent une grande quantité de données annotées et des ressources en calcul importantes, ce qui entraîne un coût énergétique élevé. Ainsi, les SNNs offrent des opportunités de recherche complémentaires par rapport aux développements des RNAs. La modélisation des SNNs se rapproche du modèle biologique du cerveau, dans lequel les neurones communiquent de manière asynchrone via des séquences d'impulsions et évoluent dans le temps. Ce type de réseau fonctionne donc avec une dimension temporelle, composé d'éléments qui peuvent traiter et communiquer l'information avec des opérations locales et parcimonieuses. Grâce à ces propriétés, les modèles basés sur les réseaux impulsionnels peuvent être implémentés et exécutés sur des architectures matérielles dédiées, très peu énergivores. De plus, les principes d'apprentissage hébbiens peuvent être mis en place sur ce type de réseaux, notamment à l'aide de méthodes bio-inspirées telles que la règle STDP. Cette dernière ne nécessite aucune supervision et conserve la propriété de localité, permettant ainsi une implémentation matérielle basse consommation. Par ailleurs, les SNNs reçoivent des entrées sous forme de séquences temporelles d'impulsions, qu'ils traitent en temps réel et en parallèle. Contrairement aux RNAs traditionnels, qui sont plutôt conçus pour traiter des données statiques et ne sont pas adaptés pour prendre en compte la dimension temporelle des données. Ainsi, les SNNs représentent une solution prometteuse pour le traitement de signaux audio, vidéo, etc. Cette caractéristique fondamentale offre des perspectives inédites sur ce type de données et peut être un complément essentiel dans le développement de systèmes embarqués temps réel, par exemple.

L'un de nos objectifs était de proposer un modèle capable d'exploiter les accélérations matérielles fournies par les puces neuromorphiques. Pour parvenir à ce résultat, notre modèle exécute ces opérations de manière locale. C'est pourquoi la règle d'apprentissage qui a été mise en œuvre pendant l'entraînement du réseau est une STDP additive. Ce choix n'a pas été motivé que par les aspects liés à la consommation énergétique. Le caractère non supervisé de l'apprentissage a été un critère prédominant dans l'élaboration de notre modèle. Nous voulions que notre modèle soit capable d'apprendre à détecter le mouvement sans supervision, avec peu d'exemples et en disposant d'une capacité d'adaptation

lorsqu'on lui présente des nouvelles classes.

Deux études ont été présentées dans ce manuscrit. Elles portent sur l'analyse du mouvement et plus particulièrement sur l'estimation de la direction du mouvement dans des données événementielles synthétiques. Ces travaux ont permis d'évaluer les capacités des SNNs à pouvoir extraire des caractéristiques spatio-temporelles en utilisant différents mécanismes offerts par ces réseaux. Une dernière partie, présentée comme une étude de cas, se concentre sur l'amélioration du modèle à caractériser des motifs spatio-temporels en explorant la possibilité d'apprendre les délais plutôt que les poids synaptiques.

Cette section du manuscrit est l'occasion de remettre en perspective les résultats obtenus et les différentes pistes qui nous semblent pertinentes à explorer à l'avenir. De plus, nous partageons notre vision globale sur les SNNs, les limites auxquelles ces modèles font face et les enjeux liés à leur développement futur.

7.1 Discussion générale des résultats

Dans le CHAPITRE 4, nous avons montré la possibilité d'entraîner un SNN de manière non supervisée à reconnaître les directions d'un mouvement induit par différents motifs dans des données événementielles synthétiques. Nous avons proposé un modèle de petite taille dont l'entraînement ne nécessite que quelques millisecondes de stimulation non supervisée pour atteindre une situation stable. De plus, même si nous poursuivons l'apprentissage, les résultats ne se dégradent pas. Ce dernier point nous a motivés à faire une étude sur la capacité d'apprentissage incrémentale du modèle (voir le CHAPITRE 5).

7.1.1 Sélectivité des motifs spatio-temporels

Dans les études présentées, la polarité des événements n'est pas prise en compte, ce qui force le réseau à se baser sur des caractéristiques temporelles uniquement. Les résultats montrent que la STDP permet de moduler les poids de façon à rendre certains neurones sensibles à des stimuli spécifiques.

Dans l'étude 4.3.6, nous avons cherché à déterminer la topologie du réseau la plus simple possible. Le but étant de caractériser des détecteurs élémentaires sensibles aux mouvements qui pourraient être utilisés par la suite dans des architectures plus complexes. Nous avons constaté qu'un réseau mono-couche feedforward était capable de reconnaître les différentes classes de mouvement, avec peu de neurones de sortie, voire dans certains cas, un nombre de neurones égal au nombre de directions possibles, chacun des neurones régissant distinctement à une classe (voir SECTION 4.3.4).

Si nous avions utilisé un RNA classique avec une topologie équivalente (mono-couche et pleinement connecté), la détection de caractéristiques temporelles n'aurait pas était possible, seule la dimension spatiale de l'information aurait pu être extraite. Par contre, avec les SNNs, une modulation des poids peut aboutir à une caractérisation des informations temporelles, puisque les neurones font évoluer leur état interne en fonction du temps. La conséquence est que les poids peuvent influencer de deux manières le réseau : une influence directe sur le choix du neurone à favoriser (filtrage spatial comme dans les RNAs) et une influence indirecte sur le moment où le neurone émet un spike. Nous parlons d'influence indirecte, car le déclenchement d'un neurone est régi par son état interne qui dépend de 2 facteurs : les précédentes stimulations entrantes et les paramètres du neurone

qui décrivent son comportement. Ces paramètres influencent le neurone sur sa sensibilité à distinguer l'information spatiale ou temporelle. Un neurone ayant une fuite rapide ne prendra que très peu en compte l'historique des impulsions qu'il a reçues alors qu'inversement, un neurone ayant une fuite lente prendra davantage en compte les événements passés. Selon le comportement du neurone, les poids peuvent représenter des caractéristiques spatiales ou temporelles (Hüning, Glünder et Palm 1998). C'est pourquoi le réglage des paramètres du SNN a des conséquences importantes sur les performances obtenues pour un problème donné. C'est l'une des principales constatations faites lors de l'étude (voir SECTION 4.4) sur le mouvement des motifs binaires. Le réglage du seuil d'activation du neurone va déterminer la complexité des motifs pour lesquels il est sensible. La complexité étant définie par le nombre d'impulsions entrantes nécessaires pour que le neurone se déclenche. Des valeurs de seuil trop grandes entraînent une sous-activation des neurones dans le réseau, rendant l'apprentissage presque impossible. En revanche, des valeurs trop faibles provoquent une sur-activation des neurones, une situation tout aussi préjudiciable puisque les impulsions ne sont pas suffisamment discriminantes et la STDP va détecter des corrélations biaisées.

En ajustant la fuite du neurone, on détermine le degré d'importance accordé entre les informations spatiales et temporelles qui caractérisent les motifs auxquels il est sensible. Une fuite trop rapide rendra le neurone insensible aux motifs temporels. Au contraire, une fuite trop grande rendra le neurone bien trop sensible aux motifs temporels, avec peu de sélectivité sur les caractéristiques spatiales. De plus, il sera plus difficile d'apprendre à détecter des caractéristiques temporelles pertinentes, la fenêtre temporelle d'acquisition des spikes étant trop grande (Bichler et al. 2012). Une remarque est cependant à ajouter. Dans notre étude mesurant l'impact du paramètre de fuite sur notre modèle (voir la SECTION 4.4.5), nous avons observé qu'à partir d'une certaine valeur de fuite, l'augmenter, même d'un facteur 5, n'impacte plus le réseau qui conserve un bon score de distinguabilité sur les caractéristiques extraites pour les différentes classes de mouvement. Ce constat s'applique à toutes les formes spatiales présentées lors des mesures. Dans notre cas, c'est la nature des données d'entrée qui permet d'expliquer cette observation. En effet, les séquences d'évènements traitées par le réseau présentent des particularités :

— une forme en mouvement se déplace seule à une vitesse constante ;
— le délai entre chaque séquence présentée est suffisamment long, permettant la réinitialisation du potentiel de membrane des neurones à la valeur de repos.

