Plasticité et codage temporel dans les réseaux impulsionnels appliqués à l'apprentissage de représentations

Adrien Fois

UNIVERSITÉ DE LORRAINE

École doctorale IAEM Lorraine

Plasticité et codage temporel dans les réseaux impulsionnels appliqués à l'apprentissage de représentations

THÈSE

présentée et soutenue publiquement le 15 Décembre 2022

pour l'obtention du

Doctorat de l'Université de Lorraine

(mention informatique)

par

Adrien Fois

Composition du jury

Président :	Jean Martinet, Professeur
Rapporteurs :	Timothée Masquelier, Directeur de recherche
	Madalina Olteanu, Professeure
Examinateurs :	Laurent Perrinet, Directeur de recherche
	Olivier Bichler, Ingénieur chercheur
	Jean Martinet, Professeur
Directeur de thèse :	Bernard Girau, Professeur

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Mis en page avec la classe thesul.

Résumé

Le calcul neuromorphique est un domaine de l'informatique en plein essor. Il cherche à définir des modèles de calculs s'inspirant des propriétés du cerveau. Le calcul neuromorphique redéfinit la nature des trois composants clés de l'apprentissage : 1) données, 2) substrat de calcul et 3) algorithmes, en se fondant sur le fonctionnement du cerveau.

Premièrement, les données sont représentées avec des événements tout ou rien distribués dans l'espace et le temps : les impulsions neuronales.

Deuxièmement, le substrat de calcul efface la séparation entre calcul et mémoire introduite par les architectures de Von Neumann en les co-localisant, comme dans le cerveau. En outre, le calcul est massivement parallèle et asynchrone permettant aux unités computationnelles d'être activées à la volée, de façon indépendante.

Troisièmement, les algorithmes d'apprentissage sont adaptés au substrat en exploitant les informations disponibles localement, au niveau du neurone.

Ce vaste remaniement dans la manière d'appréhender la représentation et le transfert de l'information, le calcul et l'apprentissage, permettent aux processeurs neuromorphiques de promettre notamment un gain d'énergie d'un facteur considérable de 100 à 1000 par rapport aux CPU.

Dans cette thèse, nous explorons le versant algorithmique du calcul neuromorphique en proposant des règles d'apprentissage événementielles répondant aux contraintes de localité et capables d'extraire des représentations de flux de données impulsionnels, épars et asynchrones. En outre, alors que la plupart des travaux connexes se basent sur des codes par taux de décharge où l'information est exclusivement représentée dans le nombre d'impulsions, nos règles d'apprentissage exploitent des codes temporels beaucoup plus efficients, où l'information est contenue dans les temps d'impulsions.

Nous proposons d'abord une analyse approfondie d'une méthode de codage temporel par population de neurones, en proposant une méthode de décodage, et en analysant l'information délivrée et la structure du code.

Puis nous introduisons une nouvelle règle événementielle et locale capable d'extraire des représentations de codes temporels en stockant des centroïdes de manière distribuée dans les poids synaptiques d'une population de neurones.

Nous accentuons ensuite la nature temporelle de l'apprentissage en proposant d'apprendre des représentations non pas dans les poids synaptiques, mais dans les délais de transmission opérant intrinsèquement dans la dimension temporelle. Cela a engendré deux nouvelles règles événementielles et locales. Une règle adapte les délais de sorte à stocker des représentations, l'autre règle adapte les poids de sorte à filtrer les caractéristiques en fonction de leurs variabilité temporelle. Ces deux règles opèrent de manière complémentaire.

Dans un dernier modèle, ces règles adaptant poids et délais sont augmentées par un nouveau neuromodulateur spatio-temporel. Ce neuromodulateur permet au modèle de reproduire le comportement des cartes auto-organisatrices dans un substrat impulsionnel, aboutissant ainsi à la génération de cartes ordonnées lors de l'apprentissage de représentations.

Enfin nous proposons une nouvelle méthode générique d'étiquetage et de vote conçue pour des réseaux de neurones impulsionnels traitant des codes temporels. Cette méthode nous permet d'évaluer notre dernier modèle sur des tâches de catégorisation.

Mots-clés: réseaux de neurones impulsionnels, apprentissage de représentations, STDP, codage temporel

Abstract

Neuromorphic computing is a rapidly growing field of computer science. It seeks to define models of computation inspired by the properties of the brain. Neuromorphic computing redefines the nature of the three key components of learning : 1) data, 2) computing substrate, and 3) algorithms, based on how the brain works.

First, the data are represented with all-or-nothing events distributed in space and time : spikes.

Second, the computational substrate erases the separation between computation and memory introduced by Von Neumann architectures by co-locating them, as in the brain. Furthermore, the computation is massively parallel and asynchronous allowing the computational units to be activated on the fly, independently.

Third, the learning algorithms are adapted to the computing substrate by exploiting the information available locally, at the neuron level.

This vast overhaul in the way information transfer, information representation, computation and learning are approached, allows neuromorphic processors to promise in particular an energy saving of a considerable factor of 100 to 1000 compared to CPUs.

In this thesis, we explore the algorithmic side of neuromorphic computing by proposing event-driven learning rules that satisfy locality constraints and are capable of extracting representations of event-based, sparse and asynchronous data streams. Moreover, while most related studies are based on rate codes where information is exclusively represented in the number of spikes, our learning rules exploit much more efficient temporal codes, where information is contained in the spike times.

We first propose an in-depth analysis of a temporal coding method using a population of neurons. We propose a decoding method and we analyze the delivered information and the code structure.

Then we introduce a new event-driven and local rule capable of extracting representations from temporal codes by storing centroids in a distributed way within the synaptic weights of a neural population.

We then propose to learn representations not in synaptic weights, but rather in transmission delays operating intrinsically in the temporal dimension. This led to two new event-driven and local rules. One rule adapts delays so as to store representations, the other rule adapts weights so as to filter features according to their temporal variability. The two rules operate complementarily.

In a last model, these rules adapting weights and delays are augmented by a new spatio-temporal neuromodulator. This neuromodulator makes it possible for the model to reproduce the behavior of self-organizing maps with spiking neurons, thus leading to the generation of ordered maps during the learning of representations.

Finally, we propose a new generic labeling and voting method designed for spiking neural networks dealing with temporal codes. This method is used so as to evaluate our last model in the context of categorization tasks.

Keywords: spiking neural networks, representation learning, STDP, temporal coding

Remerciements

Je tiens tout d'abord à remercier mon directeur de thèse, Bernard Girau, pour son soutien constant et ses précieux conseils tout au long de ce projet de recherche. Son expertise et sa disponibilité ont été essentielles à la réalisation de cette thèse. Je lui suis profondément reconnaissant pour son temps et son énergie dévoués à mon travail.

Mes remerciements vont aussi à mes chers collègues de bureau, en particulier à Pierre et Noémie. Nos discussions, nos débats, notre solidarité, ont rendus cette aventure enthousiasmante et intellectuellement stimulante.

Je remercie également ma famille et mes amis, je pense notamment à mes amis nancéiens Manon et Mehdi, pour leur soutien indéfectible et leur amitié fidèle. Leurs encouragements et leur soutien ont été une source de réconfort et de motivation précieuse pour moi. J'apprécie grandement leur présence dans ma vie.

Enfin, je tiens à remercier l'ensemble de l'équipe Biscuit ainsi que les membres de mon jury qui m'ont fait l'honneur de consacrer une partie de leur temps et de leur énergie pour l'évaluation de mes travaux.

Table des matières

Glossaire

ANN *Artificial Neural Networks.*

BMU *Best Matching Unit.*

CNN *Convolutional Neural Networks.*
CPU *Central Processing Unit.*

GPU *Graphics Processing Unit.*

IF *Integrate and Fire.*

k-WTA *k-Winners-Take-All.*

LIF *Leaky Integrate and Fire.*

MDN *Mean Distance to Neurons.*
MDS *MultiDimensional Scaling.*

NSTDP *nonweight-dependent Spike-Timing Dependent Plasticity.*

PSNR *Peak Signal to Noise Ratio.*

QV Quantification Vectorielle.

RMS *Root Mean Squared.*

SBMU *Spiking Best Matching Unit.*
SNN *Spiking Neural Networks.*
SO-TCRL *Self-Organizing Temporally Coded Representation Learning.*
SOM *Self-Organizing Map.*
SPK-POP *Spike Population.*
SSIM *Structural Similarity Index.*
STDP *Spike-Timing Dependent Plasticity.*

TMP-POP *Temporal Population.*
TPE *Tree-Structured Parzen Estimator.*

W-TCRL *Weight - Temporally Coded Representation Learning.*

WBCD *Wisconsin Breast Cancer Dataset.*

WD-TCRL *Weight Delay - Temporally Coded Representation Learning.*

WSTDP *weight-dependent Spike-Timing Dependent Plasticity.*

WTA *Winner-Take-All.*

Chapitre 1

Introduction

Scrute la nature, c'est là qu'est ton futur

Léonard de Vinci

D'un point de vue computationnel, les systèmes biologiques excellent à encoder, transférer et extraire les informations pertinentes de leur environnement. Leurs performances, fruits de millions d'années d'évolution, sont si époustouflantes qu'elles approchent dans de nombreux cas l'optimalité, i.e. les limites fixées par les lois de la physique (Bialek, 2012). Les exemples de systèmes biologiques dont les performances approchent les limites physiques sont abondants.

Un premier exemple provient des embryons de drosophiles (Dubuis et al., 2013). Dans un embryon de drosophile en développement, chaque cellule doit correctement se positionner sur son emplacement attitré. Cela permet de construire le plan rudimentaire de la drosophile. Pour correctement se positionner dans l'embryon, chaque cellule mesure localement un signal encodant l'information positionnelle. Ce signal consiste en une concentration de molécules de facteurs de transcription, concentration mesurée localement par les capteurs biologiques de la cellule. Théoriquement, la concentration de ces molécules permet à la cellule d'atteindre une erreur de positionnement d'environ 1 %. Cette erreur théorique vient du bruit de la mesure induit par la très faible concentration de molécules, de l'ordre de la nanomole. Malgré ce signal très faible, la précision est théoriquement suffisante pour obtenir une relation d'équivalence entre concentration de molécules et position (relativement à la largeur d'une cellule). Or, les données expérimentales attestent que cet optimum en termes de précision dans le positionnement des cellules dans l'embryon est atteint. Autrement dit, les performances réelles du système biologique sont proches d'un optimum théorique fixé par les limites physiques.

La rétine peut aussi être citée parmi la myriade d'exemples illustrant la quasi-optimalité de la précision et de la fiabilité atteinte par des systèmes biologiques (Pugh, 2018). La rétine des vertébrés comporte des millions de bâtonnets, récepteurs sensoriels sensibles à l'intensité lumineuse. Les bâtonnets produisent un signal continu en sortie qui est fonction de l'intensité lumineuse reçue en entrée. L'intensité lumineuse est proportionnelle au nombre de photons frappant une surface par seconde. Un bâtonnet présente la remarquable capacité de détecter fiablement un unique photon en entrée, particule élémentaire de lumière dans l'univers. Il est impossible de faire mieux que de détecter un unique photon. En parallèle de l'extraordinaire performance des

bâtonnets à collecter l'intensité lumineuse, les neurones de la rétine ont été optimisés pour encoder avec une grande fidélité le signal continu produit par les bâtonnets, via des potentiels d'actions discrets propagés dans le cerveau. Ainsi la capacité de collecter et de traiter l'intensité lumineuse est telle qu'un humain plongé dans l'obscurité est capable de détecter une poignée de photons absorbés par sa rétine.

Les incroyables performances des systèmes biologiques motivent le développement de systèmes artificiels reproduisant ou imitant leurs fonctions. Le succès de la démarche bio-inspirée a été maintes fois éprouvé, en particulier en qui concerne la discipline au coeur de ce manuscrit : l'apprentissage automatique.

Un exemple saillant provient des réseaux de neurones convolutionnels (CNN). Les CNN ont engendré une révolution pour le traitement d'images (LeCun et al., 1999). Leurs genèse est inspirée de la structure hiérarchique (ascendante) du cortex visuel et des champs récepteurs des neurones traitant une petite partie du champs visuel, plutôt que son intégralité (Hubel et Wiesel, 1959). Les CNN dépassent à présent les performances humaines pour des tâches de catégorisation visuelle, et leur domaine d'applications à été étendu à un large éventail de tâches tel que le traitement automatique du langage naturel. Les CNN représentent à présent un standard pour l'apprentissage profond (*deep learning*).

Un autre exemple de démarche bio-inspirée ayant produit une percée majeure en apprentissage automatique provient des mécanismes attentionnels. L'idée de l'attention, mécanisme permettant de sélectionner des informations et des calculs à opérer, a permis une révolution dans la traduction automatique (Bahdanau et al., 2015), réduisant très significativement l'écart avec les performances humaines. Les mécanismes attentionnels ont ensuite été appliqués à un large éventail de tâches tel que la génération de descriptions à partir d'images (*image captioning*). Tout comme les CNN, les mécanismes attentionnels sont devenus un standard pour l'apprentissage profond.

Cependant les bonnes performances délivrées par ces réseaux de neurones profonds ne transcrivent qu'une partie de l'histoire. Bien qu'ils s'inspirent conceptuellement du cerveau, la puissance de calcul ainsi que la consommation énergétique requise pour obtenir ces performances croît exponentiellement d'année en année (voir Figure 1.1).

Un exemple récent provient du modèle linguistique GPT3 de OpenAI [1]. Il s'agit d'un réseau de neurones profond constitué de 175 millards de paramètres plastiques. Le réseau a été entraîné sur une base de données de 500 milliards de mots, équivalente à un dixième des mots contenus dans la totalité des livres (130 millions) publiés depuis l'invention de l'imprimerie par Gutenberg en 1440. En outre, l'entrainement du réseau a coûté l'équivalent de la consommation énergétique d'un cerveau humain (20 W) accumulée durant 5300 ans, nous faisant remonter à l'invention de l'écriture. Cette consommation de données, de paramètres et d'énergie permet à GPT3 d'atteindre des performances proches de celles d'un humain pour des tâches linguistiques.

Les réseaux de neurones profonds délivrent ainsi de hautes performances pour des tâches spécifiques, mais pour un coût de plus en plus prohibitif. Ces limites appellent un changement de paradigme. Appeler à un changement de paradigme implique un remaniement complet de la nature des trois composants clés de l'apprentissage : 1) données, 2) substrat de calcul (calcul et mémoire) et 3) algorithmes.

Ce changement de paradigme peut être initié par une démarche poussant le curseur de la bio-inspiration bien plus loin, visant à imiter le meilleur système intelligent connu et à l'efficience

1. https ://web.archive.org/web/20210413021003/https ://lambdalabs.com/blog/demystifying-gpt-3/

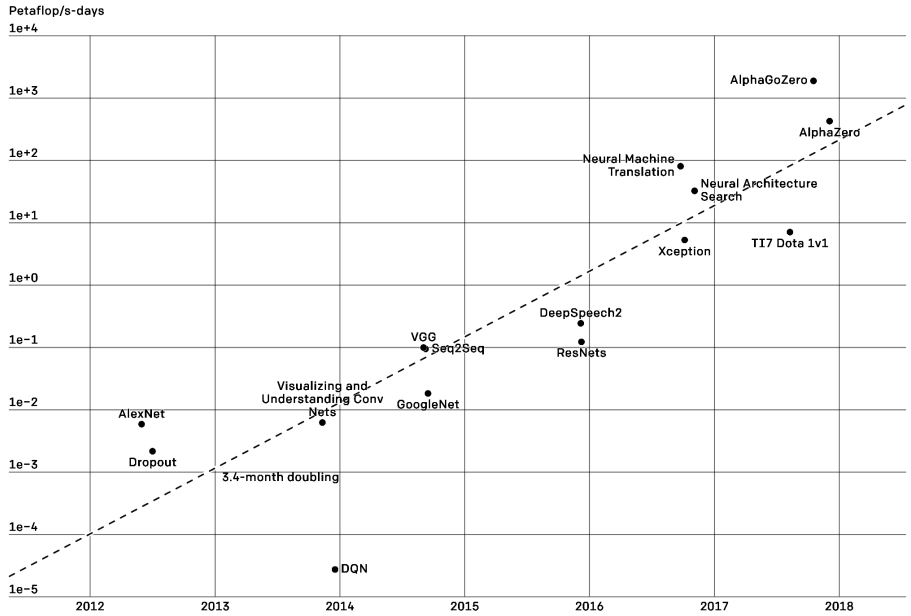

FIGURE 1.1 – La puissance de calcul requise pour l'entrainement des réseaux de neurones artificiels profonds a augmenté de façon exponentielle durant la dernière décennie. Figure tirée de `https://openai.com/blog/ai-and-compute/`

encore inégalée : le cerveau.

Imiter le cerveau est précisément l'objet du calcul neuromorphique. Le calcul neuromorphique vise à implémenter *in silico* des fonctions du cerveau. Cela est réalisé en modifiant les trois composants clés de l'apprentissage automatique :

1. Des données événementielles tout ou rien (1 ou 0) transmises de façon asynchrone et éparse (matrice creuse) : les impulsions neuronales, plutôt que des données à valeurs réelles transmises de façon synchrone et dense (matrice dense) comme dans les *Artificial Neural Networks* (ANN).
2. Un substrat de calcul massivement parallèle co-localisant calcul (neurone) et mémoire (synapse) : les processeurs neuromorphiques, plutôt que de les séparer comme dans les *Central Processing Unit* (CPU) basés sur des architectures de Von Neumann. En outre, ces processeurs neuromorphiques fonctionnent de manière asynchrone, en synergie avec les flux de données événementiels, contrairement au fonctionnement synchrone des CPU et *Graphics Processing Unit* (GPU).
3. Des algorithmes fonctionnant en synergie avec le niveau matériel co-localisant calcul et mémoire, en utilisant des informations localement disponibles pour actualiser les paramètres soumis à apprentissage, plutôt que des techniques nécessitant une connaissance globale ou partielle du réseau. En outre, ces algorithmes doivent permettre d'actualiser les paramètres à la volée, i.e. de façon asynchrone et indépendante, plutôt que d'actualiser l'ensemble des paramètres du réseau de façon synchrone à chaque itération comme pour la rétropropagation de l'erreur.

Les processeurs neuromorphiques implémentent *in silico* des millions de neurones impulsion-

nels. Les neurones impulsionnels sont considérés comme la troisième génération de neurones (Maass, 1997), dont les capacités computationnelles sont aussi puissantes, voire plus puissantes dans certains cas, que les autres types de réseaux de neurones (neurones à fonction d'activation Heaviside et sigmoïde). Contrairement aux neurones artificiels des ANN, les neurones impulsionnels des *Spiking Neural Networks* (SNN) ont une dynamique : leur évolution dépend explicitement du temps. Le temps - complètement ignoré dans les générations de neurones précédentes - introduit ainsi une nouvelle dimension au calcul, augmentant son expressivité. En outre, bien que l'évolution des variables d'états des neurones impulsionnels soit continue, leur sortie est un évènement binaire émis dans le temps, plutôt qu'un nombre réel comme pour les neurones artificiels.

Ce vaste remaniement dans la manière d'appréhender la représentation et le transfert de l'information, le calcul et l'apprentissage, permet aux processeurs neuromorphiques de promettre un gain d'énergie d'un facteur considérable de 100 à 1000 par rapport aux CPU. Ils promettent également d'envisager des architectures à très large échelle, éliminant le goulot d'étranglement des architectures de Von Neumann en co-localisant calcul et mémoire. Le rêve poursuivi est d'atteindre à terme les capacités computationnelles d'un cerveau humain, dont l'estimation basse en termes d'opérations par seconde (FLOPS) est de 600 petaFlops [2], équivalente aux plus puissants supercalculateurs du monde, pour une consommation énergétique d'à peine 20 W, équivalente à une ampoule électrique. Parmi les processeurs neuromorphiques les plus importants, nous pouvons citer SpiNNaker de l'université de Manchester (Painkras et al., 2013), TrueNorth d'IBM (Akopyan et al., 2015) et Loihi d'Intel (Davies et al., 2018). Loihi2 intègre un million de neurones impulsionnels et 120 millions de synapses programmables.

Le gain en termes de puissance de calcul et de consommation énergétique apporté par les processeurs neuromorphiques est appelé à révolutionner le domaine de l'apprentissage automatique et ouvre la voie à l'apprentissage en ligne et embarqué sur les robots ou les *smartphones*.

Cette thèse s'inscrit dans le paradigme du calcul neuromorphique, dans son versant algorithmique d'apprentissage non-supervisé de représentations.

1.1 Objectifs

Un premier défi est de développer des règles d'apprentissage événementielles permettant à des neurones impulsionnels d'extraire des représentations de motifs impulsionnels, asynchrones et épars, dans la perspective d'obtenir une faible erreur de reconstruction.

Un deuxième défi est la contrainte de localité. Le principe de localité permet une convergence *top-down* (algorithmique) et *bottom-up* (matérielle) grâce à l'utilisation de règles événementielles, spatialement et temporellement locales.

Les méthodes locales existantes capables d'extraire des représentations de motifs impulsionnels sont basées sur des données codées en taux de décharge. Le codage par taux de décharge considère uniquement le nombre d'impulsions émis comme moyen de représenter l'information. Un autre paradigme encore minoritaire, le codage temporel, considère les temps d'impulsions comme moyen de représenter l'information. La dimension temporelle devient ainsi le support clé pour représenter l'information.

La façon dont l'information est encodée est absolument cruciale. Le gain promis par les codes

2. Cette estimation est faite en calculant chaque milliseconde l'état du cerveau constitué d'environ 86 milliards de neurones et de 7000 synapses par neurone. Cela donne donc $8.6\text{E+}10 * 7.0\text{E+}3 * 1.0\text{E+}3 = 6.02\text{e+}17$ FLOPS.

temporels relativement aux codes par taux de décharge est exponentiel en termes de vitesse de transmission de l'information, ainsi que de nombre d'impulsions émis et donc de consommation énergétique induite. Tandis que les codes par taux de décharge sont la norme, l'utilisation de codes temporels permet d'envisager des gains de performances très significatifs pour les processeurs neuromorphiques. Les avantages conséquents offerts par les codes temporels nous motivent à les adopter.

Le troisième défi est ainsi de développer des règles d'apprentissage capables d'extraire des représentations de codes temporels, tout en maintenant des performances comparables aux méthodes basées sur des codes par taux de décharge.

En résumé, la conjonction des trois défis donne l'objectif de développer des règles d'apprentissage événementielles, spatialement et temporellement locales, capables d'extraire efficacement des représentations de codes temporels.

1.2 Contributions principales et organisation du manuscrit

Le chapitre 2 expose le contexte de cette thèse en commençant par une présentation et une étude comparative des principaux codes neuronaux. Puis les principaux modèles d'apprentissage auto-organisés impulsionnels et non-impulsionnels sont passés en revue. Enfin nous présentons les modèles de synapses et de neurones utilisés dans ce manuscrit, pour finir par un aperçu des processeurs neuromorphiques.

Après avoir présenté le contexte, les contributions principales sont détaillées au fil des quatre chapitres suivants :

— Le chapitre 3 propose une analyse approfondie d'une méthode de codage temporel par population de neurones, introduit une méthode de décodage, et analyse l'information délivrée et la structure du code.

— Dans le chapitre 4, un premier modèle *Weight - Temporally Coded Representation Learning* (W-TCRL) basé sur de nouvelles règles événementielles et locales est proposé. Ce modèle est capable d'extraire des représentations de codes temporels en stockant des centroïdes dans les poids synaptiques. WD-TCRL améliore significativement l'état de l'art en termes d'erreur de reconstruction, tout en opérant avec des codes temporels plutôt qu'avec des codes en taux de décharge. Il s'agit à notre connaissance du premier modèle événementiel et local capable d'extraire des représentations de codes temporels en visant une faible erreur de reconstruction.

— Dans le chapitre 5, un deuxième modèle *Weight Delay - Temporally Coded Representation Learning* (WD-TCRL) est présenté. Ce second modèle est basé sur de nouvelles règles événementielles et locales, capable d'extraire des représentations de codes temporels en stockant des centroïdes non pas dans les poids synaptiques, mais dans les délais de transmission. L'adaptation des délais active un mode opératoire clé des neurones impulsionnels : la détection de coïncidences temporelles. En outre, contrairement aux poids, les délais opèrent intrinsèquement dans la dimension temporelle, ce qui les rend particulièrement adaptés au traitement de codes temporels. Une seconde règle événementielle et locale adapte les poids synaptiques en fonction de la variabilité temporelle des caractéristiques. Les règles d'adaptation des poids et des délais opèrent de manière conjointe. WD-TCRL délivre des performances similaires à W-TCRL en termes d'erreur de reconstruction. WD-TCRL est à notre connaissance le premier modèle événementiel et local capable d'extraire

des représentations de codes temporels et de les stocker dans les délais plutôt que dans les poids synaptiques, en visant une faible erreur de reconstruction.

— Dans le chapitre 6, un troisième modèle *Self-Organizing Temporally Coded Representation Learning* (SO-TCRL) étend le modèle WD-TCRL du chapitre 5 avec un nouveau neuromodulateur, de sorte à générer des cartes topographiquement ordonnées. La génération de cartes topographiquement ordonnées permet d'étendre le principe de localité au voisinage spatial des neurones, de sorte que des neurones spatialement proches partagent des représentations proches dans l'espace d'entrée. SO-TCRL est à notre connaissance le premier modèle complet de *Self-Organizing Map* (SOM) impulsionnelle conjuguant 1) une capacité de quantification vectorielle, 2) de génération de cartes ordonnées et 3) de catégorisation pour des bases de données génériques.

— Toujours dans le chapitre 6, une nouvelle méthode générique d'étiquetage et de vote *Temporal Population* (TMP-POP) est introduite pour des SNN traitant des codes temporels. TMP-POP est une méthode utilisée pour la catégorisation assignant des scores de confiances aux neurones en fonction de leurs temps d'impulsions relatifs. Cette méthode extrait ainsi l'information contenue dans les motifs impulsionnels spatio-temporels de la couche de sortie pour des tâches de catégorisation. TMP-POP est utilisée pour évaluer notre nouveau modèle de SOM impulsionnelle, la SO-TCRL, sur des tâches de catégorisation.

Enfin nous résumons les développements de ces différents chapitres dans le chapitre 7. Nous discutons également des perspectives offertes par les différents modèles, notamment en termes d'implémentation matérielle dans des processeurs neuromorphiques.

Chapitre 2

Contexte

Sommaire

2.1 Codes par taux de décharge et codes temporels

Les stimuli nous parvenant de notre environnement extérieur sont essentiellement de nature continue. Par exemple les sons sont des variations continues de la pression de l'air.

Par contraste, le cerveau communique avec des évènements pouvant être considérés comme binaires (0 ou 1) : les impulsions neuronales. Toutes les perceptions comme les sons, les odeurs, les couleurs, sont représentées par des motifs impulsionnels se déployant dans l'espace (le réseau de neurones) et le temps (le moment où sont émises les impulsions).

Le pont entre le domaine physique et le domaine biologique est réalisé par les neurones sensoriels. Les neurones sensoriels transforment par transduction les signaux physiques en motifs impulsionnels traitables et interprétables par le cerveau. Ils transforment ainsi des entrées continues dans des motifs impulsionnels discrets. La finesse de la discrétisation des entrées implique ultimement une capacité à finement discriminer les états de l'environnement et à décider des actions adéquates. La transformation d'entrées continues en impulsions discrètes est nommée encodage neuronal. La nature de l'encodage neuronal est un problème central en neurosciences.

Il existe deux paradigmes pour l'encodage neuronal, souvent présentés de façon antagonique : le codage par taux de décharge et le codage temporel. Cette section ne vise pas tant à nourrir ce débat en neurosciences qu'à exposer les avantages et les inconvénients des principaux codes appartenant à ces deux paradigmes d'un point de vue computationnel. Ces codes neuronaux sont également couramment utilisés dans un contexte d'apprentissage automatique (*machine learning*) par apprentissage supervisé (Kheradpisheh et Masquelier, 2020 ; C. Lee et al., 2020), par renforcement (Mozafari et al., 2018 ; Patel et al., 2019), ou non-supervisé (Diehl et Cook, 2015 ; Kheradpisheh, Ganjtabesh et al., 2018).

2.1.1 Codage par taux de décharge des entrées

Une observation importante menant au codage par taux de décharge est qu'un neurone émet un nombre d'impulsions dépendant de l'intensité du stimulus qu'il reçoit en entrée. Plus l'intensité du stimulus reçu est élevée, plus le nombre d'impulsions émis est élevé.

Le codage par taux de décharge considère la dimension temporelle comme un simple medium pour le transfert d'informations. Autrement dit, les temps d'impulsions sont considérés comme aléatoires, obéissant généralement à une loi de Poisson pour un stimulus constant, et n'ont pas d'incidence sur l'information transmise. Ainsi les temps d'impulsions ne sont pas reproductibles pour un stimulus présenté répétitivement en entrée. A contrario, le nombre d'impulsions est beaucoup moins variable et donc beaucoup plus fiable. Ainsi pour le codage par taux de décharge la seule variable d'encodage considérée comme pertinente, car fiable, est le nombre d'impulsions émis (Georgopoulos, Kalaska et al., 1982).

Ce codage présente un intérêt certain dans le cas où les temps d'impulsions ne sont pas reproductibles ou fiables, e.g. si le bruit est prépondérant.

Codage par taux de décharge avec un unique neurone

Une première façon d'instancier ce code est de considérer un unique neurone encodeur. Plus précisément, il s'agit d'encoder une entrée par la moyenne temporelle ν du nombre d'impulsions n émises durant une fenêtre temporelle T (voir Fig. 2.1) :

$$\nu = n/T$$

Par exemple, pour $n = 4$ impulsions et une fenêtre temporelle $T = 0.1$ s, le taux de décharge $\nu = 40$ Hz.

Considérons un nombre de pas de temps $\Delta = T/dt$ avec T étant la fenêtre temporelle et dt le pas de temps. L'information transmise par ce code est alors de $\log_2(\Delta + 1)$ bits. Par exemple pour $T = 10$ ms et $dt = 1$ ms, le nombre de pas de temps $\Delta = 10$ et le nombre d'états disponibles pour encoder l'entrée est alors de $\Delta + 1 = 11$ en considérant le cas où le neurone n'émet aucune impulsion. Donc le nombre de bits transmis est de $b = \log_2(\Delta + 1) = \log_2(11) \approx 3.5$ bits.

L'inconvénient d'une telle approche est qu'il est nécessaire d'augmenter significativement la largeur de la fenêtre temporelle T pour augmenter la résolution en bits b de l'entrée. En effet, pour une résolution de l'entrée de b bits, il faut un nombre de pas de temps $\Delta = 2^b - 1$ et donc une fenêtre temporelle d'encodage $T = (2^b - 1)dt$. Cela signifie que la vitesse de transfert de l'information décroit exponentiellement relativement à la résolution en bits b de l'entrée désirée.

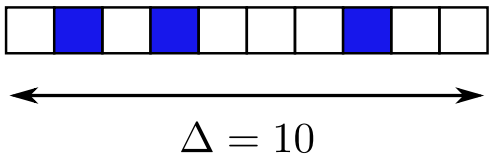

FIGURE 2.1 – Codage de l'entrée par le taux de décharge d'un unique neurone. Le taux de décharge est la moyenne du nombre d'impulsions émis par un unique neurone sur une fenêtre temporelle discrétisée en Δ pas de temps. Les cases bleues représentent les moments où une impulsion est émise au sein de la fenêtre. Notons que cette discrétisation du temps usuelle en simulation trouve son corrélat dans le domaine biologique. En effet, bien que le temps soit continu, les temps d'impulsions ont une précision limitée d'un ordre de 1 ms à 0.1 ms, justifiant la discrétisation du temps.

Ainsi, pour passer d'une résolution de $b = 11$ bits à $b = 12$ bits avec $dt = 1$ ms, il faut doubler T de $T \approx 2$ s à $T \approx 4$ s, ce qui revient à passer d'une vitesse de transfert d'environ 5.5 bits/s à 3 bits/s.

Codage par taux de décharge avec une population de neurones

Une stratégie pour atténuer le problème de la croissance exponentielle de T relativement à b est de calculer le taux de décharge non pas avec un unique neurone, mais avec une population de l neurones. Autrement dit c'est l'activité collective de la population de neurones qui rend compte du taux de décharge (voir Fig. 2.2). On parle alors de codage par population (EBITZ et HAYDEN, 2021).

Soit T la largeur de la fenêtre temporelle d'encodage. Il s'agit de compter le nombre d'impulsions $n_{\text{pop}}(T)$ émis par la population de neurones durant l'intervalle de temps T. Le nombre d'impulsions est divisé par le nombre de neurones l pour obtenir le nombre d'impulsions moyen par neurone, puis ce résultat est divisé par l'intervalle de temps T pour obtenir le taux de décharge ν_{pop} :

$$\nu_{\text{pop}} = \frac{1}{T}\frac{n_{\text{pop}}(T)}{l}$$

A présent examinons le nombre de bits transmis par un codage par taux distribué sur une population de neurones. Soit les mêmes paramètres que pour le codage avec un unique neurone, i.e. $T = 10$ ms et $dt = 0.1$ ms et donc $\Delta = 10$. Considérons à présent non pas un unique neurone $l = 1$ mais 10 neurones $l = 10$. Le nombre d'états disponibles pour encoder l'entrée est alors de $1 + l\Delta = 101$. Donc le nombre de bits transmis est de $b = \log_2(1 + l\Delta) = \log_2(101) \approx 6.6$ bits.

Analysons maintenant l'influence du nombre de neurones l sur la vitesse de transfert de l'information. Afin de se comparer avec les résultats obtenus avec un unique neurone, considérons de nouveau les mêmes paramètres. Pour une résolution de l'entrée de b bits, il faut un nombre de pas de temps $\Delta = (2^b/l) - 1$ et donc une fenêtre temporelle d'encodage $T = ((2^b/l) - 1)dt$. Ainsi, pour passer d'une résolution de $b = 11$ bits à $b = 12$ bits avec $l = 10$ neurones et $dt = 1$ ms, il faut passer de $T \approx 0.2$ s à $T \approx 0.4$ s, ce qui revient à passer d'une vitesse de transfert d'environ 55 bits/s à 30 bits/s. Comparativement à un unique neurone, nous avons ainsi augmenté la vitesse de transfert de l'information d'un facteur de 10, égal au nombre de neurones l considéré.

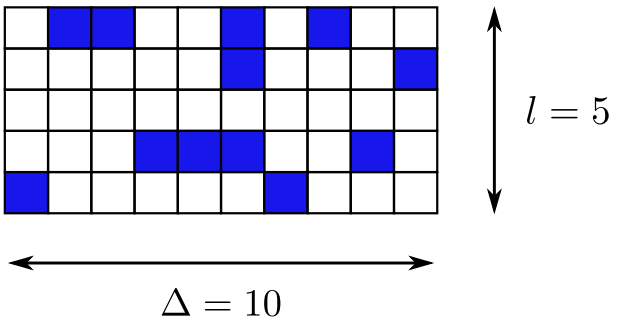

FIGURE 2.2 – Codage de l'entrée par le taux de décharge d'une population de l neurones. Le taux de décharge est la moyenne de l'activité d'une population de l neurones sur une fenêtre temporelle T discrétisée en Δ pas de temps. Les cases bleues représentent les moments où une impulsion est émise.

Pour une résolution en bits b fixée, nous pouvons conclure que la fenêtre d'encodage T nécessaire pour un codage avec un unique neurone peut être divisée par un facteur de l en utilisant une population de l neurones, puisque $T = ((2^b/l) - 1)dt$. Cela permet d'atténuer significativement le problème de la vitesse de transfert de l'information.

Cependant, un problème majeur demeure : bien qu'il soit possible de raccourcir la fenêtre d'encodage T grâce au codage par population, le nombre maximal d'impulsions émis croît toujours exponentiellement relativement à la résolution de l'entrée en bits b considérée. Rappelons que l'entrée est exclusivement codée dans une seule variable macroscopique discrète : le taux de décharge. Ainsi discrétiser l'entrée avec une résolution de b bits revient à un nombre maximal d'impulsions émis de $2^b - 1$, chaque valeur discrète de $n_{\text{pop}} \in [0, 2^b - 1]$ encodant une valeur discrète de la variable d'entrée. Le codage neuronal par taux de décharge, qu'il soit distribué ou non, devient ainsi rapidement extrêmement couteux en termes de nombre d'impulsions et donc de consommation énergétique et de besoin de communication.

2.1.2 Codage temporel des entrées

D'un point de vue biologique, la majorité des neurones déchargent de façon très éparse. La distribution des taux de décharge dans le cortex suit une distribution log-normale (BUZSÁKI et MIZUSEKI, 2014), i.e. seule une minorité de neurones déchargent à des taux importants. Le taux moyen de décharge d'un neurone dans le cortex est ainsi estimé à seulement 0.1-2 impulsions par seconde.

Le nombre d'impulsions émis par un neurone en réponse à un stimulus présenté en entrée peut descendre à une unique impulsion. Ainsi THORPE et GAUTRAIS, 1996 ont montré que le système visuel est capable de traiter des images naturelles en environ 100 ms. L'information visuelle doit traverser séquentiellement une dizaine de couches de traitements prenant chacune environ 10 ms de temps de traitement. Or, un neurone transmet entre 0 à 100 impulsions en 1 s. En considérant la contrainte d'une fenêtre temporelle de traitement de 10 ms, ils en déduisent que chaque neurone a seulement le temps d'émettre une unique impulsion, suggérant ainsi fortement un codage temporel plutôt qu'un codage par taux de décharge. De façon plus générale, un codage par taux de décharge peut être opérationnel si le stimulus extérieur est constant ou varie lentement, ce qui est rarement le cas dans des conditions écologiques.

Le nombre d'impulsions étant un facteur déterminant pour la consommation énergétique et les besoins de communication, nous souhaitons minimiser cette quantité et posons pour la suite que chaque neurone ne peut émettre qu'une unique impulsion pour encoder une entrée. Minimiser le nombre d'impulsions émis est particulièrement pertinent dans le contexte d'ultra-basse consommation du calcul neuromorphique (Akopyan et al., 2015 ; Davies et al., 2018).

Dans ce cadre, les codes temporels, basés sur les temps d'impulsions, semblent plus adaptés que les codes en taux de décharge, basés sur le nombre d'impulsions. Un autre argument en faveur des codes temporels - d'un point de vue biologique - provient de nombreuses observations empiriques démontrant que les temps d'impulsions des neurones du cortex peuvent être précis et reproductibles, avec une précision allant de 1 ms jusqu'à 0.1 ms (Brette et Guigon, 2003). En outre Petersen et al., 2001 ont estimé qu'environ 85 % de l'information dans le cortex somato-sensoriel du rat est transmise dans les temps d'impulsions, avec un rôle clé des premières impulsions.

Codage temporel avec un unique neurone

Un code temporel peut être réalisé avec un unique neurone. L'intervalle de temps pris (ou la latence) entre le moment où le stimulus est présenté et le moment où le neurone émet une impulsion encode la valeur du stimulus (voir Fig. 2.3). Plus l'intensité du stimulus est grande, plus la latence sera courte, car le neurone atteint plus rapidement son seuil de décharge. Par contraste, plus l'intensité du stimulus est faible, plus la latence sera longue. On parle de codage *first spike latency* ou *time to first spike* (Chase et Young, 2007), agissant comme un convertisseur intensité-latence. Remarquons que le temps d'impulsion n'est pas suffisant pour déterminer la latence et qu'il est nécessaire de connaitre explicitement le temps du début de la présentation du stimulus.

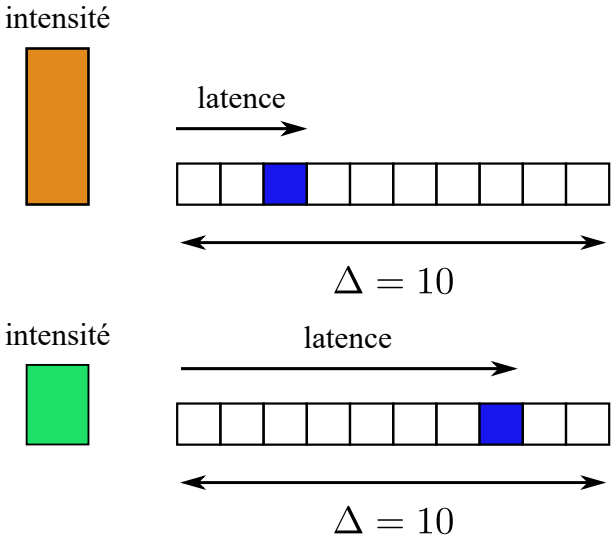

Figure 2.3 – Codage *first spike latency*. L'entrée est encodée dans la latence entre le moment où est présenté le stimulus et le moment où le neurone décharge. Plus l'intensité du stimulus est élevée, moins la latence entre le moment où le stimulus est présenté et le moment où le neurone décharge est grande.

A présent, examinons l'information transmise par ce codage temporel avec un unique neurone. Considérons un nombre de pas de temps $\Delta = T/dt$ où T est la fenêtre temporelle et dt le pas de temps. L'information transmise par ce code est alors de $\log_2(\Delta)$ bits. Par exemple pour $T = 10$ ms et $dt = 1$ ms, le nombre de pas de temps $\Delta = 10$ et le nombre d'états disponibles pour encoder l'entrée est alors de 10. Donc le nombre de bits transmis est de $b = \log_2(\Delta) = \log_2(10) \approx 3.3$ bits.

Le codage par latence avec un unique neurone nous permet donc d'obtenir un nombre de bits b équivalent au codage par taux de décharge avec un unique neurone. Un avantage clé est que l'utilisation des temps précis d'impulsions nous permet de n'émettre qu'une seule impulsion, tout en atteignant les mêmes performances.

Cependant l'inconvénient d'une telle approche est qu'il est nécessaire de réduire dt pour augmenter la résolution en bits b de l'entrée, ce qui est équivalent à dire qu'à dt constant, il faut augmenter T, tout comme pour le codage par taux de décharge avec un unique neurone. En effet, pour une résolution de l'entrée de b bits, il faut un nombre de pas de temps $\Delta = 2^b$ et donc une fenêtre temporelle d'encodage $T = 2^b dt$. Cela signifie que la vitesse de transfert de l'information décroit exponentiellement relativement à la résolution en bits b de l'entrée désirée. Ainsi, pour passer d'une résolution de $b = 11$ bits à $b = 12$ bits avec $dt = 1$ ms, il faut doubler T de $T \approx 2$ s à $T \approx 4$ s, ce qui revient à passer d'une vitesse de transfert d'environ 5.5 bits/s à 3 bits/s. Ainsi le codage *first spike latency* présente l'inconvénient d'induire des latences très longues en fonction de la résolution en bits b de l'entrée considérée.

Codage temporel avec une population de neurones

Considérons à présent l'utilisation d'une population de neurones pour augmenter la vitesse de transfert de l'information des codes temporels.

Codage par rang

Une première stratégie de codage par population est de n'utiliser les temps d'impulsions que pour déterminer l'ordre des impulsions au sein du motif impulsionnel. Cette stratégie est nommée codage par rang (Thorpe et Gautrais, 1998). Les temps précis des impulsions ne sont donc pas considérés, seul le rang des impulsions au sein du motif impulsionnel importe (voir Fig. 2.4). Cette perte d'information offre en contrepartie une invariance à des variations de contraste et de luminance d'images présentées en entrée (Thorpe et Gautrais, 1998), alors que de telles variations impacteraient significativement les temps d'impulsions.

A présent examinons le nombre de bits b transmis par un codage par rang. Considérons $l = 10$ neurones émettant chacun une impulsion. Le nombre d'états disponibles pour encoder l'entrée est alors de $l!$, impliquant un nombre de pas de temps minimal $\Delta = l$ pour encoder les $l!$ combinaisons. Remarquons qu'augmenter la largeur de T avec un codage par rang n'augmente pas le nombre de bits transmis. Ainsi il est optimal d'utiliser le plus petit T possible afin de transmettre les réalisations des $l!$ combinaisons le plus rapidement. Le nombre de bits b transmis est donc seulement dépendant du nombre de neurones l et est de $b = \log_2(l!) = \log_2(10!) \approx 21$ bits. Cela induit une vitesse de transfert de l'information d'environ 2.1 kbits/s avec $\Delta = l = 10$

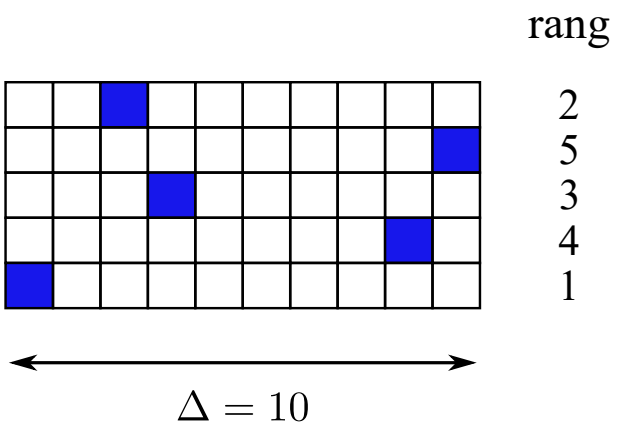

FIGURE 2.4 – Codage de l'entrée par le rangs des impulsions. Seul l'ordre ou le rang des impulsions est considéré pour encoder l'entrée, les temps précis des impulsions n'importent pas.

et un pas de temps $dt = 1$ms.

Codage par latences relatives

Une seconde stratégie est d'utiliser l'information collectivement contenue dans les temps précis des impulsions pour encoder l'entrée (voir Fig. 2.5). On parle de codage en latences relatives (GOLLISCH et MEISTER, 2008). Nous considérons ici les temps relatifs entre les impulsions, plutôt que les temps d'impulsions relativement au moment où le stimulus est présenté. On se place ainsi du point de vue d'un neurone recevant ces impulsions, plutôt que du point de vue d'un observateur extérieur au système ayant connaissance du moment où le stimulus est présenté en entrée.

Pour calculer la capacité de ce code, il est nécessaire de dénombrer le nombre d'états disponibles pour l'encodage. Des états sont redondants, e.g. un motif impulsionnel ayant subi une translation temporelle correspond aux mêmes temps d'impulsions relatifs et donc au même état. Il faut donc discerner les différents cas possibles.

Dans le cas où tous les l neurones déchargent en même temps, i.e. lorsque tout les neurones déchargent à $t + i$ avec $i = 0$, nous obtenons un unique code.

Un autre cas à considérer est lorsque tous les l neurones déchargent dans une fenêtre temporelle entre t (borne inférieure) et $t+i$ (borne supérieure), avec $i \in [1, \Delta - 1]$ étant l'intervalle de temps maximal (ou latence relative maximale) observé entre les temps d'impulsions des l neurones. Nous obtenons $(i + 1)^l$ codes au total, mais nous voulons seulement les codes qui ont au moins un neurone émettant à t, et au moins un autre émettant à $t+i$ (sinon, la latence relative maximale n'est pas de i : elle est strictement inférieure à i). Donc nous retranchons tous les codes n'utilisant pas l'instant t (borne inférieure) : il y en a $(i + 1 - 1)^l = i^l$, et nous retranchons aussi les codes n'utilisant pas $t + i$ (borne supérieure) : il y en a également $(i + 1 - 1)^l = i^l$. Cependant nous avons retranché deux fois tous les codes n'utilisant ni l'instant t (borne inférieure), ni l'instant $t + i$ (borne supérieure). Il faut donc rajouter $(i + 1 - 1 - 1)^l = (i - 1)^l$ codes au dénombrement.

En dénombrant et additionnant ces différents cas, nous obtenons le nombre d'états Z offert par un codage en latences relatives :

$$Z = 1 + \sum_{i=1}^{\Delta-1} \left((i+1)^l - 2 \cdot i^l + (i-1)^l \right)$$

Nous pouvons simplifier l'équation en séparant et changeant les indices des sommes avec $j = i+1$ et $k = i-1$:

$$\begin{aligned} Z &= 1 + \sum_{j=2}^{\Delta} j^l - 2 \cdot \sum_{i=1}^{\Delta-1} i^l + \sum_{k=0}^{\Delta-2} k^l \\ &= 1 + \Delta^l + (\Delta-1)^l - 2 \cdot (\Delta-1)^l - 2 \cdot 1^l + 1^l + 0^l \\ &= \Delta^l - (\Delta-1)^l \end{aligned}$$

Le nombre de bits b transmis par un codage en latences relatives est donc de :

$$\begin{aligned} b &= \log_2 (Z) \\ &= \log_2 \left(\Delta^l - (\Delta-1)^l \right) \end{aligned}$$

Nous pouvons réécrire cette équation dans le but d'analyser son comportement asymptotique :

$$\begin{aligned} b &= \log_2 \left(\Delta^l - (\Delta-1)^l \right) \\ &= \log_2 \left(\Delta^l \cdot \left(1 - \frac{(\Delta-1)^l}{\Delta^l} \right) \right) \\ &= \log_2(\Delta^l) + \log_2 \left(1 - \left(\frac{\Delta-1}{\Delta} \right)^l \right) \end{aligned}$$

Notons que $1 > \left(\frac{(\Delta-1)}{\Delta}\right)^l > 0$, et donc $0 < 1 - \left(\frac{(\Delta-1)}{\Delta}\right)^l < 1$, ce qui assure que le second terme $\log_2 \left(1 - \left(\frac{\Delta-1}{\Delta}\right)^l\right)$ est bien défini. Ce second terme retranche une certaine quantité de bits - car $\log_2(x) < 0$ avec $x \in]0,1[$ - au premier terme $\log_2(\Delta^l)$. Plus Δ est grand, plus le nombre de bits retranchés sera grand. En effet si $\Delta \to +\infty$, alors $\left(1 - \left(\frac{\Delta-1}{\Delta}\right)^l\right) \to 0$. Néanmoins, le premier terme va rester prépondérant, ce qui montre que $b \to +\infty$ quand $\Delta \to +\infty$, à l constant. Cela peut être vérifié avec un développement limité du premier ordre. Dans un premier temps, on factorise l'expression :

$$\begin{aligned} \frac{Z}{\Delta^l} &= \frac{\Delta^l - (\Delta-1)^l}{\Delta^l} \\ &= 1 - \frac{(\Delta-1)^l}{\Delta^l} \\ &= 1 - \left(\frac{\Delta-1}{\Delta} \right)^l \\ &= 1 - \left(1 - \frac{1}{\Delta} \right)^l \end{aligned}$$

On réalise maintenant un développement limité du premier ordre en posant $f(x) = (1-x)^l$ avec $x = \frac{1}{\Delta}$, $\Delta \to +\infty$, et donc $x_0 = 0$:

$$\begin{aligned} f(x) &\approx f(x_0) + f'(x_0) \cdot (x - x_0) \\ &\approx (1-0)^l + f'(x_0) \cdot (x - 0) \\ &\approx 1 + f'(x_0) \cdot x \end{aligned}$$

La dérivée de $f(x)$ est donnée par :

$$\begin{aligned} f'(x) &= \Big((1-x)^l\Big)' \\ &= l \cdot (1-x)^{l-1} \end{aligned}$$

On calcule $f'(x_0)$ et on remplace le résultat dans le développement limité du premier ordre de $f(x)$:

$$\begin{aligned} f(x) &\approx 1 + lx \\ &\approx 1 + \frac{l}{\Delta} \end{aligned}$$

En remplaçant ce résultat dans notre expression factorisée on obtient :

$$\begin{aligned} \frac{Z}{\Delta^l} &\approx 1 - \left(1 + \frac{l}{\Delta}\right) \\ &\approx \frac{l}{\Delta} \end{aligned}$$

Le nombre d'états Z donne ainsi :

$$\begin{aligned} Z &\approx \frac{l \cdot \Delta^l}{\Delta} \\ &\approx l \cdot \Delta^{l-1} \end{aligned}$$

Nous pouvons alors calculer le nombre de bits b :

$$\begin{aligned} b &\approx \log_2(l \cdot \Delta^{l-1}) \\ &\approx \log_2(l) + (l-1) \cdot \log_2(\Delta) \end{aligned}$$

Ce qui vérifie que b se comporte bien asymptotiquement comme $\log_2(\Delta^l) = l \log_2(\Delta)$ quand $\Delta \to +\infty$, pour un nombre de neurones l constant.

Nous avons réalisé une analyse asymptotique pour $\Delta \to +\infty$ à l constant. A présent, l'étude est réalisée en posant un nombre de neurone $l \to +\infty$ à Δ constant. Rappelons que le nombre de bits b est donné par :

$$b = \log_2(\Delta^l) + \log_2\left(1 - \left(\frac{\Delta - 1}{\Delta}\right)^l\right)$$

Analysons l'impact asymptotique du second terme en termes de perte d'informations. Cela donne $\left(\frac{\Delta-1}{\Delta}\right)^l \to 0$ et donc $\log_2\left(1-\left(\frac{\Delta-1}{\Delta}\right)^l\right) \to \log_2(1) \to 0$ bits. Asymptotiquement, nous pouvons donc conclure que le second terme ne diminue pas la quantité d'information transmise. Le premier terme $\log_2(\Delta^l)$ devient de plus en plus prédominant avec un nombre de neurones l croissant. Cela assure une croissance exponentielle du nombre d'états disponibles pour l'encodage de l'entrée.

En résumé, on obtient asymptotiquement $b = \log_2(\Delta^l)$ que ce soit pour le nombre de neurones $l \to +\infty$ ou pour le nombre de pas de temps $\Delta \to +\infty$. Cela nous permet de réaliser une comparaison asymptotique de fonctions en utilisant les notations de Landau pour le nombre de pas de temps $\Delta(b,l)$:

$$\Delta(b,l) = \Theta\left(2^{b/l}\right)$$

Ou de façon équivalente pour la fenêtre temporelle $T(b,l)$:

$$T(b,l) = \Theta\left(2^{b/l} \cdot dt\right)$$

En considérant toujours les mêmes paramètres, $l = 10$ neurones et $\Delta = 10$, on obtient ainsi $b = \log_2\left(\Delta^l - (\Delta-1)^l\right) \approx 33$ bits. Remarquons que contrairement au codage par rang, augmenter le nombre de pas de temps Δ augmente le nombre de bits b. En outre, comparativement à un codage par taux de décharge réalisé avec une population, nous avons ici réduit bien plus significativement la largeur de la fenêtre temporelle T, le nombre de neurones l se trouvant dans l'exposant $T(b,l) = \Theta\left(2^{b/l} \cdot dt\right)$, plutôt que dans le dénominateur $T = \left(\left(2^b/l\right) - 1\right) dt$.

Le codage par latences relatives est donc très intéressant pour rapidement transférer une grande quantité d'information, tout en maintenant un faible coût énergétique.

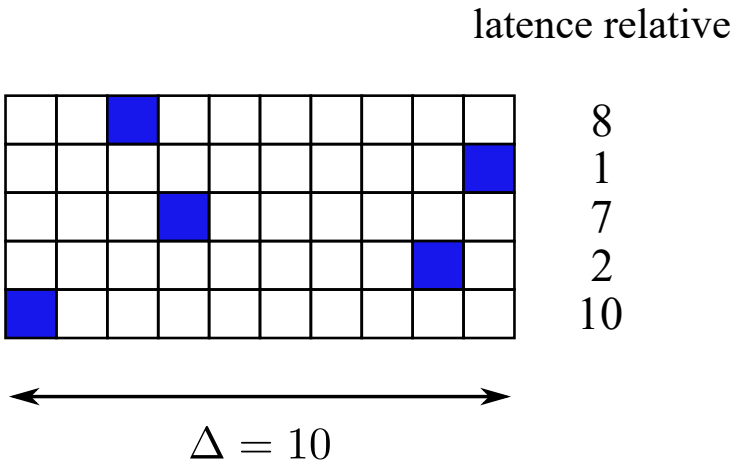

FIGURE 2.5 – Codage de l'entrée par les latences relatives des impulsions. Les latences relatives et donc les temps précis des impulsions sont utilisés pour encoder l'entrée.

2.1.3 Synthèse

Nous avons commencé par considérer un codage par taux de décharge avec un unique neurone. Augmenter la résolution en bits b de l'entrée avec un tel code nécessite une croissance

exponentielle de la largeur de la fenêtre temporelle d'encodage T, ce qui implique une réduction drastique de la vitesse de transfert de l'information.

Le codage par population du taux de décharge permet d'atténuer le problème de la chute de la vitesse de transfert de l'information. Cependant le codage par taux de décharge, qu'il soit réalisé avec un neurone ou une population de l neurones induit un nombre d'impulsions n dont la croissance est exponentielle relativement à la résolution en bits b de l'entrée désirée. Cela induit de sérieux problèmes en termes de consommation énergétique et de besoins de communication.

Ainsi, nous nous sommes tournés vers les codes exploitant les temps d'impulsions, i.e. les codes temporels, en posant la contrainte que chaque neurone ne puisse émettre qu'une unique impulsion. De façon analogue au codage par taux de décharge, un code temporel peut être réalisé avec un unique neurone. Il s'agit de coder l'entrée dans la latence entre le moment où le stimulus est présenté en entrée du neurone et le moment où le neurone décharge. Cependant, bien que le nombre de bits b transmis soit équivalent à un codage par taux de décharge, tout en émettant une unique impulsion, le problème de la croissance exponentielle de la fenêtre d'encodage T relativement à la résolution en bits b de l'entrée se repose.

Pour répondre à ce problème de croissance exponentielle de T relativement à b, nous nous sommes tournés vers les codes temporels instanciés non pas avec un unique neurone, mais avec une population de l neurones. Notre attention s'est concentrée sur le codage par rang où seul l'ordre des impulsions encode l'entrée, et le codage par latences relatives où l'entrée est encodée dans les temps précis des impulsions. Contrairement au codage par rang, le codage par latences relatives permet d'augmenter le nombre de bits b transmis en élargissant la fenêtre temporelle d'encodage T. En outre, remarquons qu'un codage en latences relatives ne requiert pas de connaître explicitement le temps initial de présentation du stimulus, contrairement à un codage *first spike latency*. L'information est ainsi entièrement contenue dans le motif impulsionnel, sans avoir explicitement besoin d'un référentiel temporel extérieur.

Notre étude comparative (voir Tab. 2.1) nous permet de conclure que le codage par latences relatives est celui qui donne les meilleures performances en termes 1) de nombre de bits transmis, 2) de nombre d'impulsions induit et 3) en termes de vitesse de transfert de l'information.

Ainsi pour une même configuration, avec $l = 10$ neurones et $\Delta = 10$ pas de temps, alors qu'un codage par population du taux de décharge transmet environ 7 bits avec jusqu'à 100 impulsions, un codage par rang transmet 21 bits avec 10 impulsions (quantité constante), et un codage par latences relatives transmet 33 bits avec 10 impulsions (quantité constante).

2.2 Apprentissage non-supervisé bio-inspiré et auto-organisé

Nous nous sommes penchés jusqu'à présent sur les différents types d'encodages neuronaux. Quid de l'apprentissage non-supervisé bio-inspiré et auto-organisé ? Partons d'un exemple concret.

Supposons que vous deviez prochainement partir en Finlande pour une collaboration avec des chercheurs finlandais. Ces derniers communiquent essentiellement en finlandais. Vous souhaitez donc apprendre les rudiments de la langue. Cependant votre temps est limité. Vous n'en avez pas suffisamment pour étudier le finlandais avec un manuel. Aussi, vous vous mettez à écouter des podcasts en finlandais en conduisant en direction de votre lieu de travail. Les jours passent et votre capacité à discriminer des phonèmes, puis des mots composés de plusieurs phonèmes, ne cesse de s'améliorer. Vous saisissez de plus en plus finement la prosodie du finlandais, i.e.

	taux décharge un neurone	taux décharge pop. neurones	latence un neurone	rang pop. neurones	latences relatives pop. neurones
$n_{\max}$	$2^b - 1$	$2^b - 1$	1	l	l
b	$\log_2\left(\frac{T}{dt} + 1\right)$	$\log_2\left(l\frac{T}{dt} + 1\right)$	$\log_2\left(\frac{T}{dt}\right)$	$\log_2(l!)$	$\log_2\left(\left(\frac{T}{dt}\right)^l - \left(\frac{T}{dt} - 1\right)^l\right)$
T	$\left(2^b - 1\right) dt$	$\left(\frac{2^b}{l} - 1\right) dt$	$2^b dt$	$\frac{T}{dt} \geq l$	$T(b, l) = \Theta\left(2^{b/l} \cdot dt\right)$

TABLE 2.1 – Synthèse des propriétés des différents codes neuronaux suivant 1) le nombre d'impulsions maximal $n_{\max}$ émis, 2) le nombre de bits b transmis, 3) la largeur de la fenêtre temporelle d'encodage T. Notons que la fenêtre d'encodage T est implicite pour un codage par rang : instancier $l!$ combinaisons de rangs nécessite que $T/dt \geq l$. La fenêtre temporelle T pour le codage en latences relatives est donnée au sens des notations de Landau de comparaison asymptotique de fonctions.

sa structure temporelle et rythmique. Votre cerveau a progressivement construit un modèle de la langue et a capturé les rudiments des règles syntaxiques la régissant. Il vous suffit alors de parfaire ce modèle et d'y inclure des éléments sémantiques de façon supervisée, e.g. à l'aide d'un manuel. Vous avez ainsi grandement optimisé le temps passé à apprendre cette langue.

Cet exemple illustre ce que permet l'apprentissage non-supervisé. Cet apprentissage non-supervisé est très schématiquement réalisé dans le cortex, tandis que l'apprentissage par renforcement est réalisé dans les ganglions de la base et le thalamus, et l'apprentissage supervisé dans le cervelet (DOYA, 1999, 2000). L'objectif de l'apprentissage non-supervisé est d'apprendre des représentations utiles, i.e. des régularités statistiques telles que des corrélations, à partir des données entrantes.

2.2.1 Carte auto-organisatrice

L'apprentissage non-supervisé permet d'extraire des caractéristiques des données via le développement de champs récepteurs. Le champ récepteur d'un neurone est défini comme la région dans l'espace d'entrée élicitant une réponse neuronale (WICKENS, 2009). L'apprentissage de champs récepteurs peut être reformulé en des termes d'apprentissage automatique comme l'apprentissage de filtres ou de vecteurs codes, se faisant généralement dans les poids synaptiques.

Un algorithme très influent d'apprentissage non-supervisé, bio-inspiré et auto-organisé de champs récepteurs est la carte auto-organisatrice (SOM), développée originellement par KOHONEN, 1990, 2013. Outre sa capacité à apprendre des représentations, cet algorithme apprend à construire des cartes topographiquement ordonnées, i.e. des points de données proches dans l'espace d'entrée sont représentés par des neurones spatialement proches dans la carte (voir Fig 2.8). Notons toutefois qu'une projection de la variété des données (*data manifold*) sur une topologie de carte est parfaite si et seulement si deux neurones adjacents dans la carte ont des vecteurs codes adjacents dans l'espace d'entrée (MARTINETZ, 1993). Cela implique généralement que la projection est parfaite si la dimension intrinsèque des données d'entrée est égale à celle de la carte.

Les cartes neuronales topographiquement ordonnées sont très communes dans le cortex sensoriel. Le cortex visuel projette l'information visuelle de la rétine dans une carte topographiquement ordonnée, où notamment des neurones spatialement proches représentent des orientations

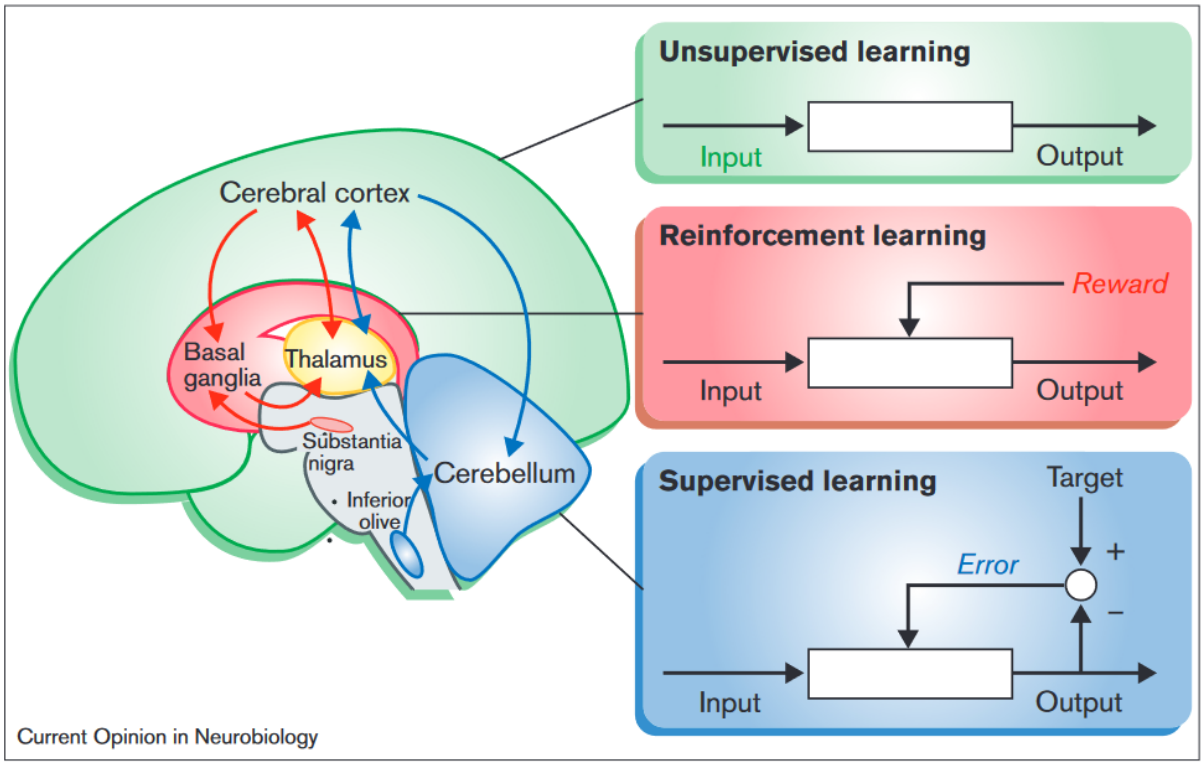

FIGURE 2.6 – Les différents types d'apprentissage sont réalisés dans différentes régions du cerveau. Figure tirée de DOYA, 2000.

du stimulus proches (MOSER et al., 2014 ; OHKI et al., 2006). Le cortex somatosensoriel projette l'information tactile de la peau dans une carte formant une représentation topographique du corps (KANDEL et al., 2000). Le cortex auditif des chats est un autre exemple de représentation topographiquement ordonnée des stimulus d'entrées (MERZENICH et al., 1975), en l'occurrence des fréquences des sons.