Il n'y a donc pas la possibilité de détecter des corrélations temporelles induites par le passage précédent d'une autre forme. Enfin, la tâche présentée lors de cette étude ne prend pas en considération la distinction entre différentes vitesses.

Par conséquent, un équilibre dans le réglage des paramètres doit être trouvé, surtout lorsque ces paramètres sont communs à l'ensemble du réseau. Les résultats de l'étude présentée dans la SECTION 4.3.4 indiquent que, pour certains réglages de paramètres, le réseau n'est en mesure de reconnaître les directions de mouvement que pour des formes spécifiques.

Ce réglage peut être complexe à obtenir : il dépend des spécificités du problème que l'on cherche à résoudre et de la nature des données d'entrée que l'on va traiter. On peut retrouver ce constat dans une autre étude (Shahsavari et Boulet 2017) traitant cette fois-ci de problème de reconnaissance. Ce réglage peut être délégué à des algorithmes évolutionnaires, mais leurs mises en place impliquent des coûts en calcul supplémentaires et

ne peuvent être envisagées que sur des problèmes simples (Bichler et al. 2012). Cependant, si nous voulons une diversification dans la reconnaissance de motifs spatio-temporels faite par le SNN, indispensable lorsqu'on veut traiter des données réelles, alors l'application d' un jeu de paramètres uniques à tout le réseau paraît, a minima difficile, voire potentiellement impossible. En effet, si nous prenons le cas de l'analyse du mouvement dans une scène, les objets en mouvement sont décrits par des séquences d'impulsions différentes selon la taille et la vitesse de l'objet. Ainsi, pour une séquence donnée, le nombre de spikes produits au même moment sera proportionnel à la taille de l'objet et l'espacement temporel entre les différents spikes sera relatif à la vitesse du mouvement. Il est donc nécessaire d'introduire une disparité dans les paramètres afin que les neurones puissent réagir à différents motifs spatio-temporels. L'étude relatée par (Rathi et Roy 2021) propose un réseau composé de neurones ayant des paramètres hétérogènes. Le réglage des paramètres se fait à l'aide d'une descente de gradient. L'apprentissage n'est donc pas fait directement par le SNN. D'autres travaux suggèrent d'avoir des neurones possédant des seuils d'activation (Falez et al. 2019b ; Paredes-Vallés, Scheper et De Croon 2019) et des fuites (X. She et al. 2021) qui s'adaptent aux distributions des stimulus entrants afin de garantir une homéostasie. Ces types de processus existent dans le cerveau (Platkiewicz et Brette 2010). C'est la raison pour laquelle nous avons décidé d'ajouter un mécanisme d'homéostasie sur le seuil d'activation dans l'étude suivante (voir CHAPITRE 5). Concernant la fuite, le choix de sa valeur à un impact plus modéré sur les performances du réseau. La seule condition à respecter est de ne pas fixer une valeur trop petite. Pour augmenter la sélectivité des neurones sur la dimension temporelle des spikes, nous avons opté pour l'utilisation des délais synaptiques. C'est un mécanisme central dans le fonctionnement de notre modèle. Lors des premiers instants du passage d'une forme en mouvement, un neurone, favorisé par l'initialisation aléatoire des poids, va se déclencher et inhiber tous les autres neurones voisins. Compte tenu du mouvement synchrone des pixels dans notre jeu de données, ce neurone sera de nouveau activé à l'étape suivante et va devenir progressivement l'unique vainqueur au sein de la compétition neuronale. Même si ce type de données ne représente qu'un cas spécifique et que le comportement des données réelles est beaucoup moins synchrone, le modèle est face à une difficulté qu'il ne parvient pas à résoudre malgré les différents mécanismes dont il dispose (neurone avec fuite, STDP, etc.). L'utilisation des délais nous a permis de pallier ce problème, en favorisant les neurones à se spécialiser sur des motifs temporels différents. Il existe différentes méthodes pour intégrer des délais à notre modèle. Une approche consiste à dupliquer les synapses, permettant ainsi l'application d'une gamme de délais prédéfinis (Paredes-Vallés, Scheper et De Croon 2019). Dans notre cas, nous avons opté pour une méthode qui attribue, pour chaque synapse, un délai défini aléatoirement sur un petit intervalle. L'utilisation de cette méthode permet de conserver la taille de la topologie.

Dans le dernier CHAPITRE 6, nous avons exploré les possibilités de caractérisations de motifs temporels en modélisant différents types de détecteurs sensibles au mouvement. En combinant les délais, la fuite du neurone et les poids, il est possible de contrôler finement la sélectivité d'un neurone à divers motifs spatio-temporels (direction précise, vitesse variable ou précise). Nos observations ont montré que la sélectivité du neurone est définie dans un premier temps par les délais, qui sont utilisés pour ajuster la direction et la vitesse du détecteur. À partir de ce réglage, le paramétrage de la fuite du neurone ajoute une certaine tolérance sur la vitesse du mouvement. Enfin, les poids permettent de moduler cette tolérance.

7.1.2 Apprentissage incrémental

Dans la première étude, nous avons entraîné notre modèle de manière non supervisée à reconnaître différentes directions d'un mouvement induit par le déplacement d'un motif binaire. En analysant les phases dans lesquelles la STDP était activée, nous avons remarqué qu'il ne fallait présenter que quelques passages d'une nouvelle classe pour que le réseau puisse apprendre à la distinguer. De plus, l'ordre de passage des différentes classes lors de l'entraînement n'a pas influencé les performances du réseau. Ces deux observations nous ont permis de démontrer des capacités d'apprentissage rapide et incrémental du modèle. Nous avons donc étudié le comportement du modèle, en introduisant séquentiellement des classes inédites et en sauvegardant l'état du réseau à chaque fois qu'une nouvelle classe a été introduite. Nous avons enrichi les données d'entrées par des séquences représentant des motifs plus variables dans la forme et la vitesse pendant leurs déplacements. Afin d'évaluer la capacité du modèle, nous avons utilisé des algorithmes de classification sur les caractéristiques extraites par le SNN. Ces méthodes ne peuvent pas traiter des séquences d'impulsions, c'est pourquoi il faut transformer ces séquences en valeurs numériques. Nous avons testé 4 stratégies de décodages (1 décodage fréquentiel, 2 décodages temporels et une fusion des 3). Les résultats montrent que le modèle est capable d'apprendre de nouvelles classes sans détériorer les performances sur les anciennes classes. Enfin, les stratégies de décodage temporelles donnent de meilleurs résultats, la meilleure stratégie étant le décodage "*rank-order*". En effet, au cours de nos expérimentations, nous avons constaté une détérioration des performances lors de l'introduction de la troisième classe due à l'ordre de présentation des différentes classes (mouvements verticaux puis mouvements horizontaux) à apprendre. Le décodage "*rank order*" est la seule stratégie qui permet de maintenir les performances du réseau sur les anciennes classes apprises. En utilisant la stratégie de décodage "*rank order*", il est donc possible de préserver plus efficacement les caractéristiques pertinentes qui définissent les classes déjà apprises.

7.2 Perspectives

Ces travaux ouvrent des perspectives quant à l'évolution de notre modèle, qui en résumé, peut être vu comme un détecteur élémentaire sensible aux caractéristiques liées aux mouvements, avec des capacités d'apprentissage non supervisé, local et incrémental. Ce détecteur permet de reconnaître des motifs spatio-temporels, paramétré par les neurones du réseau. Nous avons pu voir dans le SECTION 4.4 que les paramètres des neurones du réseau influencent la capacité du modèle à pouvoir être sensible à certains types de stimuli. Le problème est que ces paramètres sont communs à tous les neurones du réseau, et ne permettent donc pas d'augmenter la diversité des motifs spatio-temporels qui peuvent être traités par le modèle. Des solutions proposent de régler dynamiquement les paramètres du neurone en leur intégrant un mécanisme homéostatique qui régule les valeurs selon le type de données d'entrée (vitesse, taille, etc.). Malgré tout, ces mécanismes ont une influence qui reste limitée puisqu'ils se basent sur les valeurs de départ des paramètres des neurones du réseau. De plus, il peut être difficile, voire impossible, de caractériser un ensemble précis de motifs spatio-temporels au travers, uniquement, du réglage des paramètres d'un neurone. Cependant, intégrer des délais synaptiques au modèle permet de répondre à cette problématique. Les délais offrent un degré de liberté supplémentaire dans la caractérisation des motifs spatio-temporels. Lors de l'élaboration de notre modèle, nous avons eu recours à l'utilisation des délais dans le but de diversifier la spécialisation des neurones sur des motifs temporels différents. Dans notre cas, la caractérisation précise d'un motif n'était pas notre première intention, c'est pourquoi chaque synapse avait une valeur définie aléatoirement sur un petit intervalle. Dans le CHAPITRE 6, nous avons mené des études de cas théorique sur la sélectivité des neurones sur des motifs spatio-temporels.

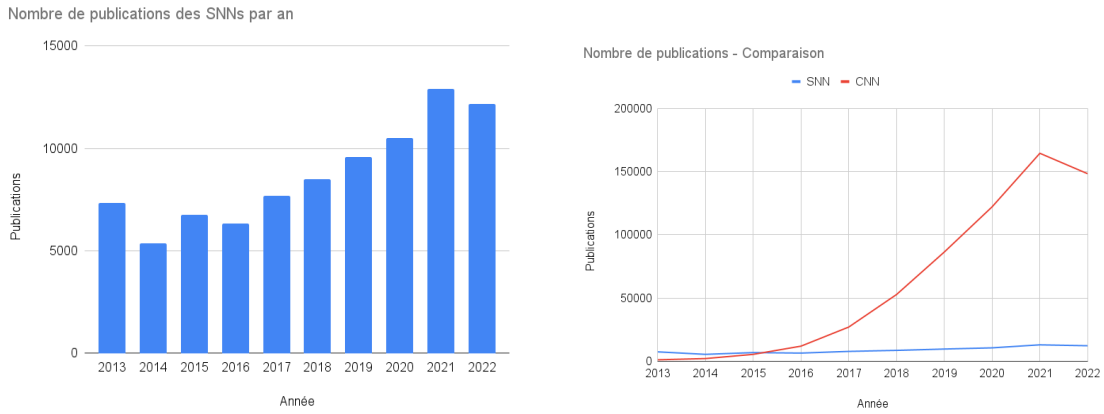
Une autre perspective est d'appliquer notre modèle à des données réelles issues d'un capteur événementiel. Ce type de données nécessite de pouvoir combiner nos détecteurs en les agrégeant dans des architectures plus complexes.

De la même manière que pour les RNAs traditionnels, l'intérêt d'élaborer des réseaux multi-couches est d'accroître l'expressivité du modèle en augmentant le pouvoir de représentation des informations du réseau. En effet, en organisant le réseau en une succession de couche de neurones, chacune va permettre l'extraction de caractéristiques de plus en plus abstraites en se basant sur les caractéristiques extraites par les couches précédentes. La conception d'architectures multi-couches pose de nouveaux défis. Plus le modèle devient complexe, plus il est délicat de résoudre les difficultés rencontrées lors des différentes études menées. Différents travaux ont montré qu'il était plus difficile de faire converger un SNN profond lors d'un apprentissage. Il existe plusieurs raisons expliquant cette problématique, l'une d'elles étant que les SNNs peuvent être plus sensibles aux hyperparamètres et nécessitent souvent plus de temps pour converger vers une solution optimale. Et pour cause, les impulsions reçues par chaque couche encodent des caractéristiques différentes, avec une distribution qui peut donc différer d'une couche à l'autre. Un jeu de paramètres communs à tout le réseau est donc plus complexe à trouver, voire impossible. Ainsi, il est plus pertinent d'avoir un jeu de paramètres hétérogènes, même si cela augmente la taille de l'espace des valeurs possibles. De plus, ce type d'architecture introduit de nouveaux paramètres tels que la stratégie de connexions entre les neurones ou encore le nombre de couches cachées. Les SNNs multi-couches sont également confrontés à d'autres problématiques, notamment liées à l'encodage de l'information par des impulsions. En

effet, l'activité des neurones diminue progressivement en fonction de leur position dans le réseau, les neurones étant de moins en moins activés au fur et à mesure que l'on s'enfonce dans les couches cachées. Cela peut poser des problèmes lors de l'apprentissage, en particulier lors de l'utilisation de la STDP dont les décisions sont basées sur les moments d'activation des neurones. Enfin, l'utilisation de délais synaptiques pour améliorer la sélectivité spatio-temporelle des neurones soulève aussi des interrogations quant à leur ajout dans toutes les couches du réseau. Faut-il les intégrer uniquement entre la couche d'entrée et la première couche cachée ou les ajouter à l'ensemble du réseau ? En effet, les couches suivantes incluent implicitement ces délais en intégrant les spikes émis par la première couche, ce qui influence la sélection des connexions avec certains neurones. L'emploi du terme "implicitement" est justifié, car les moments d'activation des neurones tiennent compte du retard introduit par les délais, qui est alors propagé à travers tout le réseau. De plus, faut-il procéder à un pré-traitement des données d'entrées afin que les couches du SNNs puissent extraire des caractéristiques pertinentes ? Certaines questions ont des réponses, comme l'utilisation de connexions convolutives dans le réseau, le réglage automatique des hyper-paramètres via des mécanismes d'homéostasie, l'introduction de règles d'apprentissage supervisé pour les délais entre les différentes couches du réseau.

7.3 Recul sur les SNNs

Actuellement, les SNNs ne bénéficient pas du même statut que les méthodes deep traditionnelles, et sont encore principalement utilisés dans des contextes expérimentaux. Toutefois, les SNNs présentent des avantages significatifs par rapport aux méthodes profondes classiques, tels que l'économie d'énergie grâce à une implémentation matérielle très efficace et la possibilité d'utiliser des règles d'apprentissage non supervisé bio-inspiré. Ces avantages ont contribué à un engouement croissant pour ce type de méthode (Tavanaei, Ghodrati et al. 2019) comme le montre la FIGURE 7.1a.



(a) Popularité des SNNs (b) Popularité des SNNs vs CNNs

Figure 7.1 – (a) Nombre d'articles publiés chaque année sur les SNNs. (b) Comparaison du nombre d'articles publiés chaque année sur les SNNs et les CNNs . Ces données ont été récoltées via l'outil `app.dimension.ai` disponible à cette URL : `https://app.dimensions.ai/discover/publication`

.