Il est remarquable que des cartes topographiquement ordonnées soient trouvées en dehors du cortex sensoriel, e.g. dans le cortex moteur des singes (GRAZIANO et AFLALO, 2007), responsable de la planification et du contrôle des mouvements. Le cortex moteur envoie ensuite les commandes motrices aux ganglions de la base, où est réalisé l'apprentissage par renforcement. Cette régularité suggère un rôle important des cartes topographiquement ordonnées dans le cortex (en charge de l'apprentissage non-supervisé), les élevant potentiellement au rang de principe organisateur et computationnel (GRAZIANO et AFLALO, 2007). Par exemple il a été suggéré que la projection d'une variété de données de haute dimension sur une surface de basse dimension - la surface du cortex - permet de minimiser le coût en termes de connexions synaptiques entre les neurones, de telle sorte que des calculs opérés sur des points de données proches dans l'espace d'entrée puissent être réalisés localement dans le cortex (DURBIN et MITCHISON, 1990).

Du point de vue de l'apprentissage automatique, la capacité de la SOM à générer des cartes topographiquement ordonnée dote l'algorithme de capacité de *clustering*[1] et de visualisation de relations entre des données de haute dimension sur une carte le plus souvent en 2D (voir Fig

1. Nous entendons par *clustering* le fait de regrouper des données non-étiquetées similaires.

2.7). En ce sens, on peut dire que la SOM réduit la dimension des données à la dimension de la carte. La SOM est un algorithme de réduction de dimensionnalité non-linéaire, permettant ainsi de traiter des variétés de données courbées (transformations non-linéaires), contrairement à des algorithmes tels que l'Analyse en Composantes Principales (PCA) limités à opérer sur la base de transformations linéaires.

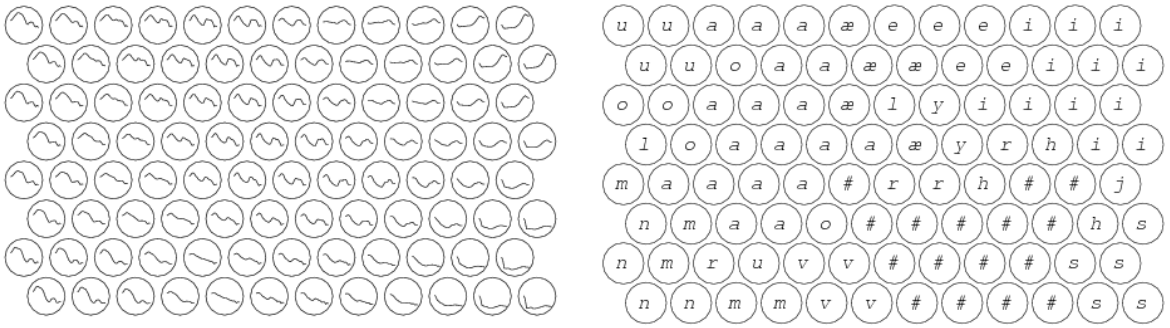

FIGURE 2.7 – Exemple d'application de *clustering* réalisé par une SOM. Une SOM avec une topologie hexagonale (chaque neurone a 6 voisins) est entraînée avec des phonèmes finnois sous la forme de signaux sonores. Les points de données sont à haute dimension. On observe à gauche une représentation des champs récepteurs (ou filtres) appris par les neurones de la SOM. La SOM a généré une carte topographiquement ordonnée : des neurones spatialement proches partagent des représentations proches. A droite sont affichés les étiquettes des neurones (e.g. a, u, o) déterminés à partir de l'activité préférentielle des neurones relativement à un ensemble de phonèmes. En ayant utilisé un apprentissage purement non-supervisé, i.e. sans utiliser d'étiquettes durant la phase d'apprentissage, on observe que les phonèmes ont globalement été correctement *clusterisés* par la SOM. Figure tirée de `http://www.scholarpedia.org/article/Kohonen_network`.

Le modèle de SOM

La carte est composée de n neurones. Ces neurones sont arrangés selon une topologie définissant les relations de voisinage entre les neurones. Il s'agit généralement d'une grille 2D de neurones, où chaque neurone a 4 voisins. D'autres topologies sont possibles comme une carte hexagonale (6 voisins) ou torique (supprimant les effets de bords).

Chaque neurone $i \in \{1, \dots n\}$ de la carte a un vecteur code de poids synaptiques $\mathbf{w}_i \in \mathbb{R}^p$ associé, où p est la dimension du vecteur code, égale à la dimension des données d'entrées. Un neurone de SOM est un modèle phénoménologique [2] de colonne corticale, brique fonctionnelle du néo-cortex, correspondant à un ensemble de neurones arrangés verticalement (en colonne) et réagissant préférentiellement à un même stimulus d'entrée.

L'algorithme

Les vecteurs codes des neurones sont généralement initialisés aléatoirement. Ensuite le processus d'auto-organisation de la carte commence. L'auto-organisation est régie par trois principes :

2. Un modèle phénoménologique fait abstraction des variables bio-physiques. Il se place à un niveau fonctionnel en cherchant à reproduire certains comportements sans considérer de mécanismes précis.

1. Compétition
2. Coopération
3. Adaptation

1) Compétition

Un point de donnée d'entrée $\mathbf{x}$ est choisi aléatoirement à un temps discret t. La distance entre l'entrée courante $\mathbf{x}$ et chaque vecteur code $\mathbf{w}_i(t)$ est calculée. La distance euclidienne, ou norme L2 est communément utilisée. Le neurone minimisant cette distance, qu'on peut interpréter comme une erreur de reconstruction, est le gagnant de la compétition. Ainsi l'espace d'entrée continu peut être projeté sur l'espace de sortie discret des neurones (ensemble fini de vecteurs codes) via une compétition entre les neurones. L'indice du neurone gagnant est nommé *Best Matching Unit*.

$$\text{bmu} = \underset{i \in \{1 \dots n\}}{\arg\min}\, d\big(\mathbf{x}, \mathbf{w}_i(t)\big) \tag{2.1}$$

2) Coopération

La quantification vectorielle vise à minimiser l'erreur de quantification moyenne (ou reconstruction) avec un nombre fini de vecteurs codes, de sorte à obtenir une compression (avec perte) des données efficiente. Cela correspond à l'apprentissage de centroïdes.

Ainsi en utilisant un critère de quantification vectorielle, le vecteur code $\mathbf{w}_{\text{bmu}}$ se rapproche de l'entrée courante $\mathbf{x}$ avec un taux d'apprentissage α :

$$\mathbf{w}_{\text{bmu}}(t+1) \leftarrow \mathbf{w}_{\text{bmu}}(t) + \alpha \cdot \big(\mathbf{x} - \mathbf{w}_{\text{bmu}}(t)\big)$$

Nous pourrions ainsi actualiser uniquement le vecteur code de la BMU, en utilisant un critère de quantification vectorielle. Cela revient à la règle d'adaptation en ligne (où les points de données suivent une séquence aléatoire) de *K-means* et à un circuit *Winner-Take-All* (WTA), où une seule unité est actualisée.

Cependant cet apprentissage purement compétitif ne permet pas de générer de carte topographiquement ordonnée, où des neurones spatialement proches partagent des représentations proches.

Pour obtenir une carte topographiquement ordonnée, une fonction de voisinage est appliquée sur le voisinage de la BMU. Cette fonction de voisinage introduit une coopération entre neurones voisins. Plus les neurones sont spatialement proches de la BMU, plus leurs vecteurs codes sont attirés par l'entrée courante. Par contraste, plus ils sont éloignés, moins leurs vecteurs codes sont attirés par l'entrée courante. Outre ce comportement, la zone d'influence de la fonction de voisinage relativement à la BMU décroît dans le temps : le rayon de voisinage décroît durant la phase d'apprentissage. Ce rayon de voisinage décroissant permet à la carte de s'organiser dans un premier temps, puis de spécialiser de plus en plus les vecteurs codes des neurones, en décorrélant

leur apprentissage. Dans le cas particulier où le rayon atteint une valeur proche ou égale à 0, on retrouve un comportement WTA et la règle d'adaptation en ligne de *K-means*.

En outre, cet apprentissage coopératif permet d'accélérer la convergence de l'algorithme et d'éviter les minima locaux (Martinetz et al., 1993). Les performances empiriques obtenues avec la SOM en termes d'erreur de reconstruction sont ainsi meilleures que les algorithmes communs de QV. L'apprentissage coopératif de la SOM lui permet également d'être tolérante aux erreurs et d'opérer comme un algorithme de QV bayésien (Luttrell, 1994 ; Yin, 2008).

3) Adaptation

Après avoir déterminé la BMU, l'actualisation des vecteurs codes se fait en tenant compte d'une fonction de voisinage Θ :

$$\mathbf{w}_i(t+1) \leftarrow \mathbf{w}_i(t) + \alpha(t) \cdot \underbrace{\Theta\big(\sigma(t), d(i, \mathrm{bmu})\big)}_{\text{fonction de voisinage}} \cdot \underbrace{\big(\mathbf{x} - \mathbf{w}_i(t)\big)}_{\text{quantif. vectorielle}} \tag{2.2}$$

Notons que le taux d'apprentissage n'est pas une constante α, mais une variable $\alpha(t) \in [0, 1]$ évoluant dans le temps t. Une fonction populaire pour la réduction du taux d'apprentissage $\alpha(t)$ est la décroissance exponentielle, dont la vitesse de décroissance est réglée par la constante de temps τ_0. La valeur initiale du taux d'apprentissage est réglée par α_0. Notons qu'à $t = \tau_0$, $\alpha(t)$ est égal à environ 37 % de sa valeur initiale α_0 : $\alpha(t = \tau_0) = \exp(-1)\alpha_0 \approx 0.37\alpha_0$.

$$\alpha(t) = \alpha_0 \exp\left(-\frac{t}{\tau_0}\right) \tag{2.3}$$

La fonction de voisinage Θ prend communément la forme d'un noyau gaussien, modèle complètement isotropique, $d(i, \mathrm{bmu})$ étant la distance euclidienne basée sur des coordonnées spatiales normalisées. Le noyau gaussien est centré sur la BMU : si $i = \mathrm{bmu}$, le noyau gaussien produit 1.0 en sortie.

$$\Theta\big(\sigma(t), d(i, \mathrm{bmu})\big) = \exp\left(-\frac{d(i, \mathrm{bmu})^2}{2\sigma^2(t)}\right) \tag{2.4}$$

Comme le taux d'apprentissage $\alpha(t)$, le rayon de voisinage $\sigma(t)$ n'est pas une constante, mais une variable dont la valeur décroît dans le temps. Une fonction populaire pour la réduction du rayon de voisinage $\sigma(t)$ est la décroissance exponentielle, dont la vitesse de décroissance est réglée par τ_1. La valeur initiale du rayon de voisinage est réglée par σ_0.

$$\sigma(t) = \sigma_0 \exp\left(-\frac{t}{\tau_1}\right) \tag{2.5}$$

Notons qu'il existe également une version *batch* de la SOM (Kohonen, 2013), où toutes les données sont présentées à l'algorithme à chaque mise à jour des vecteurs codes. L'algorithme itère alors les modifications induites par ce *batch* de données jusqu'à stabiliser les solutions obtenues.

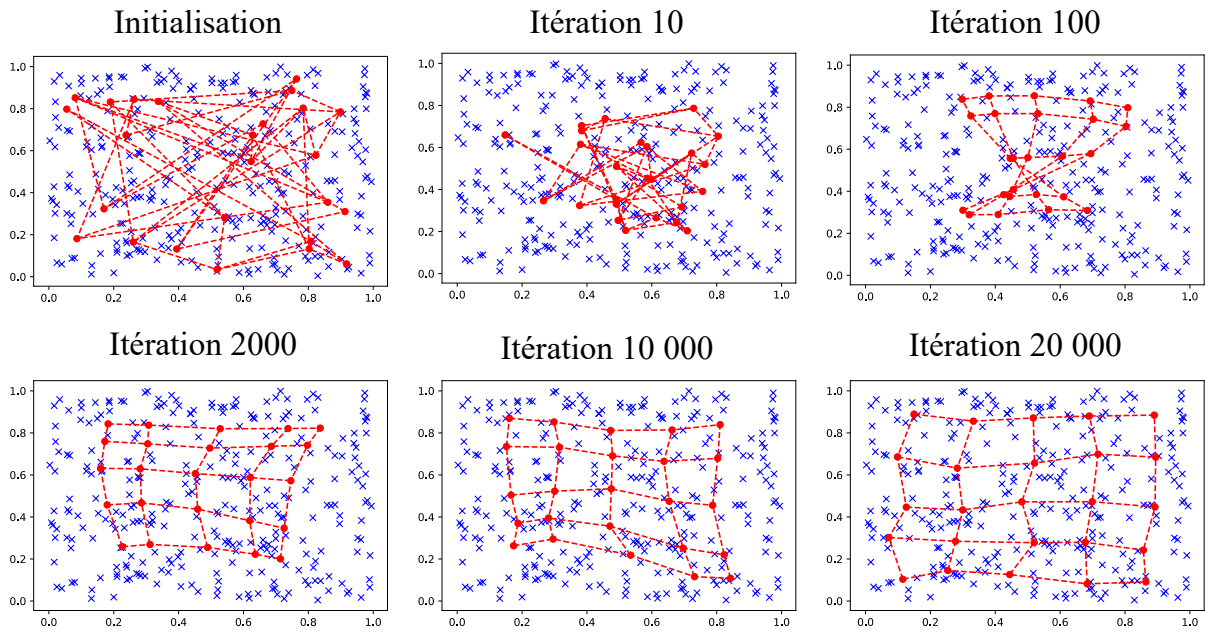

FIGURE 2.8 – Exemple de génération de carte topographiquement ordonnée par une SOM. La SOM est constituée de 5*5 neurones disposés sur une topologie en grille, i.e. chaque neurone est connecté à 4 voisins. Les vecteurs d'entrées sont en 2D et représentés par les croix bleues. Les vecteurs codes appris par les neurones sont de même dimensionnalité que les entrées (2D) et sont représentés par les points rouges. Les lignes rouges représentent les connexions entre les neurones.

2.2.2 Apprentissage avec des réseaux de neurones impulsionnels

Bien que la SOM soit d'inspiration biologique, cet algorithme ne permet pas de traiter à la volée les données asynchrones, binaires et éparses que sont les impulsions neuronales. A chaque itération de l'algorithme, les vecteurs d'entrées doivent arriver de façon synchrone et prendre des valeurs continues.

Il est donc nécessaire de s'orienter vers d'autres méthodes d'apprentissage plus adaptées au traitement de données événementielles.

Principes de l'apprentissage hebbien

En 1949, un neuropsychologue du nom de Donald Hebb (HEBB, 1949) proposait un postulat fondateur d'apprentissage biologique non-supervisé et auto-organisé, visant à expliquer la formation des champs récepteurs : une synapse connectant deux neurones est renforcée si le neurone présynaptique participe (en déchargeant) de manière fréquente à la décharge du neurone postsynaptique.

Ce principe d'apprentissage hebbien se caractérise (HAYKIN et HAYKIN, 2009) ainsi par :

1. une localité spatiale, i.e. la variation du poids synaptique dépend d'informations ou de variables accessibles au niveau de la paire de neurones connectés,

2. une dépendance à la dimension temporelle, i.e. la variation du poids synaptique dépend des dates auxquelles les neurones pré et postsynaptique sont actifs,
3. une nature interactionnelle, i.e. la variation du poids synaptique est causée par l'interdépendance entre les activités des neurones pré et postsynaptique : une prédiction ne peut pas être réalisée avec l'activité d'un seul de ces deux neurones,
4. une nature corrélationnelle, i.e. la variation du poids synaptique dépend de corrélations temporelles entre l'activité pré et postsynaptique, impliquant l'apprentissage de caractéristiques statistiquement fréquentes.

Une première règle hebbienne simple

Il n'existe pas une unique règle d'apprentissage hebbien, mais une pluralité de règles. La plasticité hebbienne est avant tout un principe guidant la formulation de règles d'apprentissage. L'apprentissage hebbien peut être formulé comme étant basé soit sur le taux de décharge soit sur les temps d'impulsions.

A partir de ce principe d'apprentissage hebbien précédemment caractérisé, peut être formulée une règle hebbienne simple : la règle du produit de l'activité. La variation du poids synaptique Δw_{ji} entre un neurone présynaptique i et postsynaptique j est fonction de leurs activités $x_i(t)$ et $y_j(t)$, modulée par un taux d'apprentissage α :

$$\Delta w_{ji} = \alpha\, x_i(t)\, y_j(t) \tag{2.6}$$

Cette règle hebbienne du produit de l'activité augmente le poids synaptique w_{ji} d'une magnitude de Δw_{ji} lorsque les deux neurones sont simultanément actifs.

Cette règle nécessite donc que les neurones soient simultanément actifs pour induire un renforcement synaptique. Or dans sa formulation, Hebb soutenait déjà que le poids synaptique est renforcé si le neurone présynaptique participe à la décharge du neurone postsynaptique. Cela introduit un caractère causal à l'interaction entre les deux neurones, qui n'est pas pris en compte par la règle du produit de l'activité : le neurone présynaptique doit décharger avant le neurone postsynaptique pour que le poids synaptique soit renforcé.

Un autre problème est l'absence de règle additionnelle pour le cas d'interactions anti-causales : le cas où le neurone postsynaptique décharge avant le neurone présynaptique.

Plasticité fonction du temps d'occurrence des impulsions (STDP)

La plasticité fonction du temps d'occurrence des impulsions ou *Spike-Timing Dependent Plasticity* (STDP) répond aux problèmes susmentionnés. La STDP est une règle hebbienne basée explicitement sur le temps d'impulsion du neurone présynaptique t_{pre} et postsynaptique t_{post}. La STDP est donc adaptée au traitement de données événementielles.

La STDP a par exemple été utilisée dans un contexte d'apprentissage non-supervisé pour une tâche de classification sur la base de données MNIST, atteignant 95 % de précision (Diehl et Cook, 2015). Tavanaei et Maida, 2019 réalisent une approximation de la rétropropagation du gradient en préservant une localité temporelle du calcul à l'aide d'une nouvelle règle STDP, ainsi dotée d'une nouvelle capacité d'apprentissage supervisé. Les performances ont notamment été testées pour des tâches de classification sur les bases de données MNIST et IRIS, atteignant des performances comparables à l'état de l'art pour des SNN.

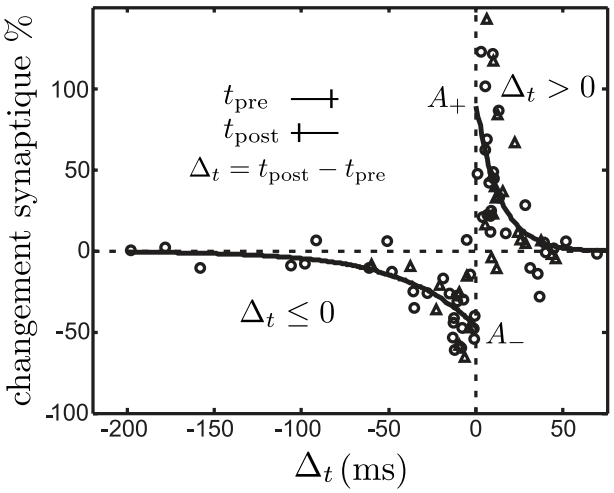

FIGURE 2.9 – Illustration du comportement de la STDP. La variation du poids synaptique est fonction de $\Delta_t = t_{\text{post}} - t_{\text{pre}}$. Le poids synaptique est renforcé (première règle) si $\Delta_t > 0$, sinon le poids est déprimé (deuxième règle) si $\Delta_t \leq 0$. Figure adaptée de FROEMKE et al., 2006.

Il existe de nombreuses règles de type STDP. Nous introduisons ici la STDP basique (voir Fig. 2.9) issue des neurosciences (SONG et al., 2000). Il s'agit d'une règle STDP additive et non-dépendante des poids synaptiques ou *Nonweight-dependent Spike-Timing Dependent Plasticity* (NSTDP). La STDP est composée de deux règles déterminant la variation du poids synaptique Δw_{ji} :

$$\Delta w_{ji} = \begin{cases} +A_+ \exp\left(-\frac{\Delta t}{\tau_{\text{pre}}}\right) & \text{si } \Delta t > 0 \\ -A_- \exp\left(+\frac{\Delta t}{\tau_{\text{post}}}\right) & \text{si } \Delta t \leq 0 \end{cases} \tag{2.7}$$
$$\text{avec } \Delta t = t_{\text{post}} - t_{\text{pre}}$$

La première règle gère le cas où l'interaction est causale : l'impulsion présynaptique intervient avant l'impulsion postsynaptique, reflétée par $\Delta t = t_{\text{post}} - t_{\text{pre}} > 0$. Le poids synaptique augmente de $\Delta w_{ji} > 0$ avec un taux d'apprentissage $A_+ \in [0, 1]$. La constante de temps τ_{pre} règle la largeur de la fenêtre temporelle pour les interactions causales, i.e. l'intervalle de temps maximal pris en compte de t_{pre} à t_{post}. Cette fenêtre temporelle introduit une localité temporelle à l'application de cette règle, appliquée pour le cas causal $\Delta t > 0$.

La seconde règle gère le cas où l'interaction est anti-causale : l'impulsion postsynaptique intervient avant l'impulsion présynaptique, reflétée par $\Delta t = t_{\text{post}} - t_{\text{pre}} \leq 0$. Le poids synaptique diminue de $\Delta w_{ji} < 0$ avec un taux d'apprentissage $A_- \in [0, 1]$. La constante de temps τ_{post} règle la largeur de la fenêtre temporelle pour les interactions anti-causales, i.e. l'intervalle de temps maximal pris en compte de t_{post} à t_{pre}. Cette fenêtre temporelle introduit une localité temporelle à l'application de cette règle, appliquée pour le cas anti-causal $\Delta t \leq 0$.

Notons que ces deux règles induisent une magnitude maximale de $\Delta w_{ji} \to +A_+$ et $\Delta w_{ji} \to -A_-$ respectivement, lorsque $\Delta t \to 0$ car $e^{\pm \Delta t}_{\Delta t \to 0} = 1$. Autrement dit, plus les temps d'impulsions sont proches dans le temps (forte corrélation temporelle), plus la variation du poids synaptique

est grande. Par contraste la magnitude est minimale $\Delta w_{ji} \to 0$ lorsque $\Delta t \to \infty$ car $e^{\pm\Delta t}_{\Delta t \to \infty} = 0$. Plus les temps d'impulsions sont éloignés dans le temps (faible corrélation temporelle), plus la variation du poids synaptique est faible.

Cependant ces règles de renforcement et d'affaiblissement du poids synaptique peuvent induire à plus long termes une divergence du poids synaptique $w_{ji} \to \pm\infty$. On souhaite maintenir les poids synaptiques dans un intervalle $w^{\min} < w_{ji} < w^{\max}$. Par simplicité [3] on fixe $w^{\min} = 0.0$ et $w^{\max} = 1.0$. Pour réaliser cet objectif on peut utiliser des limites souples (*soft bounds*) injectant une dépendance multiplicative au poids synaptique w_{ji} dans la STDP. Il s'agit ainsi d'une règle STDP multiplicative et dépendante des poids synaptiques ou *Weight-dependent Spike-Timing Dependent Plasticity* (WSTDP).

$$\Delta w_{ji} = \begin{cases} +A_+(w^{\max} - w_{ji}) \exp\left(-\frac{\Delta t}{\tau_{\text{pre}}}\right) & \text{si } \Delta t > 0 \\ -A_- w_{ji} \exp\left(+\frac{\Delta t}{\tau_{\text{post}}}\right) & \text{si } \Delta t \leq 0 \end{cases} \tag{2.8}$$

Ainsi, pour la première règle induisant un renforcement du poids synaptique $\Delta w_{ji} > 0$, lorsque $w_{ji} \to w^{\max}$ alors $(w^{\max} - w_{ji}) \to 0$ et donc $\Delta w_{ji} \to 0$. La limite supérieure $w^{\max}$ est donc bien respectée, même si $w^{\max}$ peut être momentanément dépassée.

Pour la deuxième règle induisant un affaiblissement du poids synaptique $\Delta w_{ji} < 0$, lorsque $w_{ji} \to w^{\min}$ avec $w^{\min} = 0$ alors $\Delta w_{ji} \to 0$. La limite inférieure $w^{\min} = 0$ est donc bien respectée, même si w_{ji} peut être momentanément négatif.

Règles STDP pour l'apprentissage de représentations

Cependant les deux classes de règles NSTDP et WSTDP ne permettent pas d'obtenir à la fois des champs récepteurs stables et à la fois une distribution continue de valeurs de poids synaptiques.

Les règles STDP additives et non-dépendantes des poids synaptiques ou *Nonweight-dependent Spike-Timing Dependent Plasticity* (NSTDP), e.g. Song et al., 2000, capturent des corrélations dans les entrées en produisant une distribution bimodale des poids synaptiques, où leurs valeurs saturent aux extremums $w^{\min}$ et $w^{\max}$ de l'intervalle de variation des poids (Rossum et al., 2000). Les règles NSTDP génèrent une forte compétition entre les synapses, dû au fait que la dépression est généralement plus élevée que la potentiation. Le résultat final est un ensemble de synapses maximalement activées (valeur du poids à $w^{\max}$) et les autres synapses désactivées (valeur du poids à $w^{\min}$). Malgré la bistabilité des règles NSTDP, celles-ci permettent toutefois de stocker des mémoires (ou représentations) à long terme dans les poids synaptiques (Billings et van Rossum, 2009). Autrement dit, les champs récepteurs appris sont stables.

Par contraste les règles STDP dépendantes des poids synaptiques ou *Weight-dependent Spike-Timing Dependent Plasticity* (WSTDP) capturent les corrélations dans les entrées en produisant une distribution unimodale des poids synaptiques. La distribution des poids résultante est unimodale car chaque poids synaptique est attiré vers sa valeur moyenne. Comme montré par Billings et van Rossum, 2009, l'inconvénient des règles WSTDP est que leur faible compétition induit une

3. Ce choix de limites souples est motivé par le fait que les règles STDP s'appliquent généralement à des connexions synaptiques excitatrices (poids positifs).

faible capacité de rétention, i.e. une faible capacité de stocker des mémoires (ou représentation) à long terme. Autrement dit, les champs récepteurs appris sont peu stables.

En résumé les règles NSTDP permettent de générer des champs récepteurs stables, mais sont bistables : elles induisent une saturation des poids synaptiques aux valeurs minimales et maximales. L'intervalle de variation des paramètres soumis à apprentissage que sont les poids synaptiques n'est donc pas pleinement exploité avec les règles NSTDP. Cela limite fortement l'expressivité des poids synaptiques, i.e. leur capacité à représenter finement des états extérieurs comme les états de l'environnement. Les règles WSTDP corrigent ce problème en générant des distribution unimodales et continues de poids synaptiques, tirant ainsi profit de l'intervalle de variation des poids. Cependant elles ne permettent pas de générer des champs récepteurs stables.

Tavanaei, Masquelier et al., 2018 proposent une méthode d'apprentissage asymptotiquement stable capable d'utiliser pleinement l'intervalle de variation des poids synaptiques pour extraire des représentations à partir de données impulsionnelles encodant des imagettes (ou patchs d'images). L'intensité de chaque pixel d'imagette est transformée en un train d'impulsions par un encodage en taux de décharge.

Pour extraire des représentations de ces trains d'impulsions, ils ont développé une règle STDP intégrant deux modules :

1. un module de quantification vectorielle visant à minimiser l'erreur moyenne de reconstruction en adaptant les poids synaptiques,
2. un module de régularisation favorisant les faibles valeurs de poids synaptiques.

La règle STDP développée est une adaptation de la règle hebbienne de Földiák, 1990, adaptation permettant d'extraire des représentations à partir de données impulsionnelles et asynchrones. La règle hebbienne de Földiák intègre un module de quantification vectorielle minimisant l'erreur de reconstruction en rapprochant le poids synaptique w_{ji} de l'entrée courante x_i lorsque la sortie du neurone postsynaptique y_j est active : $\Delta w_{ji} \propto y_j(x_i - w_{ji})$. Ainsi l'intervalle de variation des poids synaptiques est pleinement utilisé pour représenter les entrées.

Grâce à une formulation probabiliste basée sur les temps d'impulsions plutôt que sur une estimation du taux de décharge, cette règle STDP décode le train d'impulsions encodant l'intensité des pixels, et minimise l'erreur de reconstruction en adaptant les poids synaptiques. Les poids synaptiques tombent dans l'intervalle $[0, 1]$. Pour augmenter l'efficience de la règle STDP dans le SNN, deux contraintes supplémentaires sont intégrées régularisant le processus d'apprentissage :

1. Sparsité[4] (i.e. faible nombre d'impulsions émises par la couche de sortie), avec une règle d'ajustement du seuil de décharge V_θ des neurones. La sparsité est considérée comme un critère permettant d'obtenir de bonnes représentations (Bengio et al., 2013 ; Falez et al., 2019), e.g. seul un petit nombre de caractéristiques peuvent être trouvées dans une image ou un patch d'image.
2. Indépendance (i.e. décorréler l'apprentissage des différents vecteurs codes), avec un circuit WTA implémenté phénoménologiquement via un softmax. Cela permet d'éviter l'apprentissage de vecteurs codes redondants.

Ainsi des représentations sont apprises dans les poids synaptiques grâce à une méthode d'apprentissage respectant les contraintes de 1) minimisation de l'erreur de reconstruction de l'entrée, 2) sparsité dans le nombre d'impulsions émis en sortie et 3) indépendance dans l'apprentissage des différents vecteurs codes.

4. Les termes sparsité et parcimonie sont utilisés de façon interchangeable dans ce manuscrit.

D'autres méthodes plus anciennes - avec de moins bonnes performances en termes d'erreur de reconstruction - se situent dans le même axe de travail. King et al., 2013 ont proposé une règle spatialement locale apprenant des champs récepteurs ressemblant à des filtres de Gabor. Cependant cette règle n'opère pas à partir des temps d'impulsions, mais à partir d'une estimation du taux de décharge des neurones afférents. Burbank, 2015 a développé une méthode reproduisant le comportement d'un auto-encodeur avec une règle STDP. Cette règle STDP apprend les poids d'encodage et de décodage avec une règle hebbienne et anti-hebbienne, respectivement. La conjonction de ces deux règles approxime la fonction de coût d'un auto-encodeur.

Notons toutefois que toutes ces règles d'apprentissage opèrent sur la base d'encodage d'entrées en taux de décharge. Il n'existe pas à notre connaissance de règle de type STDP, spatialement et temporellement locale, capable d'extraire des représentations à partir d'encodages temporels dans la perspective d'obtenir une faible erreur de reconstruction.

2.2.3 Mesures pour l'évaluation des performances

Nous présentons dans cette section deux mesures utilisées dans l'ensemble du manuscrit. D'autres mesures plus spécifiques seront introduites dans les chapitres 4, 5 et 6.

Erreur de reconstruction RMS

De bonnes représentations peuvent êtres conçues comme un ensemble fini de vecteurs codes compressant le plus possible une distribution de données d'entrées. Nous souhaitons quantifier l'erreur de reconstruction entre un vecteur d'entrée $\mathbf{a}_p$ et le vecteur code $\widehat{\mathbf{a}}_p$ qui le représente. Cela peut être réalisé avec la racine de l'erreur quadratique moyenne (RMS). Il s'agit d'une mesure de similarité basée sur la distance euclidienne entre $\mathbf{a}_p$ et $\widehat{\mathbf{a}}$. Chaque vecteur d'entrée $\mathbf{a}_p$ se voit assigné un vecteur code $\widehat{\mathbf{a}}_p$ qui le représente. La racine de l'erreur quadratique moyenne agrège les erreurs de reconstruction en une seule valeur. Plus sa valeur tend vers 0, plus la reconstruction est fidèle. Elle est calculée comme suit :

$$RMS = \frac{1}{P}\sum_{p=1}^{P}\sqrt{\frac{1}{k}\sum_{i=1}^{k}(a_{i,p} - \widehat{a}_{i,p})^2} \tag{2.9}$$

où P est le nombre de vecteurs d'entrées et k est la dimension du vecteur d'entrée.

Sparsité

Nous souhaitons également évaluer la sparsité des SNN, i.e. le pourcentage de neurones actifs. Nous introduisons une mesure simple de sparsité pour un vecteur d'entrée donné :

$$\text{Sparsité} = \frac{1}{m}N_{\text{imp}} \tag{2.10}$$

où m est le nombre de neurones, et N_{imp} est le nombre d'impulsions émises par les m neurones en réponse à un vecteur d'entrée. Une faible valeur indique un grand nombre d'unités inactives et

donc une grande sparsité. La sparsité moyenne peut être calculée pour un ensemble de vecteurs d'entrées. Dans la suite de manuscrit nous utiliserons les termes sparsité et parcimonie de façon interchangeable.

2.3 Modèles neuronaux impulsionnels et synaptiques

Nous avons précédemment étudié les propriétés de différents encodages neuronaux à un niveau d'abstraction élevé ou d'apprentissage avec des règles événementielles, sans considérer le comportement de l'unité computationnelle et de stockage qu'est le neurone impulsionnel et la synapse, respectivement. Les neurones impulsionnels sont connectés entre eux par des synapses, formant ainsi un réseau de neurones impulsionnels (SNN).

Cette section vise à présenter les modèles de neurones impulsionnels et de synapses utilisés dans ce manuscrit.

2.3.1 Modèles de neurones impulsionnels

Les neurones impulsionnels sont des modèles biophysiques de neurones biologiques. Ils modélisent l'évolution temporelle continue des variables d'états interne au neurone via des équations différentielles. La sortie d'un neurone impulsionnel est assimilable à une impulsion discrète (0 ou 1), plutôt qu'une valeur continue comme pour les modèles de neurones formels, dont la sortie continue est une abstraction du codage en taux de décharge. Alors que la dimension temporelle t est totalement ignorée par les modèles de neurones classiques, celle-ci devient explicite pour les neurones impulsionnels, permettant ainsi de transmettre l'information et de calculer explicitement dans la dimension temporelle. L'intérêt de l'introduction explicite du temps a été précédemment illustré, e.g. via un encodage en latences relatives, permettant de transmettre environ 3.3 bits d'information par impulsion neuronale (de nature binaire, rappelons le) en utilisant une fenêtre temporelle de 10 pas de temps.

En outre, Maass (Maass, 1997) a montré que non seulement les SNN sont des approximateurs universels de fonctions, mais que leurs capacités computationnelles sont aussi puissantes, voire plus puissantes dans certains cas, que les autres types de réseaux de neurones (neurones à fonction d'activation Heaviside et sigmoïde). Maass réfère ainsi les SNN à la troisième génération de modèles de réseaux de neurones.

Il existe un grand nombre de modèles de neurones impulsionnels, se situant à différents niveaux de granularité biophysique (voir Fig. 2.10). Le choix du modèle de neurone est dépendant de critères fixés par le chercheur comme la volonté de reproduire fidèlement la trajectoire du potentiel membranaire, variable clé, ou encore de reproduire les formes des trains impulsions (Izhikevich, 2004).

Dans ce manuscrit, nous utiliserons le modèle Intègre et Tire à Fuite *Leaky Integrate and Fire* (LIF). Ce choix est motivé par le fait que ce modèle phénoménologique capture le comportement élémentaire d'un neurone biologique, est résoluble analytiquement, computationnellement peu coûteux, et très utilisé dans une perspective d'apprentissage automatique (*machine learning*) qui est la nôtre. En outre, le modèle LIF est couramment implémenté *in silico* dans les processeurs neuromorphiques (Davies et al., 2018).

L'état d'un neurone LIF est décrit par l'évolution dans le temps t de son potentiel de mem-

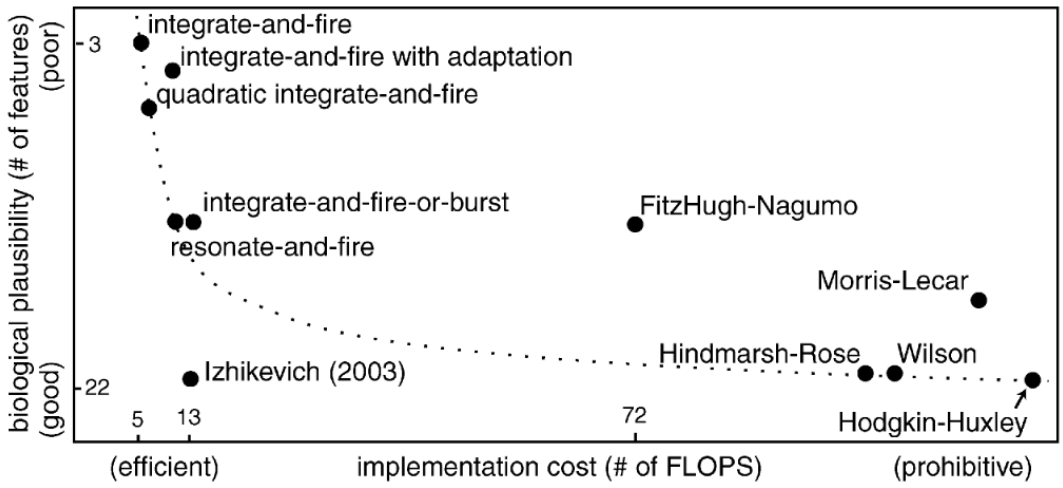

FIGURE 2.10 – Classification des modèles de neurones impulsionnels. Figure tirée de IZHIKEVICH, 2004.

brane $V(t)$ à l'aide d'une équation différentielle ordinaire :

$$\tau_m \frac{dV(t)}{dt} = -V(t) + I(t) \tag{2.11}$$

à laquelle s'ajoute la sortie $s(t)$ et la réinitialisation de $V(t)$ lorsque le seuil de décharge V_θ est atteint, $V(t)$ étant alors maintenu à 0 durant une période réfractaire T_{refrac} :

$$\text{si } V(t) < V_\theta, \quad \text{alors } s(t) = 0 \tag{2.12}$$

$$\text{si } V(t) \geq V_\theta, \quad \text{alors } \begin{cases} s(t) = 1 \\ V(u) = 0 \ \forall u \in]t, t + T_{\text{refrac}}] \end{cases} \tag{2.13}$$

Si aucune entrée n'est reçue, e.g. si $I(t) = 0$, le potentiel membranaire $V(t)$ décroît exponentiellement jusqu'à son potentiel de repos, ici implicitement à 0 par commodité. La vitesse de la décroissance exponentielle est contrôlée par la constante de temps de la membrane τ_m. Une grande (faible) valeur de τ_m implique une décroissance lente (rapide) de $V(t)$. τ_m peut également être interprété comme un taux d'oubli. Si une entrée est reçue, $V(t)$ augmente. Lorsque $V(t)$ a suffisamment intégré l'entrée pour atteindre le seuil de décharge $V(t) \geq V_\theta$, le neurone émet une impulsion $s(t) = 1$ et le potentiel membranaire $V(t)$ est réinitialisé à 0 (voir Fig. 2.11). Le neurone entre alors dans une période réfractaire absolue T_{refrac} durant laquelle il ne peut pas émettre d'impulsion, car $V(t)$ cesse d'intégrer les entrées durant cet intervalle de temps.

Neurones impulsionnels : intégrateurs ou détecteurs de coïncidences ?

Un neurone impulsionnel peut opérer selon deux modes opératoires de décharge : intégrateur ou détecteur de coïncidences (KÖNIG et al., 1996). Ces modes sont liés à la nature de l'encodage neuronal.

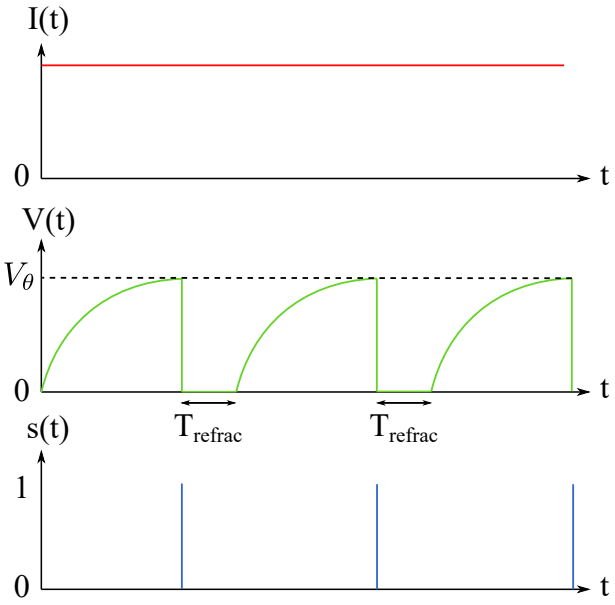

FIGURE 2.11 – Comportement d'un neurone LIF soumis à une entrée constante $I(t)$. Le potentiel membranaire $V(t)$ croît jusqu'à atteindre le seuil de décharge V_θ, à ce moment une impulsion est émise $s(t) = 1$, $V(t)$ est réinitialisé à 0 et le neurone rentre dans une période réfractaire de durée T_{refrac}.

Le mode "intégrateur" implique que le neurone accumule et mémorise l'activité reçue en entrée dans son potentiel membranaire, jusqu'à atteindre son seuil de décharge. Une structure précise des temps d'impulsions n'est alors pas nécessaire, rendant ce mode particulièrement compatible avec un encodage en taux de décharge. Du côté du neurone, le mode "intégrateur" implique généralement une valeur du paramètre τ_m élevée, pour obtenir une faible vitesse de décroissance du potentiel membranaire $V(t)$ et ainsi accumuler et mémoriser l'activité reçue sur une large fenêtre temporelle (PAUGAM-MOISY et BOHTE, 2012).

Par contraste, le mode "détecteur de coïncidences" implique que les impulsions reçues en entrée du neurone soient quasi synchrones pour que le neurone décharge. Ainsi les temps d'impulsions ont une importance cruciale, rendant ce mode particulièrement compatible avec un encodage temporel. Du côté du neurone, le mode "détecteur de coïncidences" implique généralement une valeur du paramètre τ_m faible, pour obtenir une grande vitesse de décroissance du potentiel membranaire $V(t)$, cette fuite rapide de $V(t)$ contraint les impulsions à arriver de façon quasi synchrone, i.e. sur une courte fenêtre temporelle, pour induire une impulsion en sortie (PAUGAM-MOISY et BOHTE, 2012).

Notons qu'un neurone n'est pas tenu d'être cantonné à un unique mode. Il a été montré (POUILLE et SCANZIANI, 2004) que des circuits neuronaux dans l'hippocampe du rat peuvent dynamiquement intervertir leurs modes, i.e. intégrateur ou détecteur de coincidences, pour respectivement extraire le taux de décharge ou les temps d'impulsions à partir des motifs impulsionnels reçus.

En dehors de considérations fines sur le mode opératoire des neurones, une observation générale est qu'un neurone LIF - quelle que soit la valeur de τ_m - est plus sensible à des impulsions synchrones qu'asynchrones, car elles maximisent la probabilité de produire une impulsion en

sortie (voir Fig. 2.12).

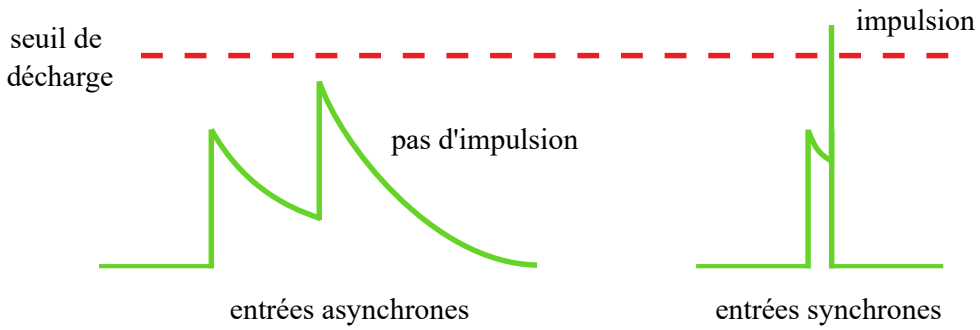

FIGURE 2.12 – Les neurones LIF sont très sensibles aux coïncidences temporelles, i.e. à la (quasi) synchronie des impulsions reçues en entrée.

2.3.2 Modèles de transmission synaptique

Les impulsions neuronales émises par un neurone atteignent d'autres neurones en traversant des connexions synaptiques. Cette section vise à présenter les différents modèles de transmission synaptique utilisés dans ce manuscrit.

Réponse instantanée

La réponse instantanée est le modèle le plus simple de transmission synaptique. Ce modèle n'est pas utilisé dans le manuscrit, mais est introduit ici à des fins didactiques.

Au moment où une impulsion est reçue, le potentiel membranaire du neurone postsynaptique $V_j(t)$ varie instantanément d'une magnitude égale à la valeur du poids synaptique w_{ji}. La connexion synaptique peut être soit excitatrice soit inhibitrice, induisant une croissance ou une décroissance de $V_j(t)$, respectivement. La dynamique du neurone n'est ainsi plus soumise à des entrées continues dans le temps, mais à des entrées événementielles.

En généralisant le comportement d'un neurone présynaptique à n neurones présynaptiques, l'actualisation de $V_j(t)$ est donnée par :

$$V_j(t) \leftarrow V_j(t) + \sum_{i=1}^{n} w_{ji} s_i(t) \tag{2.14}$$

où $s_i(t)$, rappelons le, est une fonction indicatrice indiquant la présence (1) ou l'absence (0) d'une impulsion émise par un neurone présynaptique i.

Réponse avec une décroissance exponentielle

Un modèle exponentiel de synapse peut aussi être utilisé. Il est utilisé dans le chapitre 4. D'un point de vue biologique, un modèle exponentiel de synapse correspond à une transmission synaptique rapide, due à la vitesse de réaction des récepteurs AMPA.

Lorsqu'une impulsion est émise par un neurone présynaptique, la synapse répond avec un saut instantané d'une magnitude égale au poids synaptique, suivi d'une décroissance exponentielle. Cette décroissance exponentielle, équivalente à un filtre basse-bas lissant le signal, peut être exprimée par une équation différentielle linéaire. Plutôt que de modéliser la transmission synaptique avec une équation différentielle linéaire par synapse, une seule équation différentielle est nécessaire pour l'ensemble des synapses, en vertu du principe de superposition des systèmes linéaires. Cela réduit considérablement la charge computationnelle.

Si au moins un neurone présynaptique i décharge, $I_j(t)$ change instantanément :

$$I_j(t) \leftarrow I_j(t) + \sum_{i=1}^{n} w_{ji} s_i(t)$$

sinon $I_j(t)$ décroît exponentiellement dans le temps avec une constante de temps τ_f :

$$\tau_f \frac{dI_j(t)}{dt} = -I_j(t) \tag{2.15}$$

Notons que contrairement au modèle de réponse instantanée changeant instantanément $V_j(t)$, le modèle exponentiel fournit une entrée $I_j(t)$ variant de façon plus continue [5] dans le temps au neurone postsynaptique j. Cela a pour effet de lisser la trajectoire temporelle du potentiel membranaire $V_j(t)$.

Réponse instantanée avec délai de transmission

Jusqu'à présent nous avons considéré des modèles de synapses ne prenant en compte que les poids synaptiques. Or, d'autres paramètres cruciaux sont souvent négligés à cause de la complexité induite, en particulier les délais de transmission. Les délais représentent le temps additionnel pris par une impulsion émise par un neurone présynaptique pour atteindre un neurone postsynaptique.

D'un point de vue biologique, les délais de transmission peuvent être induits par l'épaisseur de la myéline autour des axones (Fields, 2015), l'épaisseur de l'axone (Chéreau et al., 2017), la densité et la réceptivité des canaux ioniques (Steuber et Willshaw, 2004). La distribution des valeurs des délais est riche, elle se déploie sur trois ordres de grandeur de 0.1 ms à 40 ms dans le cortex des mammifères (Swadlow, 1985). Leur rôle est crucial pour de nombreuses tâches comme la localisation sonore ou encore la synchronisation de populations neuronales traitant des entrées visuelles (Sabatini et Regehr, 1999).

Les délais nous intéressent ici moins pour fidèlement rendre compte de la diversité des mécanismes biophysiques, que pour leurs propriétés computationnelles encore trop peu exploitées.

En effet, contrairement aux poids synaptiques, les délais opèrent intrinsèquement dans la dimension temporelle. Ils sont ainsi des paramètres particulièrement pertinents pour traiter des codes temporels. La reproductibilité des délais de transmission *in vivo* et leur existence même est un argument en faveur des codes temporels, comme le fait remarquer Izhikevich : "Pourquoi le cerveau maintiendrait-il différents délais avec une telle précision si le temps d'impulsion n'était pas important ?" (Izhikevich, 2006).

5. Pour être précis, il s'agit d'une continuité à droite, la continuité à gauche n'étant quand même pas assurée aux temps de réception des impulsions présynaptiques.

De plus, les délais relâchent la contrainte de synchronie des impulsions présynaptiques, pour qu'un neurone puisse opérer comme détecteur de coïncidences. Nous avons précédemment évoqué que les neurones opérant comme détecteur de coïncidences étaient adaptés aux codes temporels, puisque très sensibles aux temps d'impulsions. Cependant ce mode nécessite des entrées quasi synchrones, ce qui semble à première vue fortement restreindre l'expressivité des codes temporels compatibles, i.e. la quantité d'information délivrée. Or, les entrées se doivent d'être synchrones au niveau du terminal postsynaptique, pas nécessairement au niveau présynaptique. Un jeu de délais compensant les latences relatives d'impulsions asynchrones a pour effet de rendre ces impulsions synchrones au niveau du terminal postsynaptique. Ainsi, un neurone peut opérer comme un détecteur de coïncidences avec des entrées asynchrones, grâce au rôle de compensation des latences entre les impulsions offert par les délais.

En résumé, les délais offrent des propriétés très attrayantes encore peu exploitées :

1. Ce sont des paramètres qui semblent particulièrement adaptés au traitement de codes temporels puisqu'ils opèrent intrinsèquement dans la dimension temporelle, contrairement aux poids synaptiques.
2. Les délais permettent de généraliser le mode de détection de coïncidences à des entrées asynchrones.

Le modèle de synapse correspond à une transmission instantanée avec un délai de transmission. Il s'agit donc d'une généralisation du modèle de transmission instantanée, tenant compte des délais de transmission. Ce modèle est utilisé dans les chapitres 5 et 6. Le modèle de transmission instantanée est ainsi un cas particulier de ce modèle de transmission avec délais pour des délais nuls.

Chaque synapse possède deux paramètres, un délai d_{ji} et un poids w_{ji}. Considérons un neurone présynaptique i connecté à un neurone postsynaptique j. Le neurone présynaptique i émet une impulsion à un instant t_{pre}, reflété par $s_i(t_{\text{pre}}) = 1$. L'impulsion met un laps de temps supplémentaire d_{ji} pour atteindre le neurone postsynaptique au temps $t = t_{\text{pre}} + d_{ji}$, où elle modifie instantanément $V_j(t)$ d'une magnitude w_{ji}. En généralisant ce comportement à tous les $i = 1, 2, ..., n$ neurones présynaptiques connectés, la variation instantanée de $V_j(t)$ devient égale à la somme des activités présynaptiques retardées par les délais et pondérées par les poids synaptiques :

$$V_j(t) \leftarrow V_j(t) + \sum_{i=1}^{n} w_{ji} s_i(t - d_{ji}) \tag{2.16}$$

Il est bien sûr possible d'augmenter ce modèle avec un modèle de synapse exponentielle, aboutissant ainsi à un modèle de synapse avec une décroissance exponentielle et un délai de transmission.

Dans ce modèle, si au moins une impulsion est émise, $I_j(t)$ - entrée continue d'un neurone postsynaptique j - varie instantanément :

$$I_j(t) \leftarrow I_j(t) + \sum_{i=1}^{n} w_{ji} s_i(t - d_{ji})$$

Sinon $I_j(t)$ décroît exponentiellement dans le temps avec une constante de temps τ_f :

$$\tau_f \frac{dI_j(t)}{dt} = -I_j(t)$$

2.4 Réseaux de neurones impulsionnels in silico : processeurs neuromorphiques

Plutôt que de simuler les SNN, les processeurs neuromorphiques les implémentent *in silico*. Ainsi les différents composants des SNN précédemment présentés tels que le modèle de neurone impulsionnel LIF ou les modèles de transmission synaptique sont implémentés de façon très efficace, offrant notamment une accélération matérielle conséquente. En outre, tandis qu'une opération élémentaire dans un CPU induit un coût énergétique de l'ordre de la nanojoule (10^{-9} J), un évènement synaptique induit un coût de l'ordre d'un picojoule (10^{-12} J) dans un processeur neuromorphique (Furber, 2016). Cela permet théoriquement d'obtenir une efficacité énergétique environ 1000 fois supérieure avec un processeur neuromorphique.

Les processeurs neuromorphiques cherchent à imiter le cerveau en offrant un substrat computationnel massivement parallèle, co-localisant mémoire et calcul. Ils promettent ainsi d'envisager des architectures à très large échelle, éliminant le goulot d'étranglement des architectures de Von Neumann dûe aux coûteux besoins de communications induit par la séparation du calcul et de la mémoire. En outre, contrairement aux GPU, ils opèrent de manière asynchrone sur la base de flux de données événementiels. L'apprentissage est réalisé à la volée à l'aide de règles événementielles.

Ainsi les processeurs neuromorphiques modifient les trois composants clés de l'apprentissage automatique :

1. Des données événementielles tout ou rien (1 ou 0) transmises de façon asynchrone et éparse (matrice creuse) : les impulsions neuronales, plutôt que des données à valeurs réelles transmises de façon synchrone et dense (matrice dense) comme dans les ANN
2. Un substrat de calcul massivement parallèle co-localisant calcul (neurone) et mémoire (synapse), plutôt que de les séparer comme dans les CPU basés sur des architectures de Von Neumann. En outre, les processeurs neuromorphiques fonctionnent de manière asynchrone, en synergie avec les flux de données événementiels, contrairement au fonctionnement synchrone des CPU et GPU.
3. Des algorithmes fonctionnant en synergie avec le niveau matériel co-localisant calcul et mémoire, en utilisant des informations localement disponibles pour actualiser les paramètres soumis à apprentissage, plutôt que des techniques nécessitant une connaissance globale ou partielle du réseau. En outre, ces algorithmes doivent permettre d'actualiser les paramètres à la volée, i.e. de façon asynchrone et indépendante, plutôt que d'actualiser l'ensemble des paramètres du réseau de façon synchrone à chaque itération comme pour la rétropropagation de l'erreur.

Parmi les processeurs neuromorphiques numériques les plus importants, nous pouvons citer SpiNNaker de l'université de Manchester (Painkras et al., 2013), TrueNorth d'IBM (Akopyan et al., 2015) et Loihi d'Intel (Davies et al., 2018). Par exemple la puce Loihi 2 intègre un million de neurones impulsionnels et 120 millions de synapses programmables. La carte *Kapoho Point* combine 8 puces Loihi2 sur la surface d'une carte de crédit, offrant ainsi 8 millions de neurones

impulsionnels et environ 1 milliard de synapses. Enfin il est possible d'empiler ces cartes de sorte à encore augmenter la puissance computationnelle disponible.

Ces différents processeurs neuromorphiques sont programmables, de sorte à notamment pouvoir implémenter de nouvelles règles d'apprentissage événementielles.

Notons également que les processeurs neuromorphiques font partie d'un écosystème plus large, dont font aussi partie les capteurs neuromorphiques ou encore les mécanismes d'encodage neuronaux (voir Fig. 2.13).

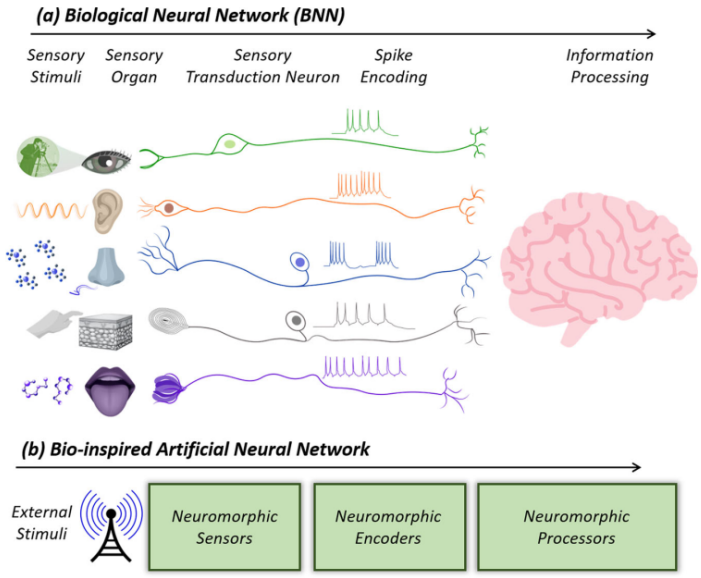

FIGURE 2.13 – Le calcul neuromorphique cherche à imiter le fonctionnement du cerveau. Son périmètre englobe capteurs neuromorphiques, mécanismes d'encodage neuronaux et processeurs neuromorphiques. Figure tirée de SUBBULAKSHMI RADHAKRISHNAN et al., 2021.

Chapitre 3

Codage et décodage des entrées avec un code temporel

> We must use time as a tool, not as a couch.
>
> John F. Kennedy

Sommaire

Dans la section 2.1 nous avons exploré les capacités théoriques maximales de différents codes neuronaux à un niveau élevé d'abstraction, i.e. sans considérer de mécanismes d'encodage précis. Nous avons retenu pour ses performances un encodage temporel par population de neurones, spécifiquement un encodage en latences relatives. Nous voulons à présent instancier cet encodage en latences relatives à l'aide de circuits neuronaux.

Ce chapitre présente une variante de la méthode d'encodage en latences relatives proposée par Bohté et al., 2002 et similaire à la variante proposée par Rumbell et al., 2014. La méthode de Rumbell et al. augmente la méthode de Bohte en permettant la présentation en continu d'entrées au SNN, tout en assurant la reproductibilité du code neuronal, cruciale pour un code temporel où l'information est contenue dans le temps précis des impulsions. La reproductibilité est assurée par la génération d'oscillations produites par des interactions excitatrices-inhibitrices

entre des neurones encodeurs recevant une entrée continue et un neurone inhibiteur. Ces oscillations permettent ainsi de présenter une entrée en continu dans le temps au SNN - sans qu'il soit nécessaire de réinitialiser l'entrée à zéro - tout en préservant les temps précis des impulsions en sortie. Ce phénomène de reproductibilité des temps relatifs des impulsions grâce au comportement oscillatoire des neurones encodeurs est appelé verrouillage de phase (*phase locking*).

Notre méthode d'encodage en latences relatives présentée dans ce chapitre offre un comportement équivalent à la méthode de Rumbell et al., 2014, i.e. assure la reproductibilité du code neuronal, tout en réduisant les besoins computationnels :

— pas de fonction alpha pour modéliser la transmission synaptique

— pas de neurone inhibiteur et donc pas de connexions multi-synaptiques entre les neurones encodeurs et le neurone inhibiteur

Cependant notre variante a pour inconvénient de ne pas permettre la présentation continue d'une entrée au SNN : elle nécessite une réinitialisation de l'entrée. Il demeure néanmoins aisé de changer la méthode d'encodage suivant les contraintes applicatives, les motifs impulsionnels produits en sortie de la couche d'encodage étant équivalents.

La méthode d'encodage temporel de Bohté et al., 2002 est utilisée dans de nombreux travaux (Hussain et Thounaojam, 2020 ; Kasabov et al., 2013 ; Rumbell et al., 2014 ; Yu et al., 2014). Cependant aucune méthode de décodage n'a été proposée, i.e. une méthode pour reconstruire l'entrée à partir de l'activité neuronale. De plus aucune étude n'a été menée sur 1) la diversité de ce code neuronal, i.e. la quantité d'information délivrée et 2) la structure du code neuronal, i.e. l'espace occupé par l'activité neuronale dans l'espace des états. Ce chapitre a pour principales contributions de combler ces manques. En résumé nous présentons :

— une variante d'une méthode simple et biologiquement plausible d'encodage des entrées sous forme de motif impulsionnel et appartenant à la classe des codes temporels, spécifiquement un encodage en latences relatives

— une méthode de décodage pour reconstruire l'entrée à partir de la réponse neuronale

— une analyse de l'information délivrée par ce code temporel et de sa structure dans l'espace des états

3.1 Méthodes pour l'encodage et le décodage

3.1.1 Codage neuronal en latences relatives

Cette section présente les mécanismes utilisés pour passer d'un signal continu multidimensionnel à une représentation de ce signal sous forme d'impulsions neuronales émises dans le domaine temporel. Cette transformation d'une entrée continue sous forme d'impulsions neuronale est appelée encodage neuronal. Une impulsion - considérée comme un évènement discret binaire (0 ou 1) - est l'unité de communication fondamentale des SNN. Le cas de l'encodage d'une unique dimension de l'espace d'entrée en motif d'impulsions sera d'abord traité, puis généralisé à un nombre de dimensions k arbitrairement grand.

Soit $a \in [0, 1]$ un scalaire normalisé à valeur réelle, qu'on veut encoder dans un motif d'impulsions. Plus précisément, on veut représenter ce scalaire $a \in [0, 1]$ avec un codage neuronal prenant la forme de latences relatives (ou temps d'impulsions relatifs). Il s'agit ainsi d'un code appartenant à la classe des codes temporels, où l'information est encodée dans le temps précis

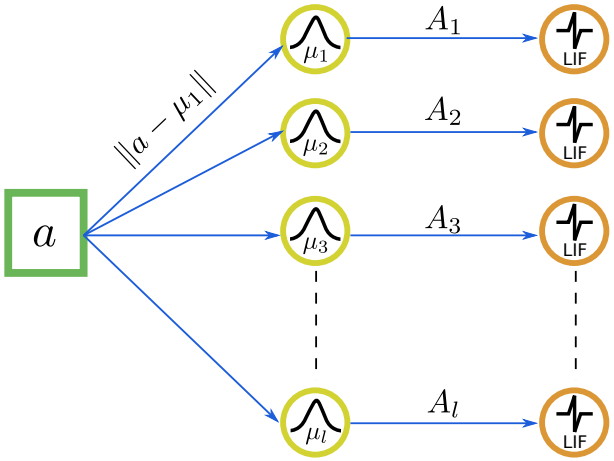

FIGURE 3.1 – Schéma fonctionnel de l'encodage par population neuronale d'une valeur réelle a. La première étape est réalisée en donnant a en entrée à l champs récepteurs gaussiens, générant l niveaux d'activations $A_1, A_2, ..., A_l$ en sortie. La seconde étape est réalisée en donnant ces niveaux d'activations $A_1, A_2, ..., A_l$ en entrée à l neurones LIF. La sortie de la population de l neurones prend la forme d'un motif impulsionnel déployé dans la dimension temporelle et constituant l'encodage neuronal de la variable d'entrée.

des impulsions. Ce code temporel peut être réalisé en distribuant le scalaire a sur une population de l neurones. On réalise ainsi un codage par population de neurones. Une population de $l = 10$ neurones a été utilisée pour collectivement représenter un scalaire. Chaque neurone de la population n'émet qu'une seule impulsion dans une fenêtre d'encodage temporelle T. Par conséquent, un scalaire $a \in [0, 1]$ est encodé dans un motif d'impulsions $\mathbf{t}$ dans le domaine spatio-temporel via l neurones, i.e. $a \in [0, 1] \rightarrow \mathbf{t} \in [0, T]^l$.

Mécanismes pour le codage

La transformation d'un scalaire $a \in [0, 1]$ en un motif d'impulsions spatio-temporel $\mathbf{t} \in [0, T]^l$ est réalisée en deux étapes chaînées (voir Fig. 3.1) :

1. La première étape implique de donner a en entrée à l champs récepteurs gaussiens, dans une relation un-à-tous. Les champs récepteurs produisent en sortie un vecteur de niveaux d'activations $\mathbf{A} \in [0, 1]^l$.
2. La deuxième étape implique de donner ce vecteur de niveaux d'activations $\mathbf{A} \in [0, 1]^l$ en entrée à l neurones LIF, dans une relation un-à-un. Les niveaux d'activations sont intégrés dans le temps par la population de neurones, produisant en sortie un motif d'impulsions, $\mathbf{t} \in [0, T]^l$. Ce motif d'impulsions spatio-temporel ou vecteur de temps d'impulsions $\mathbf{t} \in [0, T]^l$ est la représentation finale, i.e. le code neuronal, de la valeur $a \in [0, 1]$ présentée en entrée du réseau.

La première étape, i.e. la transformation d'un scalaire $a \in [0, 1]$ en niveaux d'activations $\mathbf{A} \in [0, T]^l$, est réalisée avec des champs récepteurs gaussiens. Chaque $i \in \{1, 2, ..., l\}$ champ récepteur gaussien est placé dans un espace circulaire dans $[0, 1]$ et est réglé par deux paramètres :

- sa valeur préférentielle (ou centre) μ_i. Le centre μ_i règle la valeur dans l'espace d'entrée

[0,1] pour laquelle le champ récepteur génère un niveau d'activation A_i maximal. Les centres μ_i sont uniformément distribués dans un espace circulaire. Ainsi pour une population de $l = 10$ neurones, les centres sont espacés d'un pas de $1/l = 0.1$, donc $\mu_i \in \{0.05, 0.15, ..., 0.95\}$.

- son étendue (ou écart-type) σ. L'écart-type $\sigma = 0.6$ règle l'étendue du champ récepteur dans l'espace d'entrée, et est identique pour tous les champs récepteurs. Ce large champ récepteur garantit la génération d'un niveau d'activation suffisamment élevé pour que tout neurone puisse décharger en réponse à toute valeur dans l'espace d'entrée $[0, 1]$. Les champs récepteurs se recoupent et couvrent l'intégralité de l'espace d'entrée circulaire.

Un champ récepteur gaussien est modélisé par une fonction gaussienne $G_{\mu_i,\sigma} : a \to A_i$ qui transforme le scalaire $a \in [0, 1]$ en un niveau d'activation $A_i \in [0, 1]$:

$$G_{\mu_i,\sigma}(a) = \exp\left(-\frac{d(a, \mu_i)^2}{2\sigma^2}\right) \tag{3.1}$$

La fonction gaussienne étant placée dans un espace circulaire, la distance $d(a, \mu_i)$ entre une entrée a et un centre μ_i est calculée par la différence de coordonnées circulaires :

$$d(a, \mu_i) = \begin{cases} \Delta_d & \text{si } \Delta_d < 0.5 \\ 1.0 - \Delta_d & \text{sinon} \end{cases} \tag{3.2}$$

$$\Delta_d = |\mu_i - a| \tag{3.3}$$

Le niveau d'activation A_i peut être interprété comme une mesure de similarité entre l'entrée $a \in [0, 1]$ et le centre μ_i, produite par un noyau gaussien. Un fort (faible) niveau d'activation $A_i \in [0, 1]$ traduit une forte (faible) similarité. En passant du niveau du champ récepteur individuel au niveau de la population, on obtient un vecteur de niveaux d'activations $\mathbf{A} \in [0, 1]^l$ en sortie des champs récepteurs. Utiliser cette mesure de similarité plutôt que la valeur absolue de l'entrée permet d'être insensible à la magnitude de l'entrée, et ainsi en dernière analyse de rendre le code temporel insensible à la magnitude de l'entrée.