Bien que les SNNs soient prometteurs, ils sont encore peu étudiés par rapport aux méthodes traditionnelles telles que les CNNs qui sont largement utilisées en vision par ordinateur. Pourtant, les SNNs pourraient apporter des solutions à des problématiques majeures telles que la consommation énergétique. En effet, les problématiques liées à la consommation énergétique sont devenues critiques et les préoccupations, inhérentes à ce sujet, ne cessent d'augmenter avec le contexte actuel (crise énergétique, crise écologique). Pourtant, les solutions mises en avant en vision (et dans beaucoup d'autres domaines) ne cessent de repousser les limites en termes de consommation énergétique. Elles résolvent des tâches de plus en plus complexes, comme la génération d'images via un texte descriptif par exemple (Ramesh et al. 2022). Ce type de tâche est résolu par des modèles tel que Dall-E, dont l'architecture comporte plus de 12 milliards de paramètres. On peut s'interroger sur la pertinence de la solution proposée pour résoudre ce type de tâche, car les réseaux qui en découlent nécessitent un grand nombre de paramètres et demandent des ressources considérables en termes de calcul et d'énergie. Il est donc légitime de se demander si une telle consommation de ressources est justifiée pour résoudre ce type de problème, et si celui-ci ne devrait pas nécessiter des solutions plus économes quitte à faire l'impasse sur les performances. Bien que ces solutions puissent répondre à des besoins dans certains secteurs, elles ne sont pas vitales si on les rapporte au coût global demandé en énergie.

Cela crée surtout une dépendance pour les utilisateurs recourant à ce type de service car ces modèles sont très difficilement reproductibles, ce qui donne aux entreprises ayant les moyens de proposer ce type de service un monopole. Ces entreprises génèrent un besoin, attirent de plus en plus d'utilisateurs et les rendent dépendants de leurs services, ce qui conduit à une multiplication des infrastructures nécessaires pour exécuter ces modèles et à une augmentation de l'empreinte énergétique globale. D'un point de vue économique, ce type de solution permet à certains acteurs du marché de pouvoir s'enrichir. Mais cela pose de véritables questions au niveau éthique (Verdegem 2022). Même si ces interrogations sont abordées par la communauté scientifique, elles sont encore largement sous-exposées. Pire, les grandes entreprises de la tech, qui ont une influence importante dans le domaine de l'IA et qui sont très présentes lors des conférences académiques les plus prestigieuses (NeurIPS, CVPR, etc.) peuvent orienter les recherches vers des pistes qui leurs semblent profitables, voire même propager de la désinformation (un phénomène déjà observé dans l'industrie du tabac) (Mohamed Abdalla et Moustafa Abdalla 2021). Cette désinformation peut se propager et être reliée très rapidement, malgré le système de comités de relecture. Par exemple, l'article de Google de 2016 (Konečnỳ et al. 2016) prétendait que l'apprentissage fédéré protégeait les informations sensibles, alors que de plus en plus d'articles suggèrent l'inverse (Farhadkhani, Guerraoui, Villemaud et al. 2022 ; El-Mhamdi et al. 2022 ; C. Zhang et al. 2021). Malgré tout, cette désinformation a été reprise sans justification, amplifiée et normalisée par beaucoup trop de publications scientifiques ($>$ 1600 citations), malgré le fait que l'article mentionne un aveu de faiblesse dans une section technique. Les enjeux sur le traitement des données privées sont donc cruciaux. Bien que cet exemple ne se réfère pas à des problématiques traitées par les SNNs, il illustre néanmoins les enjeux autour de l'IA et l'impact sociétal majeur que cela créait. Il n'y a pas besoin de se tourner vers des scénarios de science-fiction pour voir émerger des IAs qui peuvent régir, concrètement, une partie du monde, avec un pouvoir de décision centralisé sans précédent. Un exemple concret est la solution Aladdin développée par la société BlackRock, qui gère environ 7% des actifs financiers mondiaux, soit environ 18.000 milliards de dollars américains (T. C. Lin 2016). En conséquence, il est primordial de tendre vers des recherches plus éthiques en IA au risque de perdre tout bénéfice potentiel avec ces solutions, voire de causer de graves dommages à la société. Nous ne suggérons en aucun cas que les SNNs soient des modèles éthiques ou qu'ils puissent apporter des solutions miraculeuses aux différents défis énoncés tels que la crise énergétique. De plus, la critique que nous faisons sur certains industriels, en particulier sur les intentions commerciales qui les animent, n'exclut pas le fait qu'ils jouent un rôle majeur dans le développement de solutions liées aux SNNs (puces neuromorphiques basse consommation tel que TrueNorth d'IBM (Akopyan et al. 2015) et Loihi d'Intel (Davies et al. 2018)) ou de système de vision bio-inspiré (DVS-Gen de Samsumg (Rebecq et al. 2019b)).

Cependant, une réflexion éthique sur les différentes pistes de recherches en IA nous semble être aspect fondamental dans la démarche scientifique. Dans ce manuscrit, nous avons décrit le potentiel offert par les SNNs qui résulte de la modélisation des neurones qui communiquent à l'aide d'impulsions de manière asynchrone. Le modèle permet d'encoder l'information de manière parcimonieuse, avec un traitement massivement parallèle à l'aide d'opérations locales. La STDP est une règle d'apprentissage non supervisée qui partage des propriétés communes sur les opérations locales, parcimonieuses et paralléli-

sables. Des études suggèrent que la règle STDP nécessite beaucoup moins d'opérations lors de l'apprentissage que les RNAs classiques entraînes par une SGD, tout en atteignant des performances équivalentes (Tavanaei, Ghodrati et al. 2019)

De plus, les SNNs ont la particularité de traiter une information en prenant en compte une dimension temporelle supplémentaire. Cela vient du fait qu'une impulsion se déclenche à un instant t. Contrairement aux réseaux classiques qui ne peuvent caractériser que des motifs spatiaux, sauf lorsqu'une architecture particulière est mise en place (récurrente, avec des convolutions 3D), les SNNs peuvent détecter des motifs spatio-temporels avec un seul neurone. En effet, les neurones impulsionnels peuvent être vu comme des détecteurs de coïncidences spatiales et temporelles dont le comportement est régi par leurs paramètres internes tels que le seuil d'activation et la fuite (Ponulak et Kasinski 2011). Les synapses connectent les unités du réseau de manière topologique et transmettent les informations qui sont caractérisées par les neurones dans le domaine temporel et/ou spatial.

Les caméras événementielles permettent de fournir des données visuelles décrites par des informations similaires aux impulsions des SNNs, avec une résolution temporelle très élevée. Ces capteurs, peu énergivores, partagent les mêmes propriétés d'asynchronie que les SNNs (Gallego et al. 2020). Entre l'apprentissage non supervisé économe en nombre d'opérations et les accélérations matérielles possibles, des solutions voient le jours pour répondre aux contraintes inhérentes aux systèmes embarqués en temps réel (Amir et al. 2017 ; Paredes-Vallés, Scheper et De Croon 2019).

Malgré tout, beaucoup de travaux se concentrent sur des problèmes de reconnaissance de chiffres manuscrits sur des jeux de données tels que MNIST (Diehl et Cook 2015 ; Falez et al. 2019b ; Masquelier et Thorpe 2007). Ce type de problème permet de mettre en évidence certaines des limitations que possèdent les modèles SNNs en vision par ordinateur. Des questions restent ouvertes : comme l'encodage des données (Vigneron et Martinet 2020), le pré-traitement (Falez et al. 2019a), les topologies multi-couches (Tavanaei, Ghodrati et al. 2019), le paramétrage dynamique des neurones (Shahsavari et Boulet 2017), etc. D'un point de vue épistémologique, l'évolution des SNNs dans le domaine de la vision par ordinateur présente des similarités avec les RNAs conventionnels, notamment avec l'utilisation de MNIST pour valider les premières architectures (LeCun, Bottou et al. 1998).