La deuxième étape, i.e. la transformation du vecteur de niveaux d'activations $\mathbf{A} \in [0, 1]^l$ en motif d'impulsions $\mathbf{t} \in [0, T]^l$, est réalisée en donnant $\mathbf{A} \in [0, 1]^l$ en entrée durant une période $T/2$ aux l neurones LIF de la couche d'encodage. Pour rappel, l'évolution du potentiel membranaire d'un neurone LIF est donnée par :

$$\tau_m \frac{dV_i(t)}{dt} = -V_i(t) + I_{\text{ext}}(t) \tag{3.4}$$

où $I_{\text{ext}}(t)$ est l'entrée du neurone, avec $I_{\text{ext}}(t) = A_i$ durant l'intervalle $[0, T/2]$ et $I_{\text{ext}}(t) = 0$ durant l'intervalle $[T/2, T]$.

Le $i$ème neurone - avec $i \in \{1, 2, ..., l\}$ - dont le champ récepteur associé avec pour centre μ_i est le plus proche de la valeur d'entrée a, reçoit le niveau d'activation A_i le plus élevé. Par conséquent, le potentiel membranaire du $i$ème neurone croît plus vite que celui des autres neurones, atteint son seuil de décharge V_θ plus vite et émet une impulsion en premier. Les autres neurones reçoivent des niveaux d'activations plus faibles, atteignent plus tardivement leur seuil V_θ et déchargent donc plus tardivement. Plus la distance entre leur centre et la valeur d'entrée est grande, plus les niveaux d'activation qu'ils reçoivent sont faibles, et plus ils déchargent tard

(Fig. 3.2). Autrement dit, plus la distance entre un centre et une valeur d'entrée est grande, plus le temps d'impulsion est grand.

La séparation des temps d'émissions des impulsions est accentuée par l'intégration non linéaire (exponentielle) des niveaux d'activations par les neurones LIF. Ainsi le temps d'émission d'une impulsion croit non-linéairement en fonction du niveau d'activation reçu, niveau d'activation lui-même fonction de la distance entre un centre μ_i et la valeur d'entrée a.

En bref, grâce à ce mécanisme d'encodage neuronal, une valeur d'entrée spécifique $a \in [0, 1]$ est encodée dans un motif d'impulsions spécifique $\mathbf{t} \in [0, T]^l$ dans le domaine spatio-temporel (voir Fig. 3.2).

Mise à l'échelle du code neuronal

Ce schème d'encodage par population peut être facilement mis à l'échelle. Autrement dit nous pouvons le généraliser d'une unique dimension $a \in [0, 1]$ dans l'espace d'entrée à un nombre arbitrairement grand de dimensions k (voir Fig. 3.4). Chaque dimension étant représentée par une population de l neurones, un vecteur d'entrée à k dimensions $\mathbf{a} \in [0, 1]^k$ devient représenté par la sortie de $n = k * l$ neurones sur une fenêtre temporelle T. Par conséquent, un vecteur d'entrée de k valeurs réelles est encodé dans un motif d'impulsions $\mathbf{t}$ dans le domaine spatio-temporel via $n = k * l$ neurones, i.e. $\mathbf{a} \in [0, 1]^k \rightarrow \mathbf{t} \in [0, T]^n$.

Reproductibilité du code neuronal

Le temps précis des impulsions étant crucial pour encoder fidèlement des valeurs continues, il est nécessaire d'assurer la reproductibilité du code neuronal. Le caractère reproductible du code neuronal est assuré par la conjonction d'une suppression du stimulus donné en entrée des neurones LIF et d'une période réfractaire durant laquelle les neurones n'intègrent plus l'entrée. Grâce à ce mécanisme, les conditions initiales des neurones sont les mêmes, i.e. les potentiels membranaires des neurones sont les mêmes lors de la présentation d'une nouvelle entrée. Par conséquent la trajectoire temporelle des potentiels membranaires est reproductible, ce qui garantit la reproductibilité des temps d'impulsions (voir Fig. 3.5). De plus, ce mécanisme permet la génération d'une unique impulsion par neurone, contrairement à un code par taux de décharge. Enfin cet encodage neuronal rend le nombre d'impulsions émises par une population de neurones invariant à la valeur de l'entrée. Cela facilite la prise de décision de neurones situés en aval, e.g. le nombre d'impulsions reçues par un neurone étant constant, cela facilite le réglage du seuil de décharge V_θ qui peut être interprété comme un seuil de décision.

Code neuronal unimodal

Ce code neuronal présente la remarquable propriété d'être unimodal (voir Fig 3.6). Cela est dû au caractère gaussien des champs récepteurs ainsi qu'à l'espace circulaire dans lequel ils sont placés. Dans notre cas, plutôt que d'utiliser un code par taux de décharge unimodal où chaque neurone émet un nombre d'impulsions dépendant de la distance entre sa valeur préférentielle et une valeur d'entrée, nous utilisons un code temporel unimodal, où chaque neurone émet une unique impulsion dont la latence dépend de la distance entre sa valeur préférentielle et une valeur d'entrée. Les codes unimodaux sont utilisés par exemple par les rats pour coder leurs positions spatiale dans leurs environnements par des cellules de lieu situées dans l'hippocampe

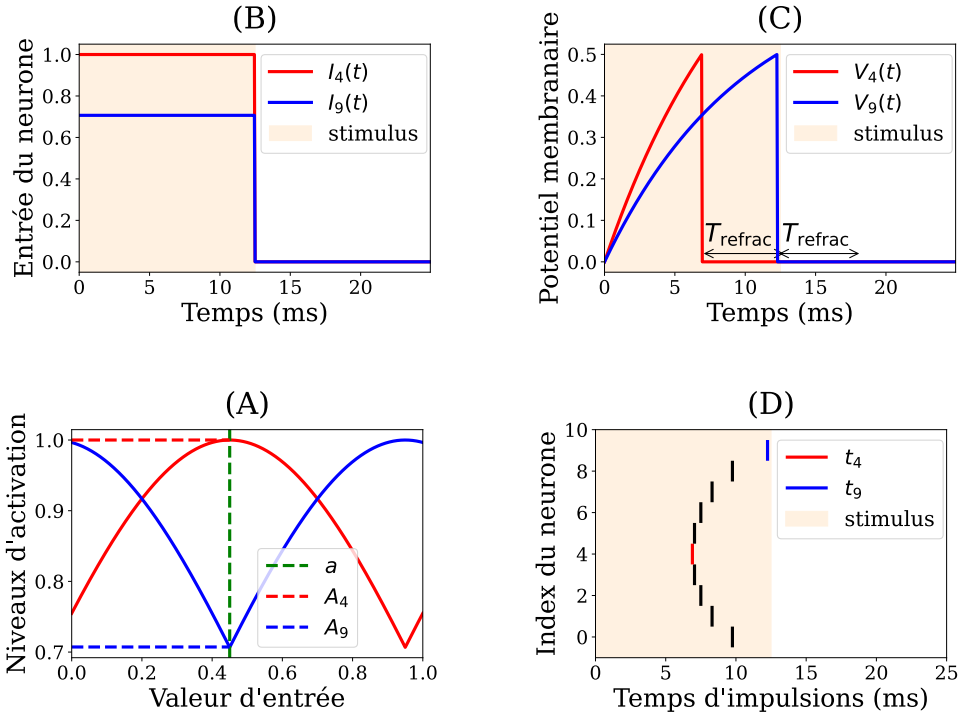

FIGURE 3.2 – Illustration des étapes successives (A)-(D) transformant une valeur d'entrée de 0.45 en un motif impulsionnel spatio-temporel. (A) Une population de $l = 10$ champs récepteurs gaussien reçoit cette entrée. Chaque $i$ème champ récepteur gaussien, avec $i = 0, 1, ..., l-1$ est centré sur sa valeur préférentielle μ_i. Les courbes d'accord de deux champs récepteurs ayant pour centres $\mu_4 = 0.45$ et $\mu_9 = 0.95$ sont affichées en ligne continue en rouge et en bleu, respectivement, ainsi que les niveaux d'activation A_4 et A_9 résultants pour une valeur d'entrée $a = 0.45$. Remarquons que le niveau d'activation A_4 est maximal (1.0) car la valeur préférentielle $\mu_4 = 0.45$ du champ récepteur est égale à la valeur d'entrée 0.45. (B) Ces niveaux d'activation A_4 et A_9 sont maintenus constants pendant un intervalle de temps de 12.5 ms, puis sont suivis d'une période de repos de même durée, ce qui génère $I_4(t)$ et $I_9(t)$, respectivement. (C) $I_4(t)$ et $I_9(t)$ sont intégrés par les potentiels membranaires $V_4(t)$ et $V_9(t)$ des neurones LIF. Le potentiel membranaire $V_4(t)$ croît plus rapidement que $V_9(t)$ car $I_4(t) > I_9(t)$. Par conséquent, $V_4(t)$ atteint plus rapidement son seuil de décharge V_θ, le neurone émet une impulsion, remet $V_4(t)$ à 0 et entre dans une période réfractaire T_{refrac}. (D) Chaque neurone émet une impulsion lorsqu'il franchit son seuil de décharge V_θ. La variabilité des niveaux d'activation reçus en entrée par la population de neurones est traduite en sortie par la variabilité des temps d'impulsions. Ainsi, par la conjonction de champs récepteurs gaussiens et de neurones LIF, la valeur d'entrée spécifique de 0.45 est transformée en un motif impulsionnel spatio-temporel spécifique.

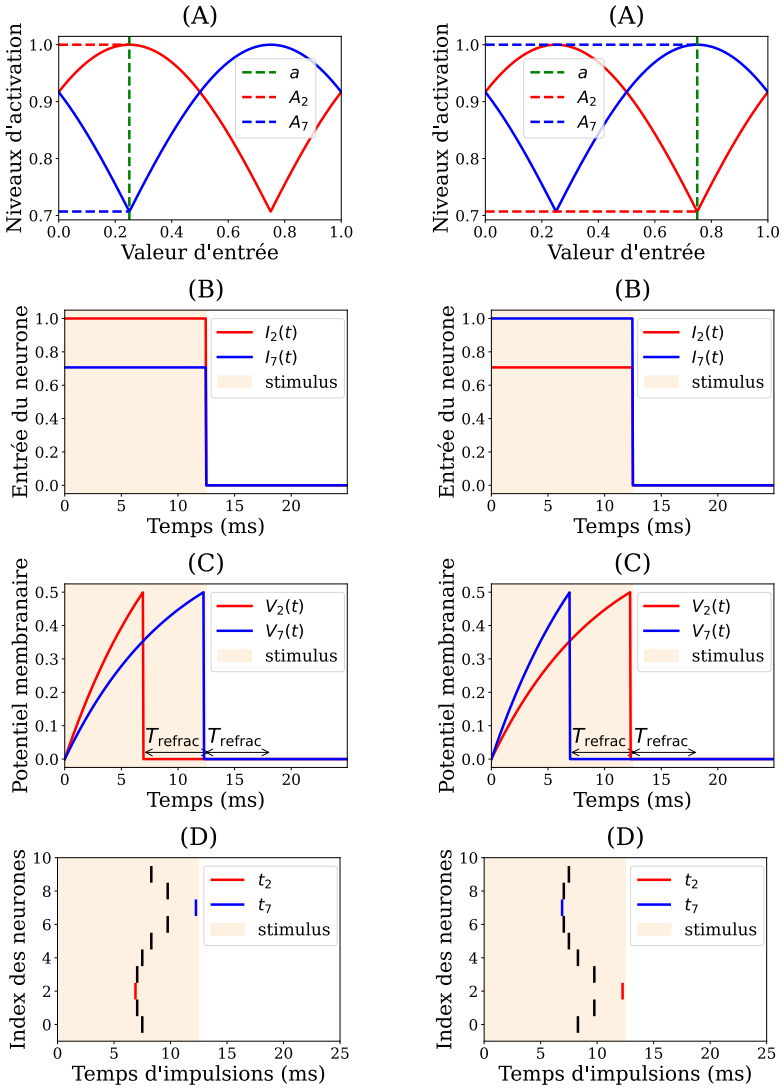

FIGURE 3.3 – Illustration d'un cas de translation spatiale d'un motif impulsionnel. La première et la seconde colonne illustrent l'encodage neuronal d'une entrée $a = 0.25$ et d'une autre entrée $a = 0.75$, respectivement. La rangée (A) montre que le neurone 2 a un champ récepteur centré sur $\mu_2 = 0.25$, tandis que le neurone 7 a un champ récepteur centré sur $\mu_7 = 0.75$. La valeur des niveaux d'activations A_2 et A_7 est inversée quand on passe d'une valeur d'entrée $a = 0.25$ à $a = 0.75$. L'inversion des niveaux d'activations A_2 et A_7 est dûe à l'inversion des distances entre les entrées $a = 0.25$ et $a = 0.75$, et les centres $\mu_2 = 0.25$ et $\mu_7 = 0.75$. (C) L'inversion des niveaux d'activations A_2 et A_7 se répercute à son tour par l'inversion de la trajectoire des potentiels membranaires $V_2(t)$ et $V_7(t)$. (D) Le résultat final est que la structure des temps de décharge représentant l'entrée $a = 0.25$ est la même que pour $a = 0.75$, dans le sens où cette structure a uniquement subi une opération de translation spatiale. Cette translation spatiale est due à la symétrie au centre des gaussiennes, ainsi qu'à l'espace circulaire empêchant les effets de bords (i.e. coupure de la gaussienne). De façon générique, toute translation d'une valeur d'entrée a par un multiple k de $1/l$ avec l étant le nombre de neurones, génère une translation spatiale du motif impulsionnel représentant a. Plus formellement, si on considère une valeur continue $a \in [0, 1]$ transformée en un motif impulsionnel $\mathbf{t} \in [0, T]^l$, alors toute valeur $a' \in (a \pm k/l) \bmod 1$ induit un motif impulsionnel $\mathbf{t}' \in [0, T]^l$ qui est une translation spatiale de $\mathbf{t} \in [0, T]^l$.

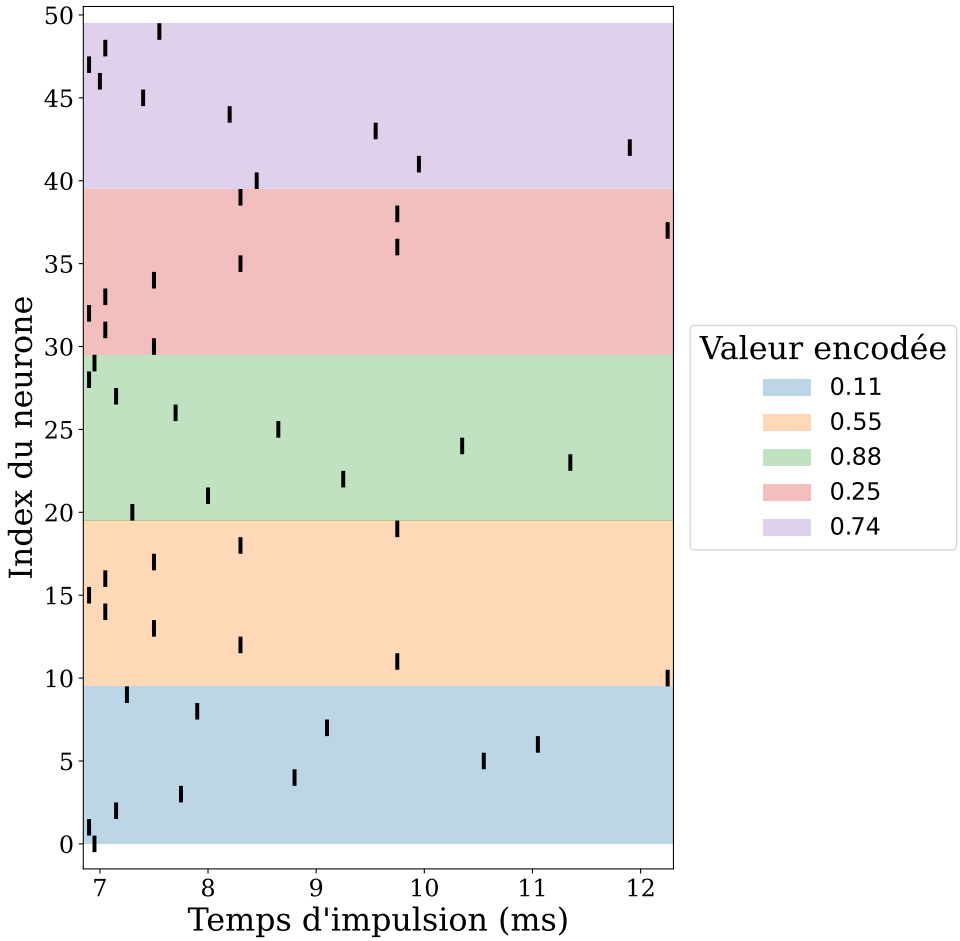

FIGURE 3.4 – Illustration de la scalabilité du mécanisme d'encodage neuronal : généralisation de l'encodage à un vecteur d'entrée. Dans cet exemple, le vecteur d'entrée est constitué de 5 dimensions. Ce vecteur de 5 dimensions est encodé par les impulsions émises par $5 * l$ neurones, avec $l = 10$. Dans cet exemple le vecteur d'entrée prend les valeurs continues $\mathbf{a} = [0.11, 0.55, 0.88, 0.25, 0.74]$. Ainsi chaque valeur du vecteur est codée par les temps d'impulsions relatifs d'une population de $l = 10$ neurones.

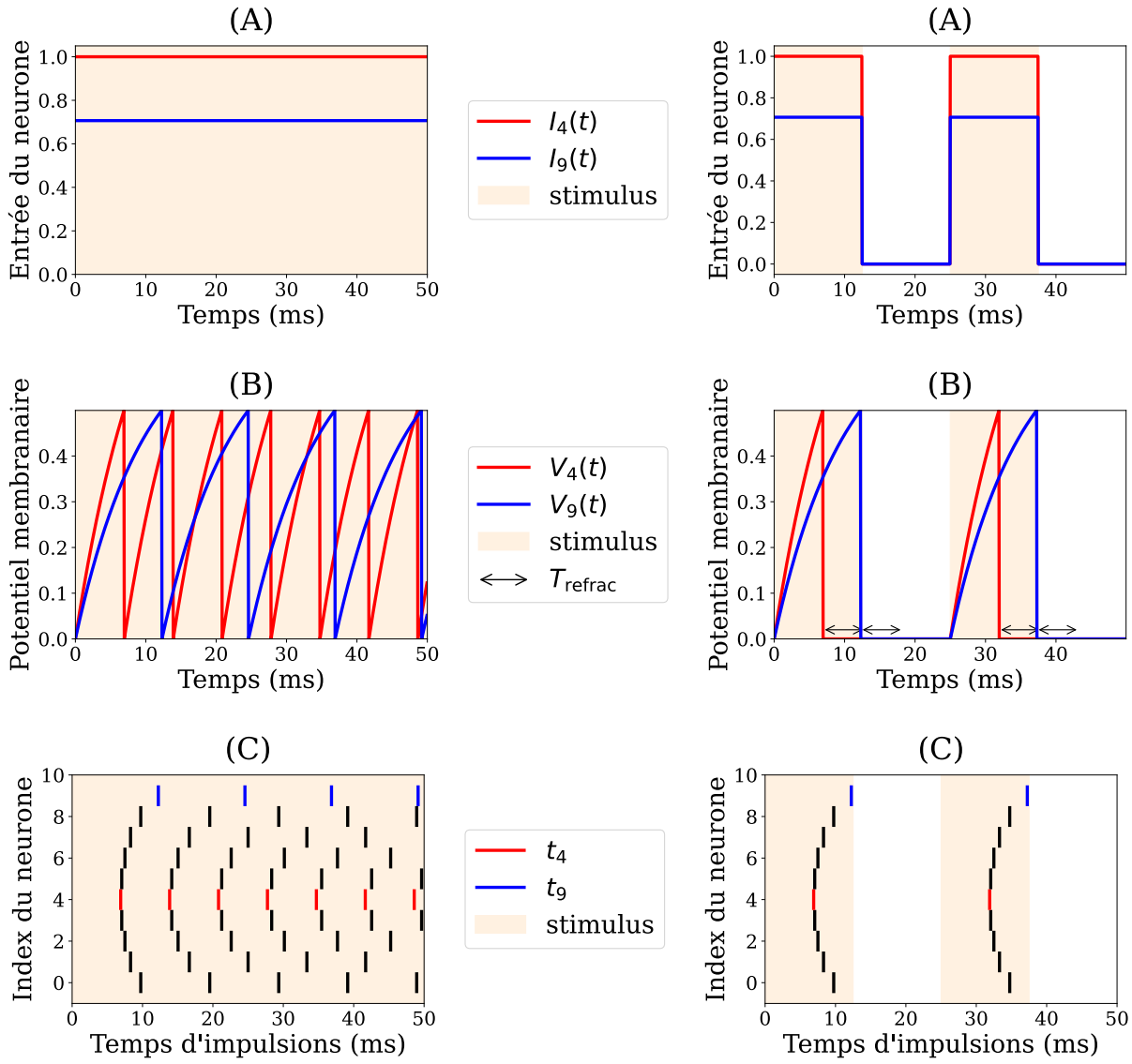

FIGURE 3.5 – Illustration du caractère reproductible du code neuronal. Dans cet exemple on présente en entrée la valeur 0.45 qu'on souhaite encoder dans un motif impulsionnel reproductible, la précision des temps d'impulsions étant cruciale pour encoder fidèlement une valeur continue. Dans la colonne de gauche, le stimulus - correspondant au niveau d'activation en sortie d'un champ récepteur - présenté en entrée d'un neurone est constant dans le temps et le neurone ne rentre pas dans une période réfractaire après avoir déchargé. Cela a pour conséquence que les temps d'impulsions relatifs se désynchronisent rapidement et que le motif impulsionnel n'est pas reproductible. Le résultat est un code par taux de décharge, non un code temporel. Dans la colonne de droite on observe comment un code temporel reproductible est réalisé par l'interaction entre la suppression du stimulus en entrée des neurones, ainsi que d'une période réfractaire. Ainsi les potentiels membranaires sont à zéro lorsqu'une nouvelle entrée est présentée. Cela assure la reproductibilité du motif impulsionnel car les conditions initiales des neurones sont les mêmes lors de la présentation d'une nouvelle entrée. De plus, contrairement à un code par taux de décharge, les neurones n'émettent qu'une seule impulsion.

(Burgess, 2014). Une méthode de décodage tirant profit du caractère unimodal du code neuronal est proposée dans la section suivante.

3.1.2 Mécanismes non-neuronaux pour le décodage des latences relatives

Après avoir présenté le mécanisme d'encodage neuronal, cette section présente la méthode de décodage non-neuronale utilisée. Cette méthode correspond au point de vue d'un observateur extérieur au système, souhaitant reconstruire le stimulus d'entrée à partir de l'activité neuronale. Nous considérons ici le cas du décodage d'une unique dimension, que nous pouvons très facilement généraliser à un nombre k arbitrairement grand de dimensions en appliquant la même procédure à chaque dimension.

Une première stratégie de décodage sous-optimale

Une première méthode de décodage naïve est de retrouver un singleton parmi l'ensemble des $l = 10$ valeurs préférentielles $\mu_i \in \{0.05, 0.15, ..., 0.95\}$ des neurones. Ce singleton est obtenu en considérant la valeur préférentielle μ_i du neurone représentant le mieux la valeur encodée. Cela correspond à la valeur préférentielle μ_i du neurone émettant la première impulsion au sein du motif impulsionnel $\mathbf{t} \in [0, T]^l$. On obtient avec cette méthode une résolution de décodage de $1/l$, l'intervalle de variation de la variable d'entrée étant de largeur 1. Le gain en termes de résolution est ainsi une fonction linéaire du nombre l de neurones utilisés. En considérant les $l = 10$ neurones utilisés expérimentalement, on obtient une résolution de 0.1, soit une résolution en bits de $\log_2(10) \approx 3.3$ bits.

Une deuxième stratégie de décodage tirant profit des latences

Cependant cette méthode ne tire pas profit de l'intégralité de l'information. Elle ne considère que l'identité du premier neurone émettant une impulsion, au lieu de considérer l'activité neuronale collective, i.e. l'information codée dans les latences relatives des impulsions. Nous présentons ici une méthode de décodage permettant d'obtenir une estimation $\hat{a} \in [0, 1]$ à partir des latences relatives au sein du motif impulsionnel $\mathbf{t} \in [0, T]^l$, encodant la valeur d'entrée $a \in [0, 1]$ (voir Fig. 3.6).

Soit $\{\Delta_{t_0}, \Delta_{t_1}, ..., \Delta_{t_{l-1}}\}$ un ensemble de l latences relatives calculées à partir des temps d'impulsions contenus dans le motif impulsionnel $\mathbf{t} \in [0, T]^l$ comme suit :

$$\Delta_{t_i} = |t_i - t_{\max}| \tag{3.5}$$

$$t_{\max} = \max_i \{t_i\} \tag{3.6}$$

Ainsi le neurone émettant la première impulsion induira la plus grande latence. Maintenant que nous avons calculé un ensemble de latences, nous réalisons le décodage avec une moyenne circulaire pondérée, ou centre de masse pour système périodique. Ce choix est motivé par le caractère unimodal du code neuronal ainsi que par l'espace circulaire dans lequel sont placés les champs récepteurs gaussiens. La première étape du processus de décodage consiste à transformer chaque valeur préférentielle $\mu_i \in \{0.05, 0.15, ..., 0.95\}$ des neurones en un angle θ_i (direction préférentielle) :

$$\theta_i = 2\pi\mu_i \tag{3.7}$$

Nous calculons ensuite les moyennes pondérées des coordonnées cartésiennes $\overline{x}$ et $\overline{y}$ à partir des angles θ_i (coordonnées angulaires) et des latences Δ_{t_i} (coordonnées radiales) :

$$\overline{x} = \frac{1}{\Delta}\sum_{i=1}^{l} \Delta_{t_i}\cos(\theta_i) \tag{3.8}$$

$$\overline{y} = \frac{1}{\Delta}\sum_{i=1}^{l} \Delta_{t_i}\sin(\theta_i) \tag{3.9}$$

$$\Delta = \sum_{i=1}^{l} \Delta_{t_i} \tag{3.10}$$

Nous calculons un nouvel angle $\overline{\theta}$ à partir des coordonnées cartésiennes $\overline{x}$ et $\overline{y}$. Cet angle $\overline{\theta}$ correspond à la moyenne circulaire pondérée :

$$\overline{\theta} = \text{atan2}\left(-\overline{y}, -\overline{x}\right) + \pi \tag{3.11}$$

Nous finissons par une transformation inverse, transformant l'angle $\overline{\theta} \in [0, 2\pi]$ en une valeur dans l'espace d'entrée $\hat{a} \in [0, 1]$:

$$\hat{a} = \frac{\overline{\theta}}{2\pi} \tag{3.12}$$

3.1.3 Etude de l'erreur de décodage

Après avoir exposé la méthode de décodage [1], nous souhaitons à présent quantifier sa précision. Une méthode adéquate doit induire une faible erreur de décodage.

1. Il est intéressant de noter qu'une moyenne circulaire maximise la vraisemblance du paramètre μ (centre) de la distribution de probabilité de von Mises (Diard et al., 2013). La distribution de von Mises ressemble à une gaussienne et a comme propriété d'être unimodale et définie sur un espace circulaire. De façon analogue, une moyenne circulaire pondérée correspond au maximum de vraisemblance de l'entrée (ou stimulus) pour des neurones avec des courbes d'accord gaussiennes et une variabilité du nombre d'impulsions suivant une loi de Poisson. Les courbes d'accord gaussiennes modélisent dans ce cas le nombre d'impulsions moyen émis en fonction de la distance de l'entrée à la direction préférentielle θ_i du neurone. On se situe donc dans le cadre d'un codage par taux de décharge. La loi de Poisson modélise de façon probabiliste la variabilité du nombre d'impulsions émis par un neurone. En bref, la moyenne circulaire des directions préférentielles θ_i des neurones, pondérée par l'activité neuronale est dans ce cas l'estimateur du maximum de vraisemblance de l'entrée. La moyenne circulaire pondérée a par exemple été utilisée pour décoder avec succès la direction du mouvement du bras à partir de l'activité de populations de neurones dans le cortex moteur des primates (Georgopoulos, Schwartz et al., 1986).

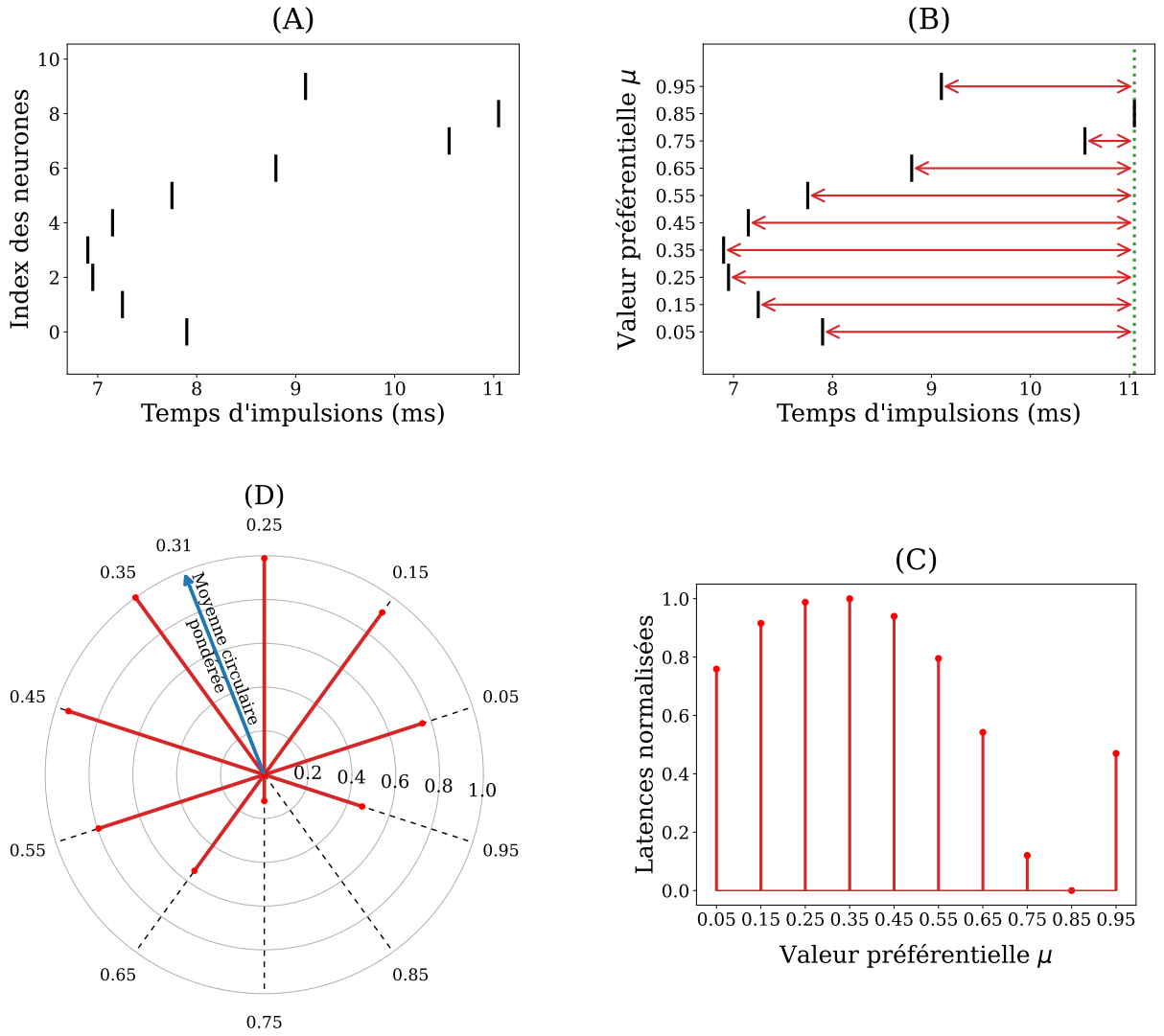

FIGURE 3.6 – Illustration de la méthode de décodage du code neuronal. (A) Une valeur $a = 0.31$ est encodée dans un motif impulsionnel. (B) Il y a correspondance entre l'indice $i \in \{0, 1..., 9\}$ d'un neurone et sa valeur préférentielle $\mu_i \in \{0.05, 0.15, ..., 0.95\}$ dans l'espace d'entrée $[0, 1]$. Les latences relatives sont affichées avec les flèches rouges. (C) Nous pouvons interpréter le motif impulsionnel comme un vecteur de poids (latences) associé à un vecteur de valeurs préférentielles. (D) L'espace des champs récepteurs étant circulaire, chaque couple latence - valeur préférentielle est transformé en deux coordonnées polaires, une coordonnée angulaire (angle) et une coordonnée radiale (rayon), respectivement. En utilisant une moyenne circulaire pondérée, nous retrouvons la valeur d'entrée 0.31. En résumé nous avons décodé le motif impulsionnel en utilisant une moyenne circulaire pondérée par la réponse (la latence) des valeurs préférentielles μ_i des neurones.

Erreur de décodage avec des temps d'impulsions parfaits

Afin de calculer l'erreur de décodage idéale, nous déterminons analytiquement le temps d'émission d'une impulsion par un neurone LIF. Cela correspond à un temps d'impulsion infiniment précis. Nous considérons que le neurone reçoit une entrée non nulle et constante dans le temps. Cela correspond à la première moitié $T/2$ de la fenêtre temporelle d'encodage, durant laquelle chaque neurone $i \in \{0, 1, ..., l-1\}$ intègre un niveau d'activation A_i (Fig. 3.2). Rappelons que l'équation différentielle régissant l'évolution du potentiel membranaire $V_i(t)$ d'un neurone LIF, pour une entrée constante A_i est :

$$\tau_m \frac{dV_i(t)}{dt} = -V_i(t) + A_i \tag{3.13}$$

La résolution analytique de cette équation différentielle ordinaire donne :

$$V_i(t) = A_i \left(1 - \mathrm{e}^{\frac{-t}{\tau_m}}\right) \tag{3.14}$$

Nous voulons déterminer analytiquement le temps d'émission d'une impulsion, i.e. déterminer le temps t_{θ_i} pris par le circuit décrit par l'équation 3.14 pour atteindre son seuil de décharge V_θ :

$$V_\theta = A_i \left(1 - \mathrm{e}^{\frac{-t_{\theta_i}}{\tau_m}}\right) \tag{3.15}$$

$$1 - \frac{V_\theta}{A_i} = \mathrm{e}^{\frac{-t_{\theta_i}}{\tau_m}} \tag{3.16}$$

$$\ln\left(1 - \frac{V_\theta}{A_i}\right) = \frac{-t_{\theta_i}}{\tau_m} \tag{3.17}$$

$$t_{\theta_i} = -\tau_m \ln\left(1 - \frac{V_\theta}{A_i}\right) \tag{3.18}$$

Equipés de l'équation 3.18, nous pouvons maintenant estimer la densité de probabilité de l'erreur de décodage idéale avec une simulation de Monte-Carlo. Nous tirons uniformément 500 000 points dans l'espace d'entrée [0,1]. Pour chaque point dans l'espace d'entrée $a \in [0, 1]$ nous calculons analytiquement 1) le vecteur de niveaux d'activations $\mathbf{A} \in [0, 1]^l$ et 2) le vecteur de temps d'impulsions $\mathbf{t}_\theta \in [0, T]^l$ résultant. Puis nous utilisons la méthode de décodage précédemment exposée pour décoder le motif impulsionnel $\mathbf{t}_\theta \in [0, T]^l$ en une valeur $\hat{a} \in [0, 1]$. L'erreur de décodage idéale $a - \hat{a}$ est calculée pour chaque point. Enfin nous estimons la densité de probabilité de l'erreur de décodage avec cet ensemble de points.

Erreur de décodage avec des temps d'impulsions approximés

La même procédure est appliquée en utilisant la méthode d'Euler pour résoudre, non pas analytiquement, mais par approximation numérique l'équation différentielle régissant l'évolution du potentiel membranaire d'un neurone LIF. La méthode d'Euler est souvent utilisée dans les simulations de SNN, ainsi que dans les processeurs neuromorphiques comme Loihi (Davies et al., 2018). Elle est computationnellement peu coûteuse en opérant itérativement par approximation linéaire, et en discrétisant le temps avec un pas de temps dt. Notons que le contraste avec la biologie n'est pas trop marqué, car bien que le temps soit continu dans les systèmes biologiques, la précision des temps d'impulsions dans les réseaux de neurones biologiques n'est pas infinie,

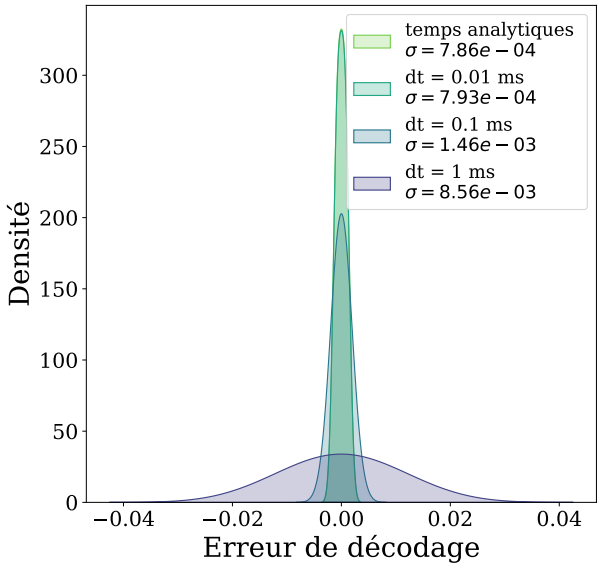

FIGURE 3.7 – Densité de probabilité de l'erreur de décodage estimée avec un noyau gaussien. On observe que la méthode de décodage est raisonnablement efficace, mais n'est pas parfaite car l'erreur n'est pas nulle pour des temps d'impulsions parfaits, i.e. infiniment précis (au sens de la précision du format de virgule flottante utilisé). Dans le cas d'une approximation numérique des temps d'impulsions avec la méthode d'Euler, on constate que la densité de probabilité se resserre autour du centre en diminuant le pas de temps dt. Pour $dt = 0.01$ ms, la densité de probabilité de l'erreur est quasiment identique à celle induite par des temps d'impulsions parfaits.

mais est bornée d'un ordre de 0.1 ms (CESSAC et al., 2010). Nous utiliserons systématiquement la méthode d'Euler dans ce manuscrit. Cependant la discrétisation de la dimension temporelle, ainsi que l'utilisation de cette méthode d'approximation numérique limitent la précision des temps d'impulsions. C'est pourquoi nous faisons varier la granularité du pas de temps dt pour étudier l'impact de la précision des temps d'impulsions sur la densité de probabilité de l'erreur de décodage.

Les densités de probabilités des erreurs de décodage avec la méthode analytique et la méthode d'Euler sont exposées dans la figure 3.7. Une erreur résiduelle demeure malgré l'utilisation de temps d'impulsions parfaits, ce qui montre que la méthode de décodage n'est pas parfaite. Une partie de cette erreur résiduelle peut être liée aux limites de précision inhérentes au format de codage des réels en virgule flottante utilisé et aux opérateurs élémentaires utilisant ce format.

Dans le cas où les temps d'impulsions sont numériquement approximés avec la méthode d'Euler, et donc limités en précision, on constate que le pas de temps dt a un impact sur l'erreur de décodage. Plus dt est petit, plus les temps d'impulsions sont précis et plus la densité de probabilité se rapproche de celle calculée à partir des temps d'impulsions parfaits (fig. 3.7).

3.2 Diversité et structure du code neuronal

Après avoir exposé et étudié les performances de la méthode de décodage, nous souhaitons à présent rendre compte des propriétés informationnelles et structurelles du code neuronal. Autre-

ment dit, nous souhaitons étudier les propriétés intrinsèques du code neuronal, indépendamment d'une quelconque méthode de décodage.

Cette section explore la diversité et la structure du code neuronal présenté, i.e. 1) la quantité d'information délivrée par le code pour un nombre de neurones et une granularité temporelle donnée et 2) l'espace occupé par l'activité neuronale dans l'espace des états, respectivement.

3.2.1 Quantifier la diversité du code avec l'entropie

Une étude analytique de la quantité d'information délivrée par les mécanismes d'encodage neuronaux présentés dans ce chapitre semble hors de portée. A la place, nous utiliserons l'entropie pour estimer la quantité d'information délivrée par les réponses neuronales induites par des entrées extérieures.

Intuitivement l'entropie mesure la variabilité d'éléments dans un ensemble, i.e. à quel point les éléments sont différents ou diversifiés dans cet ensemble. L'entropie est strictement positive. Elle est définie en bits pour un logarithme en base binaire ($\log_2$). Cette mesure nous donne le nombre de bits moyen minimal pour transmette des messages sans ambiguité. Autrement dit, l'entropie correspond à la longueur moyenne du code composé d'une séquence d'états binaires que nous devrions envoyer pour atteindre un optimum en termes de compression des messages. Plus l'entropie est grande, plus la quantité d'information transmise est grande.

Ainsi plus l'entropie du code neuronal est grande, plus le code est diversifié ou expressif. Cela implique qu'il faut une longue séquence d'états binaires pour représenter l'information délivrée par le code neuronal. Par contraste, si le nombre de bits est faible, cela signifie que le code neuronal est peu diversifié. Autrement dit, sa capacité représentationnelle est pauvre.

Le but est d'estimer l'entropie de la réponse neuronale, considérée comme un vecteur aléatoire discret de latences $\mathbf{\Delta_t}$, induit par une entrée a, elle même considérée comme une variable aléatoire discrète. Pour n évènements (ou points de données) l'entropie de la réponse $H(\mathbf{\Delta_t})$ se calcule avec l'équation suivante :

$$H(\mathbf{\Delta_t}) = -\sum_{i=1}^{n} P(\mathbf{\Delta_t} = \mathbf{\Delta_t}_i) \log_2 P(\mathbf{\Delta_t} = \mathbf{\Delta_t}_i) \tag{3.19}$$

L'entropie d'un vecteur ou d'une variable aléatoire discrète suivant une loi uniforme - autrement dit quand tous ses états sont équiprobables - est maximale. Dans ce cas, pour n états nous obtenons une entropie de $\log_2(n)$. Si les états ne sont pas équiprobables, alors l'entropie ne peut que décroître. L'entropie atteint son minimum à 0 dans le cas particulier où la variable aléatoire est constante, i.e. prend une unique valeur. Nous pouvons facilement vérifier que l'entropie est bien de $\log_2(n)$ pour une loi uniforme. Les états étant équiprobables nous avons donc $P(\mathbf{\Delta_t} = \mathbf{\Delta_t}_i) = 1/n$. Nous obtenons par substitution dans l'équation 3.19 :

$$\begin{aligned} H(\mathbf{\Delta_t}) &= -\sum_{i=1}^{n} \frac{1}{n} \log_2 \left(\frac{1}{n}\right) \\ &= n\frac{1}{n} \log_2(n) \\ &= \log_2(n) \end{aligned}$$

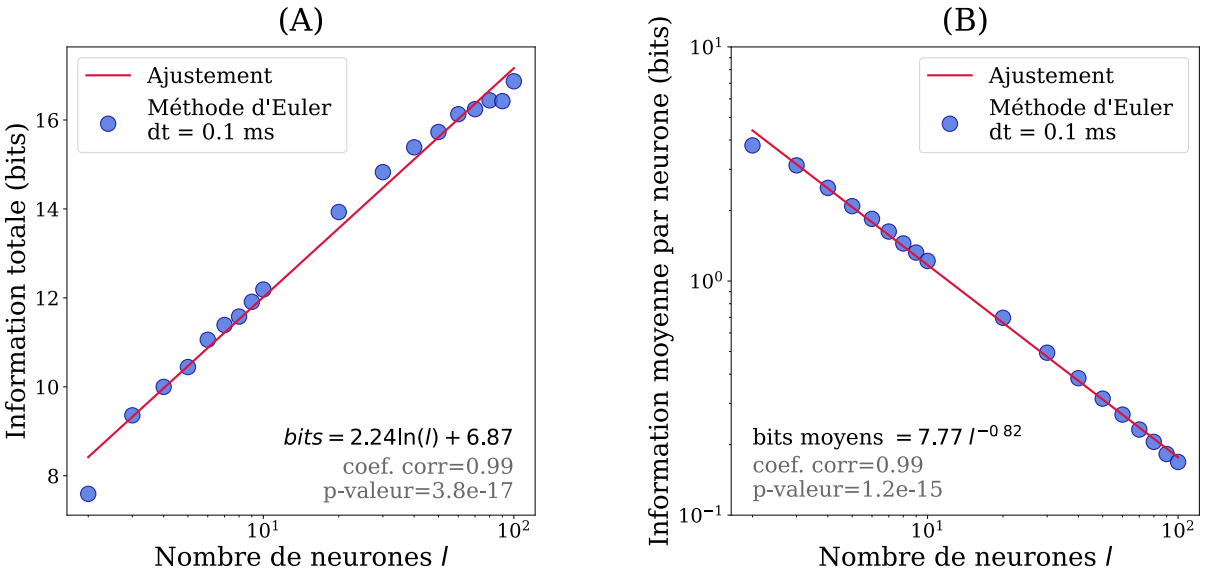

FIGURE 3.8 – (A) Information totale transmise par le code neuronal en fonction du nombre de neurones. Le nombre de bits transmis est une fonction linéaire-log du nombre de neurones. (B) Information moyenne transmise par neurone en fonction du nombre de neurones. Le nombre de bits moyen décroit suivant une loi de puissance dont la base est le nombre de neurones l.

L'objectif étant d'encoder une entrée qu'on suppose uniformément distribuée avec la plus grande précision, on souhaite par conséquent obtenir une entropie de la réponse neuronale la plus proche possible de l'entropie de l'entrée. Le cas idéal est obtenu lorsque l'entropie de l'entrée est égale à l'entropie de la sortie, i.e. de la réponse neuronale. Cela signifie que toutes les valeurs d'entrées sont codées sans ambiguité, i.e. qu'il y a bijection entre chaque point de donnée d'entrée et chaque point d'activité neuronale.

Nous considérons pour les expériences suivantes une entropie de l'entrée de 19 bits qui suit une loi uniforme discrète. Cela revient à considérer $n = 2^{19}$ valeurs uniformément espacées dans l'espace d'entrée [0,1], et donc une résolution de la variable d'entrée de $1/2^{19}$. Chacune de ces valeurs est ensuite fournie en entrée à la méthode d'encodage, et chaque vecteur de latences $\mathbf{\Delta_{t}}_i$ résultant en sortie est enregistré. Enfin l'entropie du code neuronal est estimée à partir de l'ensemble des n vecteurs de latences.

3.2.2 Relation entre information et nombre de neurones

Nous voulons étudier la relation entre l'information délivrée par le code neuronal en fonction du nombre de neurones utilisés pour encoder une variable d'entrée. L'entropie est ainsi calculée pour un nombre de neurones variable, en fixant le pas de temps $dt = 0.1$ ms.

Nous observons (fig. 3.8) que les points suivent une relation linéaire sur un graphique semi-logarithmique selon l'axe des absisses (nombre de neurones). Cela implique une relation logarithmique. Nous utilisons une regression linéaire-log pour déterminer les coefficients du modèle, i.e. le coefficient directeur et l'ordonnée à l'origine.

Un nombre de neurones pas trop important demeure intéressant en termes de bits transmis.

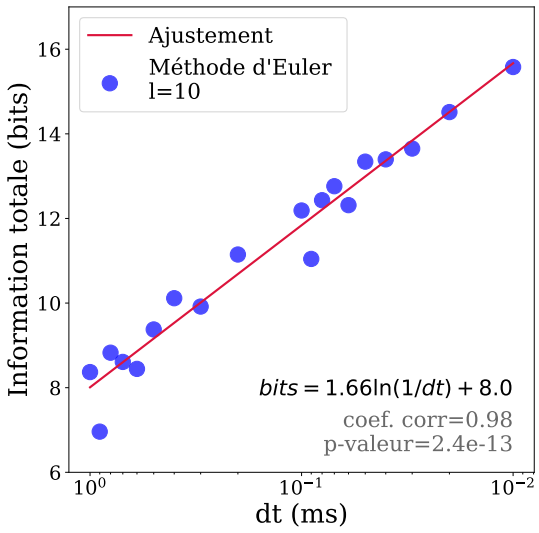

FIGURE 3.9 – (A) Information totale transmise par le code neuronal en fonction du pas de temps dt. Le nombre de bits est une fonction logarithmique du pas de temps.

Cela est mis en évidence dans le graphique (B) où on observe que jusqu'à environ $l = 10$ neurones, chaque neurone transmet en moyenne plus d'un bit d'information.

Les points suivent une fonction affine avec les deux axes gradués selon une échelle logarithmique. Cela induit une loi de puissance dont les coefficients ont été déterminés par une régression.

3.2.3 Relation entre information et temps

De la même façon, nous étudions ici la relation entre l'information délivrée par le code neuronal en fonction de la granularité du pas de temps dt utilisé. L'entropie est ainsi calculée pour un pas de temps dt variable, en fixant le nombre de neurones à $l = 10$. La fenêtre temporelle T étant fixée, diminuer dt revient à augmenter le nombre de pas de temps disponibles pour l'encodage. Par exemple passer de $dt = 1$ms à $dt = 0.1$ ms implique 10 fois plus de pas de temps disponibles par neurone pour encoder une entrée.

Nous observons (fig. 3.9) que les points suivent une relation linéaire sur un graphique semi-logarithmique selon l'axe des absisses (dt). Nous réalisons une régression linéaire-log pour trouver le meilleur ajustement des coefficients. Nous obtenons ainsi une fonction linéaire-log entre la quantité d'information transmise et le pas de temps.

Nous pouvons conclure qu'il existe deux moyens d'augmenter la quantité d'information transmise. Soit augmenter le nombre de neurones l, soit augmenter le nombre de pas de temps, revenant dans notre cas à diminuer le pas de temps dt pour une fenêtre temporelle T fixée. Le nombre de bits croît à un taux similaire (logarithmique), que ce soit pour le pas de temps dt ou pour le nombre de neurones l.

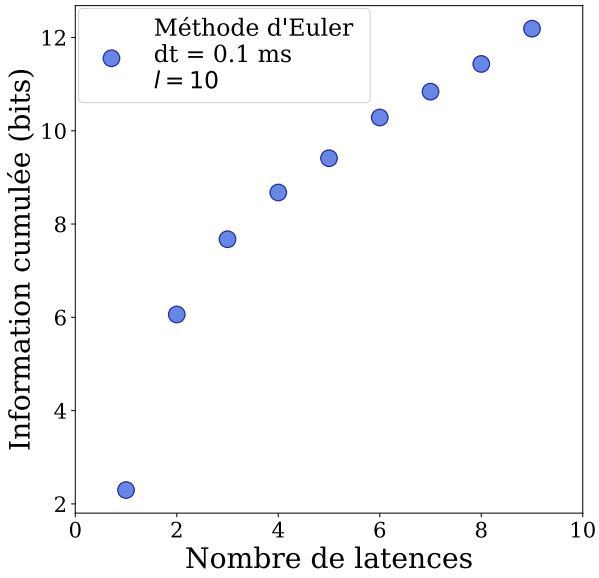

FIGURE 3.10 – Influence de l'ordre (ou rang) des impulsions sur la quantité d'information transmise. Les premiers neurones qui déchargent, et donc les premières latences, transmettent le plus grand nombre de bits.

3.2.4 Les premières impulsions transmettent le plus d'informations

A présent nous examinons si le rang, i.e. l'ordre des impulsions a un impact sur le nombre de bits transmis. Nous considérons une population de $l = 10$ neurones et un pas de temps $dt = 0.1$ ms. Nous calculons l'information cumulée suivant les n premières latences.

La conclusion (fig. 3.10) est que l'ordre des impulsions a un impact significatif sur le nombre de bits transmis. Les premiers neurones qui déchargent transmettent le plus d'information. Ainsi les 5 premiers neurones émettant une impulsion parmi une population de $l = 10$ neurones émettent près de 9 bits, soit environ 3/4 de l'information totale. Un avantage clé est que cela permet à un neurone en aval de rapidement réduire son incertitude sur l'entrée et de prendre une décision fiable, avant que tous les neurones en amont aient déchargé, augmentant ainsi la vitesse de traitement de l'information du SNN.

3.2.5 Un code neuronal structuré

Afin de pouvoir visualiser la structure du code neuronal, nous limitons notre étude à trois neurones, et donc à un espace d'états à trois dimensions. Chaque dimension correspond à la latence Δ_{t_i} d'un neurone i.

L'espace d'entrée dans [0,1] est uniformément échantillonné avec 100 000 points. Chaque entrée est ensuite fournie au circuit neuronal d'encodage, produisant en sortie un motif impulsionnel constitué de trois temps d'impulsions. Enfin trois latences sont extraites à partir de ces trois temps d'impulsions. Ainsi chaque point 1D dans l'espace d'entrée est projeté dans un espace de dimension supérieure, ici en un point 3D dans l'espace des latences.

Le résultat observé (voir Fig. 3.11) est que les états occupés par les neurones sont confinés à un

sous-espace de l'espace des états. Ce sous-espace est appelé variété neuronale (*neural manifold*). La variété neuronale prend la forme d'une boucle courbée résidant dans un espace des latences à trois dimensions (*embedding dimension*), mais dont la dimension intrinsèque est d'une seule dimension (Fig. 3.11).

Il est intéressant de noter que des topologies de variétés neuronales en forme de boucles (ou anneau) ont été trouvées dans des circuits neuronaux cruciaux. Par exemple l'activité collective des cellules de direction de la tête encode l'orientation de la tête. L'orientation de la tête correspond à une variable circulaire, chaque valeur angulaire est ainsi encodée par une activité neuronale. Bien que l'activité neuronale soit définie dans un espace contenant potentiellement plusieurs milliers de dimensions (plusieurs milliers de neurones), l'activité neuronale des cellules de direction de la tête du rat est très corrélée en étant confinée à un sous-espace, i.e. la variété neuronale. Chaudhuri et al., 2019 ont trouvé que la topologie de la variété neuronale correspond à un anneau. Cela indique que la dimension intrinsèque de l'activité collective des neurones est d'une seule dimension. Un autre exemple provient des cellules de lieux, qui représentent collectivement une carte de l'environnement. Chaque cellule de lieu décharge préférentiellement pour une position spécifique dans la carte. Gardner et al., 2022 ont montré que la topologie de la variété neuronale est torique, la dimension intrinsèque de la variété est ainsi de deux. Ainsi d'un point de vue neuronal, un animal en déplacement dans son environnement se déplace dans un tore. Un autre élément important est que cette topologie torique n'est pas induite par la nature des entrées sensorielles, mais est une propriété intrinsèque du circuit neuronal, ce qui est aussi le cas dans notre méthode d'encodage.

Par contraste avec la variété neuronale induite par notre méthode d'encodage, notons qu'un codage par latence qui maximiserait l'information transmise induirait une variété prenant la forme d'un nuage de points recouvrant uniformément l'intégralité de l'espace des états. Cet espace étant défini dans cet exemple sur trois dimensions (trois neurones), cela reviendrait à un cube. Autrement dit, la dimension intrinsèque de la variété neuronale serait ici égale à la dimension de l'espace d'état (*embedding dimension*). Ainsi ce code tirerait profit de la combinatoire du nombre d'états disponibles par dimension pour maximiser la diversité du code neuronal. Autrement dit, un tel code maximiserait le nombre de valeurs encodables (chaque point dans l'espace d'états encodant une valeur) et donc maximiserait l'information transmise. Cela implique que les mécanismes d'encodage sous-jacents devraient rendre l'activité de chaque neurone indépendante de l'activité des autres neurones de la population. Autrement dit, les variables discrètes que sont les latences $\Delta_{t_0}, \Delta_{t_1}, ..., \Delta_{t_l}$ seraient indépendantes, permettant une croissance exponentielle du nombre d'états disponibles. A titre d'illustration, un tel code utilisant 50 pas de temps, e.g. une fenêtre temporelle de 5 ms avec $dt = 0.1$ ms, et une population de $l = 3$ neurones (comme dans la Fig. 3.11), induirait un nombre d'états $n = 50^3$ et donc une information de $\log_2(n) \approx 17$ bits. Par contraste, il faudrait environ 100 neurones avec la méthode d'encodage utilisée pour atteindre 17 bits (fig. 3.10). Ainsi, sous cette configuration, il est remarquable qu'un encodage d'une entrée tombant dans $[0, 1]$ avec une résolution de $1/2^{17}$ serait atteint avec l'émission de seulement 3 impulsions neuronales, en tirant pleinement profit de la dimension temporelle pour représenter l'information.

Cependant, bien que la structure du code neuronal présenté dans ce chapitre induise une information transmise bien en deça de cet idéal théorique informationnel, elle apporte en contrepartie un avantage décisif. En effet, on observe que des points de données circulaires proches sont également proches dans la variété neuronale, en raison de la topologie en boucle de la variété neuronale. Cette propriété est particulièrement intéressante pour favoriser l'apprentissage

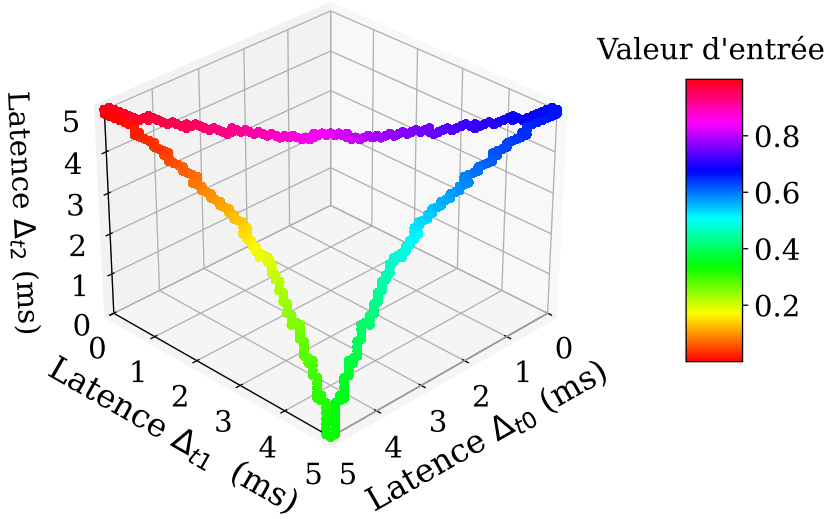

FIGURE 3.11 – Structure du code neuronal pour une population de 3 neurones. Chaque point de donnée 1D d'entrée est représenté par un point 3D d'activité neuronale, constitué de 3 latences $\Delta_{t0}, \Delta_{t1}, \Delta_{t2}$. On observe que les états occupés par l'activité neuronale sont confinés à un sous-espace, appelé variété neuronale (*neural manifold*) et formant une boucle. Notons que la boucle n'est pas induite par la structure de l'entrée mais par le circuit d'encodage neuronal. La variété neuronale préserve les relations de proximité entre des points pour une variable d'entrée circulaire. On peut le constater visuellement de par le gradient de couleur. Cette propriété est particulièrement pertinente pour favoriser l'apprentissage de caractéristiques (*features*) par des neurones situés en aval, recevant des motifs impulsionnels en entrée. En effet, nous posons une hypothèse centrale lorsque nous réalisons toute forme d'apprentissage : des données proches ont une sémantique proche (hypothèse de variété).

de caractéristiques (*features*) par des neurones situés en aval, recevant ces motifs impulsionnels en entrée. En effet, nous posons une hypothèse centrale lorsque nous réalisons toute forme d'apprentissage (e.g. *clustering*) : des points de données proches ont une sémantique proche. Cette hypothèse de variété est communément partagée en apprentissage automatique et en neurosciences computationnelles. De plus, les mécanismes de décodage devant être implémentés par des neurones en aval peuvent devenir très coûteux pour des codes trop riches ou trop complexes, où un changement mineur dans l'activité neuronale pourrait induire un changement sémantique majeur. Le mécanisme d'encodage présenté dans ce chapitre fournit donc un compromis diversité-structure raisonnable si le nombre de neurones considéré pour encoder une variable n'est pas trop élevé, de l'ordre d'une dizaine de neurones.

3.3 Synthèse

Nous avons exposé dans ce chapitre une méthode d'encodage neuronal transformant une variable d'entrée en motif impulsionnel spatio-temporel. Cette transformation est réalisée par la conjonction de champs récepteurs gaussiens placés dans un espace circulaire et d'une population de neurones LIF. L'information est ainsi contenue dans l'activité collective d'une population de neurones. Il s'agit d'un codage par population appartenant à la classe des codes temporels, où l'information est contenue dans les temps d'impulsions précis, il a donc été crucial d'assurer sa reproductibilité. Contrairement à un code par taux de décharge où une valeur est encodée dans le nombre d'impulsions émis dans une fenêtre temporelle, chaque neurone émet ici une unique impulsion. Cela permet de significativement réduire la consommation énergétique induite, ainsi que d'éviter la surcharge de la bande passante pour des processeurs neuromorphiques. En outre, ce code présente la remarquable propriété d'être unimodal.

A partir de ce code et en particulier de son caractère unimodal, nous avons proposé une méthode de décodage. Cette méthode reconstruit l'entrée grâce à une moyenne circulaire des directions préférentielles des neurones, pondérée par leurs latences. Les performances de la méthode de décodage en termes de reconstruction de l'entrée ont été testées.

Nous avons ensuite examiné les propriétés intrinsèques du code neuronal, i.e. l'information délivrée par le code ainsi que sa structure.

Il existe deux moyens d'augmenter l'information délivrée par ce code. Soit 1) utiliser une fenêtre temporelle d'encodage plus grande, soit 2) utiliser un nombre de neurones plus grand. Ainsi suivant les contraintes de l'application considérée, on peut soit 1) minimiser les ressources consommées (nombre de neurones et d'impulsions), soit 2) minimiser le temps pris pour transmettre l'information. Un compromis entre ces deux critères est permis par ce code neuronal.

L'entropie a été utilisée afin d'estimer l'information transmise - quantifiée en bits - par le code neuronal. Une étude paramétrique a été menée afin d'examiner l'impact du nombre de neurones ainsi que la granularité temporelle utilisée pour encoder une entrée. Des modèles linéaires-log ont été ajustés à l'aide de régressions pour mettre en lumière les fonctions liant ces paramètres au nombre de bits transmis. Ces modèles révèlent donc que le nombre de bits transmis croît logarithmiquement relativement au nombre de neurones l ou au pas de temps dt. La fenêtre temporelle T étant fixée, diminuer dt revient à augmenter le nombre de pas de temps disponibles pour l'encodage. Malgré cette croissance logarithmique du nombre de bits, un nombre raisonnable de neurones encodeurs demeure intéressant : un neurone transmet plus d'un bit d'information en moyenne pour une population de $l = 10$ neurones et un pas de temps de $dt = 0.1$ ms.

Nous avons également mis en évidence que l'ordre des impulsions est déterminant. Les premiers neurones émettant une impulsion transmettent la plus grande quantité d'information. Ainsi pour une population de 10 neurones, les 5 premiers neurones émettant une impulsion délivrent environ 3/4 de l'information totale. Un bénéfice clé est que cela permet une prise de décision à la fois rapide et fiable pour des neurones situés en aval, avant que tous les neurones en amont aient déchargé, augmentant ainsi la vitesse de traitement de l'information du SNN.

La structure du code révèle que l'activité neuronale est confinée à un sous-espace de l'espace d'états (*neural manifold*). Cette structure préserve les relations de proximité pour des données circulaires dans un espace spatio-temporel de haute dimension. Cette propriété est particulièrement pertinente pour toute forme d'apprentissage (e.g. *clustering*) qui pose une hypothèse essentielle : deux points proches ont une sémantique proche. La méthode d'encodage présentée dans ce chapitre offre donc un compromis raisonnable entre structure et diversité (information transmise) du code neuronal.

Ce chapitre a exposé et analysé la méthode d'encodage neuronal utilisée dans ce manuscrit. Il s'agit d'un encodage par latences relatives appartenant à la classe des codes temporels et instancié par l'activité collective d'une population de neurones. Les prochains chapitres proposent des méthodes d'apprentissage non-supervisées capables d'extraire des représentations d'un tel encodage neuronal. Les méthodes d'apprentissage de représentations développées ne sont pas spécifiques à un mécanisme d'encodage particulier, mais sont généralisables à n'importe quel encodage temporel émettant une impulsion par neurone. L'apprentissage de représentations se réfère au stockage d'informations dans des mémoires à long terme.

Chapitre 4

Apprentissage de représentations dans les poids synaptiques

Sommaire

4.1 Introduction

Les réseaux de neurones impulsionnels ou *Spiking Neural Networks* (SNN) ont fait l'objet d'une attention croissante ces dernières années. Ils ont été utilisés pour effectuer des tâches d'apprentissage supervisée (Kheradpisheh et Masquelier, 2020 ; C. Lee et al., 2020), par renforcement (Mozafari et al., 2018 ; Patel et al., 2019) et non supervisée (Diehl et Cook, 2015 ; Kheradpisheh, Ganjtabesh et al., 2018).

Les réseaux de neurones artificiels ou *Artificial Neural Networks* (ANN) émettent à chaque itération de façon synchrone des sorties à valeur réelle. Par contraste, les SNN émettent de manière éparse et asynchrone des sorties binaires : des impulsions. Ce schème de transmission événementielle de l'information allège les canaux de communication et réduit significativement l'empreinte énergétique. Tandis qu'une opération élémentaire dans un CPU induit un coût de l'ordre d'un nanojoule (10^{-9} J), un évènement synaptique induit un coût de l'ordre d'un picojoule (10^{-12} J) dans un processeur neuromorphique (Furber, 2016). Cela permet théoriquement d'obtenir une efficacité énergétique environ 1000 fois supérieure avec un processeur neuromorphique. Cette efficacité énergétique est également du fait de l'architecture massivement parallèle

des processeurs neuromorphiques, qui co-localisent la mémoire (les synapses) et l'unité de calcul (le neurone). Ce schème architectural basé sur les principes de parallélisme et de localité au niveau matériel peut ensuite être pleinement exploité au niveau algorithmique par l'utilisation de règles d'apprentissage locales telles que la STDP. Les règles d'apprentissage locales exploitent les informations accessibles au niveau du terminal présynaptique (ou entrée synaptique) et du terminal postsynaptique (ou sortie synaptique).

Utiliser des règles locales compatibles avec un substrat de calcul neuromorphique, afin d'extraire des représentations de flux de données événementielles et asynchrones, est donc très prometteur. King et al., 2013 ont proposé une règle spatialement locale apprenant des champs récepteurs ressemblant à des filtres de Gabor. Burbank, 2015 a développé une méthode reproduisant le comportement d'un auto-encodeur avec une règle STDP. Cette règle STDP apprend les poids d'encodage et de décodage avec une règle hebbienne et anti-hebbienne, respectivement. La conjonction de ces deux règles permet l'approximation de la fonction de coût d'un auto-encodeur. Plus récemment, Tavanaei, Masquelier et al., 2018 ont proposé une règle STDP, intégrant un module de Quantification Vectorielle et un module de régularisation, offrant les meilleures performances en termes d'erreur de reconstruction.

Cependant toutes ces méthodes opèrent sur la base d'un encodage des entrées en taux de décharge, où l'information est codée dans le nombre d'impulsions émises. L'encodage en taux de décharge recquiert un nombre d'impulsions élevé, peu plausible biologiquement et induisant un coût énergétique élevé, ainsi qu'une transmission lente de l'information (voir section 2.1). Par contraste, les codes temporels sont beaucoup plus économiques en transmettant l'information non pas dans le nombre d'impulsions, mais dans les temps d'impulsions. Considérons à titre illustratif l'encodage d'une entrée avec une résolution de $b = 16$ bits via un codage par population de $l = 10$ neurones et un pas de temps $dt = 1$ ms. Un codage par taux de décharge induirait jusqu'à 65 535 impulsions (32 767.5 impulsions en moyenne en considérant une distribution uniforme des entrées) pour encoder une entrée sur une fenêtre temporelle $T \approx 6.5$ s. Par contraste un encodage en latences relatives idéal induirait de manière constante 10 impulsions (une impulsion par neurone) pour encoder une entrée sur une fenêtre temporelle $T \approx 0.003$ s. Pour cet exemple un encodage temporel par latences relatives nous permet donc d'obtenir 1) une efficacité énergétique environ 3000 fois supérieure en termes de nombre d'impulsions et 2) une vitesse de transfert de l'information environ 2000 fois plus rapide. Plus généralement, plus la résolution b considérée est grande plus l'avantage en termes de coût énergétique et de vitesse de transfert est grand (exponentiellement plus avantageux) comparativement à un encodage en taux de décharge, comme mis en évidence dans la section 2.1. Les codes temporels sont donc particulièrement pertinents pour le calcul et l'apprentissage neuromorphiques.

Nous souhaitons utiliser des codes temporels, plus précisément un encodage en latences relatives (voir section 3.1.1), tout en atteignant les performances de reconstruction des modèles précédement exposés. De plus, nous voulons exploiter des règles d'apprentissage locales et événementielles, opérant avec les temps d'impulsions, afin de faciliter une future implémentation neuromorphique.

Ce chapitre introduit un SNN à deux couches, où une nouvelle règle STDP adapte les poids synaptiques entre ces deux couches à partir des temps d'impulsions et en opérant localement dans l'espace et le temps. Nous proposons la première méthode, à notre connaissance, capable d'apprendre des représentations à partir d'un encodage temporel des entrées à l'aide d'une règle de type STDP, dans la perspective d'obtenir une faible erreur de reconstruction. Les performances de l'architecture sont testées sur la base de données MNIST et sur une base d'images

naturelles. En outre, un problème courant des SNN est leur manque d'adaptabilité à des données de dimensions variées. Afin de répondre à ce problème, nous proposons également dans ce chapitre une paramétrisation générique du SNN lui permettant de traiter des données de dimension arbitraire.

4.2 W-TCRL : architecture et apprentissage

Cette section décrit les différents composants architecturaux et algorithmiques du SNN utilisé pour l'apprentissage de représentations. La nouvelle règle STDP développée pour apprendre des représentations dans les poids synaptiques est présentée. Nous nommons notre nouveau modèle *Weight - Temporally Coded Representation Learning* (W-TCRL).

4.2.1 Architecture

Le réseau de neurones impulsionnel (SNN) est composé de deux couches de neurones *Leaky Integrate and Fire* (LIF). La première couche u encode les données d'entrée en latences relatives (voir Chap. 3.1.1). La deuxième couche v extrait des représentations des motifs impulsionnels reçus en entrée. Ces deux couches sont entièrement connectées (voir Fig. 4.1). L'apprentissage de représentations est réalisé entre ces deux couches à l'aide d'une nouvelle règle STDP adaptant les poids synaptiques w_{ji}.

L'architecture a été simulée à l'aide du simulateur Brian 2 (Stimberg et al., 2019). Les sections suivantes décrivent plus précisément les différents composants architecturaux.

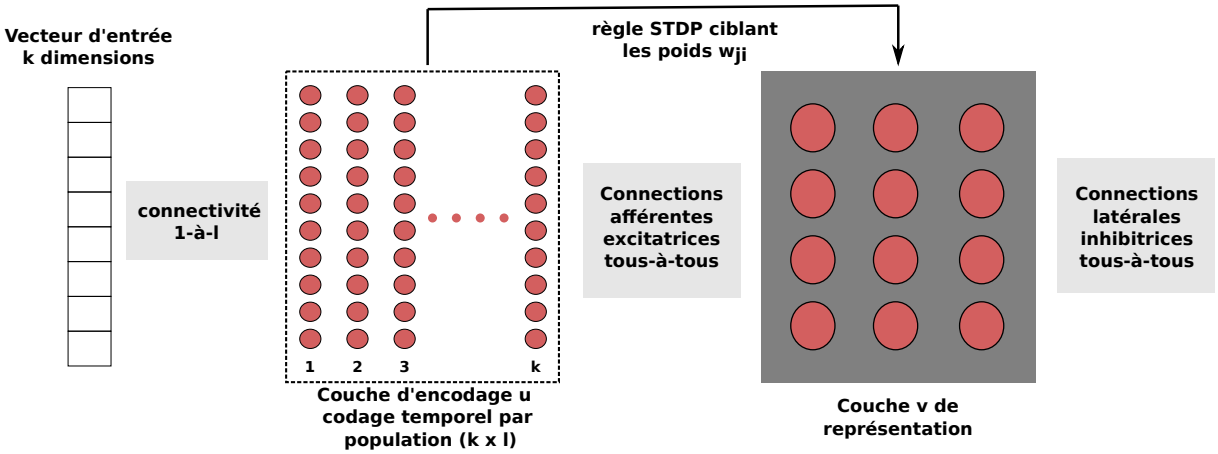

Figure 4.1 – Schéma bloc de l'architecture du SNN. Soit un vecteur d'entrée à k dimensions. Une dimension du vecteur d'entrée est encodée par l neurones. L'activité éparse des $k*l$ neurones est transmise à la couche de représentation. Entre ces deux couches s'opère l'apprentissage des poids synaptiques w_{ji} avec une nouvelle règle STDP. Des connexions latérales inhibitrices sont présentes dans la couche de représentation pour mettre les neurones en compétition, générant un apprentissage compétitif.

Modèle de neurone et de synapse

Le réseau ne contient que des neurones LIF. Ce modèle à une variable d'état (potentiel membranaire) capture le comportement basique d'un neurone biologique tout en maintenant un faible coût computationnel et une facilité d'analyse (voir section 2.3.1). Pour rappel (voir Eq. 3.4) la dynamique d'un neurone LIF est donnée par :

$$\tau_m \frac{dV(t)}{dt} = -V(t) + I(t)$$

on ajoute la sortie $s(t)$ et la réinitialisation de $V(t)$ lorsque le seuil de décharge V_θ est atteint, $V(t)$ est alors maintenu à 0 durant une période réfractaire T_{refrac} :

$$\begin{aligned} &\text{si } V(t) < V_\theta, \quad \text{alors } s(t) = 0 \\ &\text{si } V(t) \geq V_\theta, \quad \text{alors } \begin{cases} s(t) = 1 \\ V(u) = 0 \ \forall u \in]t, t + T_{\text{refrac}}] \end{cases} \end{aligned}$$

La transmission synaptique est modélisée par un modèle exponentiel de synapse (voir section 2.3.2). Si au moins un neurone présynaptique i décharge, la fonction indicatrice $s_i(t)$ prend la valeur 1 (0 sinon) et l'entrée du neurone postynaptique $I_j(t)$ change instantanément d'une magnitude égale au poids synaptique w_{ji} :

$$I_j(t) \leftarrow I_j(t) + \sum_{i=1}^{n} w_{ji} s_i(t)$$

sinon $I_j(t)$ décroît exponentiellement dans le temps avec une constante de temps τ_f :

$$\tau_f \frac{dI_j(t)}{dt} = -I_j(t)$$

4.2.2 Apprentissage de représentations dans les poids

Comme exposé dans la section 2.2.2, les règles *Nonweight-dependent Spike-Timing Dependent Plasticity* (NSTDP) permettent de générer des champs récepteurs stables, mais sont bistables : elles induisent une saturation des poids synaptiques aux valeurs minimales et maximales. L'intervalle de variation des paramètres soumis à apprentissage que sont les poids synaptiques n'est donc pas pleinement exploité avec les règles NSTDP. Cela limite fortement l'expressivité des poids synaptiques, i.e. leur capacité à stocker des représentations riches. Les règles *Weight-dependent Spike-Timing Dependent Plasticity* (WSTDP) corrigent ce problème en générant des distribution unimodales et continues de poids synaptiques, tirant ainsi profit de l'intervalle de variation des poids. Cependant elles ne permettent pas de générer des champs récepteurs stables.

Notre règle STDP est à la fois 1) stable et 2) exploite pleinement l'intervalle de variation des poids synaptiques pour stocker des représentations. L'interaction entre notre règle STDP et le code temporel unimodal génère une distribution stable et unimodale de poids synaptiques. La magnitude d'un poids synaptique est fonction de la distance temporelle entre une impulsion pré et postsynaptique. Plus un neurone présynaptique de la couche d'encodage u décharge rapidement,

mieux il représente l'entrée courante et plus la magnitude du poids synaptique induite par notre règle STDP est élevée. Cela renforce ainsi le pouvoir causal de ce neurone présynaptique sur l'émission d'une impulsion postsynaptique.

Notre nouvelle règle STDP est dérivée d'un critère de quantification vectorielle. Elle prend la forme d'une nouvelle règle STDP opérant 1) localement dans l'espace avec une seule paire de neurones pré et postsynaptiques i et j, et 2) localement dans le temps dans une fenêtre d'apprentissage temporelle.

Les règles STDP sont caractérisées par un changement de poids Δw_{ij} qui est fonction de l'intervalle de temps $\Delta_t = t_{\text{post}} - t_{\text{pre}}$ entre les impulsions post et présynaptiques, respectivement. La magnitude de la variation du poids Δw_{ij} dépend généralement d'un noyau de décroissance exponentielle fonction de l'intervalle de temps Δ_t : $e^{-\Delta_t/\tau}$. Deux constantes de temps différentes τ_+ et τ_- gèrent respectivement le cas causal $\Delta_t > 0$ et anti-causal $\Delta_t \leq 0$, et déterminent la largeur de la fenêtre temporelle d'apprentissage.

Pour que le calcul reste efficace et local, nous avons implémenté une version en ligne de la STDP (Sjöström et Wulfram Gerstner, 2010). Chaque synapse a accès à deux variables d'état locales $x_i(t)$ et $y_j(t)$. Ces variables servent de mémoires des activités pré et postsynaptiques récentes, respectivement. Lorsqu'une impulsion pré (post) synaptique est émise, $x_i(t)$ $\big(y_j(t)\big)$ saute à 1 puis décroît exponentiellement avec une constante de temps τ_x $\big(\tau_y\big)$ jusqu'à 0. De cette manière, nous reproduisons le comportement de $e^{-\Delta_t/\tau}$.

$$
\begin{aligned}
x_i(t) &\leftarrow 1, && \text{si } s_i(t) = 1 \\
\tau_x \frac{dx_i(t)}{dt} &= -x_i(t), && \text{sinon} \\
y_j(t) &\leftarrow 1, && \text{si } s_j(t) = 1 \\
\tau_y \frac{dy_j(t)}{dt} &= -y_j(t), && \text{sinon}
\end{aligned}
$$

W-TCRL : règle STDP pour l'apprentissage de représentations dans les poids

La variation du poids synaptique Δw_{ji} est calculée en fonction de la trace présynaptique $x_i(t) \in]0, 1]$ et postsynaptique $y_j(t) \in]0, 1]$. Le poids synaptique $w_{ji} \in [0, 1]$ est ensuite actualisé par ce changement $w_{ji} \leftarrow w_{ji} + \Delta w_{ji}$. Notre nouvelle règle STDP s'énonce ainsi (voir Fig 4.2) :

$$
\Delta w_{ji} = \begin{cases} +\alpha^+\big(1 - x_i(t) - w_{ji} + w_{\text{offset}}\big), & \text{si } s_j(t) = 1 \text{ et } x_i(t) > \epsilon \\ -\alpha^-\big(1 - y_j(t)\big), & \text{si } s_i(t) = 1 \text{ et } y_j(t) > \epsilon \end{cases} \tag{4.1}
$$

où α^+ et α^- sont les taux d'apprentissage. Cette règle STDP fonctionne en combinaison avec l'introduction de limites dures (*hard bounds*) bornant les poids synaptiques dans $0 \leq w_{ji} \leq 1$:

$$
w_{ji} \leftarrow \min\big(1, \max(0, w_{ji} + \Delta w_{ji})\big)
$$

Premier cas d'adaptation des poids synaptiques

Le premier cas d'adaptation est 1) déclenché de façon événementielle par une impulsion postsynaptique indiquée par $s_j(t = t_{\text{post}}) = 1$ et 2) opère localement dans le temps car la trace présynaptique $x_i(t)$ est échantillonnée au temps $t = t_{\text{post}}$ et doit être supérieure à ϵ, correspondant à une fenêtre temporelle depuis le temps de l'impulsion présynaptique t_{pre}. Ainsi, les impulsions présynaptiques arrivant en dehors de cette fenêtre temporelle ne déclenchent pas la règle. Cette règle traite le cas d'interactions causales, i.e. des impulsions présynaptiques émises avant ou au moment de l'impulsion postsynaptique, i.e. $\Delta_t \geq 0$.

Ce premier cas d'adaptation intègre un module de quantification vectorielle visant à apprendre la distribution des temps relatifs des impulsions dans la distribution des poids synaptiques, et donc à minimiser l'erreur de reconstruction car l'encodage neuronal est temporel. Nous pouvons observer, en analysant la solution à l'équilibre $\Delta_{w_{ji}} = 0$ que le poids w_{ji} converge vers une valeur dépendant de $x_i(t)$ et donc de l'intervalle de temps $\Delta_t = t_{\text{post}} - t_{\text{pre}}$ entre des impulsions pré et postsynaptiques :

$$\begin{aligned}
&\Delta w_{ji} = 0 \\
&\Leftrightarrow\ 0 = 1 - x_i(t) - w_{ji} + w_{\text{offset}} \\
\Leftrightarrow\ &w_{ji} = 1 - x_i(t) + w_{\text{offset}} \\
\Leftrightarrow\ &w_{ji} = 1 - \exp\left(-\frac{t - t_{\text{pre}}}{\tau_x}\right) + w_{\text{offset}}
\end{aligned}$$

Ce cas d'adaptation étant déclenché par une impulsion postsynaptique indiquée par $s_j(t = t_{\text{post}}) = 1$, nous obtenons finalement :

$$\Leftrightarrow\ w_{ji} = 1 - \exp\left(-\frac{\Delta_t}{\tau_x}\right) + w_{\text{offset}} \tag{4.2}$$

Considérons à titre illustratif un motif impulsionnel présenté de façon repétée en entrée à un neurone postsynaptique. w_{ji} convergera vers une valeur qui dépend de la trace présynaptique $x_i(t)$ et donc de Δ_t. Rappelons que la trace $x_i(t)$ saute à 1 lorsqu'une impulsion présynaptique est émise puis décroît exponentiellement dans le temps. Ainsi plus la valeur de $\Delta_t \to \infty$ est grande, plus la valeur de $x_i(t) \to 0$ est petite. Les premières impulsions présynaptiques émises transportent la plus grande quantité d'informations (voir section 3.2.4) sur l'entrée, qu'on peut interpréter comme une variable cachée. Les premières impulsions sont reflétées par un grand Δ_t et donc un petit $x_i(t) \in]0, 1]$. C'est pourquoi nous introduisons $1 - x_i(t)$ dans la règle, cela induit des valeurs de poids synaptiques w_{ji} élevées pour les premiers neurones présynaptiques qui déchargent, renforçant ainsi le pouvoir causal des premières impulsions présynaptiques sur le neurone postsynaptique. Ainsi, le premier neurone présynaptique qui décharge induira la valeur de poids w_{ji} la plus élevée, tandis que le dernier neurone qui décharge induira la valeur de poids w_{ji} la plus faible. Ce processus se produit pour chaque paire de neurones pré et postsynaptique remplissant les conditions du premier cas d'adaptation. L'ordre et la latence des impulsions sont ainsi représentés dans la distribution des poids synaptiques w_{ji} résultante. Il nous reste à aborder le rôle du dernier terme w_{offset}. Quand $\Delta_t \to 0$, cela est reflété par $x_i(t) \to 1$ et induit $1 - x_i(t) \to 0$ et donc $w_{ji} \to 0$. Pour éviter que $w_{ji} \to 0$ lorsqu'une impulsion présynaptique est émise à un temps $t_{\text{pre}} \approx t_{\text{post}}$, i.e. quand $\Delta_t \to 0$, nous ajoutons un décalage positif $w_{\text{offset}} > 0$ à la règle.

Deuxième cas d'adaptation des poids synaptiques

Le deuxième cas d'adaptation est 1) déclenché de façon événementielle par une impulsion présynaptique indiquée par $s_i(t = t_{\text{pre}}) = 1$ et 2) opère localement dans le temps car la trace postsynaptique $y_j(t)$ est échantillonnée au temps $t = t_{\text{pre}}$ et doit être supérieure à ϵ, correspondant à une fenêtre temporelle depuis le temps de l'impulsion post t_{post}. Ainsi, les impulsions postsynaptiques arrivant en dehors de cette fenêtre temporelle ne déclenchent pas la règle. Cette règle traite le cas d'interactions anti-causales, i.e. des impulsions présynaptiques émises après ou au moment de l'impulsion postsynaptique, i.e. $\Delta_t \leq 0$.

Nous pouvons reformuler ce cas d'adaptation de sorte à faire apparaître explicitement la dépendance à $\Delta_t = t_{\text{post}} - t_{\text{pre}}$:

$$\Delta w_{ji} \propto 1 - y_j(t)$$
$$\Delta w_{ji} \propto 1 - \exp\left(-\frac{t - t_{\text{post}}}{\tau_y}\right)$$

Ce cas d'adaptation étant déclenché par une impulsion présynaptique indiquée par $s_i(t = t_{\text{pre}}) = 1$, nous obtenons :

$$\Delta w_{ji} \propto 1 - \exp\left(-\frac{t_{\text{pre}} - t_{\text{post}}}{\tau_y}\right)$$
$$\Delta w_{ji} \propto 1 - \exp\left(+\frac{\Delta_t}{\tau_y}\right)$$

Δw_{ji} est proportionnel à $1 - \exp\left(\frac{\Delta_t}{\tau_y}\right)$, le tout est multiplié par $-\alpha^-$. Ainsi ce deuxième cas d'adaptation déprime les poids synaptiques. De façon similaire à la trace présynaptique $x_i(t)$, la trace postsynaptique $y_j(t)$ saute à 1 lorsqu'une impulsion postsynaptique est émise puis décroît exponentiellement dans le temps. Ainsi plus la valeur de $\Delta_t \to \infty$ est grande, plus la valeur de $y_j(t) \to 0$ est petite. Les dernières impulsions présynaptique émises délivrent le moins d'informations sur l'entrée (voir section 3.2.4). C'est pourquoi nous introduisons $\Delta w_{ji} \propto 1 - y_j(t)$ dans la règle : plus une impulsion présynaptique arrive tardivement relativement à une impulsion postsynaptique, plus la dépression du poids synaptique w_{ji} est élevée. Ainsi en considérant le comportement asymptotique de ce cas de dépression synaptique, le poids $w_{ij} \to 0$, car la borne inférieure du poids est fixée à 0 à l'aide d'une limite dure (*hard bound*).

Circuit Winner-Take-All

Les neurones postsynaptiques deviennent progressivement sélectifs à des motifs impulsionnels spécifiques via l'adaptation de leurs poids synaptiques afférents. En outre, les neurones postsynaptiques prennent une décision (émission d'une impulsion) avant que l'intégralité des impulsions présynaptiques ait été reçue - de façon analogue aux neurones biologiques - accélérant ainsi la vitesse de traitement du SNN. Rappelons que les premières impulsions transmettent la plus grande quantité d'informations sur l'entrée (voir Fig. 3.10) et donc réduisent le plus l'incertitude sur l'entrée.

Cette propriété est exploitée par un circuit temporel *Winner-Take-All* (WTA), dans lequel le premier neurone qui franchit son seuil de décharge $V_j(t) > V_\theta$ émet une impulsion en réponse à un

motif impulsionnel en entrée, et est déterminé comme étant le meilleur représentant de ce motif impulsionnel. Nous appelons *Spiking Best Matching Unit* (SBMU) l'index du neurone impulsionnel déchargeant en premier. Le premier neurone qui décharge (SBMU) inhibe fortement les autres neurones de la couche de représentation, les empêchant de décharger et ainsi d'apprendre le prototype courant. Ce schème d'apprentissage compétitif et auto-organisé est implémenté par des connexions inhibitrices latérales tous-à-tous. Le circuit induit une activité très éparse dans la couche de représentation et force les neurones à apprendre des vecteurs code non corrélés, deux ingrédients importants pour un apprentissage efficient (Bengio et al., 2013 ; Falez et al., 2019).

Cependant pour un vecteur d'entrée donné, un pur circuit WTA induit l'actualisation du vecteur code d'un unique neurone : la SBMU. Nous souhaitons accélérer l'apprentissage en recrutant plus d'un neurone (K-WTA) durant la présentation d'un vecteur d'entrée. Pour atteindre cet objectif, nous introduisons un mécanisme homéostatique augmentant la valeur des poids synaptiques latéraux de la couche de représentation. L'inhibition latérale est initialement faible avec des poids latéraux initialisés à $w_{ji} = -c_{min}$ puis augmente durant la phase d'apprentissage avec le mécanisme homéostatique jusqu'à atteindre la valeur cible $w_{ji} \approx -c_{max}$. Ainsi en utilisant des valeurs de $c_{\min}$ et $c_{\max}$ appropriées, le circuit passe dynamiquement d'un comportement K-WTA à WTA. Cela permet de recruter de moins en moins de neurones durant la phase d'apprentissage et d'ainsi progressivement décorréler et spécialiser les vecteurs codes appris par les neurones. Le mécanisme homéostatique est implémenté par :

$$\tau_w \frac{dw_{ji}}{dt} = -c_{\max} - w_{ji} \tag{4.3}$$

où τ_w est la constante de temps du mécanisme homéostatique. Nous fixons τ_w à 1/3 du temps de la phase d'apprentissage, de telle sorte à atteindre $w_{ji} \approx -c_{max}$ à la fin de la phase d'apprentissage. Ainsi pour une fenêtre temporelle d'encodage T et P vecteurs d'entrées, nous obtenons la relation $\tau_w = 1/3 * T * P$.

4.3 Protocole expérimental et résultats

4.3.1 Présentation des bases de données

Les bases de données d'apprentissage utilisées sont MNIST et une base de données d'images naturelles. Ce sont des bases de données fréquemment utilisées, nous permettant de comparer les performances de notre méthode avec des méthodes d'apprentissage connexes.

MNIST

La base de données MNIST (Deng, 2012) est constituée de chiffres manuscrits. Elle est composée de 60 000 images d'apprentissage et 10 000 images de tests de 28*28 pixels en niveaux de gris (voir Fig 4.4).

Images naturelles

La base de données d'images naturelles a été mise à disposition par Olshausen avec un article fondateur très influent (Olshausen et Field, 1996) visant à expliquer la formation des champs

récepteurs à partir de principes théoriques et normatifs. Ces images sont des scènes visuelles naturelles, i.e. un ensemble de photographies prises en conditions écologiques (voir Fig 4.5). La base de données est composée de 10 images de 512*512 pixels en niveaux de gris. Les images sont ensuite découpées en imagettes pour être fournies en entrée du réseau. Ainsi pour des patchs de 4*4 pixels, nous obtenons une base de 163 840 imagettes. Dans la suite de ce manuscrit les termes patchs et imagettes seront utilisés de façon interchangeable.

4.3.2 Métriques utilisées

Un premier objectif naturel est d'évaluer la qualité des représentations visuelles apprises par les neurones de la couche de la représentation v. Pour réaliser cela, nous utilisons l'erreur de reconstruction RMS entre un vecteur d'entrée et le vecteur code de la SBMU associée, afin de quantifier la similarité entre les patchs d'images originaux et reconstruits.

Le *Structural Similarity Index* (SSIM) est une autre mesure évaluant la similarité entre deux images en quantifiant le degré de structure entre deux images, plutôt que la différence pixel-à-pixel comme le fait le *Peak Signal to Noise Ratio* (PSNR) basé sur l'erreur RMS. Cependant le SSIM n'est pas utilisé dans les travaux connexes aux nôtres, ne permettant pas de comparer nos performances. En outre, des études (Dosselmann et Yang, 2011 ; Horé et Ziou, 2010) ont révélé des liens analytiques et statistiques entre le PSNR (basé sur le RMS) et le SSIM, indiquant que leurs différences proviennent essentiellement de leur sensibilité à la dégradation de l'image. Plus généralement, il n'existe pas encore de mesure de qualité visuelle satisfaisante rendant compte de la perception humaine. C'est pourquoi nous complétons notre étude quantitative utilisant l'erreur de reconstruction RMS avec une étude qualitative visuelle des représentations apprises et des images reconstruites à partir de ces dernières.

Des travaux connexes à notre méthode (Burbank, 2015 ; Tavanaei, Masquelier et al., 2018) utilisent une mesure de similarité basée sur un coefficient de corrélation pour évaluer la qualité des représentations visuelles. Cependant nous n'utilisons pas cette mesure de corrélation à cause de sa très forte sensibilité aux *outliers* (Jenkin et al., 1991 ; Yen et Johnston, 1996), la difficulté de son interprétation (Lee Rodgers et Nicewander, 1988) et à cause de raisons techniques : le coefficient n'est pas défini si un patch d'image a une intensité constante, ce qui est par exemple régulièrement le cas pour la base de données MNIST (voir Fig. 4.4).

En dehors de l'évaluation de la qualité des représentations apprises, nous évaluons d'autres caractéristiques du SNN avec la parcimonie de l'activité de la couche de représentation et l'incohérence euclidienne dans le processus d'élection auto-organisé de la SBMU. L'incohérence euclidienne est une nouvelle mesure que nous introduisons ci-dessous.

Erreur quadratique moyenne

Nous utilisons l'erreur RMS pour quantifier la différence entre un patch d'image d'entrée $\mathbf{a}_p$ et un patch reconstruit $\hat{\mathbf{a}}_p$ (4.4) :

$$RMS = \frac{1}{P}\sum_{p=1}^{P}\sqrt{\frac{1}{k}\sum_{i=1}^{k}(a_{i,p}-\hat{a}_{i,p})^2} \tag{4.4}$$

où k est le nombre de pixels d'un patch et P le nombre de patchs, et $a_{i,p}$ le ième pixel de a_p.

Rappelons que chaque $i \in \{1, 2, ..., k\}$ dimension d'un patch d'entrée est distribuée sur l neurones encodeurs dans la couche d'encodage. Chacun des $z \in \{1, 2, ..., l\}$ neurones encodeurs a une valeur préférentielle associée $\mu_z \in \{\mu_1, \mu_2, ..., \mu_l\}$. La couche d'encodage est connectée dans une relation tous-à-tous à la couche de représentation. Chaque neurone postsynaptique de la couche de représentation est ainsi connecté à $k * l$ neurones présynaptiques de la couche d'encodage.

Chaque dimension $i \in \{1, 2, ..., k\}$ du patch reconstruit $\hat{a}_{i,p}$ - correspondant dans notre cas à l'intensité d'un pixel - est décodée à partir de la distribution des l poids synaptiques $\{w_1^i, w_2^i, ..., w_l^i\}$ associée grâce à la méthode de décodage introduite dans le chapitre 3 :

Pour rappel (voir section 3.1.2) la première étape du processus de décodage consiste à transformer chaque valeur préférentielle $\mu_z \in \{\mu_1, \mu_2, ..., \mu_l\}$ des neurones présynaptiques en un angle θ_z (direction préférentielle) :

$$\theta_z = 2\pi\mu_z$$

Nous calculons ensuite les moyennes pondérées des coordonnées cartésiennes $\overline{x}_i$ et $\overline{y}_i$ à partir des angles θ_z (coordonnées angulaires) et des poids synaptiques w_z^i associés au pixel i de la SBMU (coordonnées radiales).

$$\begin{aligned}
\text{avec } w_z^i &= w_{\text{sbmu},(i-1)l+z} \\
\overline{x_i} &= \frac{1}{\Delta_i} \sum_{z=1}^{l} w_z^i \cos(\theta_z) \\
\overline{y_i} &= \frac{1}{\Delta_i} \sum_{z=1}^{l} w_z^i \sin(\theta_z) \\
\Delta_i &= \sum_{z=1}^{l} w_z^i
\end{aligned}$$

Nous calculons un nouvel angle $\overline{\theta_i}$ à partir des coordonnées cartésiennes $\overline{x}_i$ et $\overline{y}_i$. Cet angle $\overline{\theta_i}$ correspond à la moyenne circulaire pondérée :

$$\overline{\theta_i} = \text{atan2}\left(-\overline{y_i}, -\overline{x_i}\right) + \pi$$

Nous finissons par une transformation inverse, transformant l'angle $\overline{\theta_i} \in [0, 2\pi]$ en l'intensité normalisée d'un pixel du patch reconstruit $\hat{a}_{i,p} \in [0, 1]$:

$$\hat{a}_{i,p} = \frac{\overline{\theta_i}}{2\pi}$$

Sparsité

La sparsité n'est pas évaluée sur la couche d'encodage mais sur la couche de sortie, i.e. la couche de représentation. Pour rappel, la sparsité correspond au pourcentage de neurones actifs pour un vecteur d'entrée durant la fenêtre temporelle d'encodage T :

$$\text{Sparsité} = \frac{1}{m} N_{\text{imp}}$$

où m est le nombre de neurones, et N_{imp} est le nombre d'impulsions émises par les m neurones en réponse à un vecteur d'entrée sur la fenêtre temporelle T. Une faible valeur indique un grand nombre d'unités inactives et donc une grande sparsité.

Cohérence dans l'élection de la SBMU

Dans un modèle de QV traditionnel tel que la SOM, l'élection du meilleur représentant du vecteur d'entrée courant (la BMU) se fait à chaque itération en ayant un accès à la totalité de l'information du réseau. Par contraste, l'élection de la *Spiking Best Matching Unit* (SBMU) se fait de façon dynamique et auto-organisée dans le SNN via une compétition entre neurones.

Nous souhaitons quantifier le degré de cohérence du processus d'élection auto-organisé de la SBMU relativement à une élection basée sur la minimisation de la distance euclidienne. Pour atteindre ce but, nous introduisons une mesure évaluant si la SBMU fait partie des meilleurs représentants en termes de minimisation de la distance euclidienne entre le patch reconstruit à partir de la SBMU et l'entrée. Nous nommons cette mesure Incohérence et l'exposons ci-dessous.

Tout d'abord, nous introduisons la fonction d qui pour un vecteur d'entrée donné, calcule la distance euclidienne normalisée entre le vecteur d'entrée courant et le vecteur code associé à l'indice $j \in \{1, \ldots, m\}$ d'un neurone de la couche de représentation :

$$d : \{1, \ldots, m\} \to [0, 1]$$

Nous calculons ensuite toutes les distances $d(j)$.

Puis nous trions toutes les paires $\big(j, d(j)\big)$ pour $j = 1, \ldots, m$ par ordre croissant des distances $d(j)$. Nous obtenons donc une liste de paires triée par ordre croissant $\big(j_p, d(j_p)\big)$ avec $p = 1, \ldots, m$ et j_p étant l'indice d'une paire dans la liste de paires.

$$\text{avec } \forall p < m \quad d(j_{p+1}) \geq d(j_p) \quad \text{et} \quad \{j_p | p = 1, \cdots, m\} = \{1, \cdots, m\}$$

Nous vérifions que l'indice de la SBMU est bien compris dans les x % premiers indices de la liste de paires. Si c'est le cas alors nous incrémentons le nombre de SBMU cohérentes (sc), sinon nous incrémentons le nombre de SBMU paradoxales (sp). Le processus est répété pour chaque vecteur d'entrée afin de déterminer si la SBMU élue est cohérente ou paradoxale. Les x % premiers indices fixent le seuil de décision, ou de tolérance, pour passer d'une SBMU cohérente à une SBMU paradoxale.

$$\begin{aligned} &\text{si sbmu} \in \{j_p \mid p \in \{1, \ldots, \lceil mx \rceil\}\}, \quad \text{alors sc} \leftarrow \text{sc} + 1 \\ &\text{sinon sp} \leftarrow \text{sp} + 1 \end{aligned}$$

Enfin nous définissons une mesure d'incohérence, où sc et sp sont le nombre total de SBMU cohérentes et paradoxales, respectivement :

$$\text{Incohérence} = 1 - \frac{\text{sc}}{\text{sc} + \text{sp}}$$

Une grande incohérence peut potentiellement induire une grande erreur de reconstruction RMS.

4.3.3 Paramètres de W-TCRL

Paramétrisation du SNN en fonction de la dimension des données d'entrées

Un problème récurrent pour l'apprentissage dans les SNN provient de leur manque d'adaptabilité à des dimensionalités de données variées : un réglage fin des hyperparamètres est souvent nécessaire.

Pour traiter ce problème, nous présentons une paramétrisation générique du SNN afin de traiter des données de dimensions k arbitrairement grandes. Seuls trois paramètres dépendent de la dimension k des données dans la couche de représentation v : le seuil de décharge des neurones V_θ^v, le niveau d'inhibition latéral initial $c_{\min}$ et final $c_{\max}$. Par commodité nous posons l'hypothèse d'une relation linéaire entre la dimension k des données d'entrées et chacun de ces 3 paramètres.

Tout d'abord rappelons que chaque neurone encodeur émet une unique impulsion pour représenter une entrée continue. Le nombre d'impulsions émises dans la couche d'encodage est ainsi proportionnel à la dimension des données k, i.e. $k * l$ impulsions, où l est le nombre de neurones utilisés pour représenter une dimension de l'espace d'entrée. Comme les impulsions sont pondérées par des poids synaptiques tombant dans l'intervalle $[0, 1]$, un neurone peut décharger seulement si son seuil de décharge $V_\theta^v = c*k*l$ avec un coefficient $c \in [0, 1]$. De façon analogue, il est nécessaire d'adapter le niveau d'inhibition latérale à la dimension des données pour préserver un comportement WTA, impliquant que le niveau d'inhibition soit suffisamment élevé pour que les neurones postsynaptiques n'atteignent pas leur seuil de décharge V_θ^v.

Les coefficients des équations linéaires liant ces 3 paramètres à la dimension des données k et au nombre de neurones encodeurs l ont été déterminés par la méthode d'optimisation bayésienne *Tree-Structured Parzen Estimator* (TPE) (Bergstra et al., 2011). La tâche considérée était la minimisation de l'erreur de reconstruction RMS pour la base de données MNIST et la base d'images naturelles, avec des vecteurs d'entrées de dimensions différentes. Le but est d'obtenir des coefficients suffisamment génériques et robustes pour traiter des distributions de données d'entrée variées et de dimensions variées. En intégrant les valeurs des coefficients optimisés dans les équations linéaires, les valeurs des 3 hyperparamètres sont déterminées par :

$$V_\theta^v = 0.25kl \tag{4.5}$$

$$c_{\min} = 7V_\theta^v \tag{4.6}$$

$$c_{\max} = 96V_\theta^v \tag{4.7}$$

Paramètres utilisés

Les paramètres du SNN sont donnés dans le tableau 4.1. Les paramètres neuronaux de la couche d'encodage u sont communs à tout le manuscrit, ainsi que les $l = 10$ neurones utilisés dans la couche d'encodage pour représenter une dimension du vecteur d'entrée.

Dans nos expériences, un vecteur d'entrée consiste en un patch d'image de 4*4 pixels, donnant une dimension de $k = 16$. Nous utilisons la méthode précédemment exposée utilisant des coefficients génériques pour mettre à l'échelle les 3 hyperparamètres V_θ^v, $c_{\min}$ et $c_{\max}$ dépendant de la dimension des entrées. Ainsi la valeur du seuil de décharge V_θ^v des neurones de la couche de représentation v est déterminée à partir de l'équation 4.5, donnant $V_\theta^v = 0.25kl = 0.25*16*10 = 40.0$. Enfin il reste à fixer les trois paramètres du mécanisme homéostatique augmentant progressive-

ment la valeur des poids synaptique latéraux de la couche de représentation v : la constante de temps τ_w, le niveau d'inhibition latéral initial $c_{\min}$ et final $c_{\max}$. La constante de temps est égale à $1/3$ du temps de la phase d'apprentissage, avec $P = 60000$ vecteurs d'entrées et une fenêtre temporelle d'encodage $T = 25$ ms, nous obtenons $\tau_w = 1/3 * P * T = 500$ s. Le niveau d'inhibition latéral initial $c_{\min}$ dans la couche v est déterminé à partir de l'équation 4.6 donnant $c_{\min} = 7V_\theta^v = 7 * 40.0 = 280$. Le niveau d'inhibition latéral final $c_{\max}$ est déterminé à partir de l'équation 4.7 donnant $c_{\max} = 96V_\theta^v = 96 * 40.0 = 3840$.

Les autres hyperparamètres du SNN sont fixes, i.e. ne dépendent pas de la dimension des données d'entrées. Ces paramètres, e.g. les taux d'apprentissages α^+ et α^-, ont été déterminés par la méthode d'optimisation bayésienne TPE.

TABLE 4.1 – Paramètres de W-TCRL utilisés dans toutes les simulations

Paramètres neuronaux						
dt	τ_m^u	τ_m^v	V_θ^u	V_θ^v	T_{refrac}^u	T_{refrac}^v
0.1 ms	10.0 ms	1.4 ms	0.5	40.0	6 ms	6 ms
Paramètres synaptiques						
τ_f (u à v)	τ_f (v à v)	τ_x	τ_y			
2.8 ms	2.0 ms	1.7 ms	3.7 ms			
Paramètres pour la règle STDP						
α^+	α^-	w_{offset}	ϵ			
0.001	0.004	0.2	0.1			
Paramètres du mécanisme homéostatique						
τ_w	$c_{\min}$	$c_{\max}$				
500 s	280	3840				

Impact des taux d'apprentissage

Nous souhaitons vérifier que l'optimisation bayésienne TPE des taux d'apprentissage α^+ et α^- de la règle STDP n'est pas tombée dans un mauvais minimum local en termes d'erreur de reconstruction RMS.

Pour valider la solution obtenue par la méthode d'optimisation bayésienne TPE, nous effectuons une recherche de grille en basse résolution sur l'intervalle $[0.001, 0.2]$ et en haute résolution sur un intervalle $[0.001, 0.04]$ resséré autour de la solution obtenue par la méthode TPE.

Nous constatons que la solution obtenue avec $\alpha^+ = 0.001$ et $\alpha^- = 0.004$ est proche d'un optimum global en termes d'erreur de reconstruction RMS. Cela permet de considérer que la solution obtenue par la méthode d'optimisation bayésienne TPE est validée.

En outre, la recherche en grille révèle que notre règle STDP est très robuste à de fortes variations de la valeur du taux d'apprentissage α^+. A contrario une petite valeur de α^- est optimale. Ainsi pour une valeur de $\alpha^+ \in [0.001, 0.2]$ et une petite valeur $\alpha^- \approx 0.001$ l'erreur de reconstruction RMS atteint un plateau quasi-optimal.

Ce comportement peut s'expliquer par le fait que le premier cas d'adaptation utilisant α^+ peut

soit renforcer soit déprimer les poids synaptiques, contrairement aux règles STDP classiques induisant uniquement un renforcement des poids. Ce premier cas d'adaptation permet d'apprendre de manière stable la distribution des latences des premières impulsions dans la distribution des poids synaptiques.

Par contraste, le deuxième cas d'adaptation utilisant α^- a un rôle de réglage fin des poids synaptiques, il ne fait que déprimer les poids synaptiques. Cette règle déprime les poids des synapses transmettant les dernières impulsions, i.e. les impulsions présynaptiques les plus tardives. Le rôle de cette seconde règle est d'augmenter la sélectivité de la réponse neuronale relativement à des motifs impulsionnels reçus en entrée.

4.3.4 Résultats pour MNIST

Les expériences ont été menées pour plusieurs tailles de réseau avec $m \in \{16, 32, 64, 128, 256\}$ neurones dans la couche de représentation et donc m vecteurs codes soumis à apprentissage. Chaque expérience a été évaluée avec 15 simulations indépendantes. Pour une comparaison équitable avec l'état de l'art fixé par Tavanaei, Masquelier et al., 2018, nous n'avons pas effectué de validation croisée.

Les poids synaptiques entre la couche d'encodage u et de représentation v sont initialisés aléatoirement dans l'intervalle $[0.6, 0.8]$. Les entrées sont normalisées dans l'intervalle $[0.15, 0.85]$. Nous introduisons une marge due à la projection de données d'entrées linéaires sur un espace circulaire (champs récepteurs) où les valeurs extrémales deviennent équivalentes ($2\ \pi$ est équivalent à 0 radian). En outre, MNIST est essentiellement composé de valeurs extrémales (pixels noir et blanc). Nous avons sélectionné aléatoirement un sous-ensemble de 15 000 et 1000 chiffres manuscrits pour la phase d'apprentissage et de test, respectivement. Des patchs de 4*4 pixels, extraits de chiffres manuscrits de 28*28 pixels, sont fournis comme vecteurs d'entrée au SNN. 60 000 patchs d'images sont donnés en entrée au SNN durant la phase d'apprentissage. Pour la phase de test, nous avons fixé l'inhibition latérale dans la couche de représentation v à c_{max} et désactivé la plasticité.

Les trois mesures de performances présentées dans la section 4.3.2 ont été utilisées pour évaluer le SNN : l'erreur de reconstruction RMS entre un vecteur d'entrée et un vecteur code, la sparsité de l'activité de la couche de représentation, et l'incohérence dans le processus auto-organisé d'élection de la SBMU.

Phase d'apprentissage

Les différentes mesures de performances décroissent de façon consistante avec le nombre d'itérations d'apprentissage ainsi que pour un nombre de neurones $m \in \{16, 32, 64, 128, 256\}$ croissant dans la couche de représentation v (voir Fig. 4.7). L'erreur de reconstruction RMS décroît, indiquant que le SNN apprend à compresser la distribution des données d'entrées, jusqu'à atteindre un plateau relativement stable dès 8000 itérations autour d'une erreur RMS d'environ 0.08 pour $m \in \{32, 64, 128, 256\}$ neurones. La sparsité diminue également tendanciellement, indiquant que le pourcentage de neurones déchargeant dans la couche de représentation v décroît en réponse à un vecteur d'entrée. Cela est dû à la conjonction de la croissance du niveau d'inhibition latérale via le mécanisme homéostatique ainsi qu'à la spécialisation croissante des neurones. Enfin l'incohérence euclidienne dans l'élection de la SBMU décroît pour un seuil de décision Th = 5 % et Th = 10 %, indiquant une décroissance du nombre de SBMU paradoxales et ainsi une sélectivité

croissante des champs récepteurs des neurones à une région spécifique de l'espace d'entrée. Autrement dit, le neurone déchargeant en premier est en moyenne un bon représentant du vecteur d'entrée.

La figure 4.8 montre que les vecteurs codes de la couche de représentation deviennent pour la plupart progressivement sélectifs à des orientations visuelles au fil des itérations d'apprentissage.

Phase de test

Nous évaluons à présent les performances du SNN sur la base de données de test pour MNIST. Les performances obtenues augmentent toutes en augmentant la capacité du SNN, i.e. le nombre de neurones $m \in \{16, 32, 64, 128, 256\}$ et donc de vecteurs codes appris dans la couche de représentation (voir Fig. 4.9). Ainsi la sparsité moyenne pour $m = 256$ neurones atteint une valeur de 0.01 signifiant que seulement 1 % des neurones de la couche de représentation sont actifs en moyenne. L'élection auto-organisée de la SBMU est réalisée avec jusqu'à 99.8 % de SBMU cohérentes pour un seuil de décision à 5 % et $m = 256$ neurones. La meilleure erreur de reconstruction RMS atteint un plateau à 64 neurones (64, 128 et 256 neurones) avec une valeur de 0.07 (voir Tab. 4.2). Cela représente une amélioration relative de 58 % par rapport à l'état de l'art (voir Tab. 4.3) pour la base de données MNIST.

La figure 4.10 montre les vecteurs codes obtenus à la fin de la phase d'apprentissage pour un nombre de neurones $m \in \{16, 32, 64, 128, 256\}$ variable. Les vecteurs codes ont appris à devenir sélectifs à des orientations visuelles variées.

Enfin la reconstruction des images à partir des vecteurs codes élus de façon auto-organisée par le SNN produit des chiffres manuscrits reconstruits de bonne qualité, comparables aux images originales (voir Fig. 4.11).

4.3.5 Résultats pour les images naturelles

Comme pour MNIST, les expériences ont été menées pour plusieurs tailles de réseau avec $m \in \{16, 32, 64, 128, 256\}$ neurones dans la couche de représentation et donc m vecteurs codes soumis à apprentissage. Chaque expérience a été évaluée avec 15 simulations indépendantes. Pour une comparaison équitable avec l'état de l'art fixé par Tavanaei, Masquelier et al., 2018, nous n'avons pas effectué de validation croisée.

Les poids synaptiques entre la couche d'encodage u et de représentation v sont initialisés aléatoirement dans l'intervalle $[0.6, 0.8]$. Les entrées sont normalisées dans l'intervalle $[0.05, 0.95]$. Cette marge est due au fait que les données d'entrées linéaires sont projetées sur un espace circulaire (champs récepteurs) où les valeurs extrémales deviennent équivalentes ($2\ \pi$ est équivalent à 0 radian). Des patchs de 4*4 pixels, extraits d'images naturelles de 512*512 pixels, sont fournis comme vecteurs d'entrée au SNN. 60 000 patchs d'images sont donnés en entrée au SNN durant la phase d'apprentissage. Pour la phase de test, nous avons fixé l'inhibition latérale dans la couche de représentation v à c_{max} et désactivé la plasticité.

De nouveau, les mêmes trois mesures de performances présentées dans la section 4.3.2 ont été utilisées pour évaluer le SNN : l'erreur de reconstruction RMS entre un vecteur d'entrée et un vecteur code, la sparsité de l'activité de la couche de représentation, et l'incohérence euclidienne de la SBMU.

Phase d'apprentissage

Tout comme pour MNIST, les différentes mesures de performances décroissent de façon consistante avec le nombre d'itérations d'apprentissage ainsi que pour un nombre de neurones $m \in \{16, 32, 64, 128, 256\}$ croissant dans la couche de représentation (voir Fig. 4.12). L'erreur de reconstruction RMS décroît, indiquant que le SNN apprend à compresser la distribution des données d'entrées, jusqu'à atteindre un plateau stable avec une valeur d'environ 0.04 dès 2000 itérations. La sparsité diminue également tendanciellement, indiquant que le pourcentage de neurones déchargeant dans la couche de représentation v décroît en réponse à un vecteur d'entrée. Cela est dû à la conjonction de la croissance du niveau d'inhibition latérale via le mécanisme homéostatique ainsi qu'à la spécialisation croissante des champs récepteurs des neurones. Enfin l'incohérence de l'élection de la SBMU décroît pour un seuil de décision Th = 5 % et Th = 10 %, indiquant une décroissance du nombre de faux positifs et ainsi une sélectivité croissante des champs récepteurs des neurones à une région spécifique de l'espace d'entrée. Autrement dit, le neurone déchargeant en premier est en moyenne un bon représentant du vecteur d'entrée.

La figure 4.13 montre que les vecteurs codes de la couche de représentation deviennent progressivement sélectifs à des motifs visuels au fil des itérations d'apprentissage.

Phase de test

Nous évaluons à présent les performances du SNN sur la base de données de test pour les images naturelles. Les performances obtenues augmentent toutes en augmentant la capacité du SNN, i.e. le nombre de neurones $m \in \{16, 32, 64, 128, 256\}$ et donc de vecteurs codes appris dans la couche de représentation (voir Fig. 4.14). Ainsi la sparsité moyenne pour $m = 256$ neurones atteint une valeur de 0.01 indiquant que seulement 1 % des neurones de la couche de représentation sont actifs en moyenne. L'élection auto-organisée de la SBMU est réalisée avec jusqu'à 99.9 % de SBMU cohérentes pour un seuil de décision à 5 % et $m = 256$ neurones. L'erreur de reconstruction RMS atteint un plateau dès 32 neurones $m \in \{32, 64, 128, 256\}$ avec une valeur de 0.04 (voir Tab. 4.2). Cela représente une amélioration relative de 117 % par rapport à l'état de l'art (voir Tab. 4.3) pour la base de données d'images naturelles.

La figure 4.15 montre les vecteurs codes obtenus à la fin de la phase d'apprentissage pour un nombre de neurones $m \in \{16, 32, 64, 128, 256\}$ variable. Les vecteurs codes ont appris à devenir sélectifs à des orientations visuelles variées.

Enfin la reconstruction des images à partir des vecteurs codes des neurones élus de façon auto-organisée par le SNN produit des images naturelles reconstruites de très bonne qualité, comparables aux images originales (voir Fig. 4.16).

4.4 Synthèse

Les travaux connexes au nôtre utilisent un codage en taux de décharge (Burbank, 2015 ; King et al., 2013 ; Tavanaei, Masquelier et al., 2018) induisant un nombre d'impulsions élevé et donc une consommation énergétique élevée, ainsi qu'une transmission lente de l'information. Notre méthode se base sur un encodage en latences relatives répondant à ces problèmes. Dans nos expériences, seulement $l = 10$ neurones sont utilisés pour encoder une entrée, chaque neurone émettant une unique impulsion, cela génère ainsi collectivement 10 impulsions.

TABLE 4.2 – Résultats moyens obtenus sur la base de données MNIST et d'images naturelles pour un nombre de neurones $m \in \{16, 32, 64, 128, 256\}$.

Dataset	MNIST					Images Naturelles				
Neurones m	16	32	64	128	256	16	32	64	128	256
RMS	0.11	0.09	0.07	0.07	0.07	0.05	0.04	0.04	0.04	0.04
Sparsité	0.19	0.12	0.05	0.02	0.01	0.15	0.08	0.04	0.03	0.01
Incohérence 5 %	0.77	0.10	0.03	0.02	0.02	0.46	0.27	0.10	0.03	0.01
Incohérence 10 %	0.71	0.03	0.01	0.01	0.01	0.27	0.08	0.01	0.004	0.003

TABLE 4.3 – Meilleures erreurs de reconstruction RMS rapportées par TAVANAEI, MASQUELIER et al., 2018 et comparées à notre modèle W-TCRL

Dataset	Tavanaei	W-TCRL
MNIST	0.17	$\mathbf{0.07} \pm 0.001$
Images naturelles	0.24	$\mathbf{0.04} \pm 0.001$

Il manquait un mécanisme efficace capable d'extraire des représentations d'un tel codage temporel. Ce manque est comblé par notre nouvelle règle STDP capable d'apprendre de façon stable la distribution des latences relatives dans la distribution des poids synaptiques. Les poids synaptiques tombent dans l'intervalle $[0, 1]$. La règle STDP opère de façon locale dans le temps et l'espace, la rendant particulièrement compatible avec une implémentation sur un processeur neuromorphique.

Les premières impulsions présynaptiques délivrent la plus grande quantité d'information sur les variables cachées que sont les entrées. Elles déclenchent le premier cas d'adaptation des poids synaptiques, induisant asymptotiquement des poids synaptiques $w_{ji} > 0$. Plus un neurone présynaptique décharge tôt, plus le poids w_{ji} induit convergera vers une valeur élevée, impliquant un pouvoir causal élevé sur l'émission d'une impulsion postsynaptique. Par contraste, les dernières impulsions présynaptiques délivrent le moins d'information sur l'entrée, i.e. réduisent marginalement l'incertitude sur l'entrée. Celles-ci déclenchent le deuxième cas d'adaptation des poids synaptiques, induisant une dépression des poids synaptiques causant asymptotiquement la désactivation des synapses $w_{ji} \to 0$. Grâce à la conjonction de ces deux cas d'adaptations des poids synaptiques, un neurone postsynaptique apprend un vecteur code (ou filtre) dans ses poids synaptiques afférents. Cela permet au neurone postsynaptique de rapidement prendre une décision relativement à un motif impulsionnel en entrée. En d'autres termes, le neurone postsynaptique décharge avant d'avoir intégré l'ensemble des impulsions présynaptiques dans son potentiel membranaire, de façon analogue aux neurones biologiques, accélérant ainsi la vitesse de traitement du SNN.

Le SNN instancie un apprentissage compétitif entre les neurones de la couche de représentation à l'aide de connexions latérales inhibitrices. Nous avons ajouté un mécanisme permettant initialement de recruter plusieurs neurones durant la phase d'apprentissage pour accélérer la convergence du SNN, puis de progressivement recruter un nombre décroissant de neurones pour spécialiser et décorréler les vecteurs codes appris. Ce comportement est obtenu avec un mécanisme homéostatique augmentant la magnitude des poids latéraux pour faire dynamiquement

passer le circuit d'un comportement K-WTA à WTA, augmentant ainsi la sparsité de l'activité dans la couche de représentation.

Alors que l'apprentissage de représentations dans les ANN est un domaine mature, de bonnes performances restent difficiles à obtenir avec les SNN. Les performances obtenues en termes d'erreur de reconstruction RMS dépassent l'état de l'art fixé par le travail de Tavanaei, Masquelier et al., 2018 basé sur un codage par taux de décharge. Notre SNN basé sur un code temporel et une nouvelle règle STDP offre une amélioration relative de 58 % pour MNIST et de 117 % pour les images naturelles. Les images reconstruites par le SNN sont de très bonne qualité, visuellement comparables aux images originales.

Il faut cependant nuancer ces améliorations relatives en termes d'erreur de reconstruction RMS. En effet, les tailles des imagettes utilisées dans nos expériences sont inférieures à celles utilisées par Tavanaei, Masquelier et al., 2018. La taille des imagettes pour MNIST est similaire : 4*4 contre 5*5 pixels. Par contre pour la base de données d'images naturelles, elle est de 4*4 contre 16*16 pixels. Même si l'erreur RMS prend en compte la dimensionalité (division par $\sqrt{\text{dim}}$) de manière à ne pas pénaliser les hautes dimensions, il est difficile de comparer des résultats basés sur des tailles d'imagettes différentes. Les raisons du choix de taille effectué dans Tavanaei, Masquelier et al., 2018 ne sont pas explicitées. Dans les expérimentations décrites dans ce chapitre, nous avons choisi la taille 4*4 de manière à obtenir une reconstruction des images plus satisfaisante visuellement que pour des tailles plus grandes, mais cela influe également sur l'erreur RMS. Néanmoins, les performances de W-TCRL ne se dégradent que très progressivement pour des tailles d'imagettes plus grande, comme rapporté dans Fois et Girau, 2020. Ainsi les améliorations relatives obtenues pour la base d'images naturelles avec des tailles d'imagettes plus grandes restent supérieures à 50 %.

L'activité de la couche de représentation après la phase d'apprentissage est très éparse, e.g. pour $m = 256$ neurones, la sparsité est de 0.01, indiquant que seul 1% de neurones de la couche de représentation déchargent en moyenne. En outre, notons que les neurones de la couche de représentation déchargent une unique fois, contrairement aux travaux basés sur des encodages en taux de décharges. Ainsi, dans le travail de Tavanaei, Masquelier et al., 2018 la sparsité est moyennée sur une fenêtre temporelle $T = 40$ pas de temps pour un nombre de neurones m donné. La meilleure sparsité reportée de 9 % implique alors qu'en moyenne 9 % des neurones déchargent à chaque pas de temps de la fenêtre d'encodage T. Par contraste, une sparsité de 1 % de neurones dans notre SNN basé sur un code temporel implique que 1 % des neurones des neurones déchargent une unique fois durant la fenêtre d'encodage T. Par conséquent, en se basant sur les meilleurs résultats en termes de sparsité de Tavanaei, Masquelier et al., 2018, un neurone décharge en moyenne $0.09 * 40 = 3.6$ impulsions durant la fenêtre temporelle T. Par contraste, en se basant sur nos meilleurs résultats en termes de sparsité, un neurone émet en moyenne 0.01 impulsions durant la fenêtre temporelle T. Dans notre méthode un neurone de la couche de représentation émet donc en moyenne jusqu'à 360 fois moins d'impulsions, ce qui est très avantageux en termes de consommation énergétique induite et de besoin de communication, tout en maintenant une erreur de reconstruction RMS compétitive. C'est une confirmation empirique de l'intérêt de l'utilisation des codes temporels relativement aux codes par taux de décharge.

Enfin l'élection auto-organisée de la SBMU dans notre SNN produit jusqu'à 99.97 % de SBMU cohérentes (pour les images naturelles avec $m = 256$ neurones et un seuil de décision Th à 5 %), relativement à un critère de minimisation de distance euclidienne. Cela montre que les neurones apprennent à devenir très sélectifs à des régions de l'espace d'entrée, i.e. qu'ils

clusterisent efficacement l'espace d'entrée.

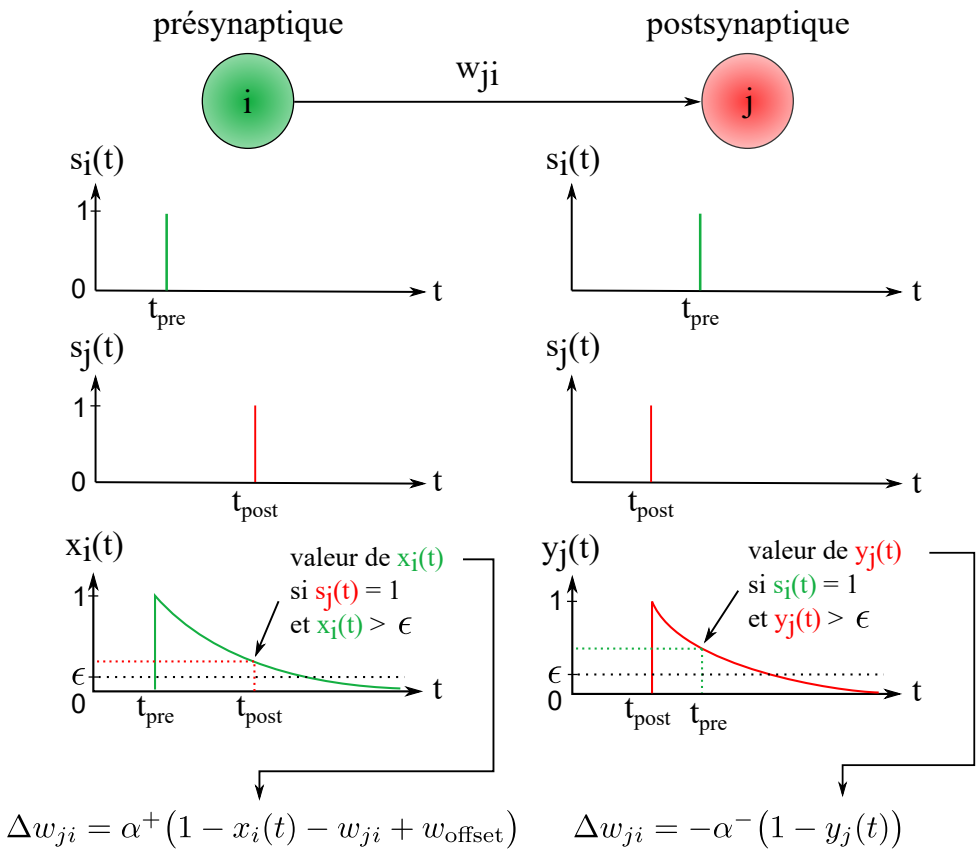

FIGURE 4.2 – Illustration des deux cas d'adaptation du poids synaptique w_{ji} de notre nouvelle STDP, spatialement et temporellement locale. La colonne de gauche illustre le premier cas d'adaptation déclenché pour le cas causal, i.e. le cas d'une impulsion présynaptique intervenant avant l'impulsion postsynaptique. La colonne de droite illustre le deuxième cas d'adaptation déclenché pour le cas anti-causal, i.e. le cas d'une impulsion postsynaptique qui intervient avant l'impulsion présynaptique. Dans les deux cas d'adaptation, la magnitude de Δw_{ji} dépend de la valeur des traces présynaptique et postsynaptique $x_i(t)$ et $y_j(t)$, respectivement. Les traces peuvent être interprétées comme des noyaux de décroissance exponentielle appliqués sur Δ_t, avec $\Delta_t = t_{\text{post}} - t_{\text{pre}}$.

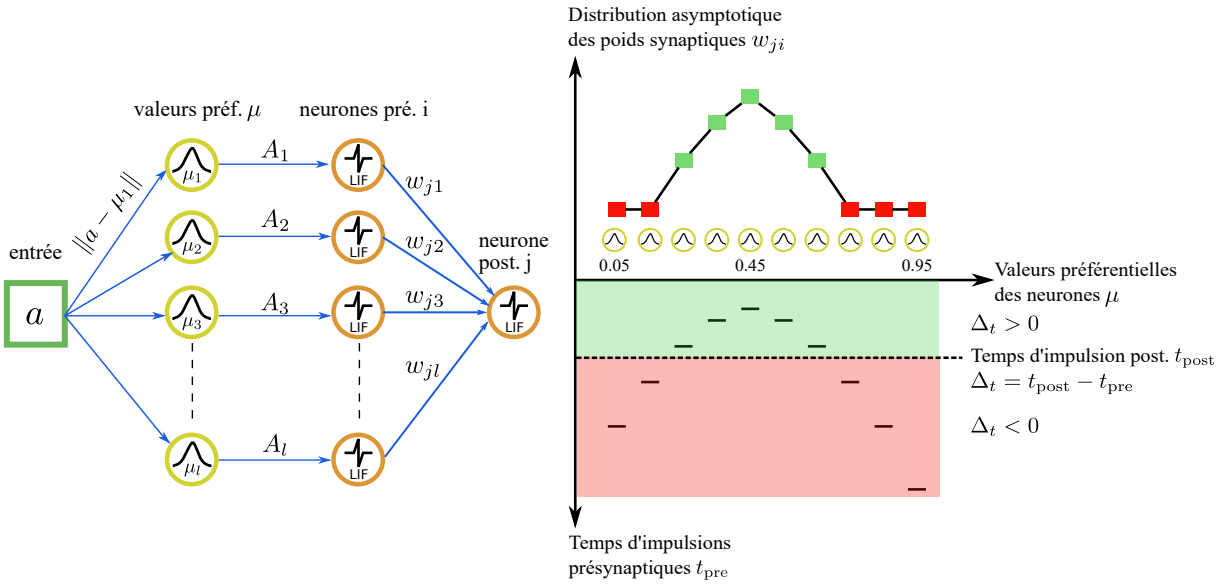

FIGURE 4.3 – Illustration de la distribution asymptotique de poids apprise avec notre règle STDP pour une valeur d'entrée $a = 0.45$. La valeur est encodée dans un motif impulsionnel avec une population de $l = 10$ neurones dans la couche d'encodage u. Le neurone dont la valeur préférentielle μ_i est la plus proche de l'entrée émet une impulsion en premier - ici le neurone ayant pour valeur préférentielle $\mu_5 = 0.45$ - suivi par les autres neurones de la population. Ce motif impulsionnel est transmis via les connexions synaptiques à un neurone postsynaptique j. Le neurone postsynaptique j de la couche de représentation intègre les entrées reçues et émet une impulsion à un temps t_{post} (ligne en pointillé), ici entre la 5e et la 6e impulsion. Le signe de la différence de temps Δ_t entre une impulsion postsynaptique et présynaptique délimite deux régimes d'apprentissage pour la règle STDP. Si $\Delta_t \geq 0$, alors le premier cas d'adaptation est déclenché. Le poids appris est un point fixe attractif. Si $\Delta_t \leq 0$, alors le deuxième cas d'adaptation est déclenché. Le poids est déprimé d'une magnitude qui croît avec Δ_t, pour tendre asymptotiquement vers 0.

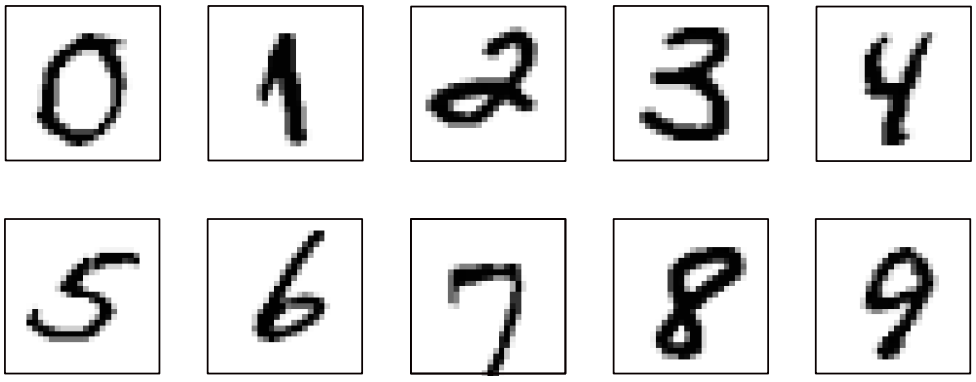

FIGURE 4.4 – Sélection de chiffres manuscrits de la base de données MNIST.