Des méthodes de conversion des RNAs vers les SNNs ont été élaborées pour réduire l'écart entre les performances des deux modèles. Bien que les avantages en termes de consommation énergétique des SNNs soient limités en raison de l'utilisation de méthodes d'apprentissage comme la SGD, la conversion permet de créer des systèmes qui bénéficient de l'accélération matérielle neuromorphique lors de l'inférence. En somme, ces méthodes de conversion permettent d'obtenir des systèmes moins énergivores (Amir et al. 2017 ; S. Kim et al. 2019). Certaines approches intermédiaires ont pour but d'appliquer la descente de gradient sur des SNNs, mais elles sont confrontées à des difficultés liées notamment à la non-dérivabilité des neurones impulsionnels (Kaiser, Mostafa et Neftci 2020 ; Tavanaei, Ghodrati et al. 2019). Ces méthodes sont basées sur une optimisation globale, qui est plus coûteuse en énergie, mais avec lesquelles on peut capitaliser sur les architectures développées pour les réseaux profonds.

Concurrencer les méthodes profondes sur des problèmes nécessitant la reconnaissance exclusive de motifs spatiaux reste difficile. En revanche, les SNNs présentent des résultats

encourageants dans le domaine du traitement vidéo, grâce à leur aptitude à caractériser à la fois les notions temporelles et spatiales. Des études utilisant des mécanismes comme la modulation des délais synaptiques, la STDP ou l'apprentissage à court terme permettent de concurrencer les RNAs classiques possédant un nombre de paramètres à optimiser équivalent (Deng et al. 2020 ; X. She et al. 2021).

En outre, les recherches faites sur la STDP permettent de développer des règles d'apprentissage locales. Certaines propositions de règles permettent de faire de l'apprentissage par renforcement (Mozafari et al. 2018), tandis que d'autres ajoutent de la supervision sans nécessiter d'opérations globales sur tous les paramètres à optimiser (Ponulak 2005).

La règle STDP est basée sur la théorie de l'apprentissage hébbien et les observations faites sur le comportement de la plasticité des synapses dans le cerveau humain. Comme mentionné précédemment dans l'état de l'art, selon Hebb (Donald O Hebb 1949), l'activité synchrone de plusieurs neurones résulterait en la formation d'une assemblée de cellules sous-tendant les processus cognitifs de haut niveau. Ce phénomène est l'un des facteurs qui contribuent à expliquer les capacités d'apprentissage du cerveau humain. Des études suggèrent que des réseaux de neurones se sont formés ainsi, dans des zones corticales précises, en devenant sélectifs à une fonction cognitive et un type particulier d'information à traiter. La transposition de ces principes aux RNAs par les SNNs et la STDP permettent d'obtenir des modèles qui résolvent des tâches spécifiques avec une certaine efficacité. Néanmoins, ces solutions n'atteignent pas les capacités d'apprentissage du cerveau humain. La plasticité des synapses biologiques est fortement influencée par le développement des synapses lors de la synaptogenèse (processus de création des synapses biologiques). Bien qu'elle se produise tout au long de la durée de vie d'une personne, une augmentation importante de la formation des synapses se produit au cours du développement précoce du cerveau (Di Cristo et Chattopadhyaya 2020). Ce développement suit des règles complexes dictées par les gènes et influencées par les processus évolutifs de la sélection naturelle. La synaptogenèse détermine donc la topologie des réseaux de neurones, le type et l'état initial des synapses (Gatto et Broadie 2010). Si nous faisons un parallèle avec les RNAs, la synaptogenèse peut être associée aux choix des architectures et à l'initialisation des poids. Cette initialisation n'est pas arbitraire dans le cerveau et suit un processus d'optimisation globale encodé dans les gènes. Cette optimisation globale peut être faite en amont par des algorithmes évolutionnistes (Elbrecht et al. 2020) ou basée sur la descente de gradient avec des jeux de données possédant une grande variété de données (Chakraborty, X. She et Mukhopadhyay 2021). Ces modèles montrent des résultats supérieurs avec des capacités de généralisation et d'évolution (apprentissage incrémental possible) sur des nouvelles données avec l'utilisation de la STDP sur des poids initialisés préalablement via une méthode d'optimisation globale. Ces perspectives permettent le développement de nouveaux modèles neuro-informatiques qui synthétisent des connaissances issues des neurosciences (comme l'apprentissage hébbien) et de l'apprentissage automatique (comme l'apprentissage par renforcement, auto-supervisé et adverse). Ces perspectives pourraient alimenter le développement de la prochaine génération d'IAs et même, à long terme, conduire à une forme de conscience artificielle. (Volzhenin, Changeux et Dumas 2022).

Bibliographie

Abdalla, Mohamed et Moustafa Abdalla (2021). « The Grey Hoodie Project : Big tobacco, big tech, and the threat on academic integrity ». In : *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society*, p. 287-297.

Akopyan, Filipp et al. (2015). « Truenorth : Design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip ». In : *IEEE transactions on computer-aided design of integrated circuits and systems* 34.10, p. 1537-1557.

Allred, Jason M. et Kaushik Roy (2020). « Controlled Forgetting : Targeted Stimulation and Dopaminergic Plasticity Modulation for Unsupervised Lifelong Learning in Spiking Neural Networks ». In : *Frontiers in Neuroscience* 14, p. 7. issn : 1662-453X. doi : 10.3389/fnins.2020.00007. url : https://www.frontiersin.org/article/10.3389/fnins.2020.00007.

Alzubaidi, Laith et al. (2021). « Review of deep learning : Concepts, CNN architectures, challenges, applications, future directions ». In : *Journal of big Data* 8.1, p. 1-74.

Amir, Arnon et al. (2017). « A Low Power, Fully Event-Based Gesture Recognition System. » In : *CVPR*, p. 7388-7397.

Bi, Guo-qiang et Mu-ming Poo (2001). « Synaptic Modification BY Correlated ». In : *Annu. Rev. Neurosci* 24, p. 139-66.

Bichler, Olivier et al. (2012). « Extraction of temporally correlated features from dynamic vision sensors with spike-timing-dependent plasticity ». In : *Neural Networks* 32, p. 339-348.

Boahen, Kwabena A (2004). « A burst-mode word-serial address-event link-I : Transmitter design ». In : *IEEE Transactions on Circuits and Systems I : Regular Papers* 51.7, p. 1269-1280.

Bowen, Jonathan P (2016). « Alan Turing : founder of computer science ». In : *School on Engineering Trustworthy Software Systems*. Springer, p. 1-15.

Brette, Romain (2015). « Philosophy of the Spike : Rate-Based vs. Spike-Based Theories of the Brain ». In : *Frontiers in Systems Neuroscience* 9, p. 151. issn : 1662-5137. doi : 10.3389/fnsys.2015.00151. url : https://www.frontiersin.org/article/10.3389/fnsys.2015.00151.

Brown, Tom et al. (2020). « Language models are few-shot learners ». In : *Advances in neural information processing systems* 33, p. 1877-1901.

Burbank, Kendra S (2015). « Mirrored STDP implements autoencoder learning in a network of spiking neurons ». In : *PLoS computational biology* 11.12, e1004566.

Cao, Yongqiang, Yang Chen et Deepak Khosla (2015). « Spiking deep convolutional neural networks for energy-efficient object recognition ». In : *International Journal of Computer Vision* 113.1, p. 54-66.

Chakraborty, Biswadeep, Xueyuan She et Saibal Mukhopadhyay (2021). « A fully spiking hybrid neural network for energy-efficient object detection ». In : *IEEE Transactions on Image Processing* 30, p. 9014-9029.

Chetlur, Sharan et al. (2014). « cudnn : Efficient primitives for deep learning ». In : *arXiv preprint arXiv :1410.0759*.

Chrislb (s. d.). *Image d'un neurone artificiel*. https://fr.wikipedia.org/wiki/R%C3%A9seau_de_neurones_artificiels, last checked Aug 18, 2018. url : https://fr.wikipedia.org/wiki/R%5C%C3%5C%A9seau_de_neurones_artificiels.