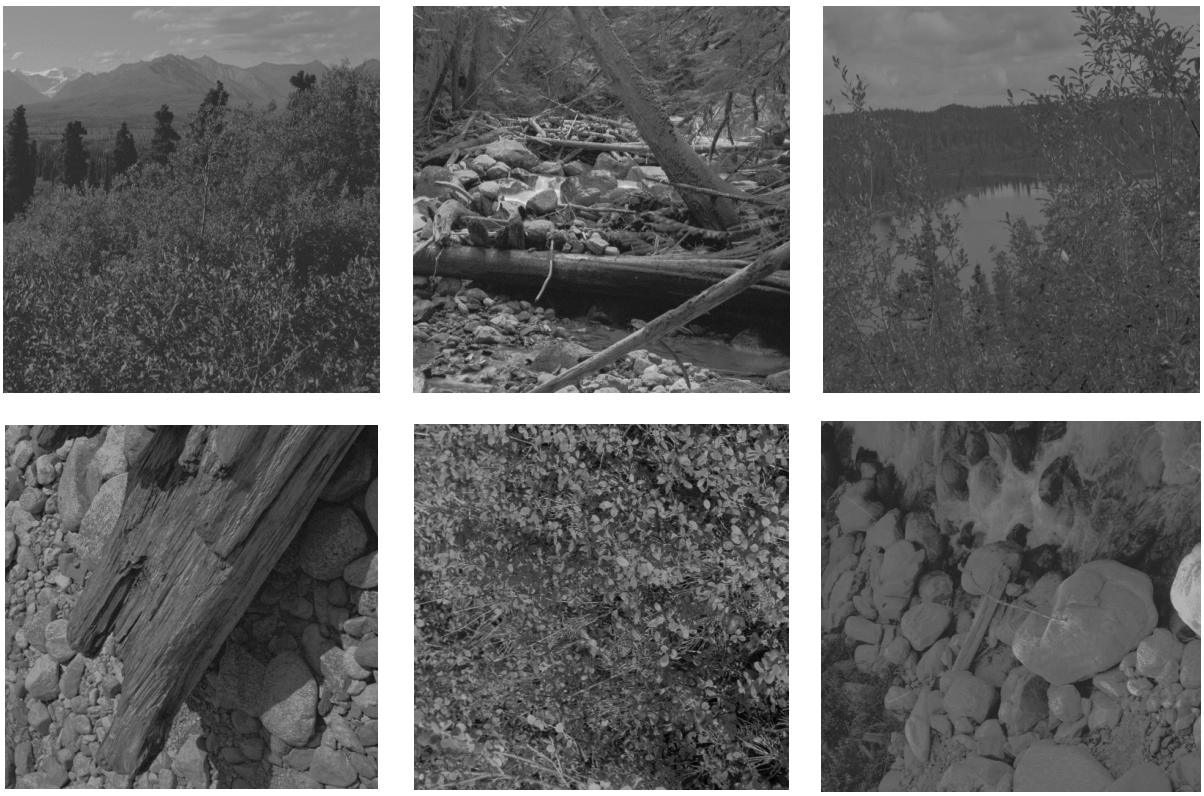

FIGURE 4.5 – Sélection d'images de la base d'images naturelles.

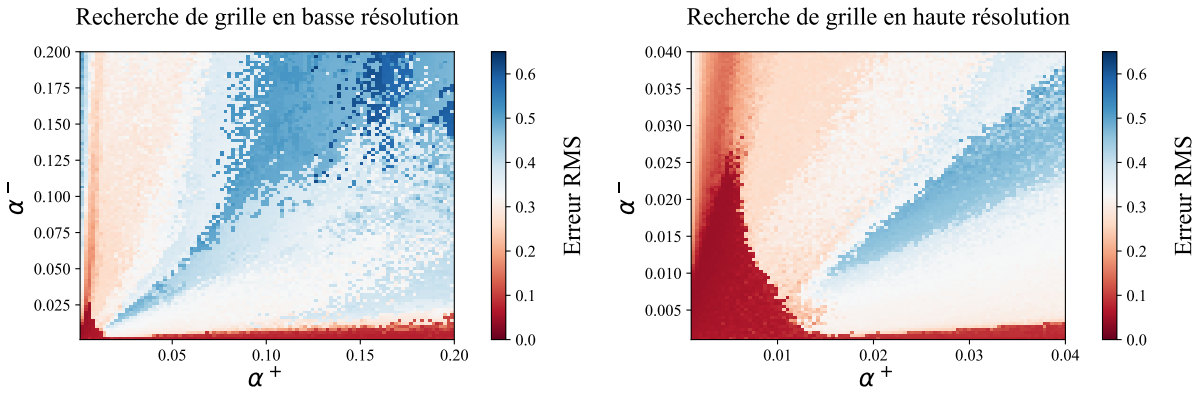

FIGURE 4.6 – Recherche de grille pour les valeurs des taux d'apprentissage α^+ et α^- sur un grand (gauche) et un petit (droite) intervalle de variation. L'erreur de reconstruction RMS est évaluée sur des images naturelles.

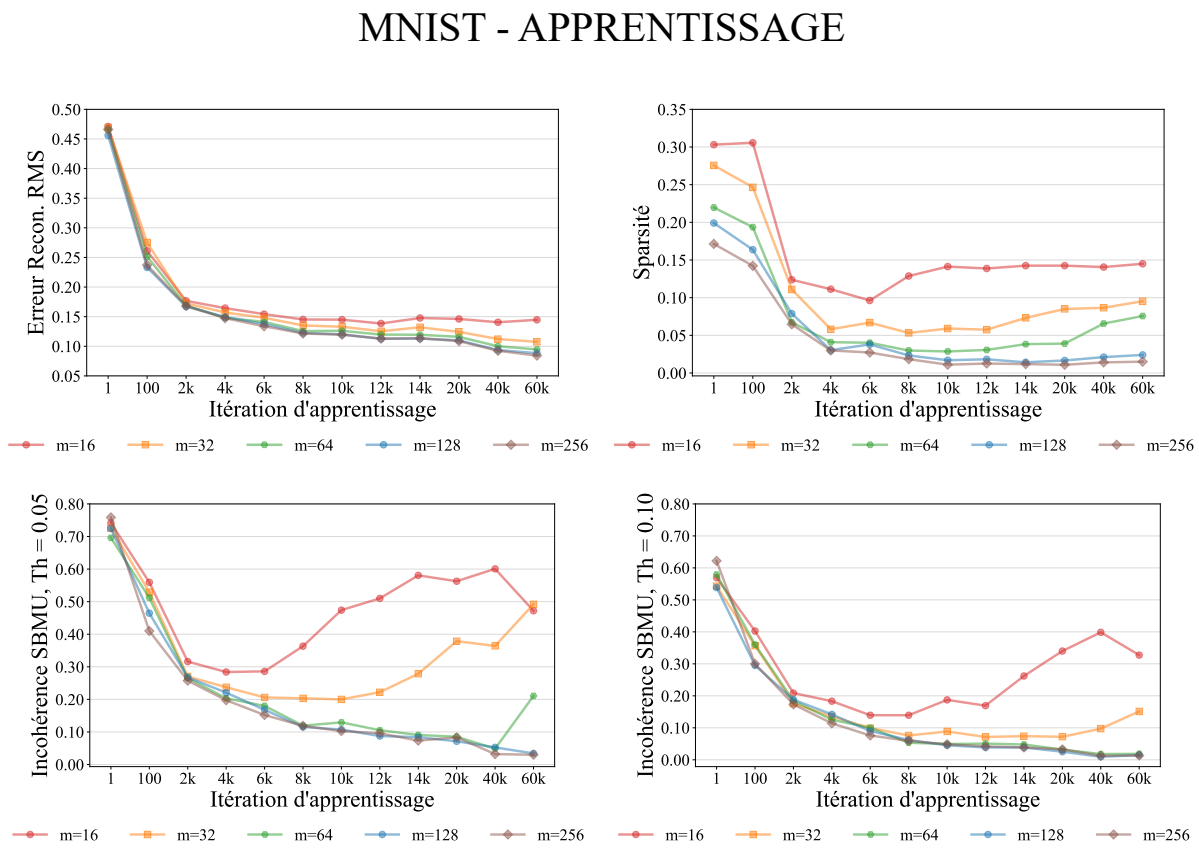

FIGURE 4.7 – Performances pour MNIST durant la phase l'apprentissage. Performances du SNN après 1, 100,..., 60k itérations pour $m \in \{16, 32, 64, 128, 256\}$ neurones, en termes d'erreur de reconstruction RMS, de sparsité, et d'incohérence dans l'élection auto-organisée de la SBMU pour un seuil de décision Th=5 % et Th=10 %.

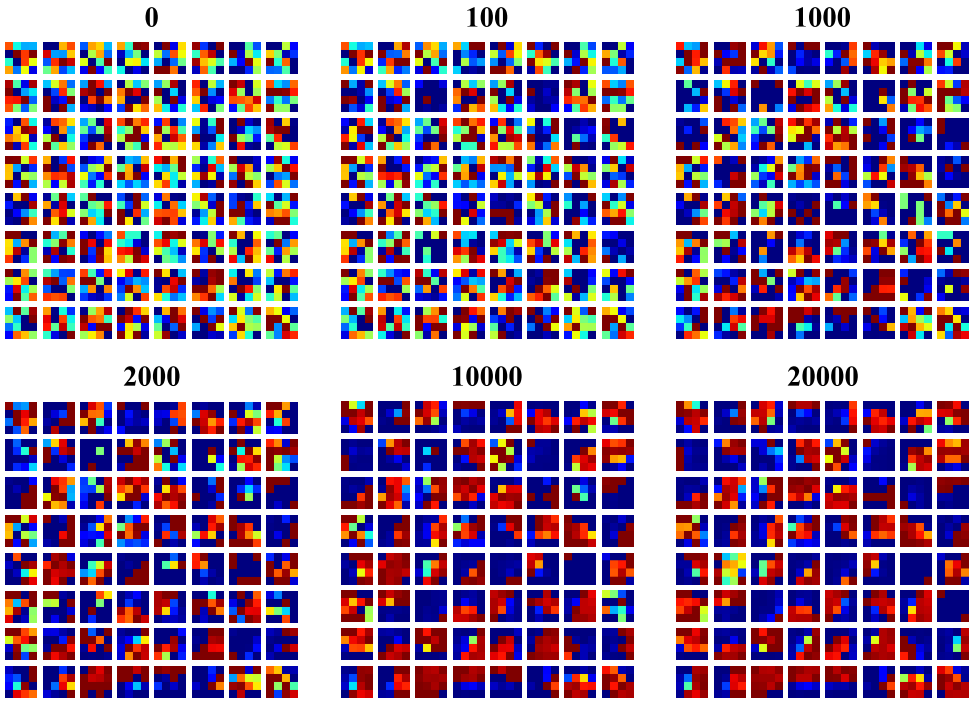

FIGURE 4.8 – Phase d'apprentissage sur MNIST : exemple d'évolution des vecteurs codes après 0, 100 ,..., 20k itérations d'apprentissage pour $m = 64$ neurones. Le gradient rouge-bleu indique les valeurs maximales-minimales de chaque pixel reconstruit.

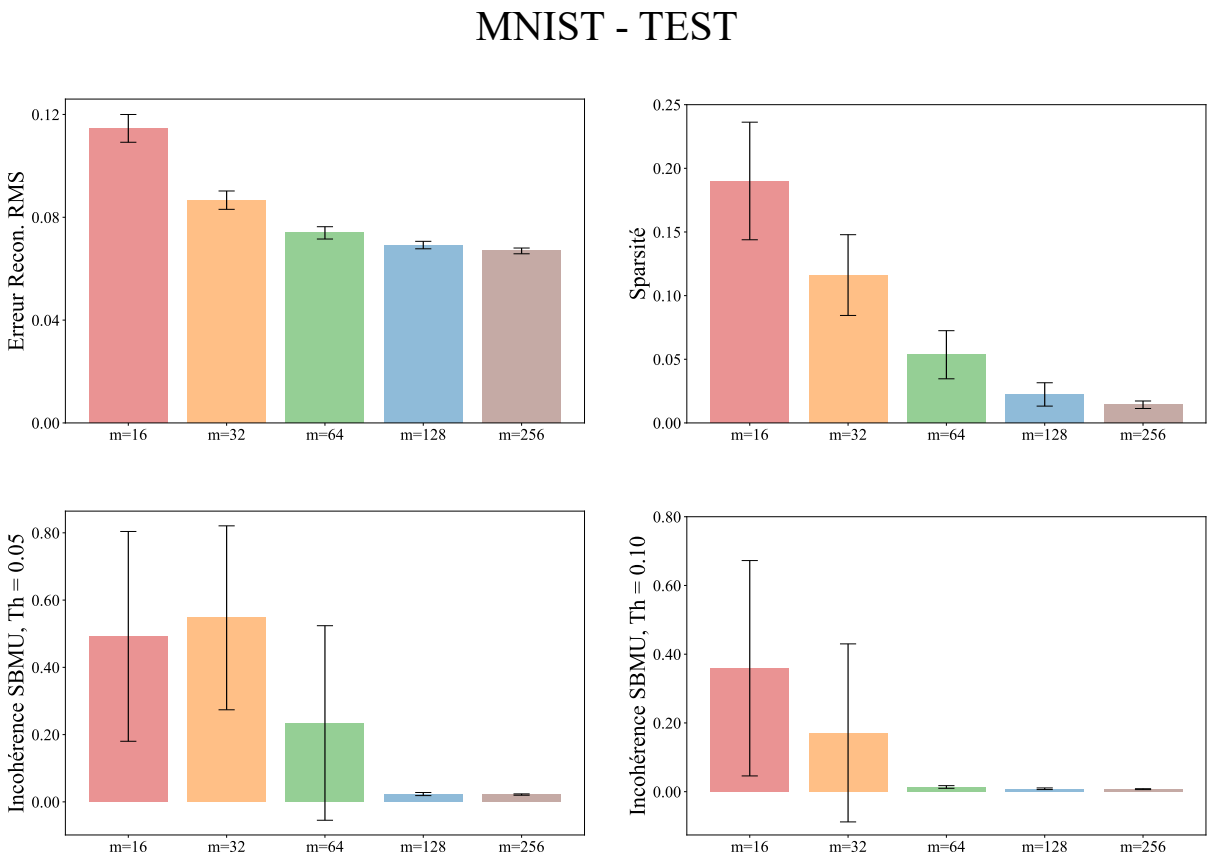

FIGURE 4.9 – Performances pour MNIST durant la phase de test. Performances pour $m \in \{16, 32, 64, 128, 256\}$ neurones, en termes d'erreur de reconstruction RMS, de sparsité, et d'incohérence dans l'élection de la SBMU pour un seuil de décision Th=5 % et Th=10 %. Les barres indiquent l'écart-type.

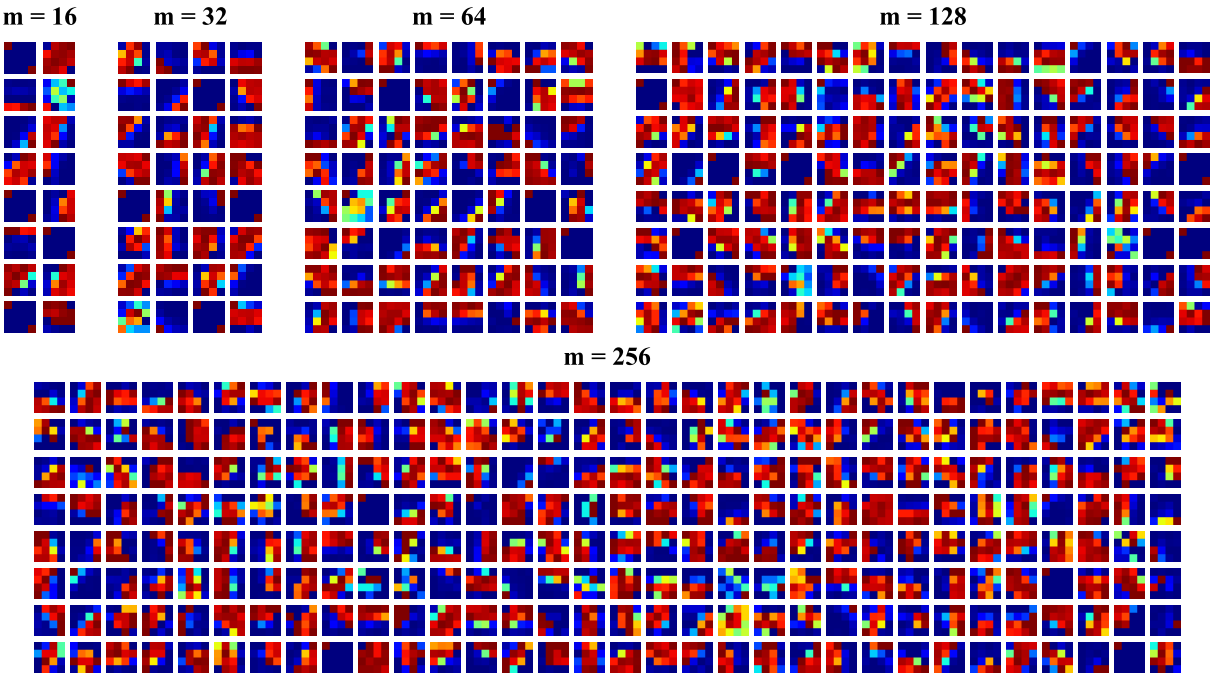

FIGURE 4.10 – Après la phase d'apprentissage sur MNIST : exemples de vecteurs codes appris dans la couche de représentation v pour $m \in \{16, 32, 64, 128, 256\}$ neurones. Le gradient rouge-bleu indique les valeurs maximales-minimales de chaque pixel reconstruit.

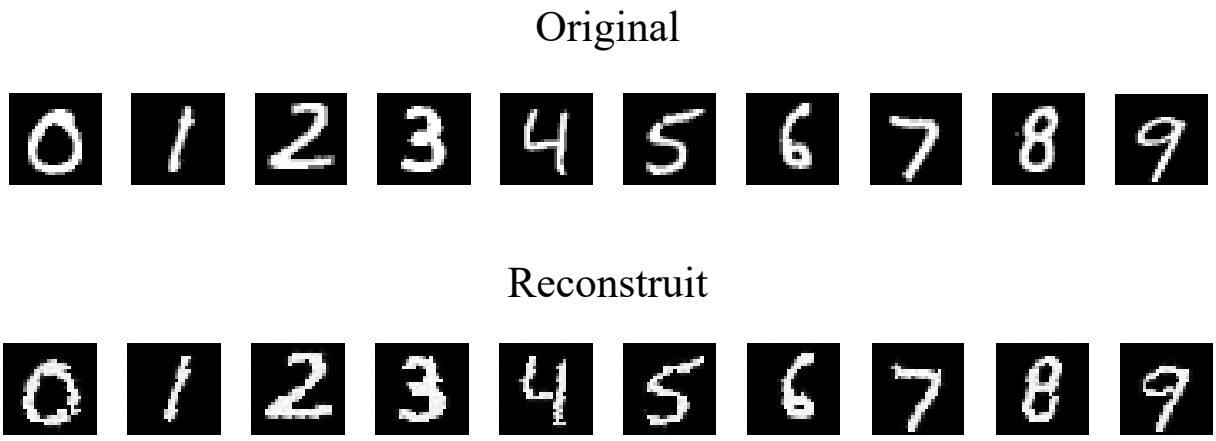

FIGURE 4.11 – Ensemble d'images de MNIST présentées en entrée du SNN et reconstruites par les vecteurs codes des neurones de la couche de représentation pour $m = 64$ neurones.

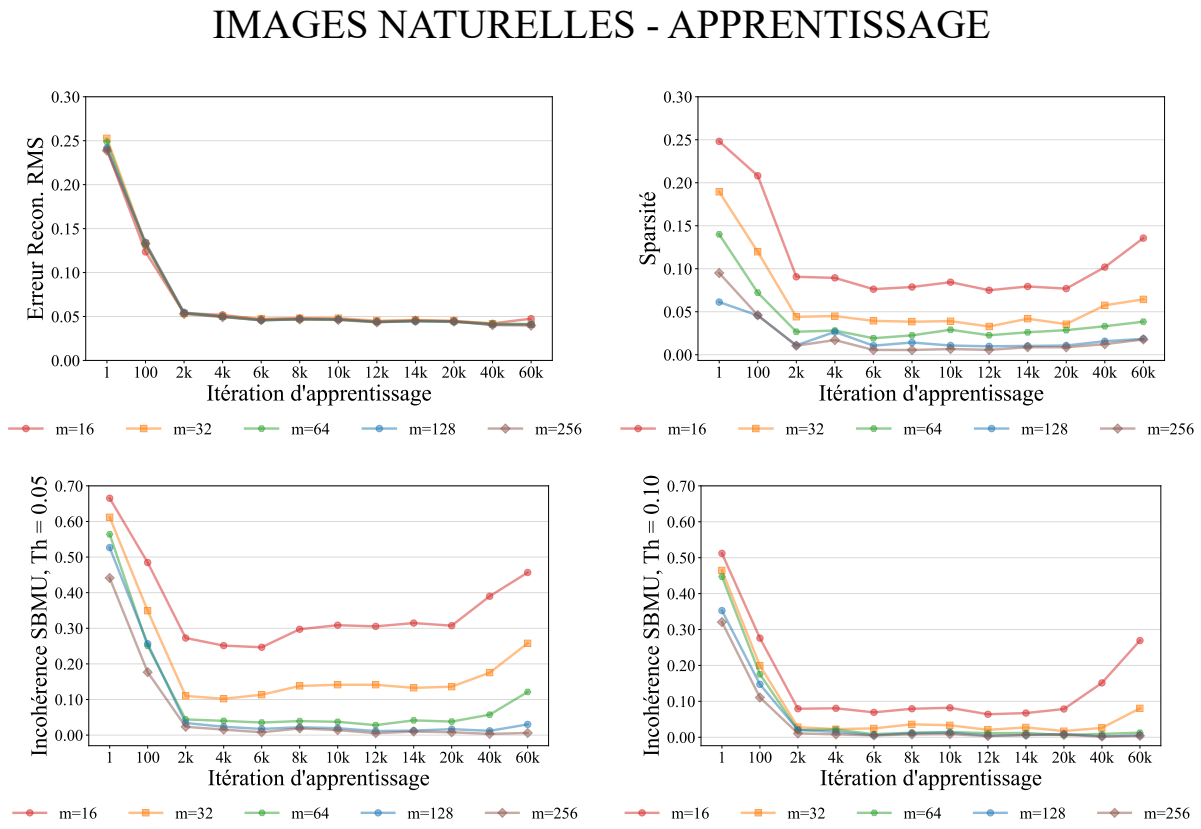

FIGURE 4.12 – Performances pour les images naturelles durant la phase l'apprentissage. Performances après 1, 100,..., 60k itérations pour $m \in \{16, 32, 64, 128, 256\}$ neurones, en termes d'erreur de reconstruction RMS, de sparsité, et d'incohérence dans l'élection auto-organisée de la SBMU pour un seuil de décision Th=5 % et Th=10 %.

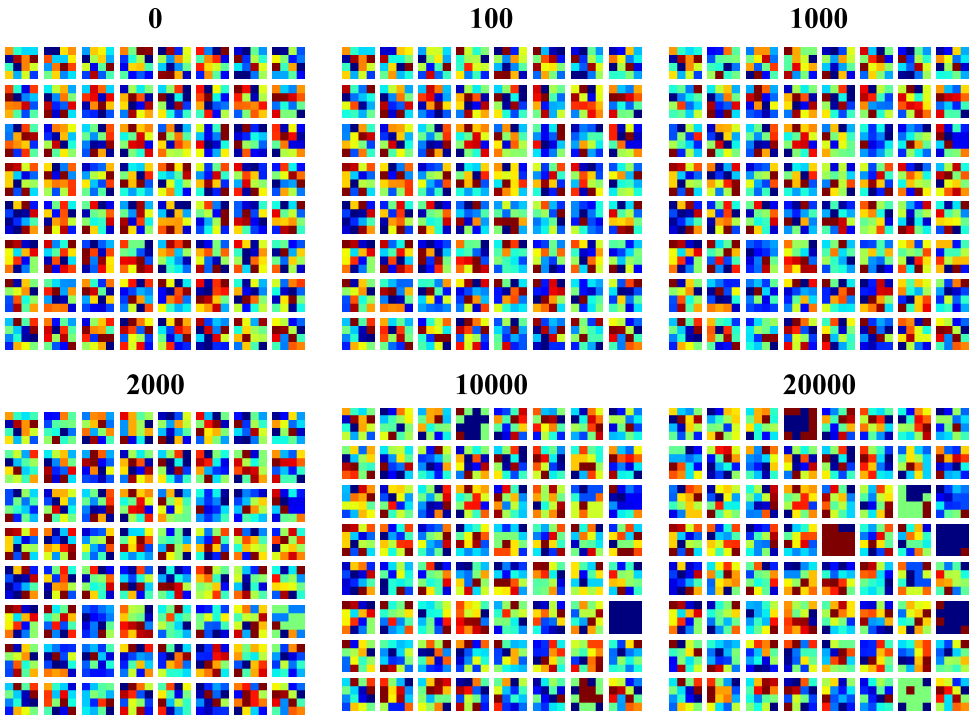

FIGURE 4.13 – Phase d'apprentissage sur des images naturelles : exemple d'évolution des vecteurs codes après 0, 100 ,..., 20k itérations d'apprentissage pour $m = 64$ neurones. Le gradient rouge-bleu indique les valeurs maximales-minimales de chaque pixel reconstruit.

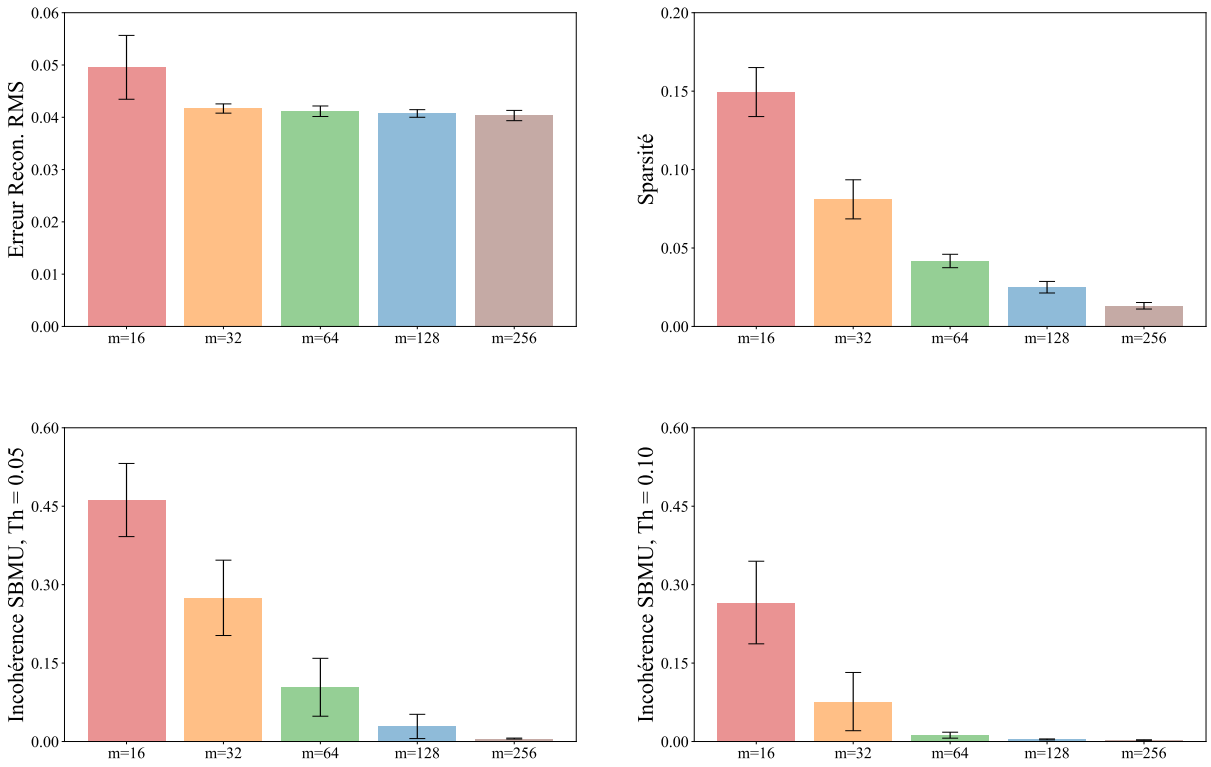

FIGURE 4.14 – Performances pour les images naturelles durant la phase de test. Performances pour $m \in \{16, 32, 64, 128, 256\}$ neurones, en termes d'erreur de reconstruction RMS, de sparsité, et d'incohérence dans l'élection de la SBMU pour un seuil de décision Th=5 % et Th=10 %. Les barres indiquent l'écart-type.

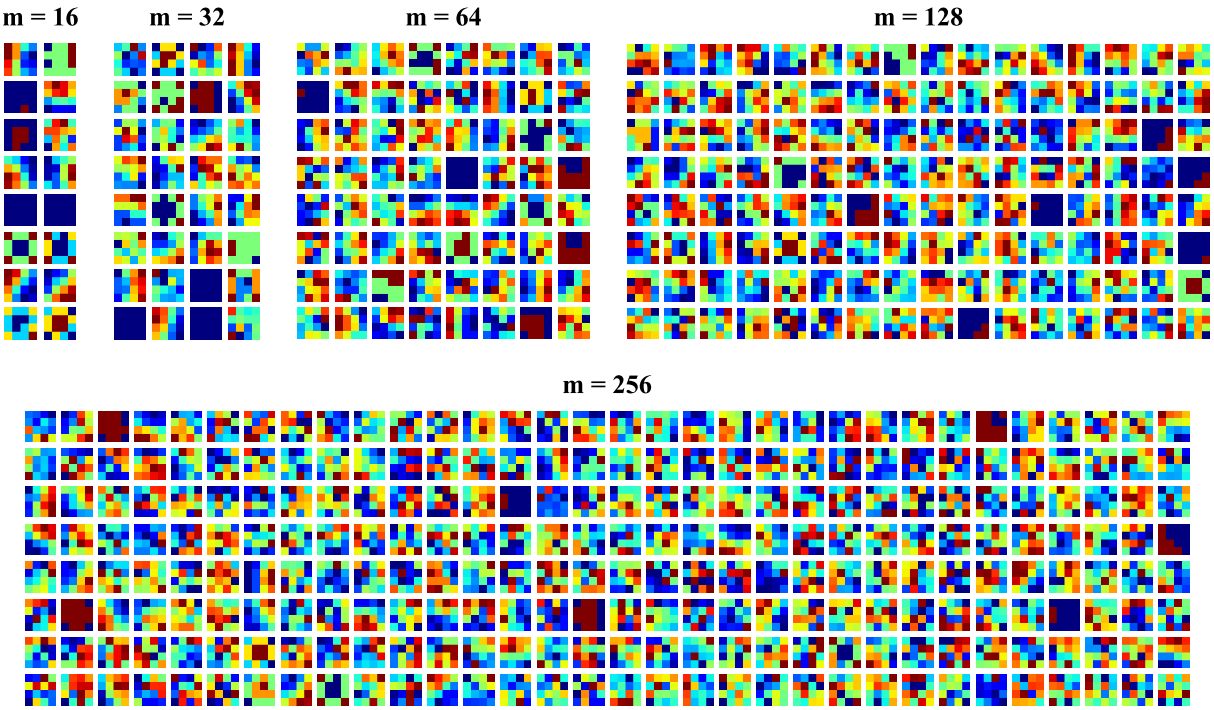

FIGURE 4.15 – Après la phase d'apprentissage sur les images naturelles : exemples de vecteurs codes appris dans la couche de représentation v pour $m \in \{16, 32, 64, 128, 256\}$ neurones. Le gradient rouge-bleu indique les valeurs maximales-minimales de chaque pixel reconstruit.

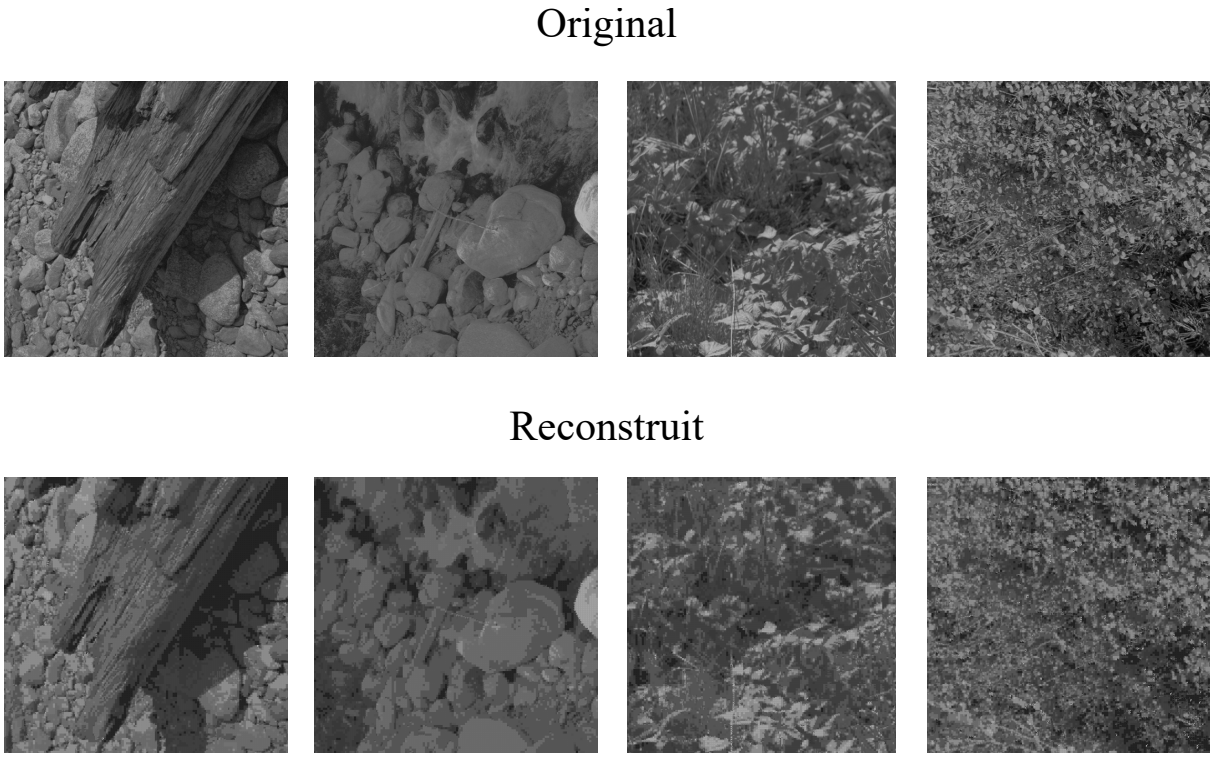

FIGURE 4.16 – Ensemble d'images naturelles originales présentées en entrée du SNN et reconstruites par les vecteurs codes des neurones de la couche de représentation pour $m = 64$ neurones.

Chapitre 5

Apprentissage de représentations dans les délais synaptiques

> Why would the brain maintain different delays with such precision if spike timing were not important ?
>
> Eugene M. Izhikevich

Sommaire

5.1 Indroduction

Dans le modèle W-TCRL présenté dans le chapitre 4, nous avons exploré comment des représentations pouvaient être apprises dans des poids synaptiques grâce à une nouvelle règle STDP fonctionnant sur la base d'un code temporel.

Cependant, cette conception de l'apprentissage est centrée sur les poids synaptiques, non sur le fonctionnement du neurone impulsionnel. Un neurone impulsionnel est un système dynamique évoluant de façon continue dans le temps, dont la fonction ultime est de produire une décision : émettre une impulsion si le seuil de décharge V_θ (ou de décision) est atteint. En déplaçant la

focale des synapses sur le neurone, une observation clé est qu'un neurone LIF est très sensible aux impulsions coïncidentes dans le temps (fig. 2.12). Ce mode de fonctionnement est nommé détecteur de coincidences (König et al., 1996).

Masquelier, Guyonneau et al., 2008 ont montré que des neurones LIF équipés d'une règle STDP adaptant les poids synaptiques étaient capables de détecter fiablement des motifs impulsionnels coïncidents plongés dans un bruit de fond, bruit de fond consistant en des impulsions émises aléatoirement dans le temps à une fréquence moyenne donnée. Bien que ce résultat soit remarquable, il nécessite que les motifs impulsionnels à détecter soient quasi-synchrones, ce qui n'est donc pas adapté au traitement de codes temporels.

Pour qu'un neurone opère en tant que détecteur de coïncidences, est-il nécessaire que les impulsions émises par les neurones présynaptiques soient synchrones ? Les délais permettent de généraliser le mode de détection de coïncidences d'entrées synchrones à des entrées asynchrones. Les délais représentent le temps pris par une impulsion émise par un neurone présynaptique pour atteindre le neurone postsynaptique. Ainsi si des impulsions sont émises de façon asynchrones, mais que les délais compensent les latences relatives, alors ces impulsions arrivent de manière synchrone au niveau du neurone postsynaptique (voir fig. 5.1). En résumé, grâce au rôle de compensation des latences entre les temps d'impulsions par les délais, un neurone peut opérer comme détecteur de coïncidences en recevant des impulsions asynchrones.

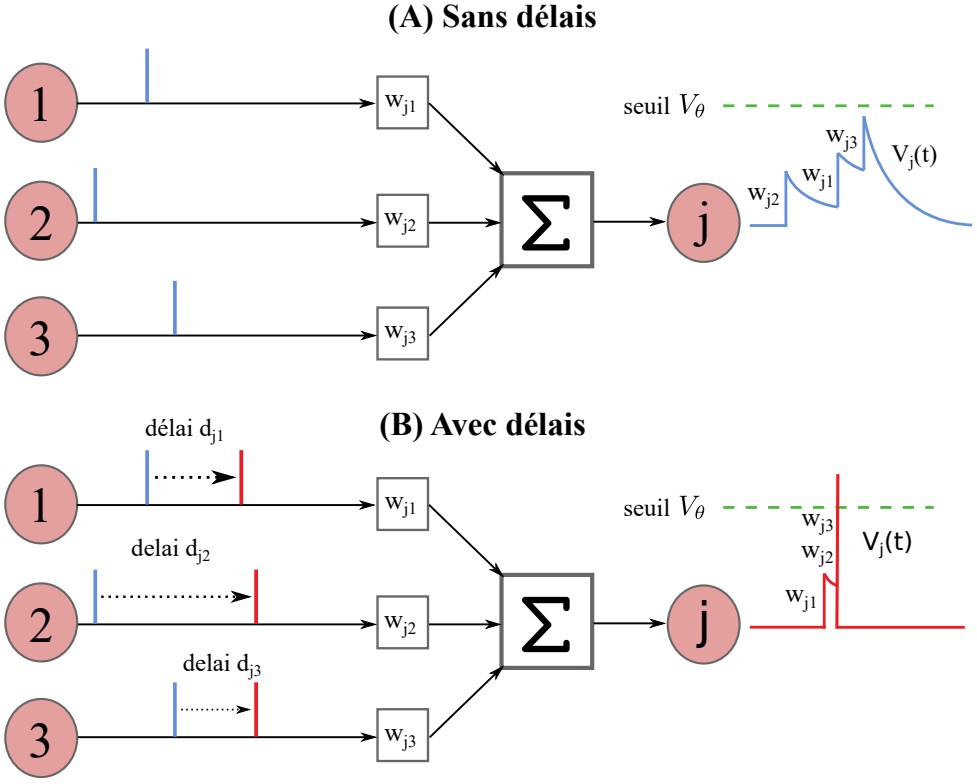

Figure 5.1 – (A) Les impulsions ne sont pas translatées dans le temps par des délais. Ainsi les impulsions sont asynchrones et restent asynchrones lorsqu'elles arrivent au niveau du neurone postsynaptique j. (B) Les délais de transmission synaptique compensent les latences entre les impulsions, résultant en une synchronisation des impulsions au niveau du neurone postsynaptique j.

D'un point de vue théorique, les délais offrent également des avantages certains. Maass et Schmitt, 1999 ont montré que la classe des fonctions apprenables avec un SNN est beaucoup plus riche avec des délais appris qu'avec des poids appris. Dans un autre contexte, Izhikevich, 2006 a montré qu'un SNN équipé de délais offre une capacité de mémoire théoriquement infinie.

Les délais offrent donc des propriétés très attrayantes encore peu exploitées :

1. Ce sont des paramètres qui semblent particulièrement adaptés aux codes temporels puisqu'ils opèrent intrinsèquement dans la dimension temporelle, contrairement aux poids synaptiques.
2. Les délais permettent de généraliser le mode de détection de coïncidences à des entrées asynchrones.
3. Ils confèrent théoriquement aux SNN une meilleure capacité computationnelle

Comment tirer profit de ces paramètres pour apprendre des représentations à partir de codes temporels ?

Une première stratégie de "sélection de délais" (Eurich et al., 1999 ; W. Gerstner et al., 1996) met à disposition un ensemble de délais fixes à chaque neurone. Pendant la phase d'apprentissage, les délais qui compensent les latences entre les entrées synchronisent les impulsions au niveau du neurone post-synaptique. Cela induit une très forte augmentation du potentiel membranaire, qui à son tour induit une probabilité plus élevée que le neurone émette une impulsion. L'émission d'une impulsion déclenche une règle d'adaptation des poids synaptiques (généralement de type STDP) renforçant les synapses qui synchronisent l'arrivée des impulsions et déprimant les autres synapses. Un poids synaptique tendant vers 0 désactive la synapse, les impulsions n'ayant plus d'impact causal sur le potentiel membranaire postsynaptique. Ainsi grâce à ce mécanisme, un sous-ensemble de délais est sélectionné parmi l'ensemble de délais disponibles, rendant le neurone sélectif à des motifs impulsionnels spécifiques.

Cependant, cette stratégie de "sélection de délais" présente deux problèmes principaux :

1. Des délais fixes peuvent mal compenser les latences relatives entre les impulsions, ce qui - en termes d'apprentissage automatique - injecte un biais élevé dans le modèle. Il s'agit ainsi de sélectionner des représentations pré-déterminées par la désactivation de connexions synaptiques.
2. Cela nécessite généralement des connexions multi-synaptiques coûteuses entre chaque paire de neurones, i.e. cette stratégie recquiert non pas une mais un nombre arbitrairement grand de synapses entre une paire de neurones.

Une deuxième stratégie "d'apprentissage des délais" (Eurich et al., 1999), considère les délais comme des paramètres soumis à apprentissage. Cette solution résout les problèmes susmentionnés du 1) biais du modèle puisqu'il y a apprentissage plutôt que sélection et 2) des connexions multi-synaptiques coûteuses en ne nécessitant qu'une seule synapse par paire de neurones.

Bien que peu nombreux, il existe quelques algorithmes ciblant l'apprentissage des délais. Cependant presque tous ces algorithmes sont supervisés (Paugam-Moisy, Martinez et al., 2008 ; Taherkhani et al., 2015 ; Wang et al., 2019 ; Zhang et al., 2020).

Parmi les rares exceptions d'apprentissage non supervisé, Matsubara, 2017 propose une méthode approximant l'algorithme Espérance-Maximisation (EM), pouvant fonctionner de manière aussi bien supervisée que non supervisée. Cette méthode utilise un modèle génératif (probabiliste) basé sur un mélange de distributions discrètes Multinoulli-Bernoulli. Néanmoins cet algorithme n'est pas basé sur des règles spatialement et temporellement locales.

Une autre méthode (Wright et Wiles, 2012) modélise les délais de transmission avec des noyaux gaussiens. Le centre et l'écart-type de la gaussienne sont ajustés de manière non supervisée pour classifier des motifs impulsionnels synthétiques. Le centre représente le délai de transmission et l'écart-type contrôle le profil de libération du courant synaptique. Si le neurone se verrouille sur un motif spécifique alors l'écart-type approche 0 et le courant est délivré instantanément. Par contraste, si l'écart-type est élevé, le courant va être délivré sur une large fenêtre temporelle. L'écart-type de la gaussienne est initialement élevé, puis est réduit durant la phase d'apprentissage avec une heuristique (recuit simulé) afin que les synapses puissent converger vers une solution.

Nadafian et Ganjtabesh, 2020 proposent des règles STDP pour ajuster les délais et les poids synaptiques. Les poids synaptiques sont renforcés pour les interactions causales (impulsion présynaptique retardée arrivant avant une impulsion postsynaptique), et déprimés pour les interactions non-causales. Les délais sont affaiblis pour les interactions causales de sorte que le neurone postsynaptique décharge plus rapidement, et augmentent pour les interactions non-causales. Cependant les règles développées ne sont pas temporellement locales. En outre, ces règles STDP n'ont pas pour objectif de minimiser l'erreur de reconstruction ou que les neurones postsynaptiques agissent comme des détecteurs de coïncidences.

Bien que ces travaux introduisent de nouveaux algorithmes d'apprentissage pour les SNN capables d'extraire des caractéristiques à partir d'entrées impulsionnelles, leurs résultats expérimentaux se limitent à tester leurs performances sur des tâches de classification. Ils ne fournissent donc pas une méthode d'apprentissage de représentations visant à obtenir une faible erreur de reconstruction. De plus, les règles développées ne sont pas spatialement et temporellement locales. Cela rend ces règles probablement incompatibles avec les processeurs neuromorphiques existants, comme Loihi (Davies et al., 2018) qui intègre des délais plastiques.

Ce chapitre propose ainsi la première méthode, à notre connaissance, capable d'apprendre des représentations dans les délais synaptiques à l'aide d'une nouvelle règle STDP, spatialement et temporellement locale, dans la perspective d'obtenir une faible erreur de reconstruction. Le choix d'apprendre des représentations à partir de codes temporels dans les délais est motivé par le fait que les délais opèrent intrinsèquement dans la dimension temporelle, ce qui n'est pas le cas des poids synaptiques. Notre nouvelle règle STDP intègre un module de quantification vectorielle pour minimiser l'erreur de reconstruction en adaptant les délais et un module de régularisation favorisant les faibles valeurs de délais. En outre, une seconde règle STDP a été développée. Elle vise à adapter les poids synaptiques en fonction de la pertinence des caractéristiques [1], pertinence quantifiée par la variabilité des caractéristiques. Ainsi un poids de pertinence est assigné à chaque caractéristique. Rappelons que dans cette architecture la fonction des poids synaptiques n'est pas de stocker des représentations - elles sont stockées dans les délais - mais de filtrer les caractéristiques en fonction de leur pertinence. Ces deux règles STDP ont des rôles complémentaires permettant d'extraire efficacement des représentations de données événementielles.

5.2 WD-TCRL : architecture et apprentissage

Cette section décrit les différents composants architecturaux et algorithmiques du SNN utilisé pour l'apprentissage de représentations. Les règles d'apprentissage ciblant les délais et les poids

1. Nous utiliserons le terme générique de caractéristiques par analogie aux *features*, ici elles se réfèrent à l'encodage temporel d'une coordonnée.

synaptiques sont également présentées. Nous nommons notre nouveau modèle *Weight Delay - Temporally Coded Representation Learning* (WD-TCRL).

5.2.1 Architecture

Le réseau de neurones impulsionnels (SNN) est composé de deux couches. La première couche encode les données d'entrée en latences relatives (voir Chap. 3.1.1) et l'autre couche extrait des représentations des motifs impulsionnels reçus. Ces deux couches sont entièrement connectées (voir Fig. 5.2). L'apprentissage est réalisé entre ces deux couches à l'aide de deux règles STDP. Une règle STDP adapte les délais d_{ji} pour stocker des représentations. Une autre règle STDP adapte les poids w_{ji} pour filtrer les caractéristiques en fonction de leurs pertinences.

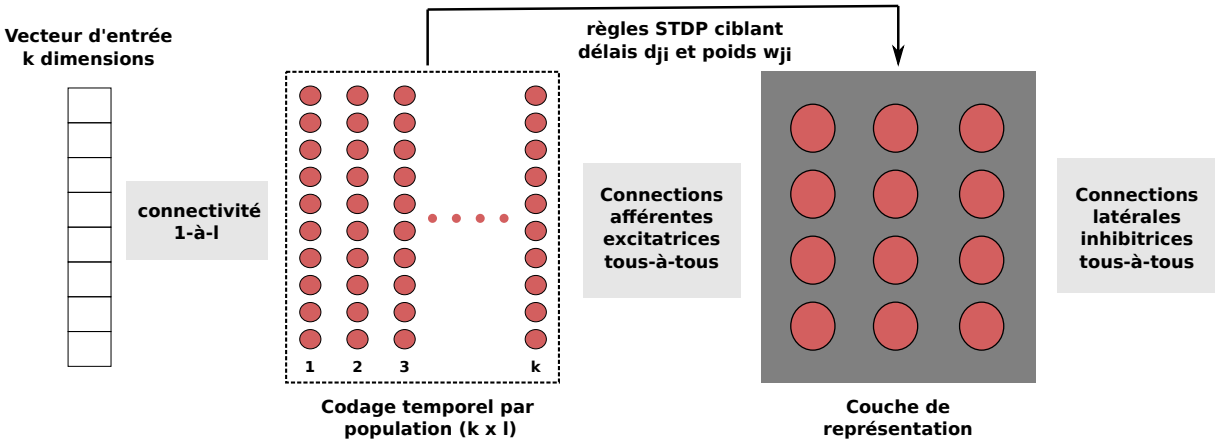

FIGURE 5.2 – Schéma bloc de l'architure du SNN. Soit un patch d'image à k pixels, correspondant à un vecteur d'entrée à k dimensions. Une dimension du vecteur d'entrée est encodée par l neurones. L'activité éparse des $k * l$ neurones est transmise à la couche de représentation. Entre ces deux couches s'opère l'apprentissage des délais d_{ji} et des poids synaptiques w_{ji} avec deux règles STDP distinctes. Des connexions latérales inhibitrices sont présentes dans la couche de représentation pour mettre les neurones en compétition, générant un apprentissage compétitif.

Modèle de neurone et de synapse

Le modèle de neurone considéré dans le SNN est le LIF régi, rappelons le (voir section 2.3.1), par l'équation :

$$\tau_m \frac{dV(t)}{dt} = -V(t) + I_{\text{ext}}(t) \tag{5.1}$$

Chaque neurone de la couche d'encodage - comme décrit dans le chapitre 3 - reçoit une entrée continue variant dans le temps $I_{\text{ext}}(t)$, similaire à l'encodage de la rétine produisant des tensions analogiques plutôt que des impulsions discrètes. Par contraste, les neurones de la couche de représentation ne reçoivent pas de tensions analogiques de la couche d'encodage, i.e. $I_{\text{ext}}(t) = 0$, mais des impulsions discrètes.

Les vecteurs d'entrée à valeur réelle sont de dimension k. Ce vecteur d'entrée est transformé en impulsions, plus précisément en latences relatives par les $n = k * l$ ($l = 10$) neurones de la couche d'encodage (voir Chapitre 3). Contrairement à un encodage en taux de décharge, l'activité est très éparse, chaque neurone émettant une unique impulsion. Le processus d'encodage est effectué sur une fenêtre temporelle $T = 25$ ms.

Ainsi les neurones de la couche de représentation reçoivent des impulsions asynchrones et éparses de la couche d'encodage. L'activité des n neurones de la couche d'encodage est transmise aux m neurones de la couche de représentation via les connexions synaptiques. Chaque synapse possède deux paramètres plastiques, un délai d_{ji} et un poids w_{ji}. Considérons un neurone présynaptique i connecté à un neurone postsynaptique j. Le neurone présynaptiques i émet une impulsion à un instant t_{pre}, reflété par $s_i(t_{\text{pre}}) = 1$. L'impulsion met un laps de temps supplémentaire d_{ji} pour atteindre le neurone postsynaptique au temps $t = t_{\text{pre}} + d_{ji}$, où elle modifie instantanément $V_j(t)$ d'une magnitude w_{ji}. En généralisant ce comportement à tous les $i = 1, 2, ..., n$ neurones présynaptiques connectés, la variation instantanée de $V_j(t)$ devient égale à la somme des activités présynaptiques retardées par les délais et pondérées par les poids synaptiques w_{ji} :

$$V_j(t) \leftarrow V_j(t) + \sum_{i=1}^{n} w_{ji} s_i(t - d_{ji}) \tag{5.2}$$

Il s'agit d'un modèle de transmission instantanée, i.e. avec un changement instantané de $V_j(t)$, tenant compte des délais de transmission. Remarquons que pour un motif impulsionnel en entrée donné, la modification des délais d_{ji} et/ou des poids w_{ji} induit un changement de la trajectoire temporelle de $V_j(t)$. Cela peut translater (ou empêcher) le moment t_{post} où le potentiel membranaire $V_j(t)$ franchit son seuil de décharge V_θ et émet une impulsion. Ce comportement est exploité par nos nouvelles règles STDP, l'une ciblant les délais d_{ji} , l'autre les poids synaptiques.

Comme dans W-TCRL, chaque synapse a accès à une variable d'état locale $x_i(t)$ (avec $i = 1, 2, \ldots n$), mémoire à court-terme de l'émission récente d'une impulsion présynaptique. Cette variable $x_i(t)$ est connue sous le nom de trace présynaptique. Elle est couramment utilisée pour des implémentations efficaces et locales des règles STDP, aussi bien en simulation (Pfister et Wulfram Gerstner, 2006) que dans les processeurs neuromorphiques (Davies et al., 2018).

De même, une trace postsynaptique $y_j(t)$ (avec $j = 1, 2, \ldots m$) accessible aux synapses mémorise l'activité postsynaptique récente. Lorsqu'une impulsion pré (post) synaptique est émise, $x_i(t)$ ($y_j(t)$) saute à 1 puis décroît exponentiellement jusqu'à 0 avec une constante de temps τ_x (τ_y) :

$$\begin{aligned} x_i(t) &\leftarrow 1, && \text{si } s_i(t) = 1 \\ \tau_x \frac{dx_i(t)}{dt} &= -x_i(t), && \text{sinon} \end{aligned} \tag{5.3}$$

$$\begin{aligned} y_j(t) &\leftarrow 1, && \text{si } s_j(t) = 1 \\ \tau_y \frac{dy_j(t)}{dt} &= -y_j(t), && \text{sinon} \end{aligned} \tag{5.4}$$

où $s(t)$ est une fonction indicatrice renvoyant 1 lorsqu'un neurone émet une impulsion au temps t, 0 sinon.

5.2.2 Apprentissage de représentations dans les délais

L'objectif visé est que les neurones de la couche de représentation apprennent des représentations à partir des latences relatives des impulsions. L'idée centrale est d'apprendre des vecteurs codes dans les délais afférents des neurones de la couche de représentation. Pour atteindre cet objectif, nous avons développé une nouvelle règle STDP intégrant un module de quantification vectorielle pour minimiser l'erreur de reconstruction et un module de régularisation pour favoriser des faibles valeurs de délais.

Représentation des temps d'impulsions

Tout d'abord, nous avons besoin d'une représentation exploitable par des règles d'apprentissage de l'activité impulsionnelle des n neurones de la couche d'encodage. Cette représentation est fournie par le vecteur $\mathbf{x}(t) \in]0,1]^n$ des n traces présynaptiques représentant le temps écoulé depuis les impulsions présynaptiques, i.e. $\mathbb{R}_+^n \rightarrow]0,1]^n$. Le vecteur des traces présynaptiques $\mathbf{x}(t)$ fournit une représentation des temps relatifs des impulsions (voir Fig. 5.3) sans horodatage, évitant l'utilisation d'une horloge globale biologiquement peu plausible et computationnellement coûteuse. De plus, $\mathbf{x}(t)$ peut être traité comme un prototype, i.e. un point dans l'espace des caractéristiques $]0,1]^n$. Notre but est d'apprendre ces prototypes avec m vecteurs codes de délais synaptiques $\{\mathbf{d_1}, \mathbf{d_2}, ..., \mathbf{d_m}\} \in [d_{\min}, d_{\max}]^n$, en utilisant une règle STDP.

Caractéristiques des règles STDP

Les règles STDP apprennent des corrélations temporelles en mettant à jour les poids synaptique en fonction de la distance temporelle $\Delta_t = t_{\text{post}} - t_{\text{pre}}$ entre les temps d'impulsions postsynaptique t_{post} et présynaptique t_{pre}. Une règle STDP est déclenchée lorsque l'impulsion présynaptique a un effet causal sur l'impulsion postsynaptique $\Delta_t \geq 0$. Une autre règle STDP est déclenchée lorsque l'impulsion présynaptique n'a pas d'effet causal $\Delta_t < 0$. De plus, les règles STDP sont généralement spatialement et temporellement locales en opérant 1) sur une seule paire i, j de neurones pré- et postsynaptiques et 2) dans une fenêtre temporelle, respectivement.

Cependant - comme exposé dans la section 2.2.2 - les règles STDP classiques ne permettent pas à la fois d'obtenir des représentations stables et d'exploiter pleinement l'intervalle de variation des paramètres soumis à apprentissage (Billings et van Rossum, 2009 ; Rossum et al., 2000).

WD-TCRL : règle STDP pour l'apprentissage de représentations dans les délais

Nous avons construit une règle STDP capable d'apprendre de manière fiable les latences relatives des impulsions, en ciblant les délais plutôt que les poids synaptiques :

$$\Delta d_{ji} = \begin{cases} \alpha^+\Big(-\tau_x \ln\big(x_i(t)\big) - (d_{ji} + \lambda d_{ji})\Big), & \text{si } s_j(t) = 1 \text{ et } x_i(t) > \epsilon \\ -\alpha^-\Big(-\tau_y \ln\big(y_j(t)\big)\Big), & \text{si } s_i(t - d_{ji}) = 1 \text{ et } y_j(t) > \epsilon \end{cases} \tag{5.5}$$

où α^+ et α^- sont les taux d'apprentissages. Cette règle STDP fonctionne en combinaison avec l'introduction de limites dures (*hard bounds*) bornant les délais dans $d_{\min} \leq d_{ji} \leq d_{\max}$:

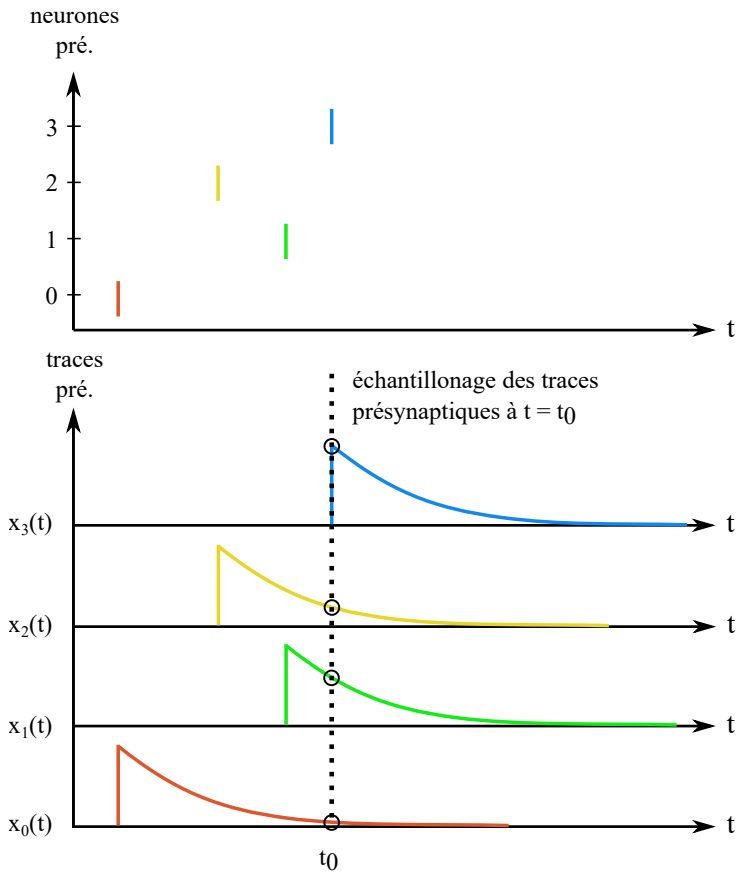

FIGURE 5.3 – Une trace présynaptique saute à 1 lorsqu'un neurone présynaptique émet une impulsion, puis décroît exponentiellement jusqu'à 0. La valeur des traces - 4 traces dans cet exemple - peut être échantillonnée à tout moment, ici à $t = t_0$. Nous obtenons alors un point (ou prototype) dans un espace de caractéristiques à 4 dimensions $\mathbf{x}(t_0) \in]0, 1]^4$. Figure inspirée de LAGORCE et al., 2017.

$$d_{ji} \leftarrow \min\big(d_{\max}, \max(d_{\min}, d_{ji} + \Delta d_{ji})\big)$$

Premier cas d'adaptation pour l'apprentissage des délais

Le premier cas d'adaptation est 1) déclenché de façon événementielle par une impulsion postsynaptique indiquée par $s_j(t = t_{\text{post}}) = 1$ et 2) opère localement dans le temps car la trace présynaptique $x_i(t)$ est échantillonnée au temps $t = t_{\text{post}}$ et doit être supérieure à $\epsilon = 0.05 \approx e^{-3\tau_x/\tau_x}$ correspondant à une fenêtre temporelle de $3\tau_x$ depuis le temps de l'impulsion présynaptique t_{pre}. Ainsi, les impulsions présynaptiques émises en dehors de cette fenêtre temporelle ne déclenchent pas la première règle. Cette règle traite le cas des impulsions présynaptiques non retardées, i.e. avant l'application du délai, émises avant ou au moment de l'impulsion postsynaptique, i.e. $\Delta_t \geq 0$.

La règle intègre un module de quantification vectorielle et un module de régularisation λd_{ji}. Le module de quantification vectorielle vise à apprendre la structure des temps relatifs des impulsions dans les délais synaptiques, et donc à minimiser l'erreur de reconstruction, l'encodage des données étant temporel. Le module de quantification vectorielle minimise l'erreur de reconstruction en minimisant la distance temporelle entre Δ_t et d_{ji}. Pour s'en convaincre, il suffit de supprimer le module de régularisation en fixant $\lambda = 0$. Maintenant, en analysant la solution à l'équilibre $\Delta_{d_{ji}} = 0$, on peut voir que d_{ji} converge vers $\Delta_t = t_{\text{post}} - t_{\text{pre}}$ qui est égal à la fonction inverse $-\tau_x \ln\big(x_i(t)\big)$ de la trace présynaptique $x_i(t)$, minimisant ainsi la distance temporelle :

$$\begin{aligned}
&\Delta_{d_{ji}} = 0 \\
\Leftrightarrow\ & 0 = -\tau_x \ln\big(x_i(t)\big) - d_{ji} \\
\Leftrightarrow\ & d_{ji} = -\tau_x \ln\big(x_i(t)\big) \\
\Leftrightarrow\ & d_{ji} = -\tau_x \ln\left(\exp\left(-\frac{t - t_{\text{pre}}}{\tau_x}\right)\right)
\end{aligned}$$

Ce cas d'adaptation étant déclenché par une impulsion postsynaptique indiquée par $s_j(t = t_{\text{post}}) = 1$, nous obtenons finalement :

$$\begin{aligned}
\Leftrightarrow\ & d_{ji} = -\tau_x \ln\left(\exp\left(-\frac{\Delta_t}{\tau_x}\right)\right) \\
\Leftrightarrow\ & d_{ji} = \Delta_t
\end{aligned} \tag{5.6}$$

Ainsi un délai apprend à prédire l'intervalle de temps Δ_t entre une impulsion présynaptique et une impulsion postsynaptique. Cela est réalisé avec une règle STDP tirant profit d'informations spatialement et temporellement locales. En passant du niveau individuel à un niveau collectif, un vecteur code de délais apprend un centroïde dans un espace temporel.

En ajoutant le module de régularisation, la solution à l'équilibre devient :

$$
\begin{aligned}
&\Delta_{d_{ji}} = 0 \\
\Leftrightarrow\ &0 = -\tau_x \ln\big(x_i(t)\big) - d_{ji} - \lambda d_{ji} \\
\Leftrightarrow\ &d_{ji} = \frac{-\tau_x \ln\big(x_i(t)\big)}{\lambda + 1} \\
\Leftrightarrow\ &d_{ji} = \frac{\Delta_t}{\lambda + 1}
\end{aligned} \tag{5.7}
$$

En observant l'équation 5.7, l'hyperparamètre λ contrôle le curseur entre le critère de quantification vectorielle et le critère de régularisation. Lorsque $\lambda \to 0$ nous retrouvons un pur critère de quantification vectorielle car $d_{ji} \to \Delta_t$. Lorsque $\lambda \to \infty$ nous obtenons un pur critère de régularisation car $d_{ji} \to 0$.

Avoir $\lambda > 0$ est recommandé pour éviter de grandes valeurs de délais, augmentant la latence de transmission de l'information et diminuant ainsi la vitesse de traitement du SNN. Sans ce module de régularisation, la règle STDP resterait sensible aux temps relatifs des impulsions mais ne serait pas sensible à la magnitude des délais. Par exemple la règle STDP n'induirait pas de différence entre un vecteur de délais (d_{11}, d_{12}, d_{13}) encodant les temps relatifs des impulsions, et ce même vecteur de délais translaté par un coefficient $c > 0$: $(d_{11} + c, d_{12} + c, d_{13} + c)$. Le module de régularisation aide à verrouiller les valeurs des délais sur la magnitude des temps relatifs entre impulsions. Cela a pour conséquence de permettre aux neurones de réagir et de prendre des décisions à la même échelle temporelle que celle des motifs impulsionnels arrivant en entrée.

Une autre interprétation de λ est de considérer un vecteur code de délais synaptiques $\mathbf{d}_j$ comme un vecteur de translation opérant dans le domaine temporel. Une grande valeur de l'hyperparamètre λ favorise alors des coefficients épars de ce vecteur de translation.

Deuxième cas d'adaptation des délais

De manière similaire au premier cas d'adaptation, le deuxième cas d'adaptation est 1) déclenché sur un mode événementiel par une impulsion présynaptique retardée, i.e. une impulsion qui est arrivée au terminal postsynaptique après l'application du délai, indiqué par $s_i(t_{\text{pre}} = t - d_{ji}) = 1$ et 2) opère localement dans le temps car la trace postsynaptique $y_j(t)$ est échantillonnée au temps $t = t_{\text{pre}} + d_{ji}$ et doit être supérieure à $\epsilon = e^{-3\tau_y/\tau_y} \approx 0.05$ qui reflète une fenêtre temporelle de $3\tau_y$ depuis le temps de l'impulsion postsynaptique t_{post}. Cette règle gère le cas des impulsions présynaptiques retardées reçues après t_{post}.

Dans ce cas, le délai d_{ji} est réduit d'une magnitude $\Delta_{d_{ji}}$ qui croît linéairement avec l'intervalle de temps $\Delta'_t = (t_{\text{pre}} + d_{ji}) - t_{\text{post}}$, proportionnelle à la fonction inverse $-\tau_y \ln\big(y_j(t)\big)$ de la trace postsynaptique $y_j(t)$:

$$
\begin{aligned}
\Delta d_{ji} &\propto -\tau_y \ln\big(y_j(t)\big) \\
\Delta d_{ji} &\propto -\tau_y \ln\left(\exp\left(-\frac{t - t_{\text{post}}}{\tau_y}\right)\right)
\end{aligned}
$$

Ce cas d'adaptation étant déclenché par une impulsion présynaptique retardée, nous obtenons :

$$\begin{aligned}
\Delta d_{ji} &\propto -\tau_y \ln\left(\exp\left(-\frac{t_{\text{pre}} + d_{ji} - t_{\text{post}}}{\tau_y}\right)\right) \\
\Delta d_{ji} &\propto -\tau_y \ln\left(\exp\left(-\frac{\Delta'_t}{\tau_y}\right)\right) \\
\Delta d_{ji} &\propto -\tau_y \ln\left(\exp\left(-\frac{d_{ji} - \Delta_t}{\tau_y}\right)\right) \\
\Delta d_{ji} &\propto d_{ji} - \Delta_t
\end{aligned}$$

Δd_{ji} est proportionnel à $d_{ji} - \Delta_t$, le tout est multiplié par $-\alpha^-$. Ainsi, si une impulsion présynaptique retardée n'a pas d'effet causal sur le neurone postsynaptique en arrivant après t_{post}, le délai d_{ji} est diminué d'une magnitude fonction de la distance temporelle $\Delta'_t = (t_{\text{pre}} + d_{ji}) - t_{\text{post}} = d_{ji} - \Delta_t$. Par conséquent, une paire de neurones n'ayant pas d'interactions causales induit un délai tendant vers $d_{\min}$.

Les neurones deviennent des détecteurs de coïncidence

En incorporant un module de quantification vectorielle minimisant la distance temporelle entre $\Delta_t = t_{\text{post}} - t_{\text{pre}}$ et d_{ji}, notre règle STDP apprend les temps relatifs des impulsions dans des vecteurs code de délais. Cela engendre la synchronisation des impulsions présynaptiques asynchrones au niveau du terminal poststynaptique. En d'autres termes, la minimisation de la distance temporelle via l'adaptation des délais maximise la synchronisation des impulsions au niveau du terminal postsynaptique. Ainsi, les neurones postsynaptiques agissent progressivement comme des détecteurs de coïncidences.

5.2.3 Adaptation des poids des caractéristiques

Une faible (grande) distance temporelle entre un motif impulsionnel représenté par un prototype $\mathbf{x}(t)$ et un vecteur code (ou centroïde) $\mathbf{d}_j$ de délais, se traduit par un degré élevé (faible) d'impulsions coïncidentes. Un degré élevé (faible) d'impulsions coïncidentes induit une valeur de crête élevée (faible) du potentiel membranaire postsynaptique $V_j(t)$. Les impulsions coïncidentes induisent une valeur de crête plus élevée que les impulsions non coïncidentes (voir fig 5.1) en raison de la décroissance exponentielle dans le temps de $V_j(t)$.

Circuit Winner-Take-All

Les neurones postsynaptiques deviennent progressivement sélectifs à des motifs impulsionnels spécifiques via l'adaptation de leurs délais afférents. En outre, les neurones postsynaptiques prennent une décision (émission d'une impulsion) avant que l'intégralité des impulsions présynaptiques soient reçues - de façon analogue aux neurones biologiques - accélérant ainsi la vitesse de traitement du SNN. Rappelons que les premières impulsions transmettent la plus grande quantité d'informations sur l'entrée (voir Fig. 3.11) et donc réduisent le plus l'incertitude sur l'entrée. Cette propriété est exploitée par un circuit temporel *Winner-Take-All* (WTA), dans lequel le premier neurone qui franchit son seuil de décharge $V_j(t) > V_\theta$ émet une impulsion en réponse à un motif impulsionnel en entrée, et est déterminé comme étant le meilleur représentant de ce motif

impulsionnel. Nous appelons *Spiking Best Matching Unit* (SBMU) l'index du neurone impulsionnel déchargeant en premier. Le premier neurone qui décharge (SBMU) inhibe fortement les autres neurones de la couche de représentation, les empêchant de décharger et ainsi d'apprendre le prototype courant. Ce schème d'apprentissage compétitif et auto-organisé est implémenté par des connexions inhibitrices latérales tous-à-tous. Le circuit induit une activité très éparse dans la couche de représentation et force les neurones à apprendre des vecteurs code non corrélés, deux ingrédients importants pour un apprentissage efficace (Bengio et al., 2013 ; Falez et al., 2019).

Cependant pour un vecteur d'entrée donné, un pur circuit WTA induit l'actualisation du vecteur code d'un unique neurone : la SBMU. Comme dans W-TCRL, nous souhaitons accélérer l'apprentissage en recrutant plus d'un neurone (K-WTA) durant la présentation d'un vecteur d'entrée. Pour atteindre cet objectif, nous introduisons un mécanisme homéostatique augmentant la valeur des poids synaptiques latéraux de la couche de représentation. L'inhibition latérale est initialement faible avec des poids latéraux initialisés à $w_{ji} = -c_{min}$ puis augmente durant la phase d'apprentissage avec le mécanisme homéostatique jusqu'à atteindre la valeur cible $w_{ji} \approx -c_{max}$. Ainsi en utilisant des valeurs de $c_{\min}$ et $c_{\max}$ appropriées, le circuit passe dynamiquement d'un comportement K-WTA à WTA. Cela permet de recruter de moins en moins de neurones durant la phase d'apprentissage et d'ainsi progressivement décorréler et spécialiser les vecteurs codes appris par les neurones. Pour rappel, le mécanisme homéostatique est implémenté par :

$$\tau_w \frac{dw_{ji}}{dt} = -c_{\max} - w_{ji} \tag{5.8}$$

où τ_w est la constante de temps du mécanisme homéostatique. Nous fixons τ_w à $1/3$ du temps de la phase d'apprentissage, de telle sorte à atteindre $w_{ji} \approx -c_{max}$ à la fin de la phase d'apprentissage. Ainsi pour une fenêtre temporelle d'encodage T et P vecteurs d'entrées, nous obtenons la relation $\tau_w = 1/3 * T * P$.

WD-TCRL : règle STDP pour l'apprentissage des poids

Cependant, des représentants erronés peuvent être élus dans ce schème compétitif, i.e. le neurone présentant la distance euclidienne minimale entre son vecteur code $\mathbf{d}_j$ et le prototype courant $\mathbf{x}(t)$ peut ne pas décharger en premier. Ce cas correspond à une SBMU paradoxale.

Notons que les poids synaptiques w_{ji} ont - comme les délais d_{ji} - un pouvoir causal sur la trajectoire du potentiel membranaire postsynaptique $V_j(t)$ et donc en dernière analyse sur le temps d'impulsion postsynaptique t_{post}. Intuitivement, nous voulons que les synapses participant à des motifs impulsionnels *clusterisés* par un neurone soient renforcées, et que les autres synapses soient affaiblies afin de moduler leur pouvoir causal. De plus, nous voulons atteindre cet objectif avec une autre règle STDP ciblant cette fois les poids synaptiques.

Il convient de rappeler qu'une faible (grande) distance temporelle proche entre un prototype $\mathbf{x}$ et un vecteur code $\mathbf{d}_j$ se traduit par un degré élevé (faible) d'impulsions coïncidentes, induisant une faible (grande) variabilité temporelle et donc une grande (faible) valeur de crête de $V_j(t)$. Autrement dit, le nombre d'impulsions coïncidentes est maximal lorsque la distribution des latences entre impulsions est égale à la distribution des valeurs des délais, ce qui induit une activation maximale du neurone postsynaptique. Par contraste, le nombre d'impulsions coïncidentes diminue lorsque les latences des entrées s'écartent de l'espérance empirique locale stockée dans le

centroïde de délais. Par conséquent, nous pouvons utiliser la variabilité temporelle pour déterminer les poids des caractéristiques dans la perspective de diminuer le taux de SBMU paradoxales et ainsi faciliter l'apprentissage. Adapter les poids synaptiques en fonction de la variabilité temporelle des caractéristiques permet d'augmenter la différence/marge dans la valeur de crête de $V_j(t)$ induite par des impulsions coïncidentes et non coïncidentes, respectivement.

Des caractéristiques densément concentrées dans une région de l'espace d'entrée sont hautement pertinentes car elles correspondent à un *cluster*, et induisent après convergence des délais un haut degré d'impulsions coïncidentes (la dispersion relativement au centroïde de délais étant faible). Par contraste, des caractéristiques très dispersées sont peu pertinentes et induisent un faible degré d'impulsions coïncidentes (la dispersion relativement au centroïde de délais étant élevée). Nous souhaitons utiliser un indicateur de dispersion pour assigner une pertinence à chaque caractéristique. Nous utilisons la variance comme indicateur de dispersion pour assigner des poids de pertinence aux caractéristiques (Domeniconi et al., 2007).

Afin d'assigner des poids de pertinence, nous introduisons d'abord un paramètre synaptique local v_{ji} pour l'estimation de la variance temporelle locale. Les dimensions (ou caractéristiques) ayant une variance faible (élevée) v_{ji} se voient attribuer des valeurs de poids élevées (faibles) $w_{ji} \in [0, 1]$. La variance temporelle locale v_{ji} est estimée en exploitant l'erreur temporelle locale e_{ji} entre une dimension d'un prototype $x_i(t)$ et la dimension associée à son vecteur code d_{ji}. L'erreur temporelle locale $e_{ji} = -\tau_x \ln\big(x_i(t)\big) - d_{ji}$ est déjà calculée et disponible dans l'équation 5.5. La règle STDP pour la variation du poids synaptique Δw_{ji} est :

$$\Delta w_{ji} = \begin{cases} \beta^+\big(\exp\big(-\frac{v_{ji}}{\sigma^2}\big) - w_{ji}\big), & \text{si } s_j(t) = 1 \text{ et } x_i(t) > \epsilon \text{ et } e_{ji} \geq 0 \\ -\beta^-\big(1 - y_j(t)\big), & \text{si } s_i(t - d_{ji}) = 1 \text{ et } y_j(t) > \epsilon \end{cases} \tag{5.9}$$

où β^+ et β^- sont les taux d'apprentissages. Cette règle STDP fonctionne en combinaison avec l'introduction d'une limite dure bornant les poids synaptique de sorte que $w_{ji} + \Delta w_{ji} \geq 0$:

$$w_{ji} \leftarrow \max\big(0, w_{ji} + \Delta w_{ji}\big)$$

La règle actualisant la variance temporelle v_{ji} est donnée plus loin.

Premier cas d'adaptation des poids synaptiques

Comme pour la règle STDP ciblant les délais, le premier cas d'adaptation est déclenché de façon événementielle par une impulsion postsynaptique $s_j(t) = 1$ et opère localement dans le temps dans une fenêtre temporelle de $3\tau_x$ depuis t_{pre}, correspondant à $x_i(t) > \epsilon$.

Nous ajoutons une contrainte supplémentaire pour tenir compte du caractère causal de l'interaction entre la paire de neurones. Le cas où 1) un temps d'impulsion présynaptique retardé $t_{\text{pre}} + d_{ji}$ arrive avant ou à t_{post} est causal et est reflété par $e_{ji} \geq 0$. L'autre cas où 2) il arrive après t_{post} n'est pas causal, ce qui est reflété par $e_{ji} < 0$. Le cas non causal $e_{ji} < 0$ est rejeté, i.e. ne déclenche pas l'application de la règle. Si les contraintes sont satisfaites, la variance temporelle locale v_{ji} est mise à jour en ligne et de manière événementielle avec une variance exponentielle mobile :

$$v_{ji} \leftarrow (1-\alpha^+\gamma)(v_{ji}+\alpha^+\gamma e_{ji}^2) \qquad \text{si } s_j(t)=1 \text{ et } x_i(t)>\epsilon \text{ et } e_{ji}\geq 0 \tag{5.10}$$

Une faible valeur de l'hyperparamètre $\gamma < 1$ et un grand nombre d'échantillons donnent une bonne estimation de la variance temporelle locale v_{ji}. Les impulsions causales qui appartiennent (n'appartiennent pas) à la région localement *clusterisée* d'un neurone induisent une faible (grande) variance temporelle v_{ji}.

La règle est stable, comme nous pouvons le voir en analysant sa solution à l'équilibre $\Delta_{w_{ji}} = 0$:

$$\begin{aligned} &\Delta_{w_{ji}} = 0 \\ \Leftrightarrow\ &0 = \exp\left(-\frac{v_{ji}}{\sigma^2}\right) - w_{ji} \\ \Leftrightarrow\ &w_{ji} = \exp\left(-\frac{v_{ji}}{\sigma^2}\right) \end{aligned} \tag{5.11}$$

Ainsi, le poids w_{ji} converge vers une valeur dans l'intervalle $]0,1]$ qui est fonction de la variance temporelle locale v_{ji}. Si $v_{ji} \to \infty$ alors le poids $w_{ji} \to 0$, sinon si la variance $v_{ji} \to 0$ alors le poids $w_{ji} \to 1$.

Les variations locales de la variance v_{ji} se répercutent de manière exponentielle dans les valeurs des poids w_{ji} correspondantes (Eq. 5.11). La pente de la fonction exponentielle est contrôlée par l'hyperparamètre σ qui s'exprime en unités de temps. Un grand (petit) σ signifie une tolérance élevée (faible) à la variabilité temporelle.

En résumé, un terminal synaptique transmettant des impulsions à caractère causal $e_{ji} \geq 0$ et induisant une faible (grande) variance temporelle v_{ji} induira une grande (faible) valeur de poids synaptique w_{ji}. Ainsi, les poids synaptiques deviennent sensibles à la variabilité temporelle et donc à la pertinence des caractéristiques locales.

Deuxième cas d'adaptation des poids synaptiques

Le deuxième cas d'adaptation est 1) déclenché de façon événementielle par une impulsion présynaptique retardée, i.e. une impulsion arrivée au terminal postsynaptique après avoir été retardée par un délai, indiqué par $s_i(t_{\text{pre}} = t - d_{ji}) = 1$ et 2) opère localement dans le temps dans une fenêtre temporelle de $3\tau_y$ depuis t_{post}, correspondant à $y_j(t) > \epsilon$. Cette règle gère le cas d'impulsions présynaptiques retardées reçues après t_{post} $(t = t_{\text{pre}} + d_{ji} > t_{\text{post}})$:

$$\begin{aligned} \Delta w_{ji} &\propto 1 - y_j(t) \\ \Delta w_{ji} &\propto 1 - \exp\left(-\frac{t - t_{\text{post}}}{\tau_y}\right) \end{aligned}$$

Ce cas d'adaptation étant déclenché par une impulsion présynaptique retardée, nous obtenons :

$$\Delta w_{ji} \propto 1 - \exp\left(-\frac{t_{\text{pre}} + d_{ji} - t_{\text{post}}}{\tau_y}\right)$$
$$\Delta w_{ji} \propto 1 - \exp\left(-\frac{d_{ji} - \Delta_t}{\tau_y}\right)$$
$$\Delta w_{ji} \propto 1 - \exp\left(-\frac{\Delta'_t}{\tau_y}\right)$$

Dans ce cas d'adaptation, le poids w_{ji} est diminué d'une magnitude $\Delta_{w_{ji}}$ qui croît de manière non linéaire avec la distance temporelle $\Delta'_t = (t_{\text{pre}} + d_{ji}) - t_{\text{post}}$. La trace postsynaptique $y_j(t)$ est utilisée pour obtenir cette non-linéarité. En effet $y_j(t)$ peut être considéré comme un noyau de décroissance exponentielle appliqué sur Δ'_t. Lorsque $\Delta'_t \to \infty$, alors $y_j(t) \to 0$ et donc la diminution du poids $\Delta_{w_{ji}} \to -\beta^-$ est maximale. Inversement, lorsque $\Delta'_t \to 0$, alors $y_j(t) \to 1$ et donc la diminution de poids $\Delta_{w_{ji}} \to 0$ est minimale. Ainsi, si une impulsion présynaptique retardée n'a pas d'effet causal sur le neurone postsynaptique en arrivant après t_{post}, le poids w_{ji} est diminué en fonction de la distance temporelle Δ'_t. Cela implique qu'une connexion synaptique entre une paire de neurones n'ayant pas d'interactions causales devient progressivement désactivée. Les poids synaptiques ne sont autorisés qu'à prendre des valeurs positives en utilisant une limite dure (*hard bound*). La limite dure est uniquement nécessaire pour la borne inférieure car cette seconde règle induit potentiellement une dérive du poids synaptique $w_{ji} \to -\infty$. La première règle introduit une limite souple à 1 via le terme w_{ji}.

5.3 Protocole expérimental et résultats

Les performances de notre modèle sont évaluées sur la base de donnée MNIST et la base de données constituée d'images naturelles déjà utilisée au chapitre précédent.

5.3.1 Paramètres de WD-TCRL

Paramétrisation du SNN en fonction de la dimension des données d'entrées

Comme il l'a été abordé dans le chapitre 4, un problème récurrent pour l'apprentissage dans les SNN provient de leur manque d'adaptabilité à des dimensionalités de données variées : un réglage fin des hyperparamètres est souvent nécessaire.

Pour traiter ce problème, nous présentons une paramétrisation générique de WD-TCRL afin de traiter des données de dimension k arbitrairement grande. Seuls trois paramètres dépendent de la dimension k des données dans la couche de représentation : le seuil de décharge des neurones V_θ, le niveau d'inhibition latéral initial $c_{\min}$ et final $c_{\max}$. Par commodité nous posons à nouveau l'hypothèse d'une relation linéaire entre la dimension k des données d'entrées et chacun de ces 3 paramètres. On se rapportera à la section 4.3.3 pour la présentation générale du raisonnement utilisé.

Les coefficients des équations linéaires liant ces 3 paramètres à la dimension des données k et le nombre de neurone encodeurs l ont encore été déterminés par la méthode d'optimisation bayésienne *Tree-Structured Parzen Estimator* (TPE) (Bergstra et al., 2011). Nous n'utilisons

pas les coefficients du modèle W-TCRL, car la méthode d'apprentissage de WD-TCRL est différente ainsi que le modèle de transmission synaptique utilisé. Pour rappel, la tâche considérée est la minimisation de l'erreur de reconstruction RMS pour la base de données MNIST et la base d'images naturelles, avec des vecteurs d'entrées de dimensions différents. Le but est d'obtenir des coefficients suffisamment génériques et robustes pour traiter des distributions de données d'entrée variées et de dimension variées.

En intégrant les valeurs des coefficients optimisés dans les équations linéaires, les valeurs des 3 hyperparamètres dans la couche de représentation sont déterminées par :

$$V_\theta = 0.5kl \tag{5.12}$$

$$c_{\min} = 0.06V_\theta \tag{5.13}$$

$$c_{\max} = V_\theta \tag{5.14}$$

Paramètres utilisés

Les paramètres du SNN sont donnés dans le tableau 5.1. Les paramètres neuronaux de la couche d'encodage sont communs à tout le manuscrit, ainsi que les $l = 10$ neurones utilisés dans la couche d'encodage pour représenter une dimension du vecteur d'entrée.

Dans nos expériences, un vecteur d'entrée consiste en un patch d'image de 4*4 pixels, donnant une dimension de $k = 16$. Nous utilisons la méthode précédemment exposée utilisant des coefficients génériques pour mettre à l'échelle les 3 hyperparamètres V_θ, $c_{\min}$ et $c_{\max}$ dépendant de la dimension des entrées. Ainsi la valeur du seuil de décharge V_θ des neurones de la couche de représentation v est déterminée à partir de l'équation 5.12, donnant $V_\theta = 0.5kl = 0.5 * 16 * 10 = 80.0$. Enfin il reste à fixer les trois paramètres du mécanisme homéostatique augmentant progressivement la valeur des poids synaptique latéraux de la couche de représentation v : la constante de temps τ_w, le niveau d'inhibition latéral initial $c_{\min}$ et final $c_{\max}$. La constante de temps est égale à $1/3$ du temps de la phase d'apprentissage, avec $P = 60000$ vecteurs d'entrées et une fenêtre temporelle d'encodage $T = 25$ ms, nous obtenons $\tau_w = 1/3 * P * T = 500$ s. Le niveau d'inhibition latéral initial $c_{\min}$ dans la couche v est déterminé à partir de l'équation 5.13 donnant $c_{\min} = 0.06V_\theta = 0.06 * 80.0 = 4.8$. Le niveau d'inhibition latéral final $c_{\max}$ est déterminé à partir de l'équation 5.14 donnant $c_{\max} = V_\theta = 80.0$.

Les autres hyperparamètres du SNN sont fixes, i.e. ne dépendent pas de la dimension des données d'entrées. Ces paramètres, e.g. les taux d'apprentissages, ont été déterminés par la méthode d'optimisation bayésienne TPE.

5.3.2 Métriques utilisées

Les métriques utilisées pour évaluer notre modèle WD-TCRL sont les mêmes que celles utilisées pour W-TCRL : erreur de reconstruction RMS, sparsité et incohérence. Nous les rappelons ci-dessous (voir section 4.3.2 pour plus de détails), avec des équations adaptées au fait que WD-TCRL apprend désormais les représentations dans les délais.

Erreur quadratique moyenne

Nous utilisons l'erreur RMS pour quantifier la différence entre un patch d'image d'entrée $\mathbf{a}_p$ et un patch reconstruit $\hat{\mathbf{a}}_p$ (5.15) :

TABLE 5.1 – Paramètres de WD-TCRL utilisés dans toutes les simulations

Paramètres neuronaux pour la couche d'encodage							
V_θ	τ_m	T_{refrac}					
0.5	10.0 ms	6 ms					
Paramètres neuronaux pour la couche de représentation							
V_θ	τ_m	T_{refrac}					
80.0	4.2 ms	6ms					
Paramètres synaptiques							
τ_x	τ_y	$d_{\min}$	$d_{\max}$				
4 ms	3 ms	0 ms	10 ms				
Paramètres des règles STDP							
λ	α^+	α^-	β^+	β^-	γ	σ	ϵ
0.41	0.08	0.16	0.05	0.45	0.37	5.0 ms	0.05
Paramètres du mécanisme homéostatique							
$c_{\min}$	$c_{\max}$	τ_w					
9.6	100.0	500 s					

$$RMS = \frac{1}{P}\sum_{p=1}^{P}\sqrt{\frac{1}{k}\sum_{i=1}^{k}(a_{i,p} - \hat{a}_{i,p})^2} \tag{5.15}$$

où k est le nombre de pixels d'un patch et P le nombre de patchs.

Rappelons que chaque $i \in \{1, 2, ..., k\}$ dimension d'un patch d'entrée est distribuée sur l neurones encodeurs dans la couche d'encodage. Chacun des $z \in \{1, 2, ..., l\}$ neurones encodeurs a une valeur préférentielle associée $\mu_z \in \{\mu_1, \mu_2, ..., \mu_l\}$. La couche d'encodage est connectée dans une relation tous-à-tous à la couche de représentation. Chaque neurone postsynaptique de la couche de représentation est ainsi connecté à $k * l$ neurones présynaptiques de la couche d'encodage.

Chaque dimension $i \in \{1, 2, ..., k\}$ du patch reconstruit $\hat{a}_{i,p}$ - correspondant dans notre cas à l'intensité d'un pixel - est décodée à partir de la distribution des l délais $\{d_1, d_2, ..., d_l\}$ associée. Dans le modèle W-TCRL d'apprentissage de représentations dans les poids synaptiques, la méthode de décodage est appliquée aux poids synaptiques (voir section 4.3.2). Par contraste, la méthode est appliquée aux délais dans WD-TCRL, les représentations étant stockées dans les délais.

Pour rappel, la première étape du processus de décodage consiste à transformer chaque valeur préférentielle $\mu_z \in \{\mu_1, \mu_2, ..., \mu_l\}$ des neurones présynaptiques en un angle θ_z (direction préférentielle) :

$$\theta_z = 2\pi\mu_z$$

Nous calculons ensuite les moyennes pondérées des coordonnées cartésiennes $\overline{x}_i$ et $\overline{y}_i$ à partir des angles θ_z (coordonnées angulaires) et des délais d_z^i associés au pixel i de la SBMU (coordonnées radiales).

$$\begin{aligned}
\text{avec } d_z^i &= d_{\text{sbmu},(i-1)l+z} \\
\overline{x_i} &= \frac{1}{\Delta_i} \sum_{z=1}^{l} d_z^i \cos(\theta_z) \\
\overline{y_i} &= \frac{1}{\Delta_i} \sum_{z=1}^{l} d_z^i \sin(\theta_z) \\
\Delta_i &= \sum_{z=1}^{l} d_z^i
\end{aligned}$$

Nous calculons un nouvel angle $\overline{\theta}$ à partir des coordonnées cartésiennes $\overline{x}$ et $\overline{y}$. Cet angle $\overline{\theta}$ correspond à la moyenne circulaire pondérée :

$$\overline{\theta_i} = \text{atan2}\left(-\overline{y_i}, -\overline{x_i}\right) + \pi$$

Nous finissons par une transformation inverse, transformant l'angle $\overline{\theta} \in [0, 2\pi]$ en l'intensité normalisée d'un pixel du patch reconstruit $\hat{a}_{i,p} \in [0, 1]$:

$$\hat{a}_{i,p} = \frac{\overline{\theta_i}}{2\pi}$$

Sparsité

La sparsité est donnée par :

$$\text{Sparsité} = \frac{1}{m} N_{\text{imp}}$$

où m est le nombre de neurones, et N_{imp} est le nombre d'impulsions émises par les m neurones sur la fenêtre temporelle d'encodage T pour un vecteur d'entrée.

Cohérence dans l'élection de la SBMU

L'incohérence de la SBMU est donnée par :

$$\text{Incohérence} = 1 - \frac{\text{sc}}{\text{sc} + \text{sp}} \tag{5.16}$$

où sc (sp) est le nombre total de SBMU cohérentes (paradoxales). On se rapportera à la section 4.3.2 pour la description de sc et sp.

5.3.3 Résultats pour MNIST

Les expériences ont été menées pour plusieurs capacités avec $m \in \{16, 64, 100, 256\}$ neurones dans la couche de représentation et donc m vecteurs codes soumis à apprentissage. Chaque expérience a été évaluée avec 10 simulations indépendantes. Pour une comparaison équitable avec l'état de l'art fixé par Tavanaei, Masquelier et al., 2018, nous n'avons pas effectué de validation croisée.

Les délais entre la couche d'encodage u et de représentation v sont initialisés aléatoirement dans l'intervalle [0,1] ms avec une gaussienne centrée sur 0.5 ms, d'écart-type de 0.2 ms. Les délais sont discrétisés avec une résolution de 0.1 ms égale au pas de temps $dt = 0.1$ ms des simulations. Les poids synaptiques sont initialisés à leurs valeurs maximale à 1.

Les entrées sont normalisées dans l'intervalle $[0.15, 0.85]$. Comme dans le chapitre 4, nous introduisons cette marge en raison de la projection de données d'entrées linéaires sur un espace circulaire (champs récepteurs) où les valeurs extrémales deviennent équivalentes (2 π est équivalent à 0 radian). En outre, MNIST est essentiellement composé de valeurs extrémales (pixels noir et blanc). Nous avons sélectionné aléatoirement un sous-ensemble de 15 000 et 1000 chiffres manuscrits pour la phase d'apprentissage et de test, respectivement. Des patchs de 4*4 pixels, extraits de chiffres manuscrits de 28*28 pixels, sont fournis comme vecteurs d'entrée au SNN. 60 000 patchs d'images sont donnés en entrée au SNN durant la phase d'apprentissage. Pour la phase de test, nous avons fixé l'inhibition latérale dans la couche de représentation v à c_{max} et désactivé la plasticité.

Les trois mesures de performances présentées dans la section 5.3.2 ont été utilisées pour évaluer le SNN : l'erreur de reconstruction RMS entre un vecteur d'entrée et un vecteur code, la sparsité de l'activité de la couche de représentation, et l'incohérence dans le processus auto-organisé d'élection de la SBMU.

Nous vérifions également le rôle complémentaire des règles d'adaptation des poids synaptiques en testant les performances 1) avec et 2) sans adaptation des poids.

Phase d'apprentissage

Dans le cas où seuls les délais sont appris et que les poids synaptiques ne sont pas adaptés, la décroissance de l'erreur de reconstruction RMS est marginale durant la phase d'apprentissage (voir Fig. 5.4). Le taux de SBMU paradoxales est supérieur à 80 %, y compris pour un seuil de tolérance Th fixé à 10 %. Cela indique que le SNN n'est pas capable d'extraire des représentations des motifs impulsionnels en entrée car les neurones sont très peu sélectifs. Ce manque de sélectivité perdure même avec un niveau d'inhibition latérale croissant, forçant une diminution du taux de neurones déchargeant dans la couche de représentation, comme indiqué avec les valeurs de sparsité décroissante.

Par contraste, en activant à la fois l'apprentissage des délais et des poids synaptiques, les différentes mesures de performances s'améliorent de façon consistante avec le nombre d'itérations d'apprentissage ainsi que pour un nombre de neurones $m \in \{16, 32, 64, 128, 256\}$ croissant dans la couche de représentation. Cela démontre empiriquement le rôle complémentaire de nos deux règles STDP adaptant délais et poids synaptiques. L'erreur de reconstruction RMS décroît, indiquant que le SNN apprend à compresser la distribution des données d'entrées, jusqu'à atteindre un plateau relativement stable dès 4000 itérations autour d'une erreur RMS d'environ 0.11 pour

$m \in \{32, 64, 128, 256\}$ neurones. La sparsité diminue également tendanciellement, indiquant que le pourcentage de neurones déchargeant dans la couche de représentation décroît en réponse à un vecteur d'entrée (voir Fig. 5.4).

Enfin la cohérence euclidienne de la SBMU décroît pour un seuil de décision Th = 5 % et Th = 10 %, indiquant une décroissance du nombre de SBMU paradoxales et ainsi une sélectivité croissante des champs récepteurs des neurones à une région spécifique de l'espace d'entrée. Autrement dit, le neurone déchargeant en premier est en moyenne un bon représentant du vecteur d'entrée.

La figure 5.5 montre que les vecteurs codes de la couche de représentation deviennent progressivement sélectifs à des orientations visuelles au fil des itérations d'apprentissage.

Phase de test

Nous évaluons à présent les performances du SNN sur la base de données de test pour MNIST. Les différences très marquées durant la phase d'apprentissage entre le cas où les poids sont adaptés et le cas où ils ne le sont pas se retrouvent dans la phase de test (voir Fig. 5.6). Ainsi sans adaptation des poids, la meilleure erreur de reconstruction RMS obtenue est de 0.17 (0.09 avec adaptation), 89 % de SBMU paradoxales (6 % avec adaptation) pour un seuil de décision Th à 5 %, et 82 % de SBMU paradoxales (2 % avec adaptation) pour $Th = 10\%$.

La meilleure erreur de reconstruction RMS est atteinte pour 64 et 128 neurones neurones avec une valeur de 0.09 (voir Tab. 4.2). Cela représente une amélioration relative de 47 % par rapport à l'état de l'art (voir Tab. 4.3) pour la base de données MNIST, légèrement inférieure aux performances obtenues avec le modèle W-TCRL (voir Tab. 5.4).

La figure 4.10 montre les vecteurs codes obtenus à la fin de la phase d'apprentissage pour un nombre de neurones $m \in \{16, 32, 64, 128, 256\}$ variable. Les vecteurs codes ont appris à devenir sélectifs à des orientations visuelles variées, avec peu de redondance entre les vecteurs codes.

Enfin la reconstruction des images à partir des vecteurs codes élus de façon auto-organisée par le SNN produit des chiffres manuscrits reconstruits de bonne qualité, comparables aux images originales (voir Fig. 5.8).

5.3.4 Résultats pour les images naturelles

De nouveau, les expériences ont été menées avec $m \in \{16, 64, 100, 256\}$ neurones dans la couche de représentation. Chaque expérience a été évaluée avec 10 simulations indépendantes. Pour une comparaison équitable avec l'état de l'art fixé par (Tavanaei, Masquelier et al., 2018), nous n'avons pas effectué de validation croisée.

L'initialisation du SNN pour la base d'images naturelles est la même que pour la base de données MNIST.

Les entrées sont normalisées dans l'intervalle $[0.05, 0.95]$. Des patchs de 4*4 pixels, extraits d'images naturelles de 512*512 pixels, sont fournis comme vecteurs d'entrée au SNN. 60 000 patchs d'images sont donnés en entrée au SNN durant la phase d'apprentissage. Pour la phase de test, nous avons fixé l'inhibition latérale dans la couche de représentation v à c_{max} et désactivé la plasticité.

Nous utilisons les mêmes trois mesures de performances pour évaluer le SNN : l'erreur de

reconstruction RMS entre un vecteur d'entrée et un vecteur code, la sparsité de l'activité de la couche de représentation, et l'incohérence dans le processus auto-organisé d'élection de la SBMU.

Nous vérifions également le rôle complémentaire des règles d'adaptation des poids synaptiques en testant les performances 1) avec et 2) sans adaptation des poids.

Phase d'apprentissage

Tout comme pour MNIST, les différences de performances entre le cas où les poids synaptiques sont adaptés et le cas où ils ne le sont pas persistent pour la base d'images naturelles (voir Fig. 5.9). Les différences sont toujours aussi marquées en termes de nombre de SBMU paradoxales, indiqué par la mesure d'incohérence. Cependant l'écart en termes d'erreur de reconstruction RMS est moins marqué. En outre, les performances en termes de RMS se rapprochent avec un nombre de neurones croissant.

La figure 5.10 montre que les vecteurs codes de la couche de représentation deviennent progressivement sélectifs à des motifs visuels au fil des itérations d'apprentissage.

Phase de test

Nous évaluons à présent les performances du SNN sur la base de données de test pour les images naturelles. L'erreur de reconstruction RMS atteint un plateau dès 32 neurones $m \in \{32, 64, 128, 256\}$ avec une valeur de 0.04 (voir Tab. 4.2). Ainsi la sparsité moyenne pour $m = 64$ neurones atteint une valeur de 0.03 indiquant que seulement 3 % des neurones de la couche de représentation sont actifs en moyenne. L'élection auto-organisée de la SBMU est réalisée avec jusqu'à 94 % de SBMU cohérentes pour un seuil de décision à 5 % et $m = 64$ neurones.

Cela représente une amélioration relative de 117 % par rapport à l'état de l'art (voir Tab. 4.3) pour la base de données d'images naturelles, équivalente à la performance obtenue par W-TCRL.

La figure 5.12 montre les vecteurs codes obtenus à la fin de la phase d'apprentissage pour un nombre de neurones $m \in \{16, 32, 64, 128, 256\}$ variable. Les vecteurs codes ont appris à devenir sélectifs à des orientations visuelles variées.

Enfin la reconstruction des images à partir des vecteurs codes des neurones élus de façon auto-organisée par le SNN produit des images naturelles reconstruites de très bonne qualité, comparables aux images originales (voir Fig. 5.13).

5.4 Synthèse

Nous avons exposé dans ce chapitre une nouvelle méthode pour apprendre conjointement les délais et les poids synaptiques afin d'extraire des représentations à partir de codes temporels. La méthode prend la forme de deux nouvelles règles STDP, une règle ciblant les délais, l'autre ciblant les poids.

La première règle STDP ciblant les délais synaptiques est capable d'apprendre les temps relatifs des impulsions dans les délais de transmission, créant une correspondance entre la dimension physique des entrées (le temps) et des paramètres plastiques (les délais). Cette règle intègre un module de Quantification Vectorielle (QV) visant à minimiser l'erreur de reconstruction et un module de régularisation favorisant les faibles valeurs de délais. Le module de QV minimise la

TABLE 5.2 – Résultats obtenus sur les bases de données MNIST et d'images naturelles pour un nombre de neurones $m \in \{16, 32, 64, 128, 256\}$.

Apprentissage poids	Sans apprentissage poids					Avec apprentissage poids				
Neurones m	16	32	64	128	256	16	32	64	128	256
RMS MNIST	0.17	0.18	0.17	0.21	0.17	0.11	0.10	0.09	0.09	0.10
RMS Nat	0.07	0.06	0.06	0.05	0.04	0.05	0.04	0.04	0.04	0.04
Sparsité MNIST	0.09	0.07	0.03	0.02	0.02	0.07	0.04	0.03	0.04	0.06
Sparsité Nat	0.07	0.03	0.02	0.01	0.04	0.13	0.09	0.06	0.05	0.04
Incoh. 5 % MNIST	0.96	0.95	0.93	0.90	0.89	0.24	0.07	0.06	0.16	0.32
Incoh. 5 % Nat	0.84	0.75	0.70	0.69	0.66	0.42	0.35	0.37	0.35	0.40
Incoh. 10 % MNIST	0.84	0.89	0.89	0.85	0.82	0.04	0.02	0.02	0.05	0.09
Incoh. 10 % Nat	0.72	0.62	0.59	0.58	0.48	0.23	0.17	0.20	0.20	0.21

TABLE 5.3 – Comparaison des meilleures erreurs de reconstruction RMS avec TAVANAEI, MASQUELIER et al., 2018.

Dataset	Tavanaei	W-TCRL	WD-TCRL
MNIST	0.17	$\mathbf{0.07} \pm 0.001$	$\mathbf{0.09} \pm 0.002$
Images naturelles	0.24	$\mathbf{0.04} \pm 0.001$	$\mathbf{0.04} \pm 0.001$

distance temporelle entre les délais et l'intervalle de temps entre une impulsion pré et postsynaptique. Cela a pour effet de maximiser la synchronie au niveau du terminal postsynaptique. Autrement dit, les impulsions asynchrones arrivent de manière synchrone après compensation des délais au niveau du neurone postsynaptique, activant un mode opératoire clé des neurones impulsionnels : la détection de coïncidences temporelles.

La deuxième règle STDP adapte les poids synaptiques. Son rôle est d'assigner des poids de pertinence aux caractéristiques en estimant leurs variabilité temporelle. La variabilité temporelle peut être estimée en tirant profit de l'écart aux centroïdes (ou espérance empirique locale) stockés dans les délais par la première règle STDP. Notre deuxième règle STDP utilise la variabilité temporelle pour distinguer les dimensions pertinentes des dimensions non pertinentes. Une variabilité temporelle élevée (faible) induit une petite (grande) valeur de poids synaptique et donc une activité postsynaptique faible (élevée), modulant ainsi le pouvoir causal d'impulsions présynaptiques sur l'émission d'une impulsion postsynaptique. Cela permet en dernière analyse aux neurones postsynaptiques de se verrouiller efficacement sur des données densément positionnées (*clusters*) dans l'espace d'entrée. Cette règle STDP adaptant les poids fonctionne en synergie avec la règle adaptant les délais, comme démontré expérimentalement. Elle permet de grandement faciliter le processus d'apprentissage de représentations dans les délais, comme indiqué par une erreur de reconstruction RMS et un nombre de SBMU paradoxales significativement plus faibles.

Le SNN instancie un apprentissage compétitif entre les neurones de la couche de représentation à l'aide de connexions latérales inhibitrices. Nous avons ajouté un mécanisme permettant initialement de recruter plusieurs neurones durant la phase d'apprentissage pour accélérer la convergence du SNN, puis de progressivement recruter un nombre décroissant de neurones pour

TABLE 5.4 – Comparaisons des performances des modèles W-TCRL et WD-TCRL

Modèle	W-TCRL					WD-TCRL				
Neurones m	16	32	64	128	256	16	32	64	128	256
RMS MNIST	0.11	0.09	0.07	0.07	0.07	0.11	0.10	0.09	0.09	0.10
RMS Nat	0.05	0.04	0.04	0.04	0.04	0.05	0.04	0.04	0.04	0.04
Sparsité MNIST	0.19	0.12	0.05	0.02	0.01	0.07	0.04	0.03	0.04	0.06
Sparsité Nat	0.15	0.08	0.04	0.03	0.01	0.13	0.09	0.06	0.05	0.04
Incoh. 5 % MNIST	0.77	0.10	0.03	0.02	0.02	0.24	0.07	0.06	0.16	0.32
Incoh. 5 % Nat	0.46	0.27	0.10	0.03	0.01	0.42	0.35	0.37	0.35	0.40
Incoh. 10 % MNIST	0.71	0.03	0.01	0.01	0.01	0.04	0.02	0.02	0.05	0.09
Incoh. 10 % Nat	0.27	0.08	0.01	0.004	0.003	0.23	0.17	0.20	0.20	0.21

spécialiser et décorréler les vecteurs codes appris. Ce comportement est obtenu avec un mécanisme homéostatique augmentant la magnitude des poids latéraux pour faire dynamiquement passer le circuit d'un comportement K-WTA à WTA, augmentant ainsi la sparsité de l'activité dans la couche de représentation.

Comparativement au modèle W-TCRL, nous atteignons des performances équivalentes en termes d'erreur de reconstruction RMS sur la base de donnée d'images naturelles et légèrement inférieures pour MNIST. Notre modèle WD-TCRL donne des performances sensiblement meilleures en termes de sparsité et d'incohérence pour des petites tailles de réseau (16 et 32 neurones). Néanmoins la tendance s'inverse pour des tailles de réseau plus importantes.

A notre connaissance, il s'agit du premier travail visant à apprendre des représentations visant une faible erreur de reconstruction dans l'espace des délais, i.e. dans le domaine temporel, plutôt que de les apprendre dans l'espace des poids. Nos résultats expérimentaux valident l'intérêt de notre approche en atteignant une erreur de reconstruction améliorant significativement l'état de l'art fixé par TAVANAEI, MASQUELIER et al., 2018 sur les bases de données MNIST et d'images naturelles, avec 47 % et 117 % d'amélioration relative, respectivement. Néanmoins les améliorations relatives pour la base de données d'images naturelles sont à nuancer, dû à la taille différente des imagettes utilisée, comme discuté dans le chapitre 4.

Un autre intérêt de ce travail provient du fait que des processeurs neuromorphiques comme Loihi (DAVIES et al., 2018) permettent d'implémenter des règles de plasticité ciblant les délais de transmission. Les règles STDP locales proposées dans ce chapitre ainsi que l'utilisation de codes temporels rendent ainsi l'architecture compatible avec une implémentation neuromorphique très efficace.

Enfin nous espérons que ce travail contribuera à considérer les délais comme des paramètres plastiques utiles pour l'apprentissage, ce qui demeure un axe de recherche encore largement inexploré. En outre les délais peuvent permettre de traiter des motifs impulsionnels se déployant sur plusieurs échelles temporelles, e.g. le cerveau maintient avec une grande précision des délais sur trois ordres de grandeurs allant de 0.1 ms à 40 ms.

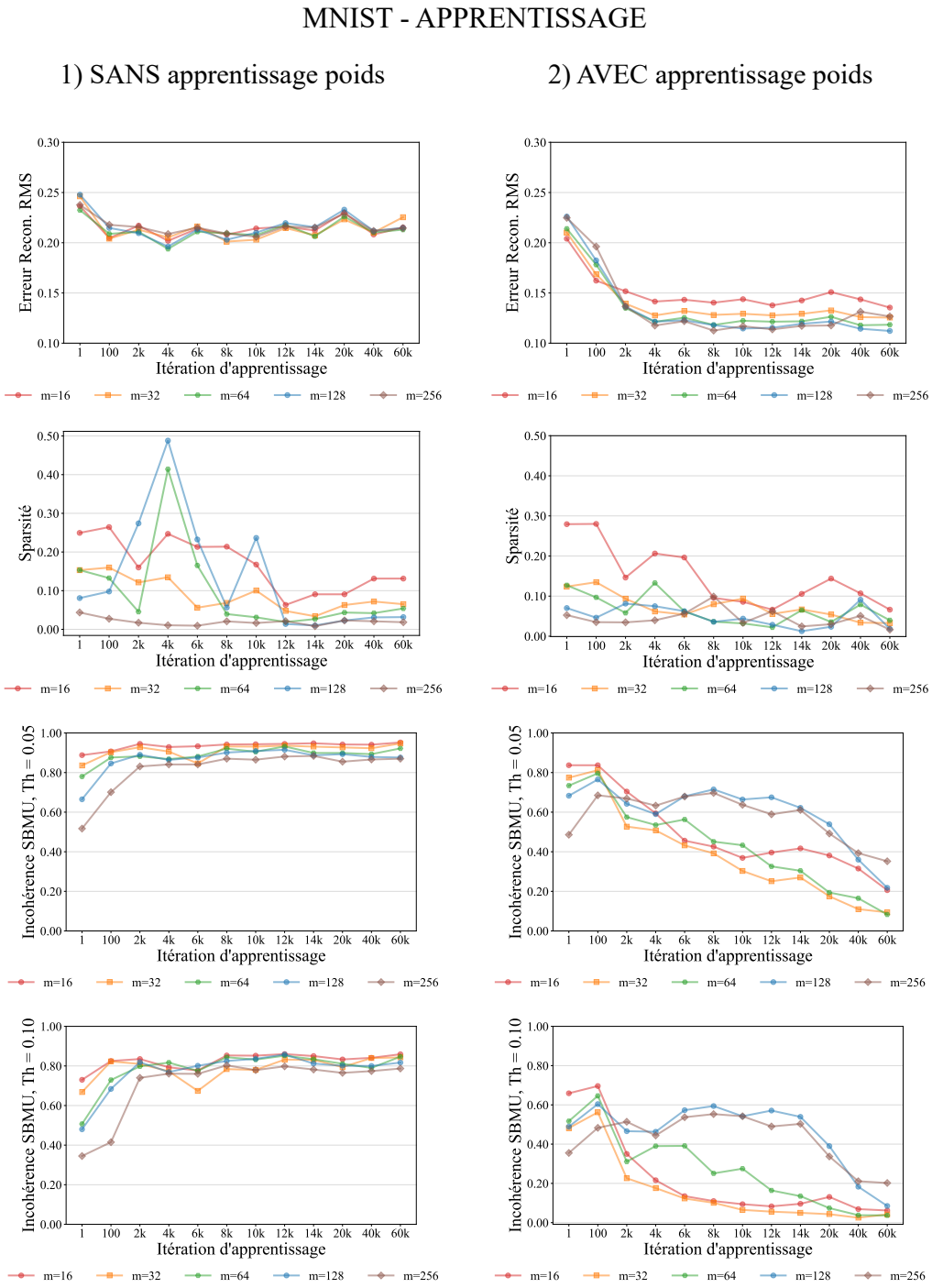

FIGURE 5.4 – Performances pour MNIST durant la phase l'apprentissage. Performances du SNN après 1, 100,..., 60k itérations pour $m \in \{16, 32, 64, 128, 256\}$ neurones, en termes d'erreur de reconstruction RMS, de sparsité, et d'incohérence dans l'élection auto-organisée de la SBMU pour un seuil de décision Th=5 % et Th=10 %.

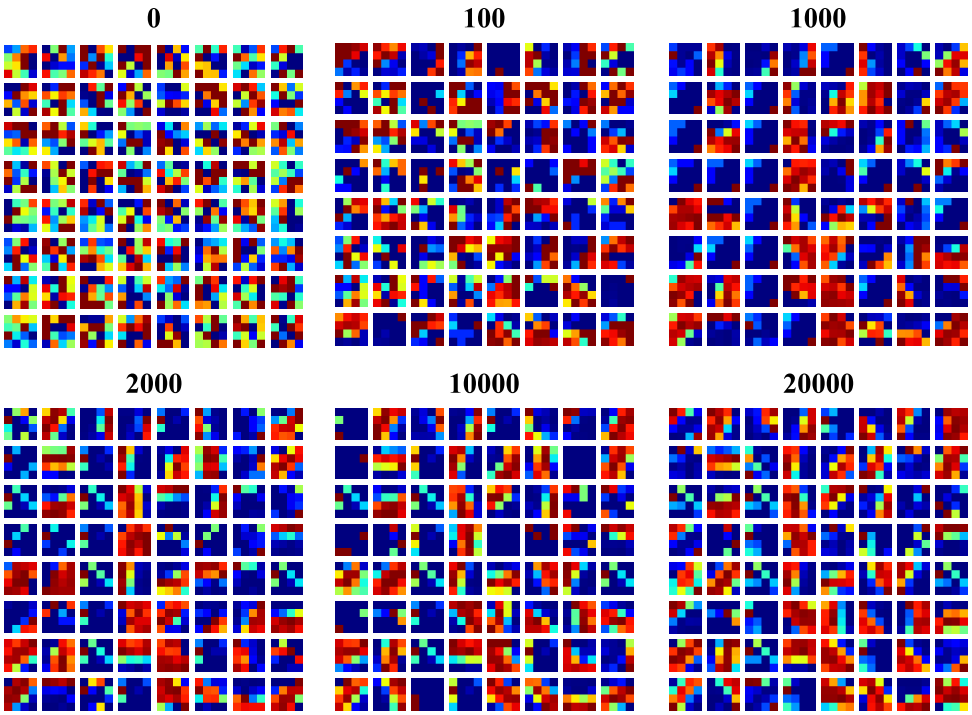

FIGURE 5.5 – Phase d'apprentissage sur MNIST : évolution des vecteurs codes après 0, 100 ,..., 20k itérations d'apprentissage pour $m = 64$ neurones. Le gradient rouge-bleu indique les valeurs maximales-minimales de chaque pixel reconstruit.

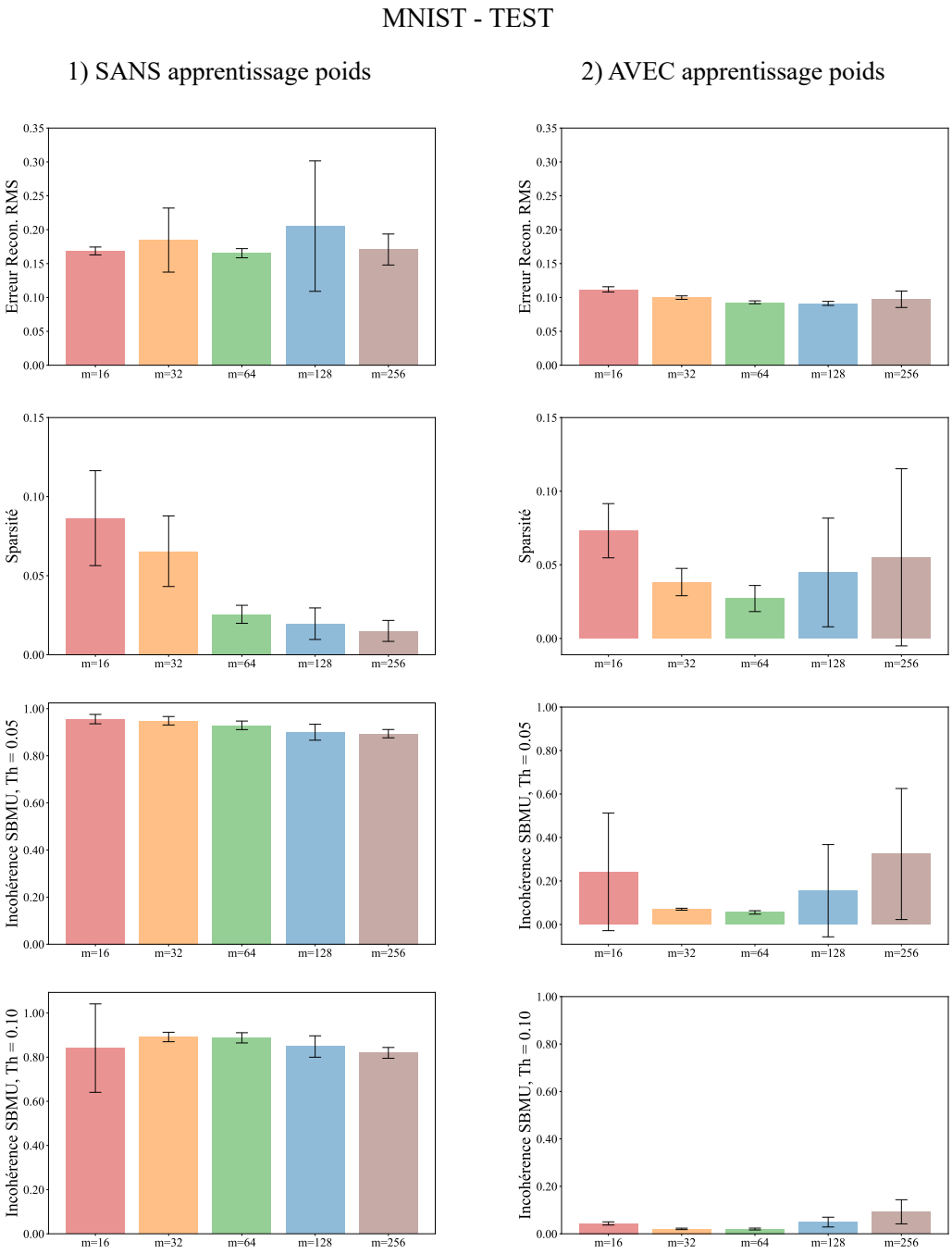

FIGURE 5.6 – Performances pour MNIST durant la phase de test. Performances pour $m \in \{16, 32, 64, 128, 256\}$ neurones, en termes d'erreur de reconstruction RMS, de sparsité, et de cohérence dans l'élection de la SBMU pour un seuil de décision Th=5 % et Th=10 %. Les barres indiquent l'écart-type.

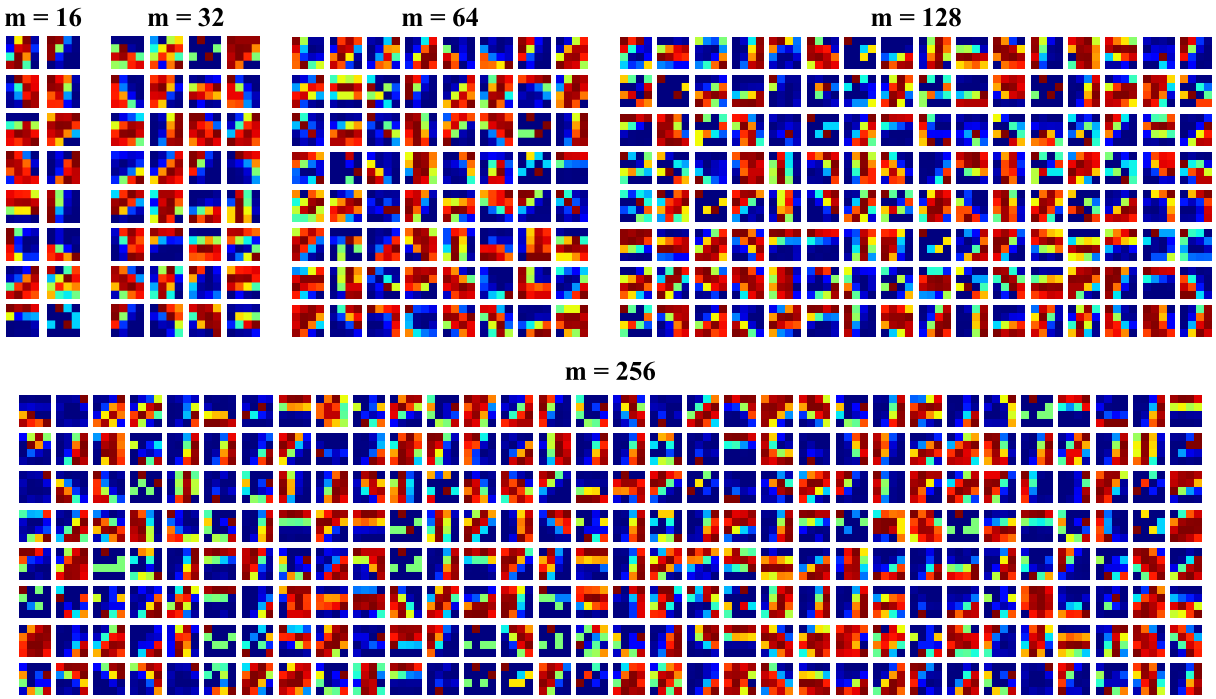

FIGURE 5.7 – Après la phase d'apprentissage sur MNIST : vecteurs codes appris dans la couche de représentation v pour $m \in \{16, 32, 64, 128, 256\}$ neurones. Le gradient rouge-bleu indique les valeurs maximales-minimales de chaque pixel reconstruit.

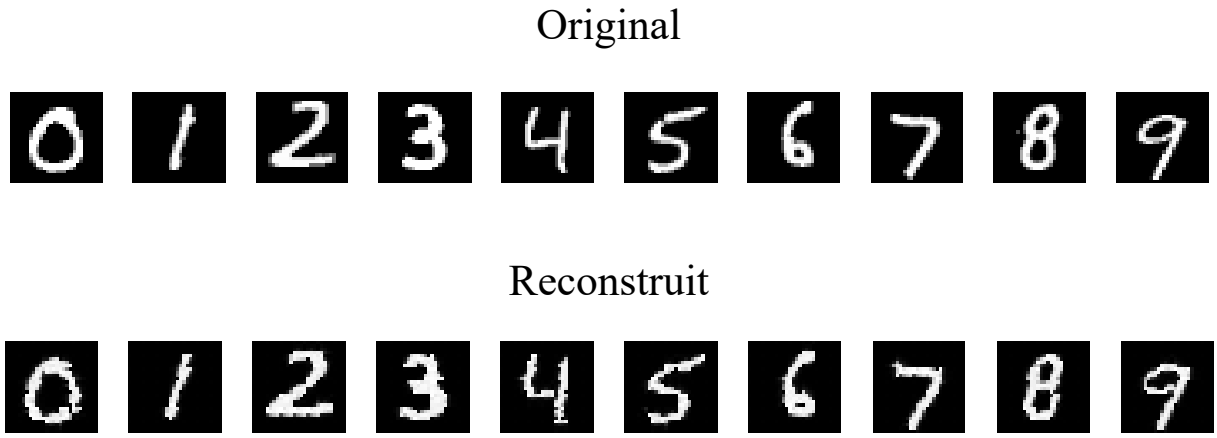

FIGURE 5.8 – Ensemble d'images de MNIST présentées en entrée du SNN et reconstruites par les vecteurs codes des neurones de la couche de représentation pour $m = 64$ neurones.

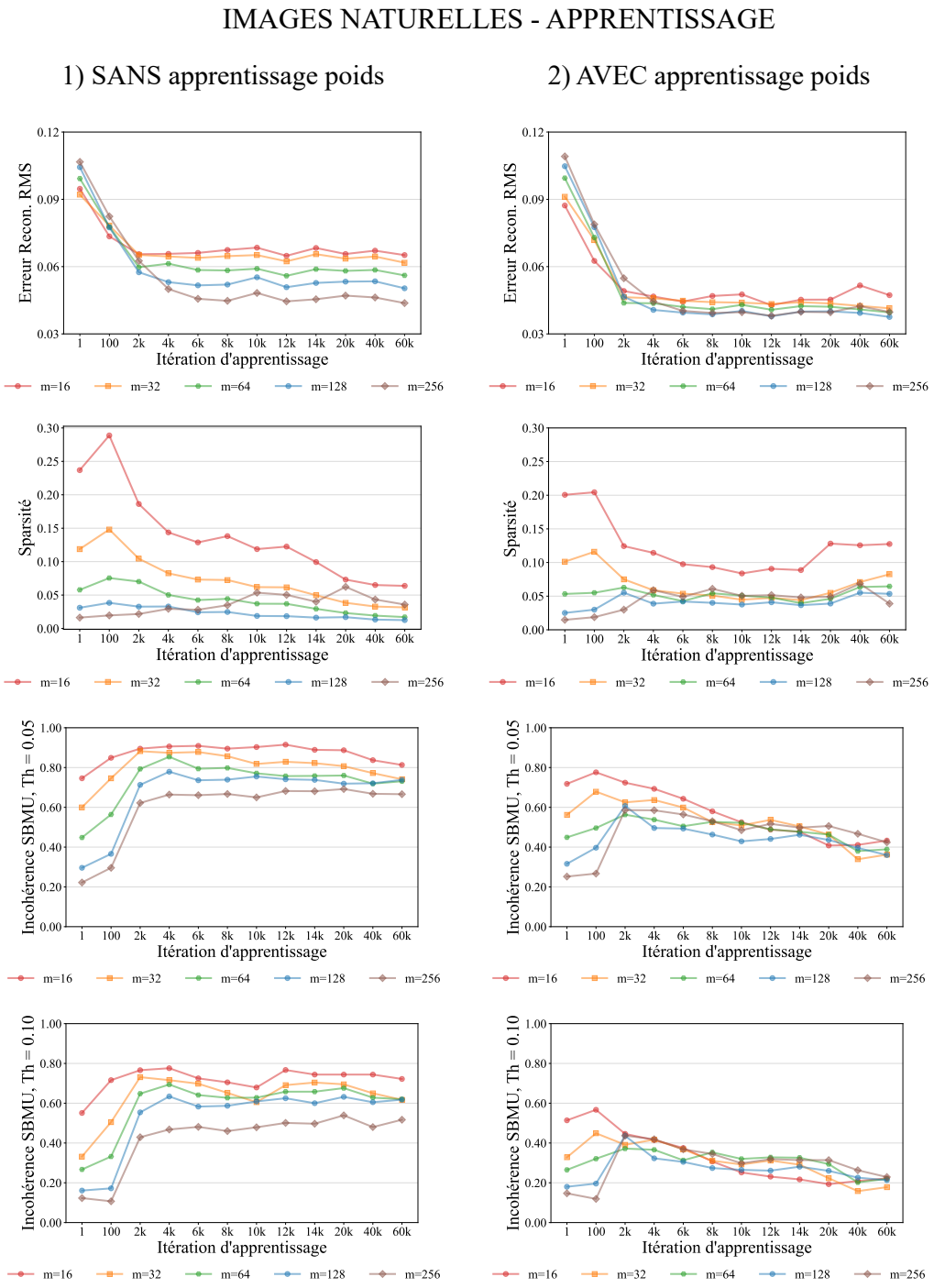

FIGURE 5.9 – Performances pour les images naturelles durant la phase l'apprentissage. Performances après 1, 100,..., 60k itérations pour $m \in \{16, 32, 64, 128, 256\}$ neurones, en termes d'erreur de reconstruction RMS, de sparsité, et de cohérence dans l'élection auto-organisée de la SBMU pour un seuil de décision Th=5 % et Th=10 %.

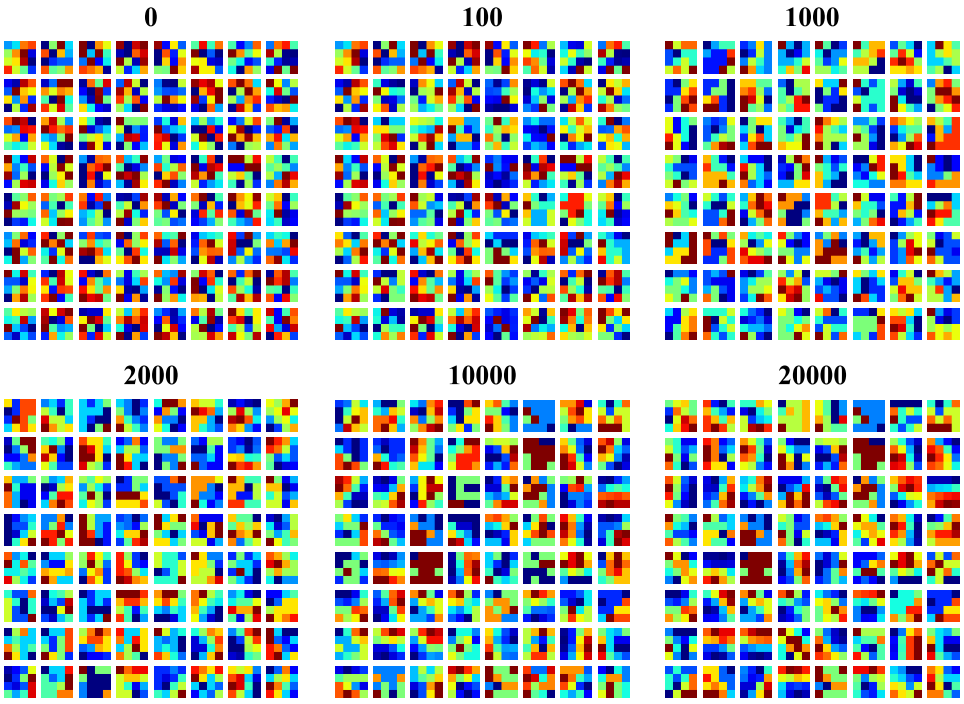

FIGURE 5.10 – Phase d'apprentissage sur des images naturelles : évolution des vecteurs codes après 0, 100 ,..., 20k itérations d'apprentissage pour $m = 64$ neurones. Le gradient rouge-bleu indique les valeurs maximales-minimales de chaque pixel reconstruit.

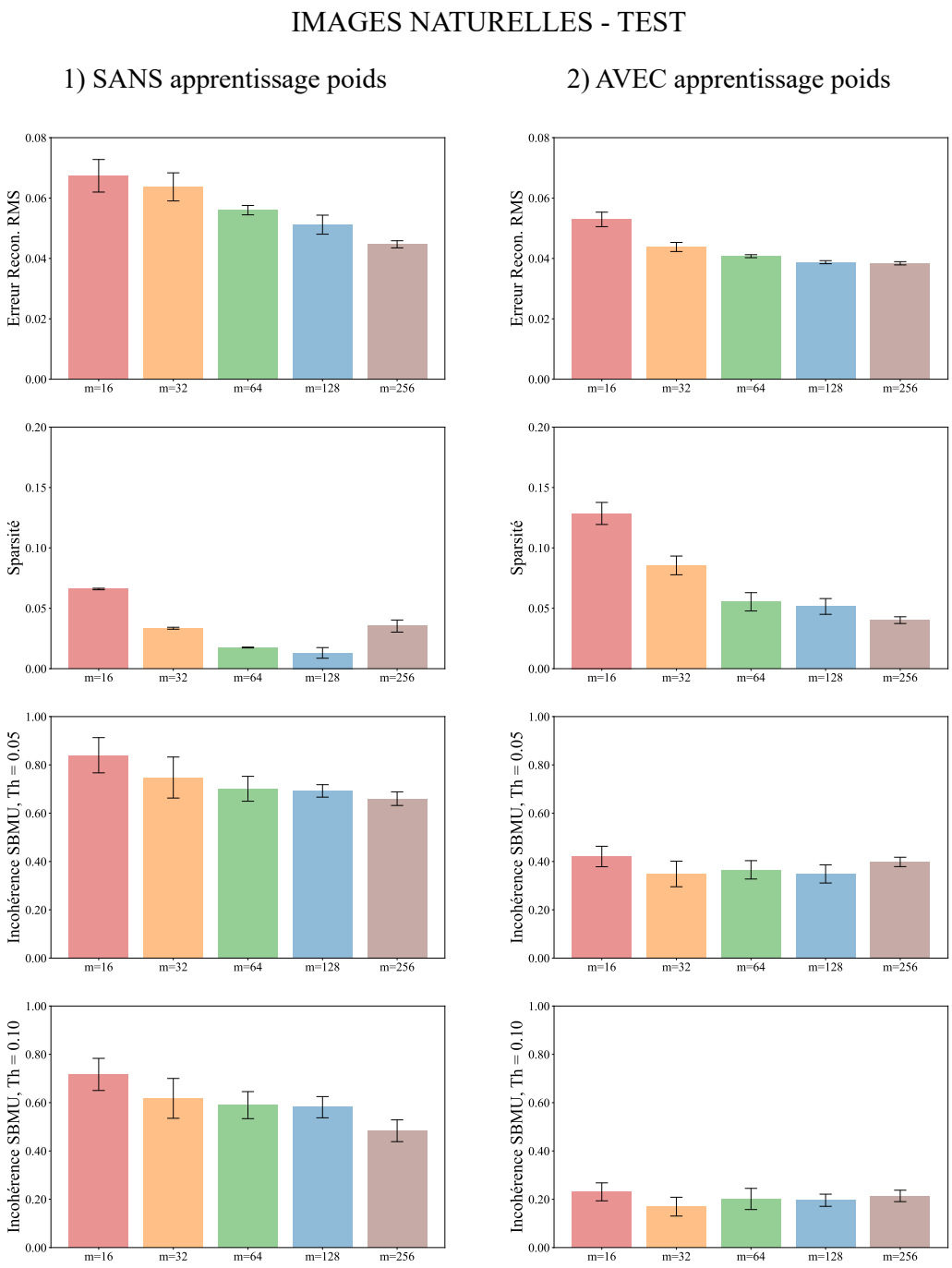

FIGURE 5.11 – Performances pour les images naturelles durant la phase de test. Performances pour $m \in \{16, 32, 64, 128, 256\}$ neurones, en termes d'erreur de reconstruction RMS, de sparsité, et de cohérence dans l'élection de la SBMU pour un seuil de décision Th=5 % et Th=10 %. Les barres indiquent l'écart-type.