Cossu, Andrea, Antonio Carta et Bacciu Davide (2020). « Continual Learning with Gated Incremental Memories for sequential data processing ». In : *IJCNN 2020*. IEEE, p. 1-8.

D Cat laz Wiki (s. d.). *Illustration tirée d'un article wikipédia sur les modèles du neurone biologique*. https://fr.wikipedia.org/wiki/ModÃÍles_du_neurone_biologique, last checked Oct 24, 2022. url : https://fr.wikipedia.org/wiki/Mod%C3%A8les_du_neurone_biologique.

Danneville, Francois et al. (2019). « A Sub-35 pW Axon-Hillock artificial neuron circuit ». In : *Solid-State Electronics* 153, p. 88-92.

Davies, Mike et al. (2018). « Loihi : A neuromorphic manycore processor with on-chip learning ». In : *IEEE Micro* 38.1, p. 82-99.

Delbruck, Tobi et Carver A Mead (1991). « Time-derivative adaptive silicon photoreceptor array ». In : *Infrared Sensors : Detectors, Electronics, and Signal Processing*. T. 1541. SPIE, p. 92-99.

Deng, Lei et al. (2020). « Rethinking the performance comparison between SNNS and ANNS ». In : *Neural networks* 121, p. 294-307.

Di Cristo, Graziella et Bidisha Chattopadhyaya (2020). « Development of neuronal circuits : From synaptogenesis to synapse plasticity ». In : *Handbook of Clinical Neurology*. T. 173. Elsevier, p. 43-53.

Diehl, Peter U et Matthew Cook (2015). « Unsupervised learning of digit recognition using spike-timing-dependent plasticity ». In : *Frontiers in computational neuroscience* 9, p. 99.

Elbrecht, Daniel et al. (2020). « Training spiking neural networks using combined learning approaches ». In : *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, p. 1995-2001.

Eurich, Christian W et al. (1999). « Dynamics of self-organized delay adaptation ». In : *Physical Review Letters* 82.7, p. 1594.

Falez, Pierre et al. (2019a). « Multi-layered Spiking Neural Network with Target Timestamp Threshold Adaptation and STDP ». In : *arXiv preprint arXiv :1904.01908*.

— (2019b). « Unsupervised visual feature learning with spike-timing-dependent plasticity : How far are we from traditional feature learning approaches ? » In : *Pattern Recognition* 93, p. 418-429.

Farabet, Clément et al. (2012). « Comparison between frame-constrained fix-pixel-value and frame-free spiking-dynamic-pixel convnets for visual processing ». In : *Frontiers in neuroscience* 6, p. 32.

Farhadkhani, Sadegh, Rachid Guerraoui, Oscar Villemaud et al. (2022). « An Equivalence Between Data Poisoning and Byzantine Gradient Attacks ». In : *International Conference on Machine Learning*. PMLR, p. 6284-6323.

French, Robert M (1999). « Catastrophic forgetting in connectionist networks ». In : *Trends in cognitive sciences* 3.4, p. 128-135.

Furber, Steve (2016). « Large-scale neuromorphic computing systems ». In : *Journal of neural engineering* 13.5, p. 051001.

Furber, Steve B et al. (2012). « Overview of the spinnaker system architecture ». In : *IEEE Transactions on Computers* 62.12, p. 2454-2467.

G. Hocquet O. Bichler, D. Querlioz (2020). « OvA-INN : Continual Learning with Invertible Neural Networks ». In : *IJCNN 2020*. IEEE, p. 1-8.

Gallego, Guillermo et al. (2020). « Event-based vision : A survey ». In : *IEEE transactions on pattern analysis and machine intelligence* 44.1, p. 154-180.

Gatto, Cheryl L et Kendal Broadie (2010). « Genetic controls balancing excitatory and inhibitory synaptogenesis in neurodevelopmental disorder models ». In : *Frontiers in synaptic neuroscience*, p. 4.

Gehrig, Daniel et al. (juin 2020). « Video to Events : Recycling Video Datasets for Event Cameras ». In : *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*.

Goodfellow, Ian J. et al. (2014). *Generative Adversarial Networks*. doi : 10.48550/ARXIV.1406.2661. url : https://arxiv.org/abs/1406.2661.

Guise, Mira, Alistair Knott et Lubica Benuskova (2015). « Enhanced polychronization in a spiking network with metaplasticity ». In : *Frontiers in Computational Neuroscience* 9, p. 9. issn : 1662-5188. doi : 10.3389/fncom.2015.00009. url : https://www.frontiersin.org/article/10.3389/fncom.2015.00009.

Hebb, DO (1949). « The organization of behavior ; a neuropsychological theory. » In.

Hebb, Donald O (1949). « The first stage of perception : growth of the assembly ». In : *The Organization of Behavior* 4, p. 60-78.

Hodgkin, Allan L et Andrew F Huxley (1952). « Currents carried by sodium and potassium ions through the membrane of the giant axon of Loligo ». In : *The Journal of physiology* 116.4, p. 449-472.

Hüning, Harald, Helmut Glünder et Günther Palm (1998). « Synaptic delay learning in pulse-coupled neurons ». In : *Neural computation* 10.3, p. 555-565.

Izhikevich, Eugene M (2003). « Simple model of spiking neurons ». In : *IEEE Transactions on neural networks* 14.6, p. 1569-1572.

Jing, Longlong et Yingli Tian (2020). « Self-supervised visual feature learning with deep neural networks : A survey ». In : *IEEE transactions on pattern analysis and machine intelligence* 43.11, p. 4037-4058.

Kaiser, Jacques, Hesham Mostafa et Emre Neftci (2020). « Synaptic plasticity dynamics for deep continuous local learning (DECOLLE) ». In : *Frontiers in Neuroscience* 14, p. 424.

Kartable (s. d.). *Illustration tirée d'un cours sur le réflexe myotatique*. https://www.kartable.fr/ressources/svt/cours/le-reflexe-myotatique/19361, last checked Aug 18, 2018. url : https://www.kartable.fr/ressources/svt/cours/le-reflexe-myotatique/19361.

Kheradpisheh, Saeed Reza et al. (2018). « STDP-based spiking deep convolutional neural networks for object recognition ». In : *Neural Networks* 99, p. 56-67. issn : 0893-

6080. DOI : https://doi.org/10.1016/j.neunet.2017.12.005. URL : http://www.sciencedirect.com/science/article/pii/S0893608017302903.

KIM, Hanme, Stefan LEUTENEGGER et Andrew J DAVISON (2016). « Real-time 3D reconstruction and 6-DoF tracking with an event camera ». In : *European conference on computer vision*. Springer, p. 349-364.

KIM, Seijoon et al. (2019). « Spiking-YOLO : Spiking Neural Network for Real-time Object Detection ». In : *arXiv preprint arXiv :1903.06530*.

KOLMOGOROV, Andrei N (1963). « On tables of random numbers ». In : *Sankhyā : The Indian Journal of Statistics, Series A*, p. 369-376.

KONEČNÝ, Jakub et al. (2016). « Federated optimization : Distributed machine learning for on-device intelligence ». In : *arXiv preprint arXiv :1610.02527*.

KRIZHEVSKY, Alex, Ilya SUTSKEVER et Geoffrey E HINTON (2012). « Imagenet classification with deep convolutional neural networks ». In : *Advances in neural information processing systems*, p. 1097-1105.

— (2017). « Imagenet classification with deep convolutional neural networks ». In : *Communications of the ACM* 60.6, p. 84-90.

KRUNGLEVICIUS, Dalius (août 2016). « Modified STDP Triplet Rule Significantly Increases Neuron Training Stability in the Learning of Spatial Patterns ». In : *Advances in Artificial Neural Systems* 2016, p. 1-12. ISSN : 1687-7594. DOI : 10.1155/2016/1746514.