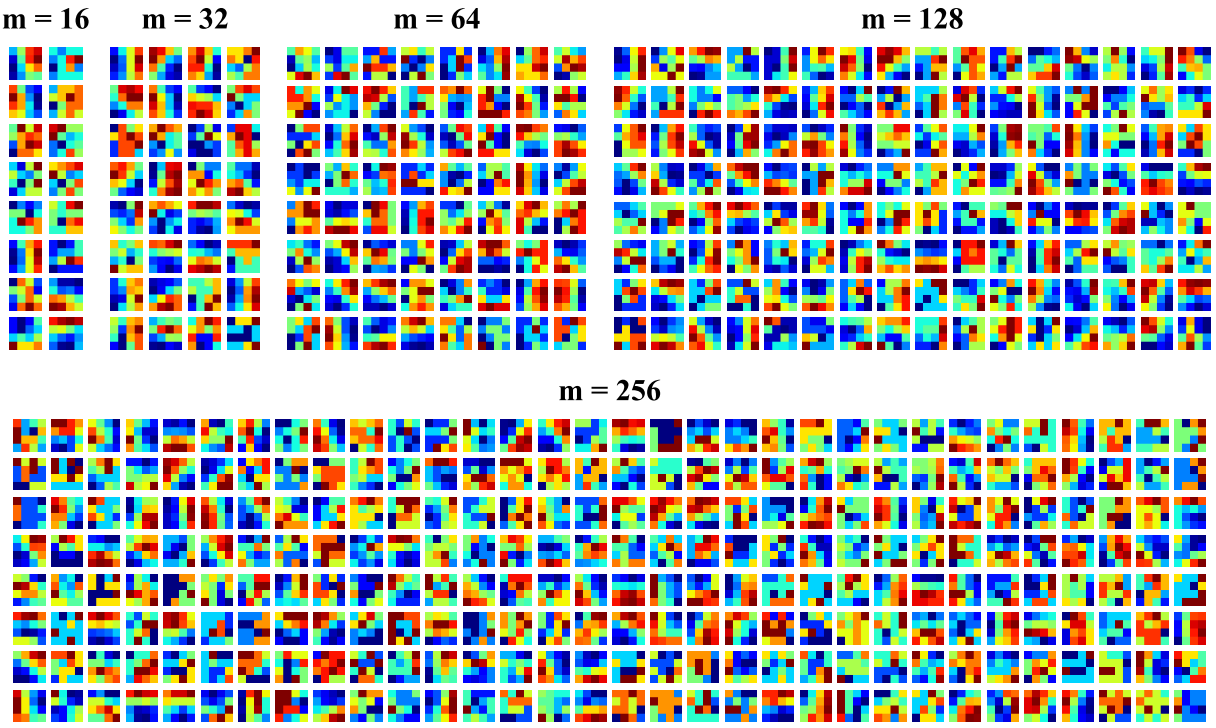

FIGURE 5.12 – Après la phase d'apprentissage sur les images naturelles : vecteurs codes appris dans la couche de représentation v pour $m \in \{16, 32, 64, 128, 256\}$ neurones. Le gradient rouge-bleu indique les valeurs maximales-minimales de chaque pixel reconstruit.

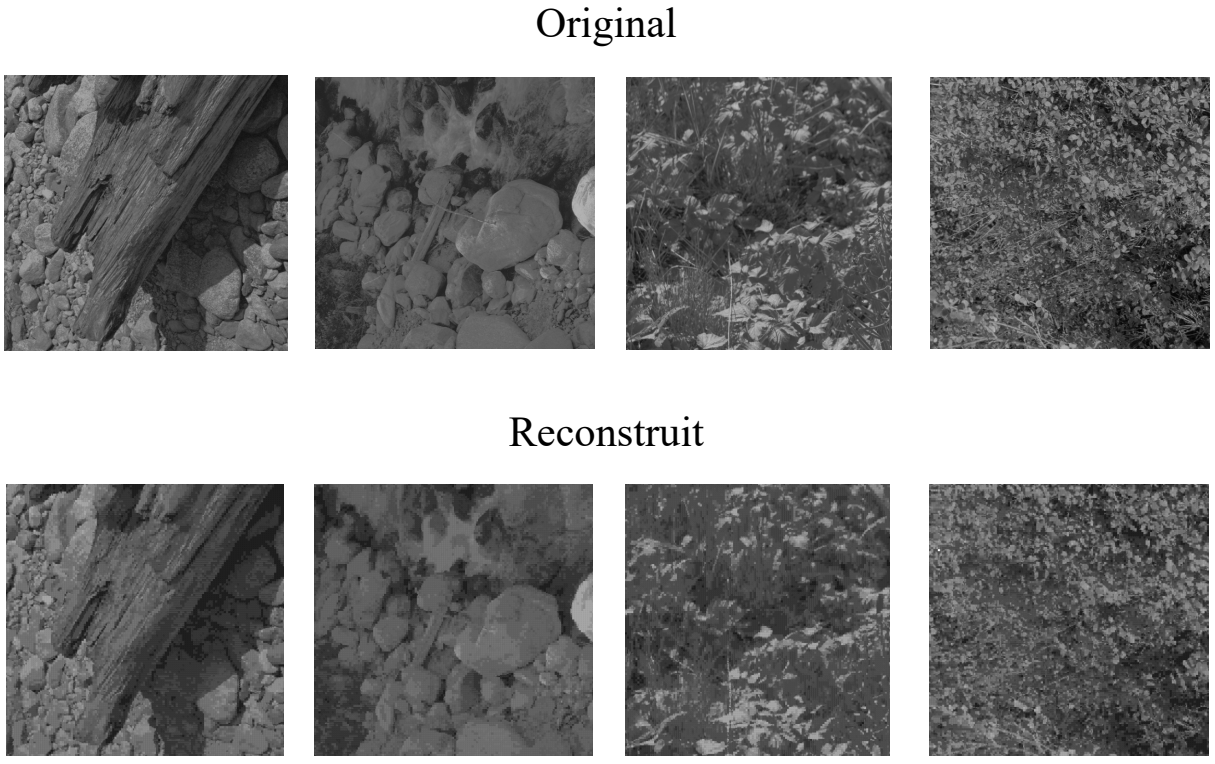

FIGURE 5.13 – Ensemble d'images naturelles originales présentées en entrée du SNN et reconstruites par les vecteurs codes des neurones de la couche de représentation pour $m = 64$ neurones.

Chapitre 6

Une carte auto-organisatrice impulsionnelle

Sommaire

6.1 Introduction

Les modèles W-TCRL et WD-TCRL sont capables d'extraire efficacement des représentations de codes temporels en apprenant des centroïdes dans l'espace des paramètres. L'apprentissage de centroïdes est essentiellement du fait d'un module clé au coeur de nos nouvelles règles STDP : un module de Quantification Vectorielle (QV) adapté au traitement de codes temporels. Les paramètres (et les méthodes) considérés pour le stockage de centroïdes diffèrent entre ces deux modèles. Tandis que W-TCRL apprend des centroïdes dans les poids synaptiques, WD-TCRL apprend des centroïdes dans les délais et donc dans un espace temporel. Contrairement aux poids synaptiques, les délais opèrent intrinsèquement dans la dimension temporelle. Ainsi le modèle WD-TCRL facilite le processus d'apprentissage en créant une correspondance entre la dimension physique temporelle dans laquelle se déploient les motifs impulsionnels et les paramètres soumis à

apprentissage que sont les délais. En outre l'adaptation des délais par le module de QV active un des modes opératoires clé des neurones impulsionnels : la détection de coïncidences temporelles.

Nous souhaitons augmenter le modèle WD-TCRL en le dotant d'une capacité à générer une carte topographiquement ordonnée. Générer des cartes topographiquement ordonnées présente plusieurs intérêts :

— Les cartes topographiquement ordonnées sont ubiquitaires dans le cortex sensoriel (Graziano et Aflalo, 2007 ; Kandel et al., 2000 ; Moser et al., 2014 ; Ohki et al., 2006), les élevant au niveau de principe organisateur et computationnel. La génération de cartes topographiquement ordonnées permet d'étendre le principe de localité au niveau du voisinage des neurones. En effet, projeter un espace d'entrée de haute dimension sur une surface de basse dimension - la surface du cortex - permet de minimiser le coût en termes de connexions synaptiques entre les neurones, de telle sorte que des calculs opérés sur des points de données proches dans la variété des données d'entrées puissent être réalisés localement dans le cortex (Chklovskii et Koulakov, 2004 ; Durbin et Mitchison, 1990).

— Ce principe de localité du calcul entre neurones partageant des représentations proches peut potentiellement être exploité par les processeurs neuromorphiques pour augmenter leurs efficience. Les processeurs neuromorphiques imitent l'organisation du cortex par leurs architectures massivement parallèles implémentant des centaines de cœurs neurosynaptiques, où mémoire (synapse) et calcul (neurones) sont co-localisés (Akopyan et al., 2015 ; Davies et al., 2018). Ce principe de localité peut permettre par exemple d'éviter de coûteuses communications à longue distance entre cœurs neurosynaptiques.

— Une hypothèse communément admise (hypothèse de variété) en apprentissage automatique et en neurosciences computationnelles est que des points de données proches dans la variété de données partagent la même valeur de variable catégorielle (Bengio et al., 2013). Autrement dit, des variations locales dans la variété de données tendent à préserver la catégorie d'un objet. De façon complémentaire, passer d'une catégorie à une autre catégorie implique généralement de passer par une région à faible densité de probabilité, séparant ainsi une catégorie d'une autre. Cela signifie que des catégories peuvent être découvertes par l'extraction de la structure statistique latente des données d'entrées. La génération de carte topographiquement ordonnée permet d'extraire la structure latente des données en générant des *clusters* de neurones spatialement ségrégués correspondant à une même catégorie, leurs représentations étant proches dans la variété de données.

Le modèle canonique pour la génération de carte topographiquement ordonnée est la *Self-Organizing Map* (SOM) de Kohonen (Kohonen, 2013). Ce modèle intègre un module de QV introduisant un apprentissage compétitif pour la représentation de l'entrée courante, et une fonction de voisinage introduisant un apprentissage coopératif entre neurones. Le module de QV permet d'apprendre des centroïdes et de projeter un espace d'entrée continu sur un espace de sortie discret, composé d'un ensemble finis de neurones et donc de vecteurs codes. La fonction de voisinage permet de projeter des distances entre des points de données d'entrées sur un espace de sortie discret, composé des positions des neurones dans la carte. La fonction de voisinage module le taux d'apprentissage des neurones en fonction de leurs distance avec la BMU (pour un rappel voir la section 2.2.1) dans la carte. Plus la distance entre un neurone et la BMU est grande, plus le taux d'apprentissage de ce neurone est faible. Par contraste, plus la distance entre un neurone et la BMU est faible, plus le taux d'apprentissage de ce neurone est élevé. Le rayon de voisinage et le taux d'apprentissage diminuent de façon monotone de façon à progressivement stabiliser et spécialiser l'organisation de la carte et les vecteurs codes obtenus, respectivement. Ainsi

durant le processus d'apprentissage, des neurones spatialement proches se mettent à progressivement partager des représentations proches dans l'espace d'entrée. Ce processus local permet progressivement de générer une carte topographiquement ordonnée à un niveau global. Ainsi en se déplaçant dans le continuum de l'espace d'entrée, les neurones qui répondent le mieux à ces entrées se trouvent sur un continuum spatial.

Cependant la SOM n'est pas adaptée au traitement des données événementielles et asynchrones que sont les impulsions. Pour répondre à ce problème, plusieurs modèles impulsionnels cherchant à reproduire les fonctions de la SOM ont été développés.

Ruf et Schmitt, 1998 ont développé un modèle influent composé de deux couches de neurones *Integrate and Fire* (IF) et utilisant un codage temporel. La première couche transforme des vecteurs d'entrées continus en impulsions neuronales avec un encodage temporel. Le codage temporel utilisé est réalisé avec un unique neurone (*first spike latency*) plutôt que via un codage par population plus efficient. Ces impulsions atteignent la deuxième couche de neurones représentant la SOM. Les neurones intègrent les impulsions reçues, pondérées par la valeur des poids synaptiques. Comme il s'agit de neurones IF sans fuite du potentiel membranaire qu'on trouve dans les LIF, cela revient à calculer une somme pondérée correspondant à une mesure de similarité entre les vecteurs codes et le vecteur d'entrée courant. Le premier neurone qui décharge est celui dont le vecteur code est le plus similaire au vecteur d'entrée, et correspond à la SBMU. Le vecteur code de la SBMU est adapté par une règle hebbienne sensible aux temps d'impulsions. Plus l'impulsion est émise tôt, plus la variation du poids synaptique induite par la règle hebbienne est élevée. Par contraste, plus l'impulsion est émise tardivement, plus la variation du poids synaptique est faible. Une contrainte spatiale est ajoutée à l'aide d'un noyau d'interaction latéral dans le but de générer une carte topographiquement ordonnée. Le noyau d'interaction consiste en des connexions excitatrices à courte distance et des connexions inhibitrices à longue distance. Les connexions excitatrices favorisent l'apprentissage des neurones voisins à la SBMU en les faisant décharger dans une proximité temporelle, tandis que les connexions inhibitrices retardent les temps d'impulsions des neurones plus distants, engendrant un faible apprentissage. L'interaction entre la règle d'adaptation des poids synaptiques basée sur les temps d'impulsions et le noyau d'interaction latérale agissant sur les temps d'impulsions permet ainsi de générer une carte topographiquement ordonnée. Enfin, de façon similaire à la SOM de Kohonen, le taux d'apprentissage est réduit de façon monotone au fil des itérations d'apprentissage, ainsi que la fonction de voisinage. Au lieu de réduire le rayon de voisinage comme dans la SOM de Kohonen, le niveau d'inhibition latéral est augmenté de telle sorte à obtenir asymptotiquement un comportement WTA, dans lequel seul le vecteur code de la SBMU est actualisé. Les résultats expérimentaux démontrent une capacité à générer une carte topographiquement ordonnée pour des données synthétiques en une et deux dimensions. Cependant ce modèle n'utilise pas de règle de type STDP, spatialement et temporellement locale. Une horloge globale est utilisée, coûteuse et biologiquement peu plausible. En outre, l'erreur de quantification, propriété clé de la SOM, n'est pas évaluée, ni sur les données synthétiques, ni sur des bases de données réelles et de plus haute dimension. Enfin, la capacité de catégorisation n'est pas non plus évaluée.

Veredas et al., 2008 ont développé une architecture de 3 couches de neurones IF connectés de façon ascendante. Le SNN est équipé d'une règle de type STDP adaptant les poids synaptiques pour produire une carte sélective à l'orientation. Ce modèle n'inclut pas d'interactions latérales permettant d'instancier un apprentissage compétitif et/ou coopératif entre neurones. La production de carte ordonnée est due aux propriétés des champs récepteurs et à leurs conditions initiales, e.g. la taille des champs récepteurs ou encore l'initialisation des poids synaptiques

en fonction de la distance dans la carte. Des expériences sur des bases de données variées n'ont pas été menées. Ainsi il est probable que ce modèle nécessite un réglage fin des hyperparamètres pour générer des cartes ordonnées à partir de distributions de données variées, ne rendant pas le modèle suffisamment générique. En outre, les capacités de QV et de catégorisation ne sont pas non plus évaluées.

Pham et al., 2007 proposent un SNN opérant avec un code temporel. Le codage temporel utilisé est à nouveau réalisé avec un unique neurone (*first spike latency*) plutôt que via un codage par population plus efficient. Le SNN est constitué de deux couches connectées par des synapses dotées de délais de transmission plastiques. Contrairement au modèle WD-TCRL, ce modèle n'intègre pas de poids synaptiques soumis à apprentissage. Les délais sont adaptés durant la phase d'apprentissage avec une règle hebbienne. L'adaptation des délais permet aux neurones de progressivement agir comme des détecteurs de coïncidences. Cependant la règle d'adaptation des délais n'est pas temporellement locale, i.e. nécessite une horloge globale. Pour un vecteur d'entrée donné le premier neurone déchargeant tend à être son meilleur représentant, car il tend à recevoir un nombre élevé d'impulsions synchrones. Cependant la SBMU n'est pas déterminée localement par les neurones. Une fonction de voisinage module le taux d'apprentissage des délais des neurones de la carte en fonction de leurs distance spatiale à la SBMU. Les performances de l'architecture ont été testées pour une tâche de *clustering* de séries temporelles. Cependant ni la capacité à générer une carte ordonnée, ni la capacité de QV n'ont été évaluées.

De façon plus proche du travail de Ruf et Schmitt, 1998, Hazan et al., 2020 ont développé un SNN à 3 couches. La première couche correspond à un encodage en taux de décharge (plutôt que temporel) projetant l'activité sur la deuxième couche de représentation (la SOM). Une règle STDP adapte les poids synaptiques entre ces deux couches. La couche de représentation est connectée de façon bidirectionnelle avec une couche inhibitrice (troisième couche). Le rôle de cette couche inhibitrice est d'imposer une contrainte topologique en inhibant les neurones voisins au premier neurone déchargeant (SBMU) dans la couche de représentation. Le niveau d'inhibition croît avec une distance à la SBMU croissante. En outre, le profil d'inhibition est initialement faible de telle sorte à accélérer l'apprentissage (K-WTA), puis croît au fil des itérations d'apprentissage pour spécialiser les vecteurs codes obtenus. Le modèle montre une capacité à *clusteriser* des catégories dans la carte pour la base de données MNIST et Atari Breakout. Les performances sont également évaluées pour une tâche de catégorisation, en testant plusieurs schèmes non-supervisés d'étiquetages de neurones. Cependant le modèle est basé sur un codage par taux de décharge peu efficient. La capacité à générer des cartes topographiquement ordonnées et à réaliser de la QV n'est pas évaluée.

Rumbell et al., 2014 proposent un modèle de SOM impulsionnelle pouvant être considéré comme une variante améliorée de Ruf et Schmitt, 1998. Les variations proviennent de plusieurs éléments. L'encodage temporel est réalisé par une population de neurones LIF plutôt que par un unique neurone IF. Des interactions excitatrices-inhibitrices génèrent un phénomène de verrouillage de phase dans la couche d'encodage permettant de présenter des entrées de façon continue au SNN, i.e. sans avoir besoin de réinitialiser le réseau comme dans Ruf et Schmitt, 1998. En outre, une règle STDP spatialement et temporellement locale adapte les poids synaptiques entre les deux couches. Le modèle démontre une capacité à générer des cartes ordonnées à partir de distribution de données en deux dimensions ainsi qu'une capacité d'adaptation dynamique à des distributions de données non-stationnaires. Le modèle montre aussi une capacité de catégorisation, avec des performances testées sur les bases de données IRIS et WBCD. Cependant une des fonctions cruciale de la SOM, la QV, n'est pas évaluée. Les auteurs ne présentent

pas non plus de méthode explicite permettant de décoder la distribution des poids synaptiques apprise. En outre, les auteurs ne proposent pas de méthode pour régler l'intervalle de variation des poids synaptiques entre la couche d'encodage et de représentation (SOM) en fonction de la dimension des données d'entrées. Le modèle nécessite ainsi un réglage manuel de l'intervalle de variation des poids suivant la dimension des données d'entrée, limitant son caractère générique.

Ce chapitre propose un modèle de SOM impulsionnelle que nous nommons *Self-Organizing Temporally Coded Representation Learning* (SO-TCRL). Le modèle SO-TCRL augmente les capacités de QV du modèle WD-TCRL avec une nouvelle capacité de génération de carte topographiquement ordonnée. Par contraste avec la SOM de Kohonen (Kohonen, 2013), la SO-TCRL ne nécessite ni de réduire le rayon de voisinage, ni de réduire le taux d'apprentissage durant la phase d'apprentissage. Le fait que ces hyperparamètres soient constants permet au modèle d'apprendre en continu comme dans le modèle DSOM (Rougier et Boniface, 2011). Certains modèles connexes précédemment exposés montrent une capacité à générer des cartes ordonnées, mais aucun ne démontre de capacité de QV. Notre modèle est ainsi à notre connaissance la première SOM impulsionnelle conjuguant ces deux fonctions clés de la SOM de Kohonen (Kohonen, 2013). Le modèle SO-TCRL est basé sur les règles STDP spatialement et temporellement locales du modèle WD-TCRL, augmentées par un nouveau neuromodulateur.

La SO-TCRL relâche la contrainte asymptotique de comportement WTA intégrée dans le modèle WD-TCRL. Contrairement aux autres modèles de SOM impulsionnelles présentés, la génération de carte ordonnée n'est pas due à un profil d'excitation-inhibition latéral agissant sur les temps d'impulsions des neurones relativement à la position de la SBMU. Plutôt que de moduler l'apprentissage des neurones en agissant sur les temps d'impulsions des neurones, nous introduisons un neuromodulateur modulant les taux d'apprentissage des neurones de la carte. Les neuromodulateurs sont des signaux cruciaux, e.g. pour l'apprentissage par renforcement, agissant sur des ensembles de synapses pour guider l'apprentissage. Les neuromodulateurs modulent plusieurs propriétés clés de l'apprentissage tel que le taux d'apprentissage ou des propriétés plus complexes comme le profil temporel de la STDP (Frémaux et Wulfram Gerstner, 2016 ; Pawlak et al., 2010 ; Yu et al., 2014). Notons qu'on peut trouver des neuromodulateurs dans les processeurs neuromorphiques, notamment sous le nom de trace d'éligibilité (Davies et al., 2018). Notre neuromodulateur consiste en un produit de deux facteurs, un facteur de modulation spatiale, fonction de la distance spatiale entre la SBMU et un neurone, et un facteur de modulation temporelle, fonction de la distance temporelle entre le temps d'impulsion de la SBMU et d'un neurone. Il s'agit ainsi d'un neuromodulateur spatio-temporel. On retrouve un facteur de modulation spatiale dans la SOM de Kohonen. Le facteur de modulation temporelle est une extension spécifique aux SNN et aux codes temporels, tirant profit de l'information contenue dans les temps d'impulsions. La distance temporelle entre deux temps d'impulsions est utilisée comme un indicateur de similarité entre le vecteur d'entrée courant et les deux vecteurs codes des neurones ayant déchargé. Une petite (grande) distance temporelle implique une grande (petite) similarité entre les vecteurs codes des deux neurones. Un bon algorithme de *clustering* minimise la distance intra-cluster et maximise la distance inter-cluster. Le rôle de ce facteur de modulation temporelle est de minimiser la distance intra-cluster en induisant une grande valeur de modulation, et de maximiser la distance inter-cluster en induisant une faible valeur de modulation.

Relâcher la contrainte WTA dans notre modèle SO-TCRL, et permettre potentiellement à tous les neurones de décharger en supprimant l'inhibition latérale, est utile dans deux mesures. La première raison est empirique, un niveau de sparsité trop important peut dégrader les performances, e.g. dans des tâches de classification (Falez et al., 2019). La deuxième raison est

d'un ordre plus théorique (Bengio, 2009 ; Bengio et al., 2013 ; Khanmohammadi et al., 2017 ; Masquelier et Kheradpisheh, 2018). Un circuit WTA force un unique neurone à répondre à une unique région de l'espace d'entrée ou à une unique catégorie. Il s'agit d'une représentation localiste, impliquant une relation un-à-un. Autrement dit, utiliser m neurones dans une représentation localiste implique de pouvoir discriminer $n = m$ régions de l'espace d'entrée. La plupart des algorithmes de *clustering* non-impulsionnels tels que K-Means ou les machines à vecteurs support (SVM) sont localistes (Bengio et al., 2013), car utilisant des représentations mutuellement exclusives ou des champs récepteurs qui ne se recoupent pas. Par contraste, une représentation distribuée (i.e. par population de neurones) implique qu'un neurone réponde à plusieurs régions de l'espace d'entrée ou à plusieurs catégories. En prenant uniquement en compte la sortie binaire d'un neurone impulsionnel, sans considérer les temps d'impulsions, on obtient alors jusqu'à $n = 2^m$ régions discriminables, ce qui offre un gain exponentiel par rapport à une représentation purement localiste. De façon équivalente, cette représentation distribuée utilise $m = \log_2 n$ neurones pour représenter n régions, plutôt que $m = n$ neurones dans une représentation localiste. Si on considère à présent la dimension temporelle et qu'on suppose que chaque neurone décharge une fois dans une fenêtre temporelle constituée de Δ pas de temps, alors on obtient $n = \Delta^m$. Notons que l'introduction de la dimension temporelle a permis de paramétrer la base, nous avons changé la base de 2 (dû à la nature binaire de la sortie d'un neurone impulsionnel) à Δ, augmentant ainsi la flexibilité et l'expressivité de ce mode de représentation distribué. Nous passons ainsi d'une représentation spatialement distribuée à une représentation spatio-temporellement distribuée. Enfin, en conjuguant la dimension temporelle et le fait qu'un neurone puisse décharger ou non, nous obtenons un nouveau gain supplémentaire en termes de nombre n de régions discriminables :

$$n = \sum_{k=0}^{m} \binom{m}{k} \Delta^{m-k}$$

Notre modèle d'apprentissage de représentations distribuées SO-TCRL est capable de traiter des données génériques. Il ne nécessite pas de réglage manuel par essai et erreur d'hyperparamètres en fonction de la dimension des données tel que la valeur maximale des poids synaptiques dans Rumbell et al., 2014. Dans les sections suivantes nous évaluons de façon approfondie les différentes propriétés attendue d'un modèle de SOM impulsionnelle, reproduisant les propriétés classiques de la SOM de Kohonen, i.e. capacité de QV, de génération de carte ordonnée et de catégorisation. Nous testons les performances en termes d'erreur de quantification (ou reconstruction) sur les bases de données d'images naturelles et MNIST, ainsi que la préservation de continuité locale entre neurones voisins. Nous testons également la qualité des cartes générées par notre SOM impulsionnelle en utilisant une mesure globale quantifiant la qualité de la projection de la variété de données (espace d'entrée) sur la carte (espace de sortie). Nous testons également la capacité du modèle à *clusteriser* des catégories dans la carte. Enfin nous proposons un nouveau schème d'étiquetage (*labelling*) tirant profit des représentations spatio-temporellement distribuées offertes par notre modèle SO-TCRL.

En résumé les principales contributions de ce chapitre sont :

— Le premier modèle de SOM impulsionnelle conjuguant à la fois une capacité de QV et de génération de carte topographiquement ordonnée sur des bases de données génériques : la SO-TCRL.

— Une augmentation des règles STDP spatialement et temporellement locales du modèle WD-TCRL via un nouveau neuromodulateur consistant en un produit d'un facteur de

modulation spatiale, basé sur les distances dans la carte, et d'un facteur de modulation temporelle basé sur les distances temporelles entre les temps d'impulsions postsynaptiques.

— Une nouvelle méthode d'étiquetage et de vote non-supervisée, tirant profit de la représentation spatio-temporellement distribuée de la SO-TCRL pour prédire les étiquettes des catégories. Cette méthode n'est pas cantonnée à opérer avec la SO-TCRL, elle est compatible avec toute architecture dont la sortie est impulsionnelle et où les temps d'impulsions transportent de l'information.

— Une méthode robuste permettant de traiter des données d'entrées de dimension arbitraire, sans nécessité de réglage fin d'hyperparamètres.

— Une étude expérimentale approfondie des fonctions attendues par un modèle de SOM impulsionnel complet, i.e. QV, génération de carte ordonnée et catégorisation.

6.2 Méthodes

Cette section décrit les différents composants architecturaux et algorithmiques de notre modèle de SOM impulsionnelle nommé SO-TCRL basé sur le modèle WD-TCRL. Cette section met l'accent sur les différences entre le modèle SO-TCRL et le modèle WD-TCRL, en particulier sur le nouveau mécanisme de neuromodulation. Le lecteur·rice est invité à se référer au chapitre 5 pour une analyse plus détaillée des composants algorithmiques communs aux deux modèles.

La SO-TCRL est une extension du modèle WD-TCRL, intégrant un nouveau neuromodulateur spatio-temporel permettant de générer des cartes ordonnées, ainsi que de minimiser la distance intra-cluster et maximiser la distance inter-cluster. Le neuromodulateur remplace les interactions latérales inhibitrices et le mécanisme homéostatique du modèle WD-TCRL.

Nous présentons également un nouveau schème d'étiquetage et de vote pour la prédiction de classes, tirant profit de la représentation spatio-temporellement distribuée de l'activité de la SO-TCRL.

6.2.1 Architecture

Le réseau de neurones impulsionnel (SNN) est composé de deux couches. La première couche encode les données d'entrée en latences relatives (voir Chap. 3.1.1) et l'autre couche extrait des représentations des motifs impulsionnels reçus, tout en intégrant une capacité à générer une carte ordonnée.

Ces deux couches sont entièrement connectées (voir Fig. 6.1). L'apprentissage est réalisé entre ces deux couches à l'aide de deux règles STDP. Une règle STDP adapte les délais d_{ji} pour stocker des représentations. Une autre règle STDP adapte les poids w_{ji} pour filtrer les caractéristiques en fonction de leurs pertinences. Ces deux règles STDP présentées dans le chapitre 5 sont soumises à un nouveau neuromodulateur agissant dans la carte (couche de représentation). Le neuromodulateur conjugue deux contraintes : 1) une contrainte topologique statique similaire à la SOM de Kohonen (Kohonen, 2013) pour la génération de carte ordonnée avec 2) une nouvelle contrainte temporelle dynamique utilisant l'intervalle de temps entre les impulsions émises par les neurones de la SOM impulsionnelle comme indicateur de similarité.

Contrairement à la SOM de Kohonen, notre modèle SO-TCRL n'utilise pas de schème de réduction du rayon de voisinage et des taux d'apprentissage, autorisant ainsi un apprentissage

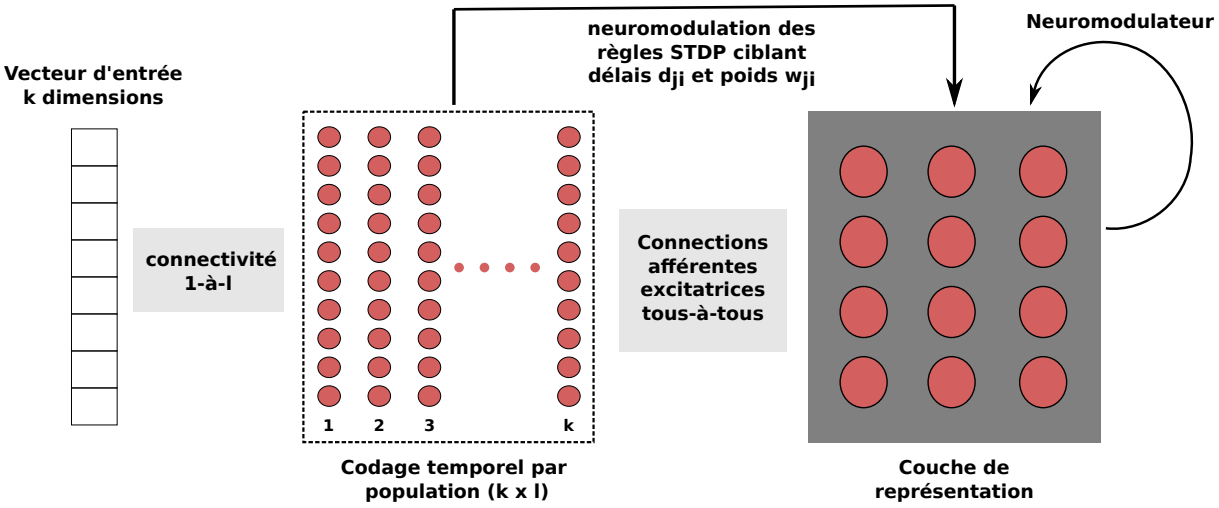

FIGURE 6.1 – Schéma bloc de l'architecture du SNN. Soit un vecteur d'entrée à k dimensions. Une dimension du vecteur d'entrée est encodée par l neurones. L'activité éparse des $k * l$ neurones est transmise à la couche de représentation. Entre ces deux couches s'opère l'apprentissage des délais d_{ji} et des poids synaptiques w_{ji} avec deux règles STDP distinctes. Un nouveau neuromodulateur opère à partir de l'activité de la couche de représentation et module l'apprentissage des deux règles STDP.

continu comme dans ROUGIER et BONIFACE, 2011.

Modèle de neurone et de synapse

Le modèle de neurone et de synapse utilisé pour la SO-TCRL est le même que pour WD-TCRL.

Le modèle de neurone utilisé dans le SNN est le LIF (voir section 2.3.1) régi par l'équation :

$$\tau_m \frac{dV(t)}{dt} = -V(t) + I_{\text{ext}}(t) \tag{6.1}$$

La variation instantanée de $V_j(t)$ est égale à la somme des activités présynaptiques retardées par les délais d_{ji} et pondérées par les poids synaptiques w_{ji} :

$$V_j(t) \leftarrow V_j(t) + \sum_{i=1}^{n} w_{ji} s_i(t - d_{ji}) \tag{6.2}$$

Il s'agit d'un modèle de transmission instantanée, i.e. avec un changement instantané du potentiel membranaire $V_j(t)$, tenant compte des délais de transmission.

Chaque synapse a accès à une trace présynaptique $x_i(t)$ (avec $i = 1, 2, \ldots n$) et deux traces postsynaptiques $y_j(t)$ et $z_j(t)$ (avec $j = 1, 2, \ldots m$). $z_j(t)$ est une nouvelle trace postsynaptique utilisée par le neuromodulateur. Les traces sont régies par les équations suivantes :

$$
\begin{aligned}
x_i(t) &\leftarrow 1, && \text{si } s_i(t) = 1 \\
\tau_x \frac{dx_i(t)}{dt} &= -x_i(t), && \text{sinon} \\
y_j(t) &\leftarrow 1, && \text{si } s_j(t) = 1 \\
\tau_y \frac{dy_j(t)}{dt} &= -y_j(t), && \text{sinon} \\
z_j(t) &\leftarrow 1, && \text{si } s_j(t) = 1 \\
\tau_z \frac{dz_j(t)}{dt} &= -z_j(t), && \text{sinon}
\end{aligned}
$$

où $s(t)$ est une fonction indicatrice renvoyant 1 lorsqu'un neurone émet une impulsion au temps t, 0 sinon.

6.2.2 Un nouveau neuromodulateur spatio-temporel

Nous introduisons dans cette section notre nouveau neuromodulateur, que nous intégrons ensuite aux règles STDP. Il consiste en un produit de deux facteurs de modulation non-linéaires (voir Figure 6.2) :

— un facteur de modulation spatiale, dépendant de la distance dans la carte entre un neurone et la SBMU

— un facteur de modulation temporelle, dépendant de la distance temporelle entre le temps d'impulsion d'un neurone et celui de la SBMU

La modulation du taux d'apprentissage d'un neurone j de la couche de représentation est déterminée par ce neurone j de façon événementielle et locale dans l'espace et le temps. Dans un premier temps, l'indice de la SBMU est déterminé par un neurone $j \in \{1, 2, \ldots, m\}$ au moment où il émet une impulsion, indiqué par $s_j(t) = 1$. La SBMU est l'indice du premier neurone ayant déchargé. Le neurone j détermine la SBMU à partir de l'activité récente des m neurones de la couche de représentation contenue dans les traces poststynaptiques $z_h(t)$ avec $h = 1, 2, ..., m$. Le neurone trouve la SBMU de façon temporellement locale en déterminant la trace postsynaptique avec la plus faible valeur au sein d'une fenêtre temporelle de $3\tau_z$ depuis le temps d'une impulsion postsynaptique, indiqué par $z_h(t) > \epsilon$, avec $\epsilon = 0.05 \approx e^{-3\tau_z/\tau_z}$.

$$
\text{sbmu} = \underset{h=1,\ldots,m}{\arg\min}\ z_h(t) \quad \text{si } z_h(t) > \epsilon
$$

Ensuite la valeur de la modulation mod_j du neurone j est déterminée par le produit de deux facteurs : un facteur de modulation spatiale et un autre facteur de modulation temporelle. Le facteur de modulation spatiale est un noyau gaussien fonction de la distance euclidienne normalisée entre la SBMU et le neurone j dans la carte. Plus (moins) les neurones sont proches dans la carte, plus (moins) la valeur de ce facteur est élevée. Cela permet à deux neurones spatialement proches de progressivement partager des représentations proches dans l'espace d'entrée. Le facteur de modulation temporelle est donné par la valeur de la trace postsynaptique de la SBMU $z_{\text{sbmu}}(t)$. Il correspond à un noyau de décroissance exponentielle appliqué à l'intervalle de temps séparant le temps d'impulsion t_{sbmu} de la SBMU et le temps d'impulsion t_j du neurone j. Le facteur de modulation temporelle vise à minimiser la distance intra-cluster et maximiser la distance inter-cluster. Deux neurones déchargeant de façon proche (éloignée) dans le temps tendent à partager des représentations proches (éloignées) dans la variété de données, et ainsi à faire partie d'un même cluster ou de deux clusters différents, respectivement. Le facteur de modulation temporelle induit une grande valeur pour deux neurones déchargeant en proximité temporelle et une faible valeur pour deux neurones déchargeant de façon temporellement distante. Le facteur de modulation temporelle peut être interprété comme un facteur dynamique dépendant de la distribution des données et des vecteurs codes appris par les neurones. Par contraste, le facteur de modulation spatiale peut être interprété comme un facteur statique imposant une contrainte topologique dans la carte.

$$\text{mod}_j = \Theta \underbrace{\big(d(j, \text{sbmu})\big)}_{\text{mod. spatiale}} \cdot \underbrace{z_{\text{sbmu}}(t)}_{\text{mod. temporelle}}$$

avec $\Theta(d)$ étant le noyau gaussien, modèle complètement isotropique dont le profil est réglé par l'hyperparamètre r, correspondant au rayon de voisinage centré sur la position de la SBMU. Un grand (petit) r implique un grand (petit) rayon de voisinage.

$$\Theta(d) = \exp\left(-\frac{d^2}{r^2}\right)$$

En résolvant l'équation différentielle régissant une trace postsynaptique, il devient explicite que la trace postsynaptique de la SBMU est équivalente à un noyau de décroissance exponentielle appliqué sur l'intervalle de temps séparant les temps d'impulsions t_{sbmu} de t_j. La vitesse de décroissance de la trace est réglée par la constante de temps τ_z. Tandis que l'hyperparamètre r règle le rayon de voisinage spatial centré sur la position de la SBMU, τ_z peut être interprété comme un rayon de voisinage temporel centré sur le temps d'impulsion de la SBMU. Un grand (petit) τ_z implique un grand (petit) rayon de voisinage temporel.

$$z_{\text{sbmu}}(t = t_j) = \exp\left(-\frac{t_j - t_{\text{sbmu}}}{\tau_z}\right)$$

Enfin, notons que le produit des deux facteurs de modulation tombe dans l'intervalle ⌋0,1]. En effet $\text{mod}_j \in]0, 1]$ car $\Theta(d) \in]0, 1]$ et $z_{\text{sbmu}}(t) \in]0, 1]$. Dans le cas particulier où le neurone j considéré est la SBMU, la valeur du neuromodulateur est maximale (1.0) car $d = 0$ et $t_j - t_{\text{sbmu}} = 0$.

6.2.3 Modulation de l'apprentissage des délais

Nous augmentons à présent les règles d'apprentissage du modèle WD-TCRL avec notre nouveau neuromodulateur spatio-temporel. Le lecteur·rice est invité à se référer au chapitre 5 pour une analyse aprofondie des règles d'apprentissage.

En intégrant le neuromodulateur à la règle STDP adaptant les délais nous obtenons :

$$\Delta d_{ji} = \begin{cases} \text{mod}_j \cdot \alpha^+ \Big(-\tau_x \ln\big(x_i(t)\big) - (d_{ji} + \lambda d_{ji}) \Big), & \text{si } s_j(t) = 1 \text{ et } x_i(t) > \epsilon \\ -\text{mod}_j \cdot \alpha^- \Big(-\tau_y \ln\big(y_j(t)\big) \Big), & \text{si } s_i(t - d_{ji}) = 1 \text{ et } y_j(t) > \epsilon \end{cases} \tag{6.3}$$

Puisque le neuromodulateur est une variable $mod_j \in]0, 1]$ les taux d'apprentissages dans notre règle STDP adaptant les délais varient à présent entre $]0, \alpha^+]$ et $]0, \alpha^-]$, respectivement.

Notons que le second cas d'adaptation des délais afférents à un neurone postsynaptique j est déclenché si ce neurone j a précédemment émis une impulsion postsynaptique dans une fenêtre temporelle réglée par $y_j(t) > \epsilon$ relativement au temps courant t. Cela implique que l'utilisation du neuromodulateur mod_j est valide car mod_j a précédemment été actualisé dans une proximité temporelle, au moment de l'émission d'une impulsion postsynaptique par le neurone j.

6.2.4 Modulation de l'apprentissage des poids

Le neuromodulateur est appliqué à tous les mécanismes adaptatifs du modèle WD-TCRL. Ainsi en intégrant le neuromodulateur à la règle STDP adaptant les poids synaptiques nous obtenons :

$$\Delta w_{ji} = \begin{cases} \text{mod}_j \cdot \beta^+ \big(\exp\big(-\frac{v_{ji}}{\sigma^2}\big) - w_{ji} \big), & \text{si } s_j(t) = 1 \text{ et } x_i(t) > \epsilon \text{ et } e_{ji} \geq 0 \\ -\text{mod}_j \cdot \beta^- \big(1 - y_j(t)\big), & \text{si } s_i(t - d_{ji}) = 1 \text{ et } y_j(t) > \epsilon \end{cases} \tag{6.4}$$

De nouveau, puisque le neuromodulateur est une variable $mod_j \in]0, 1]$ les taux d'apprentissages dans notre règle STDP adaptant les poids varient à présent entre $]0, \beta^+]$ et $]0, \beta^-]$, respectivement. L'erreur temporelle locale $e_{ji} = -\tau_x \ln\big(x_i(t)\big) - d_{ji}$ est déjà calculée et disponible dans l'équation 6.3.

L'utilisation du neuromodulateur dans le second cas d'adaptation est valide pour les mêmes raisons exposées dans la règle STDP ciblant les délais.

Lorsque les contraintes du premier cas d'adaptation sont satisfaites, non seulement le premier cas d'adaptation est déclenché, mais également l'estimation événementielle et en ligne de la variance temporelle v_{ji}. Le taux d'adaptation de la variance mobile est lui aussi soumis au neuromodulateur. Une faible valeur du neuromodulateur implique une faible actualisation de la variance v_{ji}, permettant de préserver une relative stabilité. Par exemple un neurone spatialement éloigné de la SBMU reçoit une faible valeur de neuromodulation, lui permettant de rester verrouillé sur la région de l'espace d'entrée qu'il *clusterise*.

$$v_{ji} = \ (1 - \text{mod}_j \cdot \alpha^+ \cdot \gamma) \cdot (v_{ji} + \text{mod}_j \cdot \alpha^+ \cdot \gamma e_{ji}^2) \tag{6.5}$$

6.2.5 Schèmes d'étiquetage et de vote pour la catégorisation

Outre la capacité de QV et de génération de carte ordonnée, nous souhaitons évaluer la capacité de notre modèle SO-TCRL à tirer profit des représentations apprises pour catégoriser des données. Soulignons que la SO-TCRL est un algorithme d'apprentissage non-supervisé, i.e. n'utilisant pas les étiquettes (ou *labels*) des données durant la phase d'apprentissage pour optimiser les représentations apprises. Notons que le fait que les étiquettes ne soient pas utilisées durant l'apprentissage rend les performances obtenues nécessairement moins compétitives qu'un apprentissage supervisé. Néanmoins cela permet à minima de valider la capacité du modèle à extraire la structure statistique latente des données dans une perspective de catégorisation, où des points proches (éloignés) dans la variété de données tendent à préserver (séparer) la catégorie d'un objet. En outre, les représentations apprises peuvent être fournies en entrées à un classifieur, e.g. il a été montré que K-Means - algorithme simple d'apprentissage non-supervisé de centroïdes à une seule couche - utilisé en conjonction avec un simple classifieur linéaire est capable d'atteindre de très bonnes performances sur les bases de données CIFAR-10 et NORB (Coates et al., 2011). Afin d'ajouter une capacité de catégorisation au modèle, nous introduisons une phase d'étiquetage (ou *labelling*) survenant entre la phase d'apprentissage et la phase de test (Diehl et Cook, 2015 ; Hazan et al., 2020) :

1. Phase d'apprentissage de représentations. La phase d'apprentissage non-supervisée utilise les données brutes (sans leurs étiquettes) pour extraire des représentations.
2. Phase d'étiquetage des neurones. Nous réutilisons la base d'apprentissage, en tirant cette fois ci profit des étiquettes (*labels*) des points de données. Des étiquettes sont assignées aux neurones en fonction de leurs réponses, i.e. des motifs impulsionnels spatio-temporels produits, à ces données étiquetées.
3. Phase de vote des neurones. Les neurones étiquetés votent pour prédire la catégorie de chaque point de donnée de test présenté en entrée du réseau.

Il existe de nombreux schèmes d'étiquetage et de vote. Nous introduisons ici informellement plusieurs schèmes d'étiquetage et de vote populaires que nous testons dans la section suivante. Le premier et le troisième schème peuvent être notamment retrouvés dans Hazan et al., 2020 sous le nom de *distance-based* et *confidence weighting*, respectivement. Le dernier schème *Temporal Population* (TMP-POP) est une innovation que nous proposons, tirant profit du caractère spatio-temporellement distribué de la réponse neuronale de la SO-TCRL.

— BMU. Il s'agit d'un schème localiste (ou individuel) où la réponse neuronale (les impulsions) n'est pas considérée. Seul le neurone minimisant la distance euclidienne entre son vecteur code et l'entrée est considéré. Notons que contrairement à la SOM, les représentations ont une dimensionnalité supérieure à celle des entrées, nous passons ici par une moyenne circulaire pondérée rappelée dans la section 5.3.2. Phase d'étiquetage : pour chaque point de donnée la BMU est calculée et son accumulateur de classe est incrémenté. Phase de vote : la BMU est calculée pour chaque point de donnée de test, la classe prédite correspond à la classe la plus représentée dans l'accumulateur de classe de la BMU.

— SBMU. Il s'agit aussi d'un schème localiste. Cependant contrairement au schème BMU, il est auto-organisé : nous considérons la réponse neuronale, i.e. le premier neurone qui décharge, plutôt qu'une décision prise à partir d'une connaissance globale des vecteurs codes des neurones. Phase d'étiquetage : la SBMU est déterminée et son accumulateur de classe est incrémenté. Phase de vote : la SBMU prédit la classe avec le plus haut score dans son accumulateur de classe.

— SPK-POP. Nous passons à présent d'un schème localiste à un schème distribué (par population). Celui-ci est basé sur l'activité spatialement distribuée des neurones dans la carte, sans considérer la dimension temporelle, i.e. les temps d'impulsions. Ce schème est populaire pour les codes en taux de décharge. Phase d'étiquetage : l'accumulateur de classes de tous les neurones ayant déchargé est incrémenté. Phase de vote : un ensemble de neurones décharge en réponse à une entrée, une somme pondérée de leurs accumulateurs de classes est utilisée. La classe prédite est celle obtenant le plus haut score.

— **TMP-POP**. Il s'agit de notre nouveau schème distribué et spatio-temporel. Il tire profit de l'information contenue dans l'identité des neurones ayant déchargé, ainsi que de l'information contenue dans la dimension temporelle, i.e. dans les temps d'impulsions. Ce schème introduit un nouveau concept de score de confiance assigné aux neurones en fonction de leurs latences relatives. Ce score est basé sur un *softmax*[1], fréquemment utilisé comme classifieur pour des tâches d'apprentissage supervisé dans les ANN. Plus un neurone décharge tôt relativement aux autres neurones, plus son score de confiance est élevé. Par contraste, plus il décharge tard, plus son score de confiance est faible. Ce score prend ainsi en compte non seulement le rang (ou ordre) des impulsions (Thorpe et Gautrais, 1998) mais également les temps précis des impulsions. Comparativement aux autres schèmes, nous passons ainsi d'une incrémentation tout ou rien (1 ou 0) de l'accumulateur de classe à une variation dans un continuum dans l'intervalle $[0, 1]$. Phase d'étiquetage : l'accumulateur de classe accumule les scores de confiance. Phase de vote : un ensemble de neurones décharge en réponse à une entrée, les scores de confiance des neurones sont pondérés par une fonction softmax de leurs latences relatives, donnant un plus grand poids de vote aux premiers neurones déchargeant. La classe prédite est celle obtenant collectivement le plus haut score de confiance.

Nous exprimons plus formellement ces différents schèmes d'étiquetage et de votes ci-dessous.

BMU

Phase d'étiquetage. La BMU est déterminée pour un point de donnée a_p de la base d'apprentissage composée de $p = 1, 2, ...P$ points. La BMU est déterminée en cherchant le vecteur code minimisant la distance euclidienne avec l'entrée courante a_p. Les vecteurs codes sont déterminés comme dans la section 5.3.2. Chaque point de donnée a un numéro de classe h associé. L'accumulateur de classe $\text{acc}(i, h)$ de la BMU ayant pour indice i est alors incrémenté de 1 pour le numéro de classe h. L'incrémentation des accumulateurs de classes des BMU $\text{acc}(i, h)$ se fait itérativement sur la base d'apprentissage constituée d'instances de classes $C_1, C_2, ...$

$$\text{acc}(i, h) = \#\Big\{p \mid a_p \in C_h \text{ et } \text{BMU}(a_p) = i\Big\}$$

Phase de test. La BMU ayant pour indice i prédit une classe c_i correspondant à la classe la plus représentée dans son accumulateur de classe. L'accumulateur est normalisé par le nombre d'instances (ou cardinal) des différentes classes C_h de la phase d'étiquetage.

$$c_i = \arg\max_h \frac{\text{acc}(i, h)}{\#C_h}$$

1. Il s'agit plus précisément d'une distribution de Boltzmann, similaire à une fonction *softmax*.

SBMU

Tandis que la BMU est déterminée en cherchant le vecteur code minimisant la distance euclidienne avec l'entrée courante a_p, la SBMU correspond à l'indice du premier neurone i qui décharge en réponse à a_p. Hormis cette différence, le schème pour la SBMU est identique à celui de la BMU.

Phase d'étiquetage. La SBMU est déterminée pour un point de donnée a_p de la base d'apprentissage composée de $p = 1, 2, ...P$ points. Chaque point de donnée a un numéro de classe h associé. L'accumulateur de classe $\text{acc}(i, h)$ de la SBMU ayant pour indice i est alors incrémenté de 1 pour le numéro de classe h. L'incrémentation des accumulateurs de classes des SBMU $\text{acc}(i, h)$ se fait itérativement sur la base d'apprentissage constituée des classes $C_1, C_2, ...$

$$\text{acc}(i, h) = \#\Big\{p \mid a_p \in C_h \text{ et } \text{SBMU}(a_p) = i\Big\}$$

Phase de test. La SBMU ayant pour indice i prédit une classe c_i correspondant à la classe la plus représentée dans son accumulateur de classe. L'accumulateur est normalisé par le nombre d'instances (ou cardinal) des différentes classes C_h de la phase d'étiquetage.

$$c_i = \arg\max_h \frac{\text{acc}(i, h)}{\#C_h}$$

SPK-POP

Spike Population (SPK-POP) est basé sur l'activité des neurones, sans considérer leurs temps d'impulsions. Nous introduisons la fonction indicatrice $s_i(a)$ retournant 1 si une impulsion a été émise par un neurone i pour une entrée a présentée dans la fenêtre temporelle d'encodage T, 0 sinon.

$$s_i(a) = \begin{cases} 1, & \text{si } s_i(t) = 1 \text{ pour } t \in [0, T] \text{ pour l'entrée } a \\ 0, & \text{sinon} \end{cases}$$

Phase d'étiquetage. Tout les neurones ayant déchargé pour un point de donnée a_p ayant h pour numéro de classe, voient leurs accumulateurs de classe être incrémentés de 1 au numéro de classe h correspondant.

$$\text{acc}(i, h) = \#\Big\{p \mid a_p \in C_h \text{ et } s_i(a_p) = 1\Big\}$$

Phase de test. Soit a_t un point de donnée de test. Soit acc_pop un nouvel accumulateur de classe agrégeant les votes des neurones. L'agrégation est réalisée en sommant les valeurs des accumulateurs de classes des neurones ayant déchargés pour un point de test a_t.

$$\text{acc\_pop}(h) = \sum_{\substack{i \\ s_i(a_t)=1}} \text{acc}(i, h)$$

avec i étant l'indice d'un neurone qui décharge pour a_t (indiqué par $s_i(a_t) = 1$).

La classe c d'un point de donnée a_t prédite collectivement par les neurones est celle qui est la plus représentée dans l'accumulateur de population acc_pop. L'accumulateur de population est normalisé par le cardinal des classes de la phase d'étiquetage.

$$c = \arg\max_h \frac{\text{acc\_pop}(h)}{\#C_h}$$

TMP-POP

Contrairement à SPK-POP, notre nouveau schème *Temporal Population* (TMP-POP) tire profit de l'information contenue dans les temps d'impulsions de la couche de représentation. Comme pour SPK-POP, nous introduisons la fonction indicatrice $s_i(a)$ retournant 1 si une impulsion a été émise par un neurone i pour une entrée a présentée dans la fenêtre temporelle d'encodage T, 0 sinon.

$$s_i(a) = \begin{cases} 1, & \text{si } s_i(t) = 1 \text{ pour } t \in [0, T] \text{ pour l'entrée } a \\ 0, & \text{sinon} \end{cases}$$

Phase d'étiquetage. Tous les neurones ayant déchargé pour un point de donnée a_p ayant h pour numéro de classe, voient leurs accumulateurs de classe être actualisés par un score de confiance normalisé tombant dans un continuum dans $[0, 1]$, au numéro de classe h correspondant. Le premier neurone qui décharge obtient le score de confiance le plus élevé. Plus un neurone décharge tardivement relativement aux autres neurones, plus son score de confiance est bas. Le score de confiance est calculé avec un *softmax* que nous adaptons au traitement de données impulsionnelles.

$$\text{acc}(i, h) = \sum_{a_p \in C_h} \text{softmax}_i \quad \text{(calculé sur la réponse neuronale à l'entrée } a_p)$$

La fonction softmax permet à la fois de préserver le rang des impulsions et de régler la sensibilité à la magnitude des latences relatives. Autrement dit, la fonction softmax est sensible au rang et à la distribution des temps des impulsions. La fonction softmax prend en entrée les latences relatives entre les temps d'impulsions t_i des neurones et produit en sortie un score de confiance attribué à chacun des neurones ayant déchargé. La sensibilité à la magnitude des latences relatives est réglée par l'hyperparamètre τ_{soft}. Cet hyperparamètre est connu sous le nom d'hyperparamètre de température, réglant l'entropie de la distribution de sortie. L'entropie est maximale pour une température infinie car la distribution devient uniforme. Dans notre cas, τ_{soft} est exprimé en unité de temps. Si $\tau_{\text{soft}} \to \infty$, la distribution des scores de confiance devient uniforme, assignant ainsi des scores identiques aux neurones. Si $\tau_{\text{soft}} \to 0$, la distribution des scores de confiance est concentrée sur un unique point, correspondant au premier neurone qui décharge.

$$\text{softmax}_i(a) = \frac{\exp\left(-\frac{\Delta_{t_i}}{\tau_{\text{soft}}}\right)}{\sum_i^m \exp\left(-\frac{\Delta_{t_i}}{\tau_{\text{soft}}}\right)}$$

$$\text{avec } \Delta_{t_i} = t_i - t_{\min}$$

Pour obtenir un accumulateur de classe équitable fair_acc tenant compte du nombre d'instances par classes, nous le divisons par le cardinal des classes :

$$\text{fair\_acc}(i,h) = \frac{\text{acc}(i,h)}{\#C_h}$$

Enfin nous normalisons l'accumulateur de classe, de sorte que pour chaque neurone la somme des scores de confiance accumulés sur les différentes classes donne 1. Le score de confiance normalisé $\text{conf}(i,h)$ d'un neurone i pour une classe h est ainsi donné par :

$$\text{conf}(i,h) = \frac{\text{fair\_acc}(i,h)}{\sum_k \text{fair\_acc}(i,k)}$$

avec k indiçant les classes représentées dans l'accumulateur.

Phase de test. Soit a_t un point de donnée de test. Soit acc_pop un nouvel accumulateur de classe agrégeant les votes des neurones ayant déchargé. L'agrégation est réalisée par une somme des scores de confiance des neurones pour les différentes classes, pondérée par la sortie de la fonction softmax. La fonction softmax prend en entrée les latences relatives des neurones induites pour un point de donnée a_t. Son rôle est de pondérer l'importance du vote des neurones : les scores de confiance des classes des neurones déchargeant en premier sont plus considérés que ceux des neurones déchargeant en dernier.

$$\text{acc\_pop}(h) = \sum_i \text{softmax}_i(a_t) \cdot \text{conf}(i,h)$$

La classe c d'un point de donnée a_t collectivement prédite par les neurones est celle qui est la plus représentée dans l'accumulateur de population acc_pop :

$$c = \arg\max_h \ \text{acc\_pop}(h)$$

6.3 Protocole expérimental et résultats

Dans cette section nous évaluons expérimentalement les différentes fonctions clés attendues pour un modèle de SOM impulsionnelle complet, i.e. les propriétés de QV, la capacité à générer des cartes ordonnées et l'efficacité en catégorisation.

Dans nos expériences, nous utilisons une topologie torique, éliminant les effets de bords.

Nous commençons par tester la QV en reproduisant les expériences menées dans les chapitres précédents sur les bases de données MNIST et d'images naturelles. Une propriété additionnelle est évaluée : la préservation des relations locales de voisinage entre neurones dans leurs vecteurs codes, induisant que deux neurones voisins tendent à partager des représentations proches.

Ensuite, nous évalons la qualité globale et la robustesse de la carte pour un jeu de données synthétique, i.e. la capacité à préserver les distances entre des points de données par les distances entre les positions des neurones représentant ces données.

Enfin nous évaluons la capacité de catégorisation de la SO-TCRL en testant les différents schèmes d'étiquetage et de vote présentés dans la section 6.2.5. Nous évaluons les performances obtenues sur les bases de données IRIS et WBCD.

Un réglage fin des hyperparamètres n'est pas nécessaire : nous utilisons le même jeu d'hyperparamètres pour mener toutes ces expériences. Seul le seuil de décharge des neurones V_θ de la couche de représentation doit être adapté à la dimension des données à cause du nombre d'impulsions afférentes induit. Nous utilisons une méthode simple et efficace pour déterminer de façon automatique la valeur du seuil de décharge V_θ.

6.3.1 Paramètres du SNN

Paramétrisation du SNN en fonction de la dimension des données d'entrées

Nous présentons une paramétrisation générique du modèle SO-TCRL pour des données de dimensions arbitraires. Seul un paramètre dépend de la dimension k des données dans la couche de représentation v : le seuil de décharge des neurones V_θ. Les paramètres réglant le profil d'inhibition latérale utilisés dans WD-TCRL ne sont plus nécessaires dans SO-TCRL, la fonction de l'inhibition latérale étant remplacée par le neuromodulateur. Par commodité nous posons l'hypothèse d'une relation linéaire entre la dimension k des données d'entrées et le seuil de décharge des neurones V_θ.

Le coefficient de l'équation linéaire liant V_θ à la dimension des données k et au nombre de neurone encodeurs l a été déterminé par la méthode d'optimisation bayésienne *Tree-Structured Parzen Estimator* (TPE) :

$$V_\theta = 0.44kl \tag{6.6}$$

Paramètres utilisés

Les paramètres du modèle SO-TCRL sont donnés dans le tableau 6.1. Rappelons que les paramètres de la couche d'encodage sont communs à l'ensemble des modèles présentés dans ce manuscrit, $l = 10$ neurones encodeurs sont utilisés par dimension d'entrée. Le même jeu de paramètres est utilisé pour l'ensemble des expériences menées pour évaluer les performances de la SO-TCRL. Seul le seuil de décharge V_θ des neurones de la couche de représentation doit être adapté à la dimension k des données. Sa valeur est déterminée de façon automatique à l'aide de l'équation 6.6.

6.3.2 Métriques utilisées

Nous introduisons dans cette section deux mesures additionnelles pour évaluer la régularité de la carte générée, la première est de nature locale (MDN), la seconde mesure est globale (EMDS). L'EMDS est couramment utilisée dans des travaux connexes, l'erreur MDN est davantage réservée aux SOM.

MDN

Comme dans la SOM de Kohonen, la SO-TCRL essaie de préserver les relations locales de voisinage entre neurones dans leurs vecteurs codes. Autrement dit, deux neurones voisins tendent à partager des représentations proches, générant ainsi une continuité locale dans la carte. Pour

TABLE 6.1 – Paramètres de SO-TCRL utilisés dans toutes les simulations

Paramètres neuronaux pour la couche d'encodage							
V_θ	τ_m	T_{refrac}					
0.5	10.0 ms	6 ms					
Paramètres neuronaux pour la couche de représentation							
V_θ	τ_m	T_{refrac}					
0.44 kl	5.3 ms	6ms					
Paramètres synaptiques							
τ_x	τ_y	τ_z	$d_{\min}$	$d_{\max}$			
4 ms	3 ms	3 ms	0 ms	10 ms			
Paramètres des règles STDP							
λ	α^+	α^-	β^+	β^-	γ	σ	ϵ
0.58	0.07	0.042	0.18	0.036	0.24	10.0 ms	0.05
Paramètres du neuromodulateur							
r							
0.10							
Paramètre du schème d'étiquetage et de vote TMP-POP							
τ_{soft}							
0.1 ms							

quantifier cette propriété de continuité locale dans les vecteurs codes des neurones, nous utilisons la *Mean Distance to Neurons* (MDN).

La MDN est basée sur l'agrégation des distances entre le vecteur code d'un neurone et les vecteurs codes de ses neurons voisins, dans notre cas ses 4 voisins pour une topologie torique. La MDN évalue ainsi la préservation locale des relations de voisinage dans les vecteurs codes des neurones. Il s'agit d'une mesure intrinsèque à la carte, i.e. sans référentiel extérieur.

$$\text{MDN} = \frac{1}{m}\sum_{j=1}^{m}\sum_{k=1}^{m}\begin{cases}||c_k - c_j||^2, & \text{si dist}(j,k) = 1\\ 0, & \text{sinon}\end{cases} \tag{6.7}$$

Cette formulation est basée sur des vecteurs codes c_j décodés à partir de chaque neurone j selon la méthode utilisée pour décoder les délais de la SBMU exposée dans la section 5.3.2.

EMDS

La MDN est une mesure locale et intrinsèque, basée sur les vecteurs codes des neurones. Nous souhaitons à présent utiliser une mesure globale et extrinsèque, évaluant la capacité à préserver les distances entre des points de données d'entrées dans la carte.

Cela est réalisé avec la métrique *MultiDimensional Scaling* (MDS) (GOODHILL et SEJNOWSKI, 1997). Cette métrique se base sur un calcul de dissimilarité entre la distance dans l'espace d'entrée

pour 2 points de données, et la distance spatiale des deux neurones les représentant. Ainsi plutôt que de considérer les vecteurs codes des neurones comme dans la MDN, l'EMDS considère les coordonnées spatiales des neurones dans la carte.

Une valeur de 0 signifie que la préservation topologique est parfaite, i.e. que chaque distance entre une paire de points de données d'entrée est égale à la distance spatiale (position spatiale) entre la paire de neurones les représentant (SBMU). Autrement dit, les distances dans l'espace d'entrée (points de données) sont préservées dans l'espace de sortie (coordonnées des neurones). Les distances dans l'espace d'entrée et dans la carte sont normalisées afin de pouvoir être comparées.

$$\text{MDS} = \sum_{p=1}^{P} \sum_{j<p} \Big(F(\mathbf{a}_p, \mathbf{a}_j) - G\big(M(\mathbf{a}_p), M(\mathbf{a}_j)\big) \Big)^2 \tag{6.8}$$

où P est le nombre de vecteurs d'entrées. F et G sont des mesures de similarité symétriques dans l'espace d'entrée et l'espace de la carte, respectivement. $F(\mathbf{a}_p, \mathbf{a}_j)$ mesure la similarité entre une paire de vecteurs d'entrées $\mathbf{a}_p$ et $\mathbf{a}_j$ à l'aide d'une distance euclidienne normalisée. Ces deux vecteurs d'entrées $\mathbf{a}_p$ et $\mathbf{a}_j$ sont représentés par deux neurones dans la carte dont les positions spatiales sont données par $M(\mathbf{a}_p)$ et $M(\mathbf{a}_j)$. $G\big(M(\mathbf{a}_p), M(\mathbf{a}_j)\big)$ mesure la similarité entre ces deux positions $M(\mathbf{a}_p)$ et $M(\mathbf{a}_j)$ avec une distance euclidienne.

Enfin, nous divisons la valeur MDS par le nombre de paires de vecteurs d'entrées afin d'obtenir une valeur moyenne insensible au nombre de vecteurs d'entrées P :

$$E_{\text{MDS}} = \frac{2 \text{ MDS}}{P(P-1)} \tag{6.9}$$

6.3.3 Evaluation de la capacité de quantification vectorielle

Nous évaluons dans cette section si la propriété de QV du modèle WD-TCRL est préservée dans SO-TCRL. Cela est mesuré avec l'erreur de reconstruction RMS équivalente à l'erreur de quantification, sur les mêmes bases de données MNIST et d'images naturelles qu'utilisées dans les chapitre précédents.

Nous mesurons également des propriétés intrinsèques (ou internes) au SNN, tel que la cohérence euclidienne des SBMU.

Comme dans la SOM de Kohonen, deux neurones voisins dans la SO-TCRL tendent à partager des représentations proches, générant ainsi une continuité locale dans la carte. C'est une propriété qui n'est pas partagée avec les modèles W-TCRL et WD-TCRL. Pour quantifier cette propriété de continuité locale dans les vecteurs codes des neurones, nous utilisons la *Mean Distance to Neurons* (MDN).

Nous évaluons la capacité de QV sur les bases de données MNIST et d'images naturelles. Chaque configuration est testée avec 10 exécutions indépendantes. Nous évaluons également l'ajout du facteur de modulation temporel au facteur spatial, i.e. spatial et spatio-temporel. Les expériences ont été menées pour plusieurs tailles de carte neuronales : (4,4), (8,8), (10,10) et (16,16) donnant $m \in \{16, 64, 100, 256\}$ neurones dans la couche de représentation et donc m

vecteurs codes soumis à apprentissage. Les délais entre la couche d'encodage et de représentation sont initialisés aléatoirement dans l'intervalle [0,0.4] ms avec une gaussienne centrée sur 0.2 ms, d'écart-type de 0.1 ms. Les poids synaptiques sont initialisés à leurs valeurs maximale à 1.