LAPICQUE, Louis (1907). « Recherches quantitatives sur l'excitation electrique des nerfs traitee comme une polarization ». In : *Journal de Physiologie et de Pathologie Generalej* 9, p. 620-635.

LAROCHELLE, Hugo (s. d.). *Cours sur les techniques d'apprentissage automatique*. https://info.usherbrooke.ca/hlarochelle/cours/ift603_H2015/contenu.html, last checked Oct 24, 2022. URL : https://info.usherbrooke.ca/hlarochelle/cours/ift603_H2015/contenu.html.

LECUN, Yann (2022). « A Path Towards Autonomous Machine Intelligence Version 0.9. 2, 2022-06-27 ». In.

LECUN, Yann, Yoshua BENGIO et Geoffrey HINTON (2015). « Deep learning ». In : *nature* 521.7553, p. 436-444.

LECUN, Yann, Léon BOTTOU et al. (1998). « Gradient-based learning applied to document recognition ». In : *Proceedings of the IEEE* 86.11, p. 2278-2324.

LEE, Jun Haeng, Tobi DELBRUCK et Michael PFEIFFER (2016). « Training deep spiking neural networks using backpropagation ». In : *Frontiers in neuroscience* 10, p. 508.

LI, Hongmin et al. (2017). « CIFAR10-DVS : an event-stream dataset for object classification ». In : *Frontiers in neuroscience* 11, p. 309.

LICHTSTEINER, Patrick, Christoph POSCH et Tobi DELBRUCK (2008). « A 128×128 120 dB 15μ s Latency Asynchronous Temporal Contrast Vision Sensor ». In : *IEEE journal of solid-state circuits* 43.2, p. 566-576.

LIN, Tom CW (2016). « Compliance, technology, and modern finance ». In : *Brook. J. Corp. Fin. & Com. L.* 11, p. 159.

MAASS, Wolfgang (1997). « Networks of spiking neurons : the third generation of neural network models ». In : *Neural networks* 10.9, p. 1659-1671.

— (2015). « To spike or not to spike : that is the question ». In : *Proceedings of the IEEE* 103.12, p. 2219-2224.

Machado, Pedro et al. (2018). « Bio-Inspired Ganglion Cell Models for Detecting Horizontal and Vertical Movements ». In : *International Joint Conference on Neural Networks (IJCNN)*. IEEE, p. 1-8.

Masquelier, Timothée et Simon J Thorpe (2007). « Unsupervised learning of visual features through spike timing dependent plasticity ». In : *PLoS computational biology* 3.2, e31.

McCulloch, Warren S et Walter Pitts (1943). « A logical calculus of the ideas immanent in nervous activity ». In : *The bulletin of mathematical biophysics* 5.4, p. 115-133.

Merolla, Paul A et al. (2014). « A million spiking-neuron integrated circuit with a scalable communication network and interface ». In : *Science* 345.6197, p. 668-673.

El-Mhamdi, El-Mahdi et al. (2022). « SoK : On the Impossible Security of Very Large Foundation Models ». In : *arXiv preprint arXiv :2209.15259*.

Minsky, Marvin et Seymour Papert (1969). « Perceptrons. » In.

Mollahosseini, Ali, Behzad Hasani et Mohammad H Mahoor (2017). « Affectnet : A database for facial expression, valence, and arousal computing in the wild ». In : *arXiv preprint arXiv :1708.03985*.

Mozafari, M. et al. (2018). « First-Spike-Based Visual Categorization Using Reward-Modulated STDP ». In : *IEEE Transactions on Neural Networks and Learning Systems* 29.12, p. 6178-6190. issn : 2162-237X. doi : 10.1109/TNNLS.2018.2826721.

Nadafian, Alireza et Mohammad Ganjtabesh (2020). « Bio-plausible Unsupervised Delay Learning for Extracting Temporal Features in Spiking Neural Networks ». In : *arXiv preprint arXiv :2011.09380*.

Nunes, João D et al. (2022). « Spiking Neural Networks : A Survey ». In : *IEEE Access* 10, p. 60738-60764.

Orchard, Garrick et Ralph Etienne-Cummings (2014). « Bioinspired visual motion estimation ». In : *Proceedings of the IEEE* 102.10, p. 1520-1536.

Orchard, Garrick, Ajinkya Jayawant et al. (2015). « Converting static image datasets to spiking neuromorphic datasets using saccades ». In : *Frontiers in neuroscience* 9, p. 437.

Ororbia, Alexander (2020). « Spiking Neural Predictive Coding for Continual Learning from Data Streams ». In : *ArXiv* abs/1908.08655.

Oudjail, Veis et Jean Martinet (2018). « Étude du mouvement d'un motif à l'aide d'un réseau de neurones impulsionnels ». In : *CORESA*.

— (2019). « Bio-inspired Event-based Motion Analysis with Spiking Neural Networks. » In : *VISIGRAPP (4 : VISAPP)*, p. 389-394.

— (2020). « Meta-parameters exploration for unsupervised event-based motion analysis ». In : *15th International Conference on Computer Vision Theory and Applications*. SCITEPRESS-Science et Technology Publications, p. 853-860.

— (2021). « Classification of motion directions with spiking neurons : a continual learning case ». In.

Paredes-Vallés, Federico, Kirk Yannick Willehm Scheper et Guido Cornelis Henricus Eugene De Croon (2019). « Unsupervised learning of a hierarchical spiking neural network for optical flow estimation : From events to global motion perception ». In : *IEEE transactions on pattern analysis and machine intelligence*.

Patterson, David et al. (2021). « Carbon emissions and large neural network training ». In : *arXiv preprint arXiv :2104.10350*.

Paugam-Moisy, Hélene et Sander Bohte (2012). « Computing with spiking neuron networks ». In : *Handbook of natural computing*. Springer, p. 335-376.

Pfister, Jean-Pascal et Wulfram Gerstner (2006). « Triplets of spikes in a model of spike timing-dependent plasticity ». In : *Journal of Neuroscience* 26.38, p. 9673-9682.

Platkiewicz, Jonathan et Romain Brette (2010). « A threshold equation for action potential initiation ». In : *PLoS computational biology* 6.7, e1000850.

Poli, Roberto (2010). « The many aspects of anticipation ». In : *Foresight*.

Ponulak, Filip (2005). *ReSuMe-new supervised learning method for spiking neural networks. Institute of Control and Information Engineering, Poznoń University of Technology*. Rapp. tech. Tech. rep.

Ponulak, Filip et Andrzej Kasinski (2011). « Introduction to spiking neural networks : Information processing, learning and applications. » In : *Acta neurobiologiae experimentalis* 71.4, p. 409-433.

Posch, Christoph, Daniel Matolin et Rainer Wohlgenannt (2010). « A QVGA 143dB dynamic range asynchronous address-event PWM dynamic image sensor with lossless pixel-level video compression ». In : *2010 IEEE International Solid-State Circuits Conference-(ISSCC)*. IEEE, p. 400-401.

Ramesh, Aditya et al. (2022). « Hierarchical text-conditional image generation with clip latents ». In : *arXiv preprint arXiv :2204.06125*.

Rathi, Nitin et Kaushik Roy (2021). « DIET-SNN : A low-latency spiking neural network with direct input encoding and leakage and threshold optimization ». In : *IEEE Transactions on Neural Networks and Learning Systems*.

Rebecq, Henri et al. (2019a). « Events-to-video : Bringing modern computer vision to event cameras ». In : *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, p. 3857-3866.

— (2019b). « High speed and high dynamic range video with an event camera ». In : *IEEE transactions on pattern analysis and machine intelligence* 43.6, p. 1964-1980.

Reichardt, Werner (1961). « Autocorrelation, a principle for the evaluation of sensory information by the central nervous system ». In : *Sensory communication*, p. 303-317.

Roh, Yuji, Geon Heo et Steven Euijong Whang (2019). « A survey on data collection for machine learning : a big data-ai integration perspective ». In : *IEEE Transactions on Knowledge and Data Engineering* 33.4, p. 1328-1347.

Rosenblatt, Frank (1958). « The perceptron : a probabilistic model for information storage and organization in the brain. » In : *Psychological review* 65.6, p. 386.

Rumelhart, David E, Geoffrey E Hinton et Ronald J Williams (1985). *Learning internal representations by error propagation*. Rapp. tech. California Univ San Diego La Jolla Inst for Cognitive Science.

— (1986). « Learning representations by back-propagating errors ». In : *nature* 323.6088, p. 533.

Rumsey, Clifton C et LF Abbott (2004). « Synaptic equalization by anti-STDP ». In : *Neurocomputing* 58, p. 359-364.

Scheerlinck, Cedric et al. (2020). « Fast image reconstruction with an event camera ». In : *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, p. 156-163.

Serrano-Gotarredona, Teresa et Bernabé Linares-Barranco (2015). « Poker-DVS and MNIST-DVS. Their history, how they were made, and other details ». In : *Frontiers in neuroscience* 9, p. 481.

Shahsavari, Mahyar et Pierre Boulet (2017). « Parameter exploration to improve performance of memristor-based neuromorphic architectures ». In : *IEEE Transactions on Multi-Scale Computing Systems* 4.4, p. 833-846.

She, Xueyuan et al. (2021). « A heterogeneous spiking neural network for unsupervised learning of spatiotemporal patterns ». In : *Frontiers in Neuroscience*, p. 1406.

Singer, Uriel et al. (2022). « Make-A-Video : Text-to-Video Generation without Text-Video Data ». In : *arXiv preprint arXiv :2209.14792*.

Sourikopoulos, Ilias et al. (2017). « A 4-fJ/spike artificial neuron in 65 nm CMOS technology ». In : *Frontiers in neuroscience* 11, p. 123.

Storrs, Katherine R, Barton L Anderson et Roland W Fleming (2021). « Unsupervised learning predicts human perception and misperception of gloss ». In : *Nature Human Behaviour* 5.10, p. 1402-1417.

Suh, Yunjae et al. (2020). « A 1280× 960 dynamic vision sensor with a 4.95-μm pixel pitch and motion artifact minimization ». In : *2020 IEEE international symposium on circuits and systems (ISCAS)*. IEEE, p. 1-5.

Tappert, Charles C (2019). « Who is the father of deep learning ? » In : *2019 International Conference on Computational Science and Computational Intelligence (CSCI)*. IEEE, p. 343-348.

Tavanaei, Amirhossein, Masoud Ghodrati et al. (2019). « Deep learning in spiking neural networks ». In : *Neural networks* 111, p. 47-63.

Tavanaei, Amirhossein et Anthony S Maida (2017). « Multi-layer unsupervised learning in a spiking convolutional neural network ». In : *Neural Networks (IJCNN), 2017 International Joint Conference on*. IEEE, p. 2023-2030.

Tavanaei, Amirhossein, Timothée Masquelier et Anthony S Maida (2016). « Acquisition of visual features through probabilistic spike-timing-dependent plasticity ». In : *2016 International Joint Conference on Neural Networks (IJCNN)*. IEEE, p. 307-314.

Turing, Alan M et J Haugeland (1950). « Computing machinery and intelligence ». In : *The Turing Test : Verbal Behavior as the Hallmark of Intelligence*, p. 29-56.

Vaswani, Ashish et al. (2017). « Attention is all you need ». In : *Advances in neural information processing systems* 30.

Verdegem, Pieter (2022). « Dismantling AI capitalism : the commons as an alternative to the power concentration of Big Tech ». In : *AI & society*, p. 1-11.

Vigneron, Alex et Jean Martinet (2020). « A critical survey of STDP in Spiking Neural Networks for Pattern Recognition ». In : *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE, p. 1-9.

Vithursan Thangarasa Thomas Miconi, Graham W. Taylor (2020). « Enabling Continual Learning with Differentiable Hebbian Plasticity ». In : *ArXiv* abs/2006.16558.

Volzhenin, Konstantin, Jean-Pierre Changeux et Guillaume Dumas (2022). « Multilevel development of cognitive abilities in an artificial neural network ». In : *Proceedings of the National Academy of Sciences* 119.39, e2201304119.

Waldrop, M Mitchell (2016). « The chips are down for Moore's law ». In : *Nature News* 530.7589, p. 144.

Wang, Zhengwei, Qi She et Tomas E Ward (2021). « Generative adversarial networks in computer vision : A survey and taxonomy ». In : *ACM Computing Surveys (CSUR)* 54.2, p. 1-38.

Yger, Pierre, Marcel Stimberg et Romain Brette (2015). « Fast learning with weak synaptic plasticity ». In : *Journal of Neuroscience* 35.39, p. 13351-13362.

Yu, Jiahui et al. (2022). « Coca : Contrastive captioners are image-text foundation models ». In : *arXiv preprint arXiv :2205.01917*.

Zhang, Chiyuan et al. (2021). « Understanding deep learning (still) requires rethinking generalization ». In : *Communications of the ACM* 64.3, p. 107-115.

Zhang, Wei et David J Linden (2003). « The other side of the engram : experience-driven changes in neuronal intrinsic excitability ». In : *Nature Reviews Neuroscience* 4.11, p. 885.

Zhao, Bo et al. (2015). « Feedforward Categorization on AER Motion Events Using Cortex-Like Features in a Spiking Neural Network. » In : *IEEE Trans. Neural Netw. Learning Syst.* 26.9, p. 1963-1978.

Chapitre 8

Annexe

Données événementielles synthétiques

La structure des données événementielles peut être représentée par 2 tableaux parallèles, l'un symbolisant les coordonnées spatiales sous formes d'indices, l'autre correspondant aux moments où les évènements se sont produits. Si on représente des données événementielles sous forme d'une séquence d'images matricielles (voir FIGURE 4.1), les coordonnées spatiales (x, y) peuvent être facilement aplatit sous forme d'indices par le calcul $i = y.w + x$, la largeur de la fenêtre étant noté w. Le TABLEAU 1 montre un exemple basé sur la FIGURE 1.

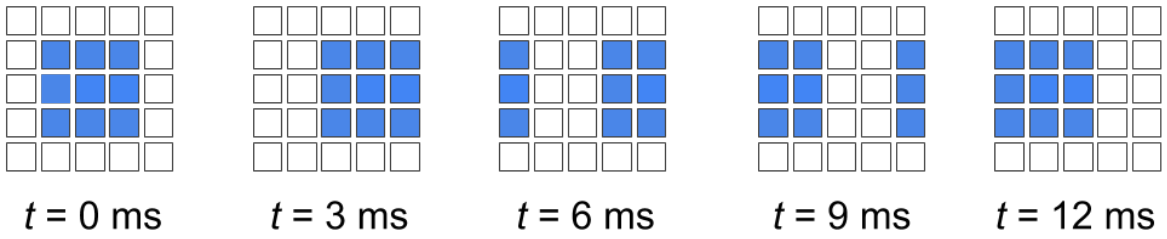


FIGURE 1 – Mouvement vers l'EST d'un rectangle 1×2 sur 15 ms dans une fenêtre torique de résolution 5×5.

Indices (i)	12	13	13	14	14	10	10	11	11	12
Temps (t)	3 ms	3 ms	6 ms	6 ms	9 ms	9 ms	12 ms	12 ms	15 ms	15 ms

TABLEAU 1 – Données AER représentant la séquence illustrée dans la FIGURE 1.