Résultats pour MNIST

Pour MNIST, les entrées sont normalisées dans l'intervalle $[0.15, 0.85]$. Nous introduisons une marge due à la projection de données d'entrées linéaires sur un espace circulaire (champs récepteurs) où les valeurs extrémales deviennent équivalentes ($2\ \pi$ est équivalent à 0 radian). En outre, la base MNIST est essentiellement composée de valeurs extrémales (pixels noir et blanc). Nous avons sélectionné aléatoirement un sous-ensemble de 15 000 et 1000 chiffres manuscrits pour la phase d'apprentissage et de test, respectivement. Des patchs de 4*4 pixels, extraits de chiffres manuscrits de 28*28 pixels, sont fournis comme vecteurs d'entrée au SNN. 60 000 patchs d'images sont donnés en entrée au SNN durant la phase d'apprentissage.

Nous évaluons les performances en termes d'erreur de reconstruction RMS entre un vecteur d'entrée et un vecteur code et de cohérence euclidienne de la SBMU. Nous évaluons également la continuité locale dans les vecteurs codes des neurones avec la mesure MDN.

Phase d'apprentissage pour MNIST

Les différentes mesures de performances décroissent de façon consistante avec le nombre d'itérations d'apprentissage (voir Fig. 6.3). L'erreur de reconstruction RMS décroît, indiquant que le SNN apprend à compresser la distribution des données d'entrées, jusqu'à atteindre un plateau relativement stable dès 8000 itérations autour d'une erreur RMS d'environ 0.13 pour $m \in \{16, 64, 100, 256\}$ neurones. La MDN diminue également, indiquant que la distance entre les vecteurs codes des neurones et les vecteurs codes de leurs voisins diminue. Enfin l'incohérence euclidienne dans l'élection de la SBMU décroît pour un seuil de décision $\mathrm{Th} = 5\ \%$ et $\mathrm{Th} = 10\ \%$, indiquant une décroissance du nombre de SBMU paradoxales et ainsi une sélectivité croissante des champs récepteurs des neurones à une région spécifique de l'espace d'entrée.

Les trajectoires des différentes mesures au cours des itérations d'apprentissage sont très proches (Fig. 6.3) pour une modulation purement spatiale et une modulation spatio-temporelle.

Phase de test pour MNIST

Nous évaluons à présent les performances de la SO-TCRL sur la base de données de test pour MNIST. Les résultats tendanciels sur la base d'apprentissage se trouvent confirmés sur la base de test (Fig. 6.4). Le facteur de modulation spatio-temporel offre de légères améliorations en termes d'erreur de reconstruction RMS et d'incohérence pour de plus grandes capacités (e.g. 256 neurones). Par exemple l'erreur RMS passe de 0.11 à 0.10 en ajoutant un facteur temporel au facteur spatial pour $m = 256$ neurones (Tab. 6.2).

La figure 6.5 montre que les vecteurs codes de la couche de représentation sont devenus sélectifs à des orientations visuelles variées. De plus, contrairement aux modèles W-TCRL et WD-TCRL, nous pouvons constater que ces orientations sont disposées de façon ordonnée dans la carte : des neurones spatialement proches partagent des orientations proches.

Enfin la reconstruction des images à partir des vecteurs codes élus de façon auto-organisée

par la SO-TCRL produit des chiffres manuscrits reconstruits de bonne qualité, comparables aux images originales (voir Fig. 6.6).

Résultats pour les images naturelles

Pour la base de données d'images naturelles, les entrées sont normalisées dans l'intervalle $[0.05, 0.95]$. Cette marge est due au fait que les données d'entrées linéaires sont projetées sur un espace circulaire où les valeurs extrémales deviennent équivalentes. Des patchs de 4*4 pixels, extraits d'images naturelles de 512*512 pixels, sont fournis comme vecteurs d'entrée au SNN. 60 000 patchs d'images sont donnés en entrée au SNN durant la phase d'apprentissage.

De nouveau, les mêmes trois mesures de performances ont été utilisées pour évaluer le SNN : l'erreur de reconstruction RMS entre un vecteur d'entrée et un vecteur code, l'incohérence dans le processus auto-organisé d'élection de la SBMU, et la continuité locale dans les vecteurs codes des neurones avec la mesure MDN.

Phase d'apprentissage pour les images naturelles

Tout comme pour MNIST, les différentes mesures de performances décroissent de façon consistante avec le nombre d'itérations d'apprentissage (voir Fig. 6.7). L'erreur de reconstruction RMS décroît, indiquant que le SNN apprend à compresser la distribution des données d'entrées, jusqu'à atteindre un plateau stable avec une valeur d'environ 0.05 dès 2000 itérations. La MDN diminue également, indiquant que la distance entre les vecteurs codes des neurones et les vecteurs codes de leurs voisins diminue. L'incohérence de l'élection de la SBMU décroît pour m = 16 neurones, mais pas pour les autres capacités. Cependant notons que cela n'affecte pas significativement l'erreur de reconstruction RMS. Une explication probable est qu'un grand nombre de neurones partagent des représentations proches, induisant qu'ils peuvent décharger avant et ainsi augmenter le nombre de SBMU paradoxales, tout en maintenant une faible erreur de reconstruction.

Phase de test pour les images naturelles

Nous évaluons à présent les performances de la SO-TCRL sur la base de données de test pour les images naturelles (Fig. 6.8). La modulation spatio-temporelle offre de légères améliorations en termes d'incohérence, en particulier pour de plus grandes capacités (100 et 256 neurones).

La figure 6.9 montre que les vecteurs codes de la couche de représentation sont devenus sélectifs à des orientations visuelles variées. Ces orientations sont disposées de façon ordonnée dans la carte : des neurones spatialement proches partagent des orientations proches.

Enfin la reconstruction des images à partir des vecteurs codes produit des images reconstruites de bonne qualité, comparables aux images naturelles originales (voir Fig. 6.10).

6.3.4 Evaluation de la capacité de génération de cartes ordonnées

Dans la section précédente, nous avons évalué la préservation locale des relations de voisinage dans les vecteurs codes des neurones avec la MDN. Il s'agit d'une mesure intrinsèque à la carte, i.e. sans référentiel extérieur.

TABLE 6.2 – Résultats moyens obtenus sur les bases de données MNIST et d'images naturelles pour un nombre de neurones $m \in \{16, 64, 128, 256\}$.

Modulation	Spatiale				Spatio-Temporelle			
Neurones m	16	64	100	256	16	64	100	256
RMS MNIST	0.11	0.11	0.11	0.11	0.11	0.11	0.10	0.10
RMS Nat	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
MDN MNIST	0.33	0.18	0.17	0.12	0.32	0.19	0.17	0.13
MDN Nat	0.36	0.15	0.12	0.08	0.36	0.15	0.12	0.08
Incoh. 5 % MNIST	0.16	0.16	0.30	0.62	0.16	0.15	0.29	0.4
Incoh. 5 % Nat	0.40	0.63	0.71	0.72	0.39	0.64	0.69	0.69
Incoh. 10 % MNIST	0.08	0.10	0.10	0.11	0.09	0.09	0.10	0.09
Incoh. 10 % Nat	0.11	0.45	0.48	0.49	0.10	0.45	0.47	0.48

Nous souhaitons à présent utiliser une mesure globale et extrinsèque, évaluant la capacité de la carte à préserver les distances entre des points de données d'entrées dans la carte. Cela peut être réalisé avec la métrique *MultiDimensional Scaling* (MDS), basée sur un calcul de dissimilarité entre la distance dans l'espace d'entrée pour 2 points, et la distance spatiale des deux neurones les représentant (SBMU). Une projection parfaite des distances entre des points d'entrées et des distances spatiales des neurones les représentant induit ainsi une erreur MDS de 0.

Nous construisons une expérience contrôlée permettant d'atteindre idéalement une erreur MDS de 0, fixant ainsi un minimum global théorique. L'expérience contrôlée est la même que celle de RUMBELL et al., 2014. Elle consiste à fournir un ensemble de points de données uniformément espacés égal au nombre de neurones utilisé par dimension de la carte. Dans nos expériences, nous utilisons une carte en deux dimensions de 10*10 neurones, induisant ainsi 10*10 points uniformément espacés dans $[0, 1]^2$ comme base d'apprentissage comme dans RUMBELL et al., 2014. Ainsi la carte est théoriquement capable de projeter parfaitement toute distance entre deux points d'entrées sur la distance spatiale entre deux neurones.

Nous entraînons la SO-TCRL avec 120 000 point de données choisis aléatoirement et évaluons la qualité des cartes avec la MDN et l'EMDS. Les délais entre la couche d'encodage u et de représentation v sont initialisés aléatoirement dans l'intervalle [0,0.4] ms avec une gaussienne centrée sur 0.2 ms, d'écart-type de 0.1 ms. Les poids synaptiques sont initialisés à leurs valeurs maximale à 1.

Nous évaluons la qualité des cartes obtenues avec 100 exécutions indépendantes pour 1) un facteur de modulation spatial ainsi 2) que pour un facteur de modulation spatio-temporel afin d'évaluer le rôle de ces facteurs dans le processus de génération de cartes ordonnées. Notons que l'espace des champs récepteurs de la couche d'encodage est circulaire, induisant que deux points de données sont proches dans un espace non pas linéaire, mais circulaire.

Les mesures EMDS et MDN décroissent de façon consistante avec le nombre d'itérations d'apprentissage (Fig. 6.11). Notons que les modulations purement spatiale d'une part et spatio-temporelle d'autre part, produisent des trajectoires similaires en terme d'évolution de l'EMDS et de la MDN. La EMDS décroît jusqu'à atteindre un plateau à 80 000 itérations autour d'une valeur de 0.005, indiquant que la distance spatiale entre deux SBMU ayant déchargé pour deux points

de donnée converge vers la distance entre ces deux points dans l'espace d'entrée. Cela montre que la carte apprend à préserver les distances entre les données dans les distances spatiales des neurones.

La MDN décroît jusqu'à atteindre un plateau à 80 000 itérations autour d'une valeur de 0.11, indiquant que la distance moyenne entre le vecteur code d'un neurone et les vecteurs codes de ses quatres voisins décroît. Cela montre que des neurones spatialement proches partagent progressivement des représentations proches. Nous pouvons mettre en relation la MDN obtenue - mesure locale et intrinsèque à la carte - avec les données d'entrées grâce à la nature de leur distribution. Nous pouvons ainsi offrir une interprétation globale et extrinsèque de la MDN. Dans nos expériences, les points de données 2D sont uniformément espacés d'un pas de 0.10 dans chaque dimension. Cela signifie que les quatre plus proches voisins d'un point de donnée dans la variété des données sont espacés d'une distance normalisée de 0.10. Ainsi le MDN idéal est de 0.10 et d'écart-type de 0, de sorte que la distance moyenne entre chaque vecteur code d'un neurone et les vecteurs codes de ses 4 voisins soit égale à la distance entre un point de donnée et ses 4 voisins. La valeur finale de la MDN est de 0.11 ± 0.01, très proche de cet idéal théorique de 0.10 ± 0.00.

Ces résultats se confirment après la phase d'apprentissage (voir Fig. 6.14 et Tab. 6.3) avec des valeurs de MDN identiques entre modulation spatiale et spatio-temporelle en atteignant 0.11 ± 0.001. La valeur moyenne d'EMDS est de 0.00521 ± 0.00663 et de 0.00599 ± 0.00635 pour une modulation spatiale et spatio-temporelle, respectivement. Les valeurs d'EMDS obtenues pour les deux configurations sont similaires à celle rapportée par Rumbell et al., 2014 de 0.00554 ± 0.00483 (Tab. 6.4).

Table 6.3 – Résultats moyens obtenus pour la génération de cartes ordonnées en termes d'EMDS et de MDN

Mesure	Modulation Spatiale	Modulation Spatio-Temporelle
EMDS	0.00521 ± 0.00663	0.00599 ± 0.00635
MDN	0.11 ± 0.001	0.11 ± 0.001

Table 6.4 – Meilleures valeurs d'EMDS rapportées par Rumbell et al., 2014 et comparées à notre modèle SO-TCRL

Rumbell et al., 2014	SO-TCRL
0.00554 ± 0.00483	$\mathbf{0.00521} \pm 0.00663$

Nous utilisons deux représentations additionnelles pour visualiser le processus de génération de cartes ordonnées :

1. La première représentation se réalise dans un espace linéaire, les vecteurs codes des neurones sont projetés dans l'espace d'entrée en 2 dimensions, ainsi que les connexions des neurones à leurs voisins. Cette représentation permet de vérifier si la carte se déplie correctement.
2. La deuxième représentation permet de visualiser la distance moyenne entre le vecteur code d'un neurone et les vecteurs codes de ses voisins. Elle est basée sur la MDN. Elle permet de vérifier si des neurones spatialement proches partagent des représentations proches.

La figure 6.12 montre que les cartes se déplient correctement au fil des itérations d'apprentissage. Non seulement les vecteurs codes des neurones ont correctement appris la distribution

uniforme des données, mais les neurones spatialement proches ont également appris à partager des représentations proches (dans un espace circulaire). Notons que la topologie de la carte est torique et que les champs récepteurs des neurones encodeurs sont situés dans un espace circulaire, induisant que deux valeurs proches sont proches dans un espace circulaire. Ainsi dans la figure 6.12 offrant une représentation linéaire, deux vecteurs codes proches des extremums (0 ou 1) d'une dimension de l'espace d'entrée sont proches dans un espace circulaire. La génération de cartes ordonnées induit que deux neurones dont les représentations sont proches sont spatialement proches, c'est pourquoi nous pouvons observer des connexions connectant les côtés opposés de l'espace d'entrée.

La figure 6.13 offre une autre représentation prenant la forme d'une carte thermique et basée uniquement sur la distance moyenne et normalisée du vecteur code d'un neurone avec les vecteurs codes de ses 4 voisins. La distance est calculée en tenant compte de l'espace circulaire des entrées et la relation de voisinage tient compte de la topologie torique de la carte. Nous pouvons observer que les vecteurs codes des neurones deviennent progressivement uniformément espacés dans la carte en convergeant vers une distance proche de 0.10, la couleur de la carte thermique devenant uniforme dans la carte.

6.3.5 Evaluation de la capacité de catégorisation

Après avoir évalué les deux propriétés clés de QV et de génération de carte ordonnée, nous terminons notre étude expérimentale en évaluant la capacité de catégorisation de la SO-TCRL. Nous évaluons cette capacité de catégorisation avec les différents schèmes d'étiquetage et de vote présentés dans la section 6.2.5.

Nous évaluons chaque expérience avec 10 exécutions indépendantes. En outre, pour une comparaison équitable avec le modèle de SOM impulsionnelle de Rumbell et al., 2014, nous effectuons lors de chacune des 10 exécutions indépendantes une validation croisée avec la méthode *K-fold* en séparant les bases de données IRIS et WBCD en $K = 5$ blocs, ces bases de données étant celles utilisées dans Rumbell et al., 2014. Les bases de données utilisées sont de complexité modeste, mais cela est à rapporter au degré encore peu mature des SNN dans ce domaine. L'apprentissage est réalisé avec 60 000 points de données choisis aléatoirement.

Table 6.5 – Résultats moyens obtenus sur la base de données IRIS et WBCD en termes d'exactitude.

Dataset	IRIS		WBCD	
Modulation	Spatiale	Spatio-temporelle	Spatiale	Spatio-temporelle
BMU	73.1 ±13.9	74.9 ±14.2	93.2 ±3.3	93.2 ±3.7
SBMU	85.7 ±11.7	86.1 ±10.1	85.3 ±3.8	85.9 ±3.6
SPK-POP	70.0 ±25.4	75.3 ±20.5	65.2 ±10.1	65.0 ±10.3
TMP-POP	92.5 ±6.8	93.0 ±6.3	96.5 ±0.7	96.7 ±0.7

Résultats pour IRIS

IRIS est la première base de données utilisée pour tester les performances en termes de catégorisation de la SO-TCRL. IRIS catégorise les tailles des fleurs de l'iris en trois classes, *Setosa*, *Versicolor* et *Virginica*. Chaque catégorie est représentée avec 50 points de données et chaque point de donnée est défini avec 4 valeurs, induisant un espace d'entrée de 4 dimensions. Nous déterminons de façon automatique le seuil de décharge V_θ des neurones de la couche de représentation avec l'équation 6.6 pour $k = 4$ dimensions.

Les meilleures performances de catégorisation en termes d'exactitude (Fig. 6.15) sont obtenues par notre nouveau schème de vote TMP-POP distribué (93 %) tirant profit de l'activité spatio-temporellement distribuée de la SO-TCRL, suivi du schème localiste SBMU (86.1 %), puis du schème localiste BMU (74.9 %) et enfin du schème distribué, mais ne tenant pas compte de la dimension temporelle SPK-POP (75.3 %). Notons toutefois que SPK-POP pourrait probablement délivrer de meilleures performances si la sparsité était plus élevée dans la couche de représentation, étant donné que SPK-POP est basé sur le nombre d'impulsions émis. En effet dans SO-TCRL, la majorité des neurones de la couche de représentation déchargent (une unique fois) en réponse à une entrée.

Les performances en termes d'exactitude pour IRIS (93 %) sont compétitives vis à vis des travaux connexes (Tab. 6.6), notamment par rapport à la valeur de 90.9 % rapportée par Rumbell et al., 2014.

Nous pouvons constater que l'apport de la modulation temporelle devient plus significatif pour des tâches de catégorisation. En effet, la modulation spatio-temporelle délivre une meilleure exactitude pour tout les schèmes de votes considérés, qu'une pure modulation spatiale. Rappelons que le rôle de ce facteur de modulation temporelle est de minimiser la distance intra-cluster en induisant une grande valeur de modulation, et de maximiser la distance inter-cluster en induisant une faible valeur de modulation. En suivant l'hypothèse de variété, des points de données proches dans la variété de données partagent la même valeur de variable catégorielle (Bengio et al., 2013). L'introduction du facteur de modulation temporelle permet ainsi de mieux regrouper les mêmes valeurs de variable catégorielle et de mieux séparer les valeurs différentes.

Nous pouvons constater dans la figure 6.16 que la SO-TCRL a extrait la structure latente des données en générant des *clusters* de neurones spatialement ségrégués correspondant à une même catégorie (*Setosa*, *Versicolor* et *Virginica*), leurs représentations étant proches dans la variété de données. Rappelons à cet endroit que la topologie de la carte est torique, induisant que les neurones aux extrémités de la carte sont connectés.

Résultats pour WBCD

Wisconsin Breast Cancer Dataset (WBCD) est la deuxième base de données utilisée pour tester les performances de catégorisation du modèle. WBCD est constituée de 683 points de données répartis en deux classes : tumeur bénigne (444 points de données) et tumeur maligne (239 points de données). Chaque point de donnée est défini avec 9 valeurs, induisant un espace d'entrée de 9 dimensions. Nous déterminons de façon automatique le seuil de décharge V_θ des neurones de la couche de représentation avec l'équation 6.6 pour $k = 9$ dimensions.

Les meilleures performances de catégorisation en termes d'exactitude (Fig. 6.15) sont de

2. Tiré de Rumbell et al., 2014.

TABLE 6.6 – Comparaison des meilleures performances de catégorisation en termes d'exactitude pour IRIS. [2]

Non-impulsionnel	% exac.	Impulsionnel	% exac.
k-Means (BOHTÉ et al., 2002)	88.6	RBF (BOHTÉ et al., 2002)	92.6
SOM (BOHTÉ et al., 2002)	85.33	SpikeProp (WU et al., 2006)	96.1
Matlab BP (WADE et al., 2010)	95.5	SWAT (WADE et al., 2010)	95.3
Matlab LM (WADE et al., 2010)	95.7	RBF (GUEORGUIEVA et al., 2006)	89
TEST (YOON et S.-Y. LEE, 1999)	91.7	SNN_{Bako} (BAKÓ, 2010)	83.4
		Spiking SOM (RUMBELL et al., 2014)	90.9
		SO-TCRL	**93.0**

nouveau obtenues par notre nouveau schème de vote TMP-POP (96.7 %), suivi du schème localiste BMU (93.2 %), puis du schème localiste SBMU (85.9 %) et enfin du schème spatialement distribué SPK-POP (65.0 %).

Les performances en termes d'exactitude pour WBCD (96.7 %) sont compétitives vis à vis des travaux connexes (Tab. 6.7), notamment par rapport à la valeur de 97 % rapportée par RUMBELL et al., 2014.

TABLE 6.7 – Comparaison des meilleures performances de catégorisation en termes d'exactitude pour WBCD. [3]

Non-impulsionnel	% exac.	Impulsionnel	% exac.
Matlab BP (WADE et al., 2010)	96.3	SpikeProp (WU et al., 2006)	97
Matlab LM (WADE et al., 2010)	96.7	SWAT (WADE et al., 2010)	95.3
		SNN_{Bako} (BAKÓ, 2010)	89.5
		Spiking SOM RUMBELL et al., 2014	97
		SO-TCRL	**96.7**

Nous pouvons constater dans la figure 6.16 que la SO-TCRL a extrait la structure latente des données en générant des *clusters* de neurones spatialement ségrégués correspondant à une même catégorie (tumeur bénigne et tumeur maligne), leurs représentations étant proches dans la variété de données. C'est une propriété importante des SOM reproduite par notre modèle SO-TCRL.

6.4 Synthèse

Nous avons présenté dans ce chapitre le premier modèle de SOM impulsionnelle complet, la SO-TCRL, conjuguant pour la première fois les trois fonctions clés d'une SOM (KOHONEN, 2013) : une capacité de quantification vectorielle, de génération de cartes ordonnées et de catégorisation sur des bases de données génériques.

La SO-TCRL est basée sur le modèle WD-TCRL opérant à partir de codes temporels et

3. Tiré de RUMBELL et al., 2014.

stockant des représentations dans les délais, plutôt que dans les poids synaptiques. La SO-TCRL étend le modèle WD-TCRL avec un nouveau neuromodulateur spatio-temporel, remplaçant les profils d'excitation-inhibition latéraux des modèles connexes (Hazan et al., 2020 ; Ruf et Schmitt, 1998 ; Rumbell et al., 2014). Le neuromodulateur spatio-temporel est le produit de deux facteurs, un facteur spatial et un facteur temporel. Le facteur spatial impose une contrainte topologique sur le voisinage de la SBMU dans le but de générer des cartes ordonnées, de façon similaire à la SOM de Kohonen. Le facteur temporel est une innovation spécifique aux SNN imposant une contrainte temporelle à partir du temps d'impulsion de la SBMU. Son rôle est de minimiser la distance intra-cluster en induisant une grande valeur de modulation, et de maximiser la distance inter-cluster en induisant une faible valeur de modulation.

Une étude expérimentale approfondie a été menée pour tester les différentes fonctions du modèle. Nos résultats expérimentaux dans la section 6.3.3 attestent que les performances en terme de QV sont préservées par la SO-TCRL, tout en intégrant des fonctions supplémentaires.

Dans la section 6.3.4, la SO-TCRL démontre une bonne capacité de génération de cartes ordonnées. Nous atteignons des performances similaires en termes d'EMDS avec le travail de référence de Rumbell et al., 2014. En outre, ces performances sont obtenues sans avoir recours à une décroissance du rayon de voisinage et/ou du taux d'apprentissage comme dans la SOM de Kohonen. Cela rend possible un apprentissage continu et en ligne (Rougier et Boniface, 2011).

Enfin nous avons terminé notre étude expérimentale en évaluant la capacité de catégorisation de la SO-TCRL dans la section 6.3.5. Afin d'évaluer la catégorisation, nous avons testé plusieurs schèmes d'étiquetage et de vote. Nous avons également introduit à cette fin notre nouveau schème de vote TMP-POP. Notons que TMP-POP est suffisamment générique pour pouvoir être appliqué à tout SNN traitant des codes temporels. Il s'agit d'un schème distribué tirant profit de l'information spatio-temporellement distribuée dans les motifs impulsionnels produits par la couche de représentation. Le shème d'étiquetage et de vote TMP-POP est basé sur une adaptation d'un softmax au traitement de codes temporels.

L'utilisation de TMP-POP en conjonction avec la SO-TCRL délivre les meilleures performances en termes d'exactitude parmi les différents schèmes testés. En outre, l'utilisation d'une modulation spatio-temporelle plutôt que purement spatiale, augmente sensiblement l'exactitude des prédictions, validant ainsi le rôle assuré par la modulation temporelle de minimisation de la distance intra-cluster et de maximisation de la distance inter-cluster. La SO-TCRL démontre de bonnes capacités de catégorisation, compétitives avec les travaux connexes. En outre, nos expériences montrent que la SO-TCRL est capable d'extraire la structure latente des données en générant des *clusters* de neurones spatialement ségrégués correspondant à une même catégorie.

Enfin notre modèle SO-TCRL est à notre connaissance le premier modèle de SOM impulsionnelle générique. Par exemple il n'est pas nécessaire de régler manuellement par essais et erreurs la magnitude maximale des poids synaptiques en fonction de la dimension des données d'entrée comme dans Rumbell et al., 2014. Dans la SO-TCRL, seul un hyperparamètre doit être adapté à la dimension des données, et nous avons proposé une méthode simple et efficace pour le déterminer automatiquement. La méthode a été empiriquement validée par les performances obtenues. Ainsi nous avons utilisé le même jeu d'hyperparamètres pour toutes les expériences menées sur des distributions et des données de dimensions variées. En outre, nous sommes en mesure de décoder explicitement la distribution des paramètres appris (dans notre cas des délais), sans avoir recours à des méthodes d'inférences basées sur des associations stimulus-réponse. Ces méthodes peuvent notamment souffrir de la malédiction de la dimension des données d'entrées.

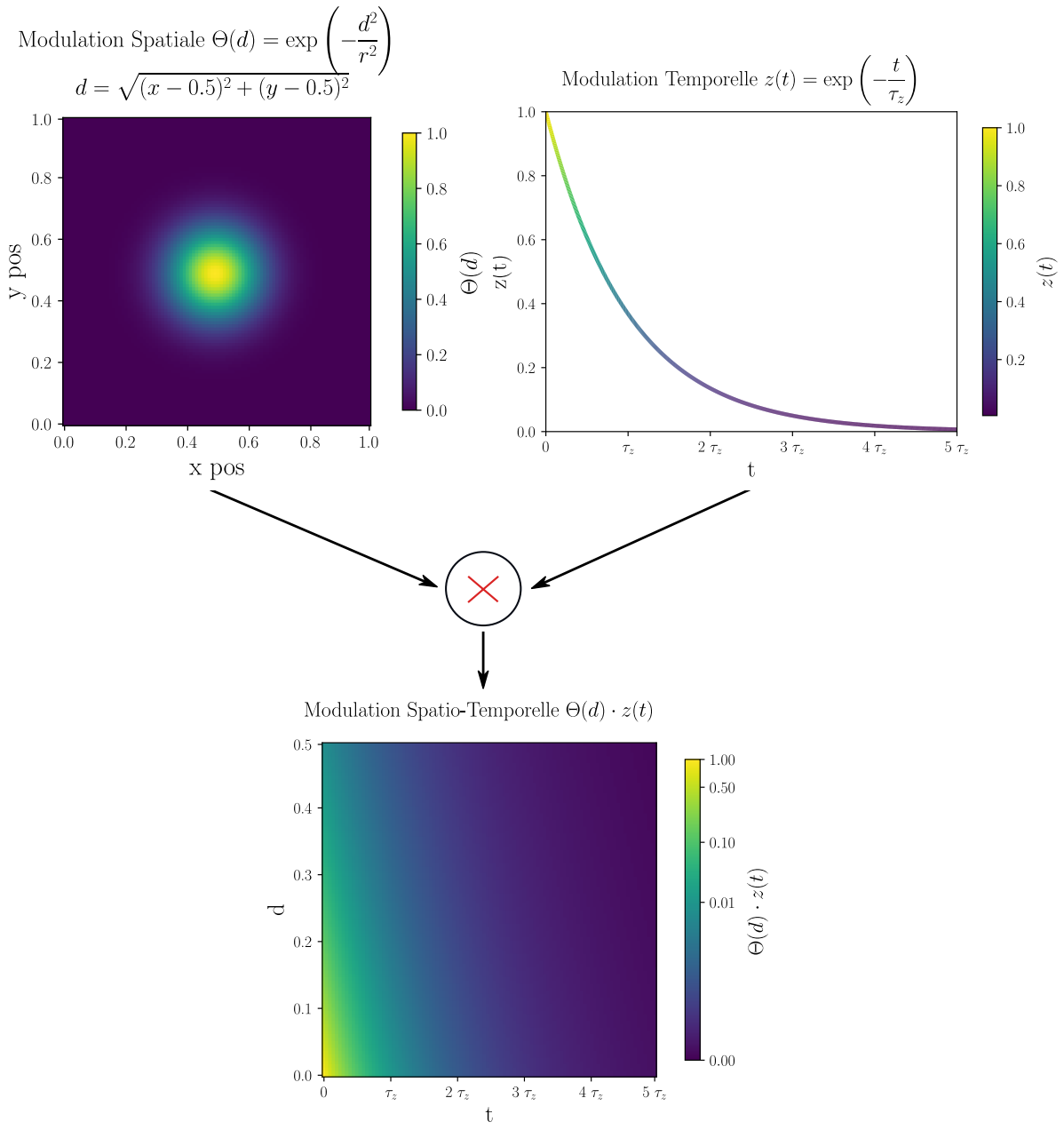

FIGURE 6.2 – Le neuromodulateur est le produit de deux facteurs de modulations : un facteur spatial et un facteur temporel. Le neuromodulateur est donc spatio-temporel. Le facteur spatial est un noyau gaussien centré sur la position de la SBMU dans la carte. Il impose une contrainte topologique sur le voisinage spatial de la SBMU. La distance d est mesurée par rapport à la position de la SBMU, ici supposée être en coordonnée (0.5,0.5). Le facteur temporel est un noyau de décroissance exponentielle implémenté avec les traces postsynaptiques $z(t)$. Il impose une contrainte de proximité temporelle relativement au temps d'impulsion de la SBMU. La valeur maximale des deux facteurs de modulation (1.0) correspond à la position de la SBMU dans la carte, et au temps d'impulsion à $t = 0$ de la SBMU, respectivement.

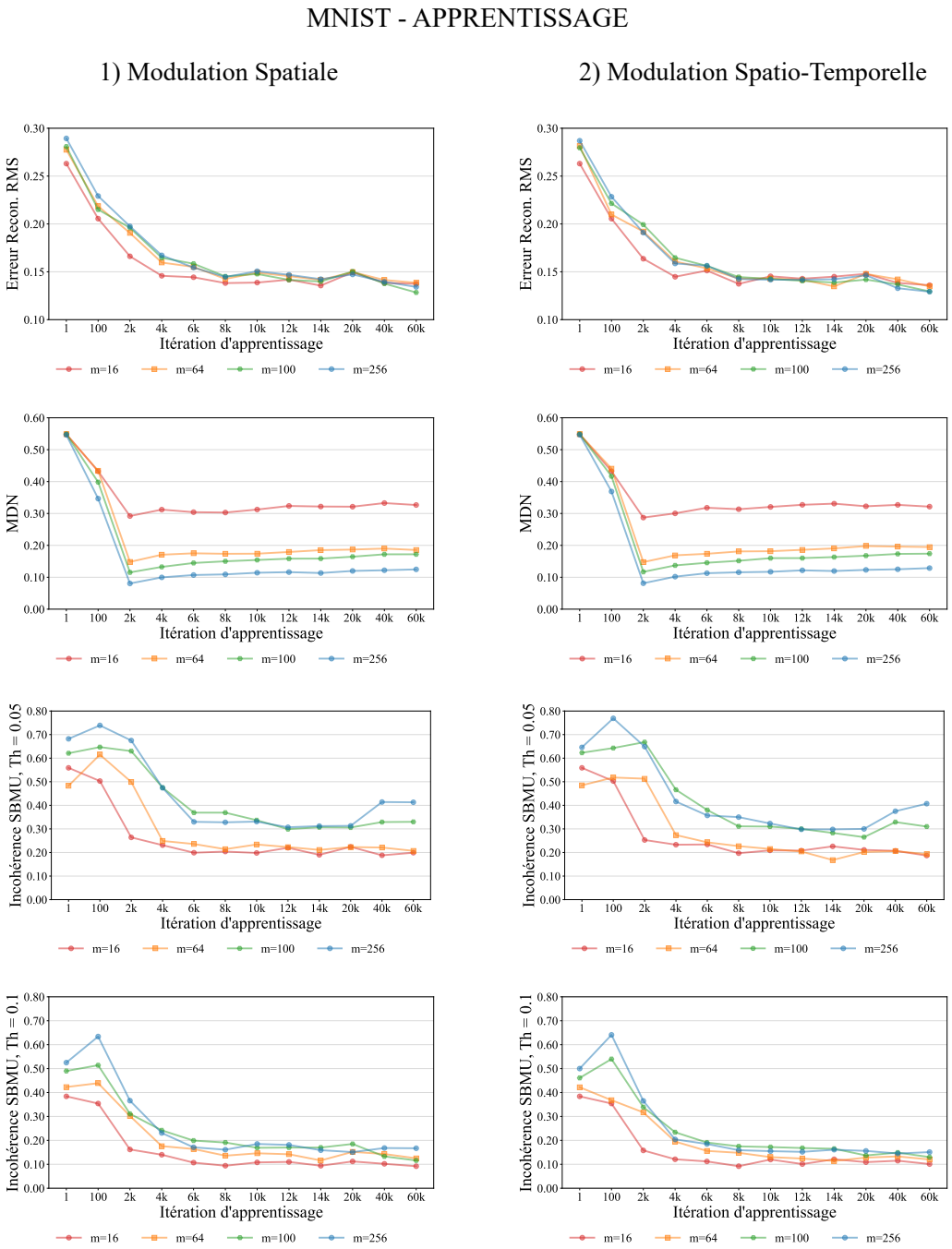

FIGURE 6.3 – Performances pour MNIST durant la phase l'apprentissage. Performances du SNN après 1, 100,..., 60k itérations pour $m \in \{16, 32, 64, 128, 256\}$ neurones, en termes d'erreur de reconstruction RMS, de MDN, et d'incohérence dans l'élection auto-organisée de la SBMU pour un seuil de décision Th=5 % et Th=10 %.

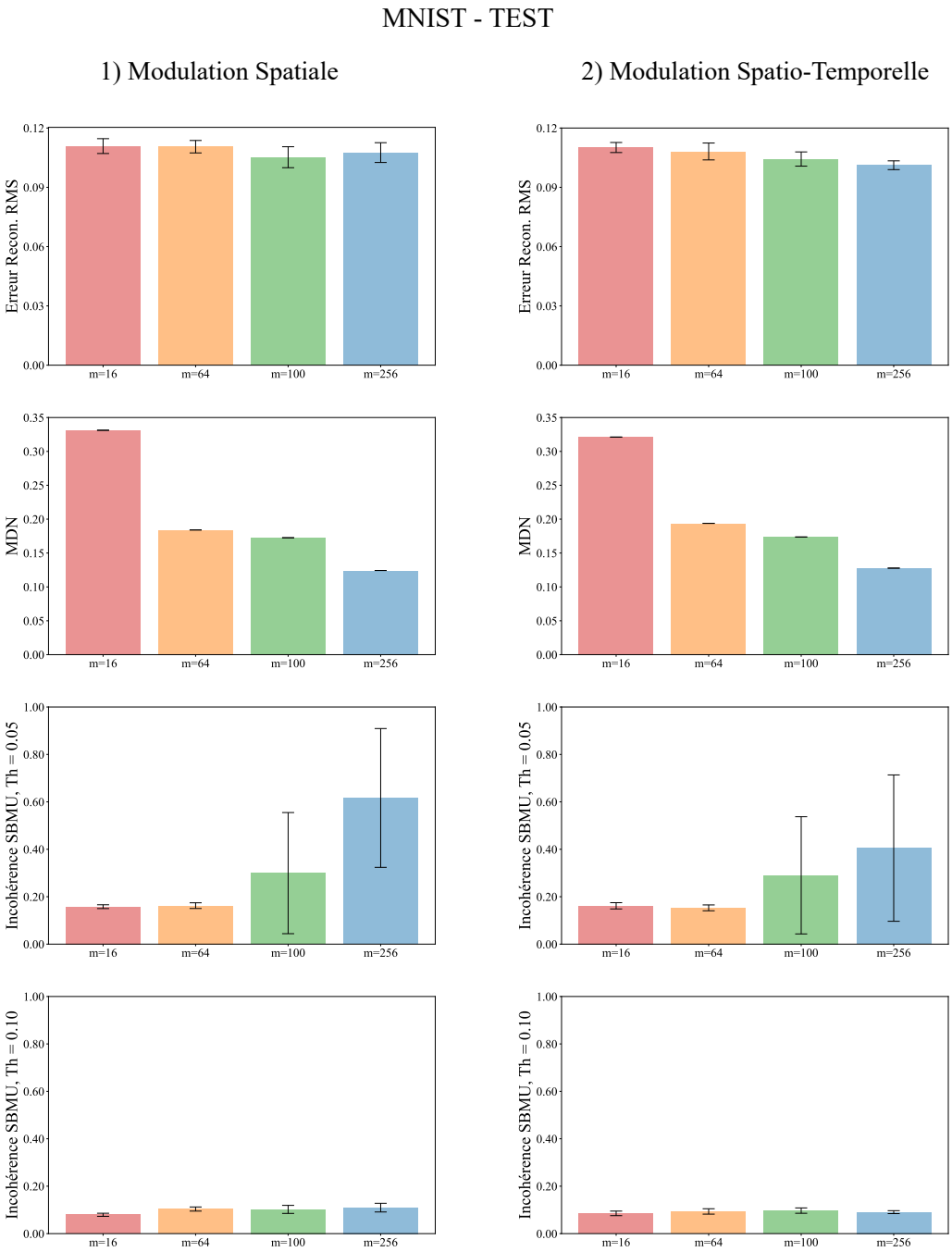

FIGURE 6.4 – Performances pour MNIST durant la phase de test. Performances pour $m \in \{16, 64, 100, 256\}$ neurones, en termes d'erreur de reconstruction RMS, de MDN, et de cohérence dans l'élection de la SBMU pour un seuil de décision Th=5 % et Th=10 %. Les barres indiquent l'écart-type.

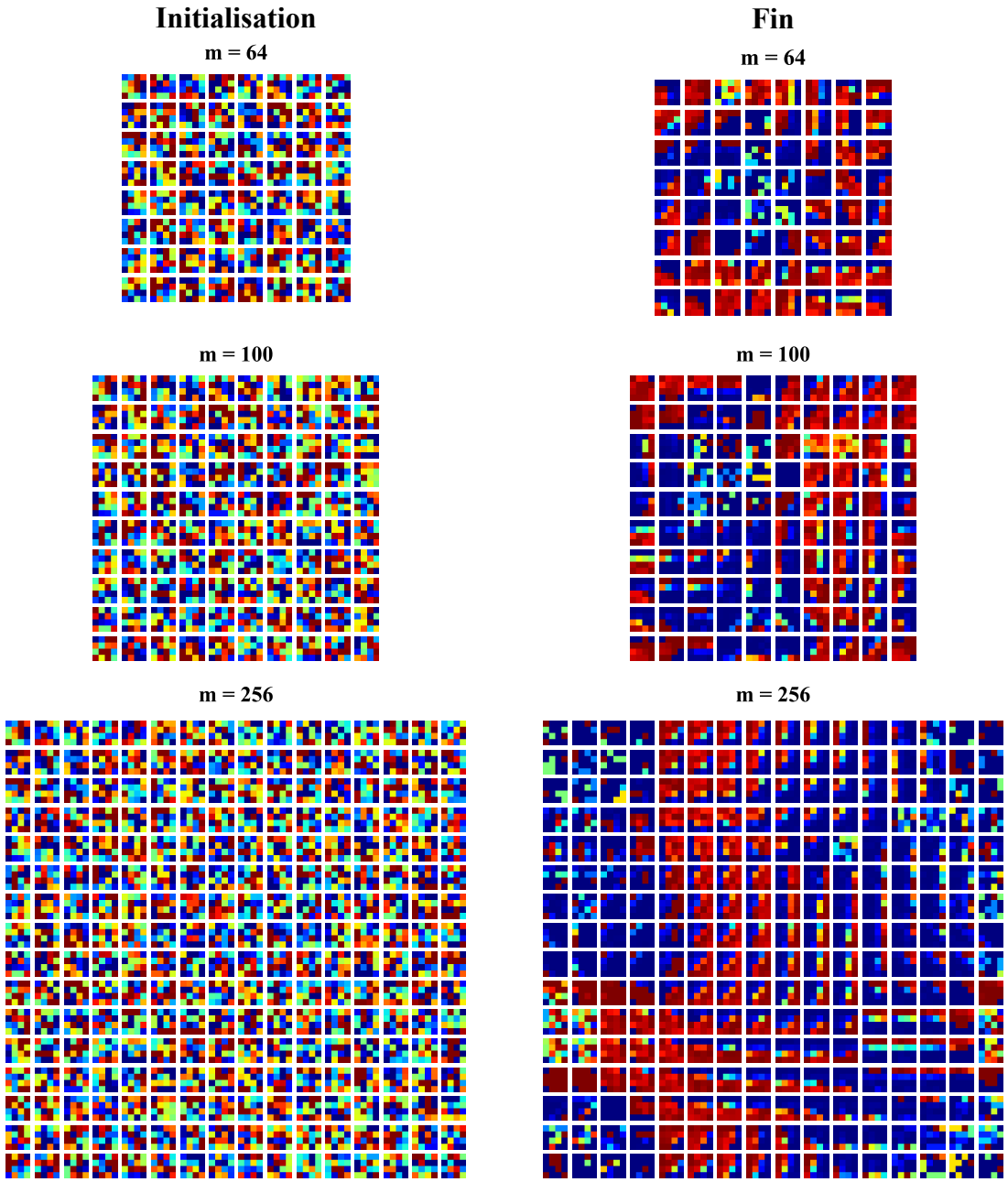

FIGURE 6.5 – Vecteurs codes des neurones pour des cartes de 8*8, 10*10 et 16*16 neurones à l'initialisation et après la phase d'apprentissage sur la base de données MNIST avec une modulation spatio-temporelle. Le gradient rouge-bleu indique les valeurs maximales-minimales de chaque pixel reconstruit.

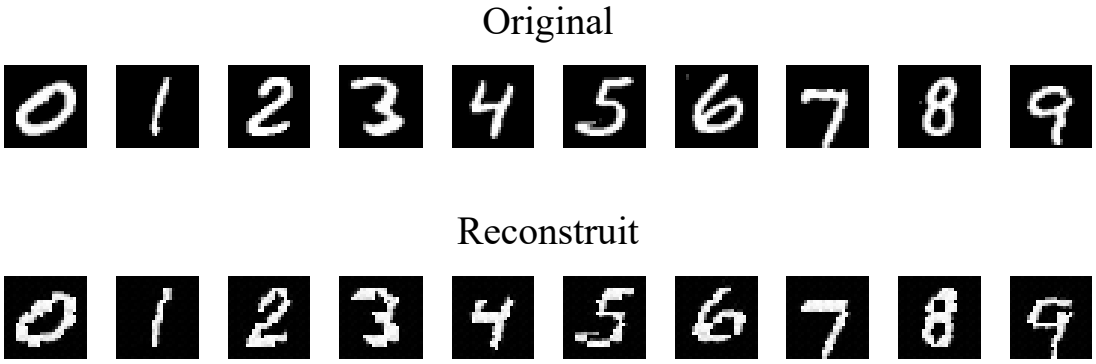

FIGURE 6.6 – Ensemble d'images de MNIST présentées en entrée de la SO-TCRL et reconstruites par les vecteurs codes des neurones de la couche de représentation pour $m = 100$ neurones.

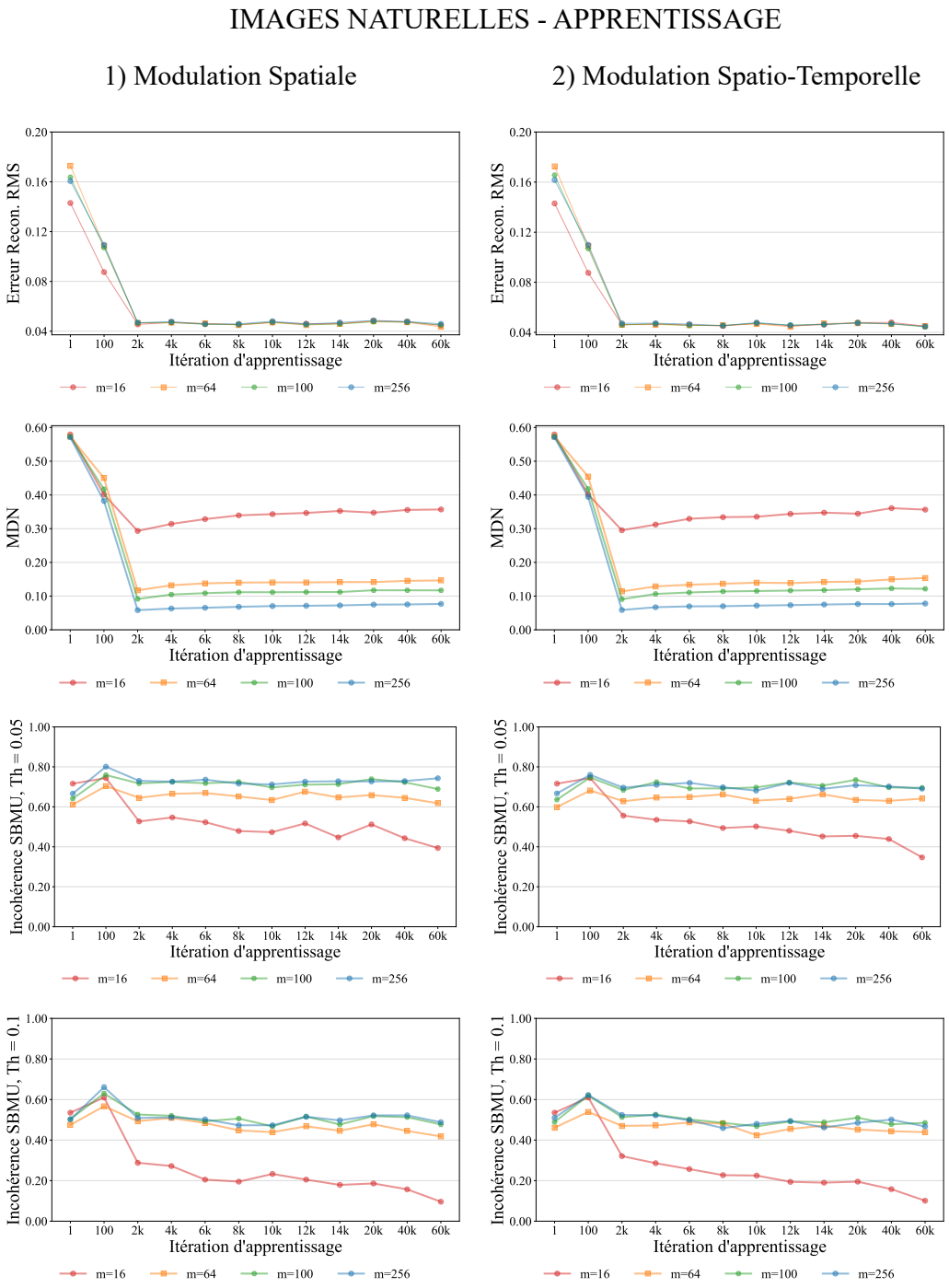

FIGURE 6.7 – Performances pour les images naturelles durant la phase l'apprentissage. Performances pour $m \in \{16, 64, 100, 256\}$ neurones, en termes d'erreur de reconstruction RMS, de MDN, et de cohérence dans l'élection de la SBMU pour un seuil de décision Th=5 % et Th=10 %.

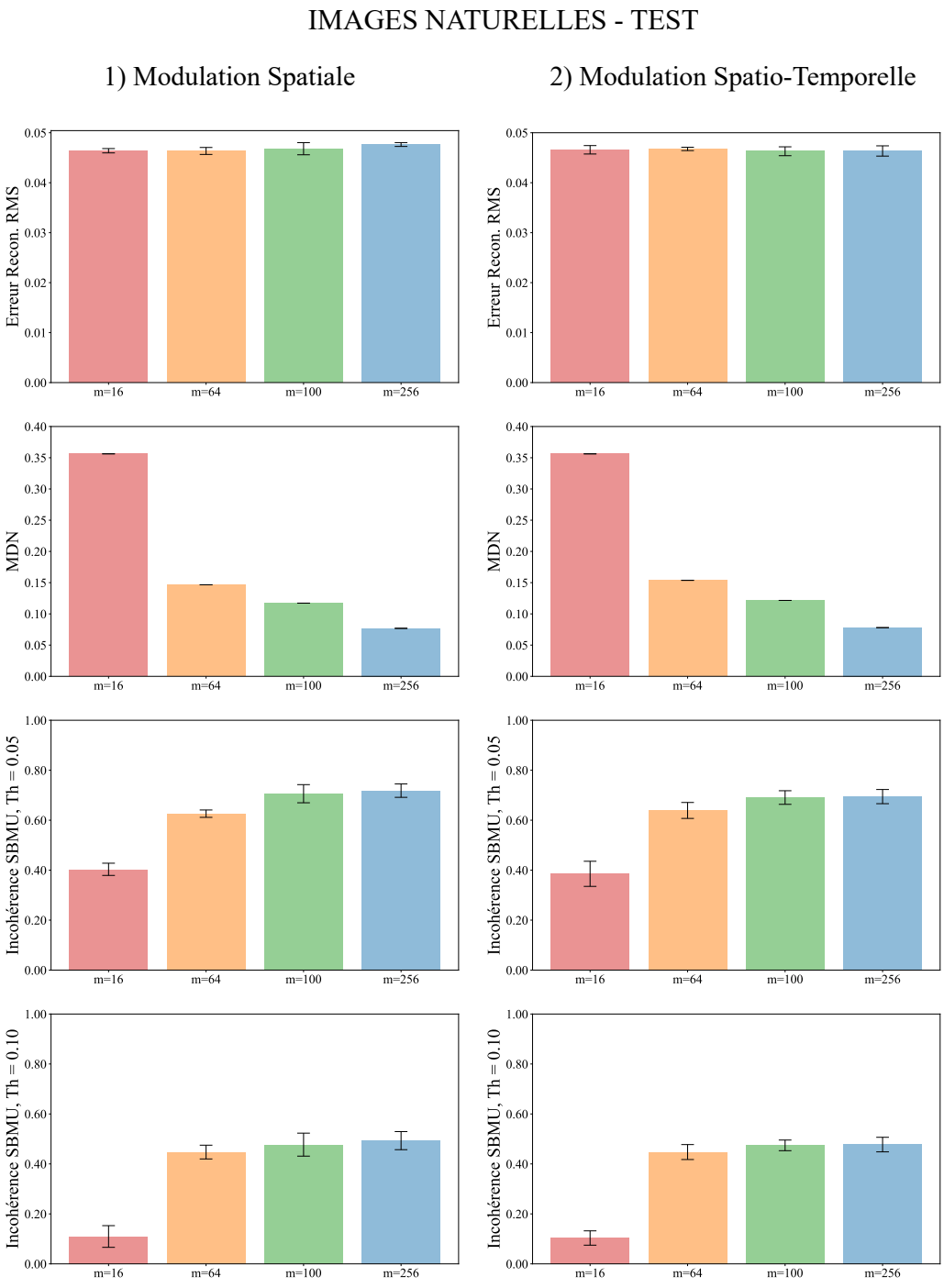

FIGURE 6.8 – Performances pour la base d'images naturelles durant la phase de test. Performances pour $m \in \{16, 64, 100, 256\}$ neurones, en termes d'erreur de reconstruction RMS, de MDN, et de cohérence dans l'élection de la SBMU pour un seuil de décision Th=5 % et Th=10 %. Les barres indiquent l'écart-type.

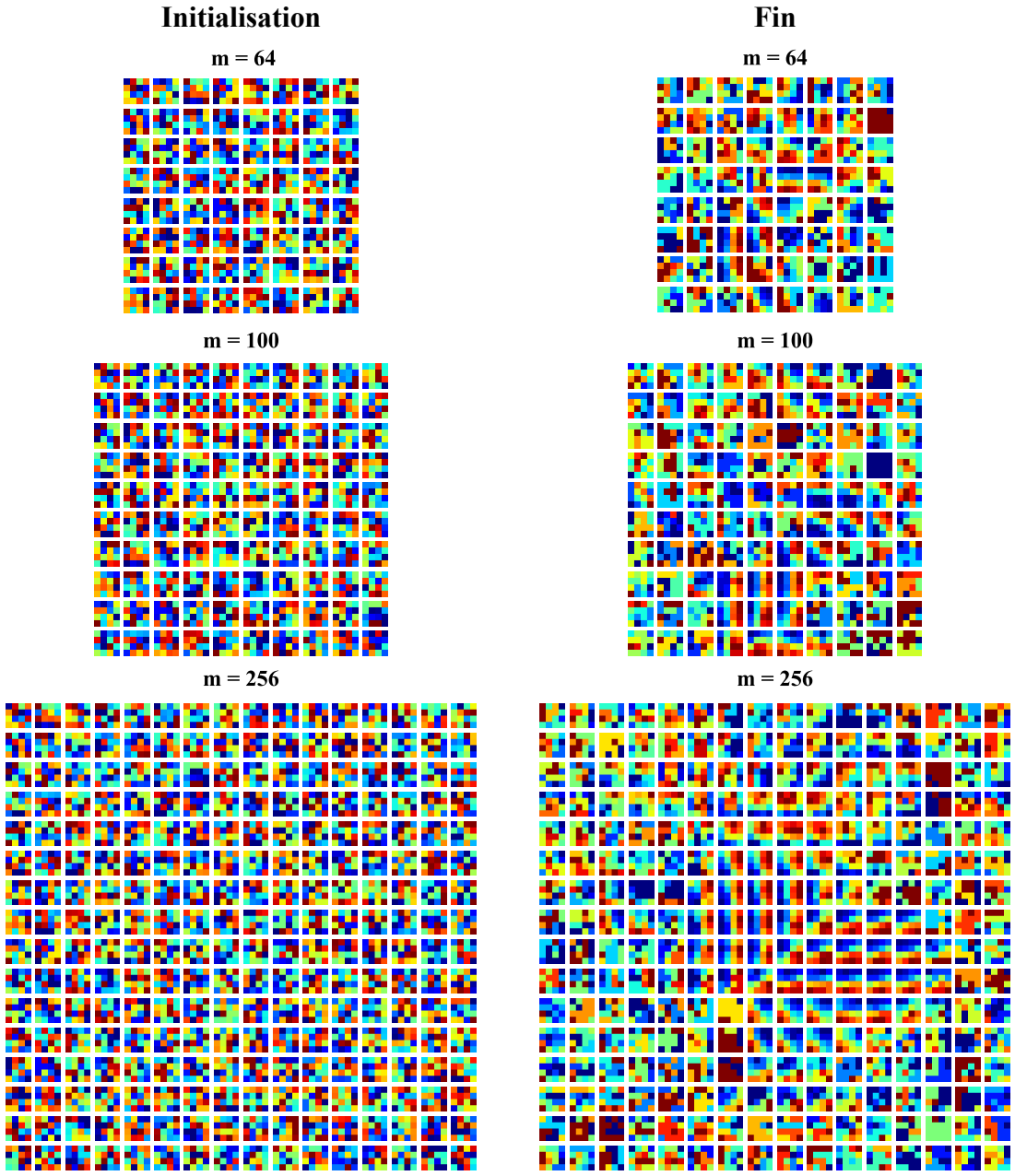

FIGURE 6.9 – Vecteurs codes des neurones pour des cartes de 8*8, 10*10 et 16*16 neurones à l'initialisation et après la phase d'apprentissage sur la base de données d'images naturelles avec une modulation spatio-temporelle. Le gradient rouge-bleu indique les valeurs maximales-minimales de chaque pixel reconstruit.

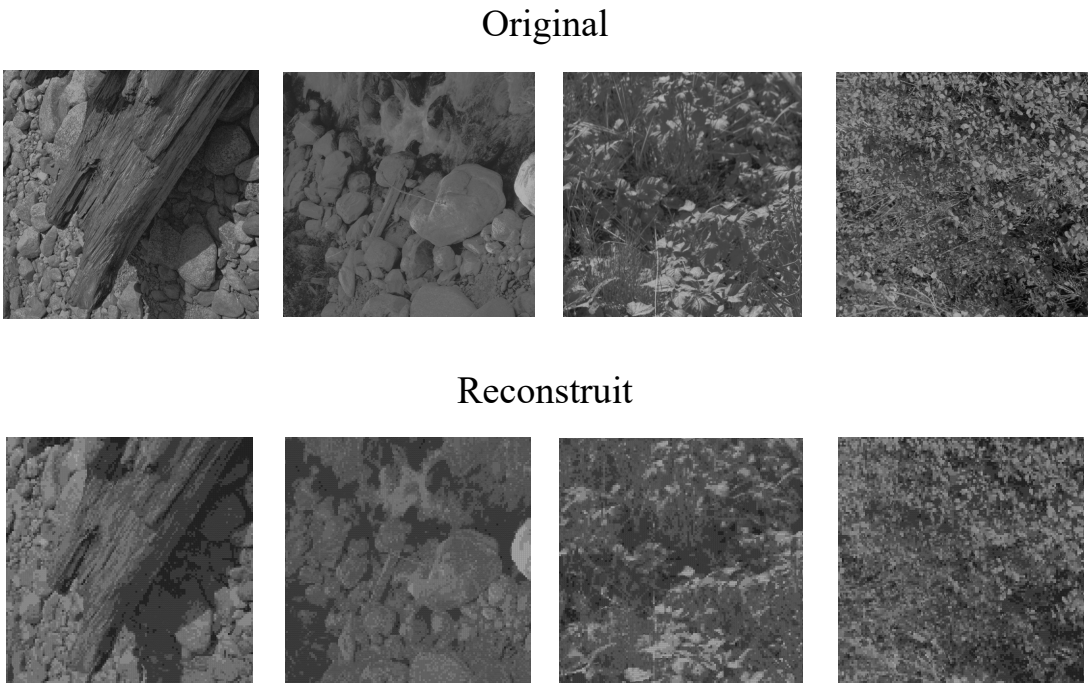

FIGURE 6.10 – Ensemble d'images naturelles présentées en entrée de la SO-TCRL et reconstruites par les vecteurs codes des neurones de la couche de représentation pour $m = 100$ neurones et un facteur de modulation spatio-temporel.

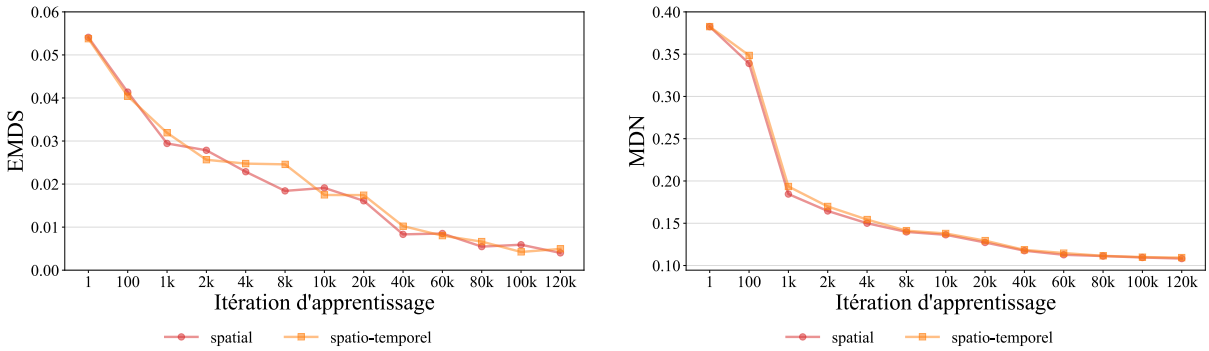

FIGURE 6.11 – Phase d'apprentissage. Evaluation de la qualité des cartes générées après 1, 100,..., 120k itérations en termes de EMDS et MDN avec une modulation spatiale et une modulation spatio-temporelle.

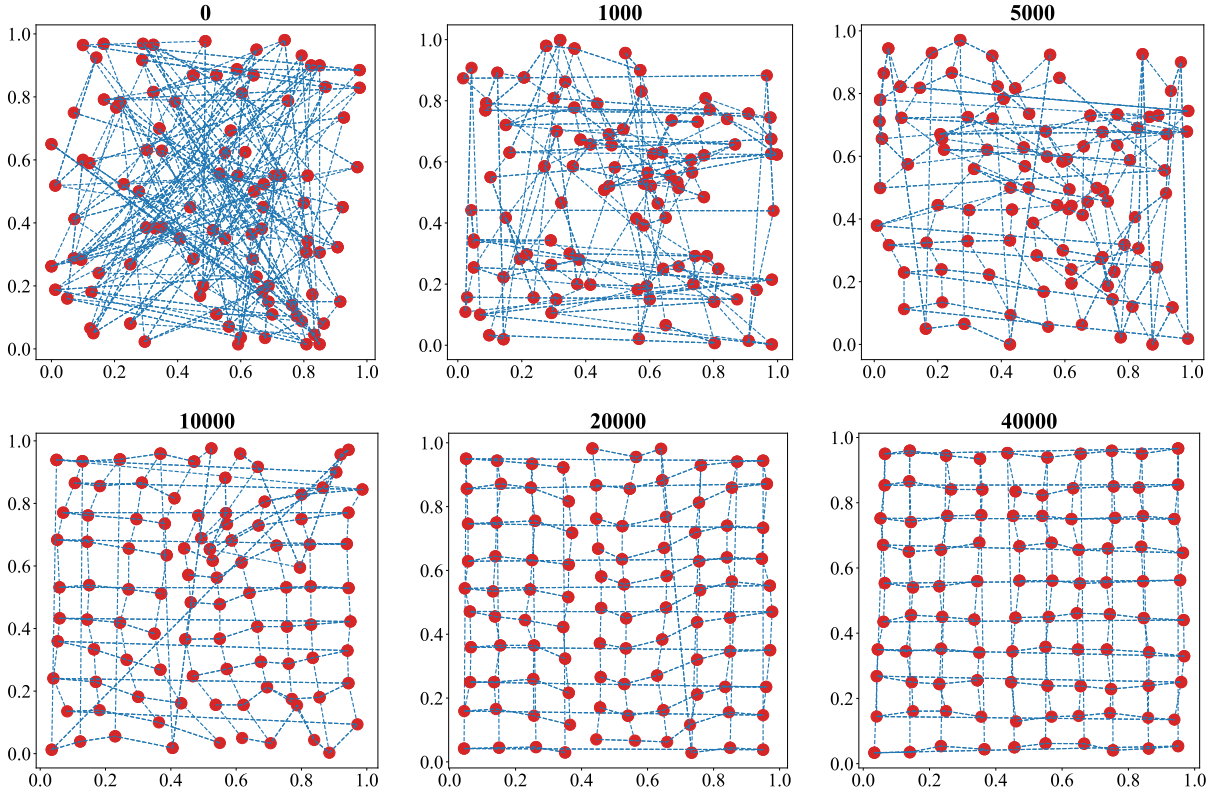

FIGURE 6.12 – Phase d'apprentissage. Exemple de génération de carte ordonnée après 0, 1000, 5000, ..., 40000 itérations pour une topologie torique. Les vecteurs codes des neurones sont projetés dans l'espace d'entrée en 2D (points rouges). Les connexions des neurones avec leurs 4 neurones voisins sont affichés avec les lignes pointillées bleues.

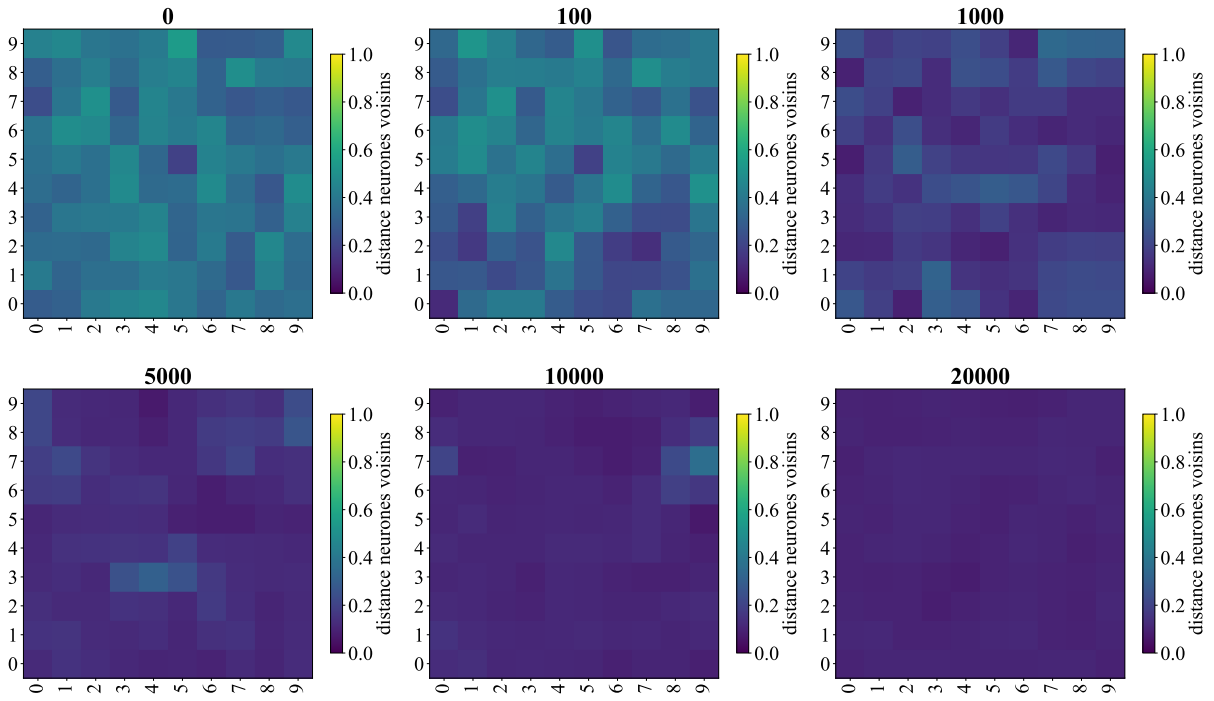

FIGURE 6.13 – Phase d'apprentissage. Exemple de génération de carte ordonnée après 0, 100, 1000, ..., 20000 itérations. La distance entre chaque neurone et ses 4 voisins décroît, jusqu'à créer un continuum uniforme dans la carte.

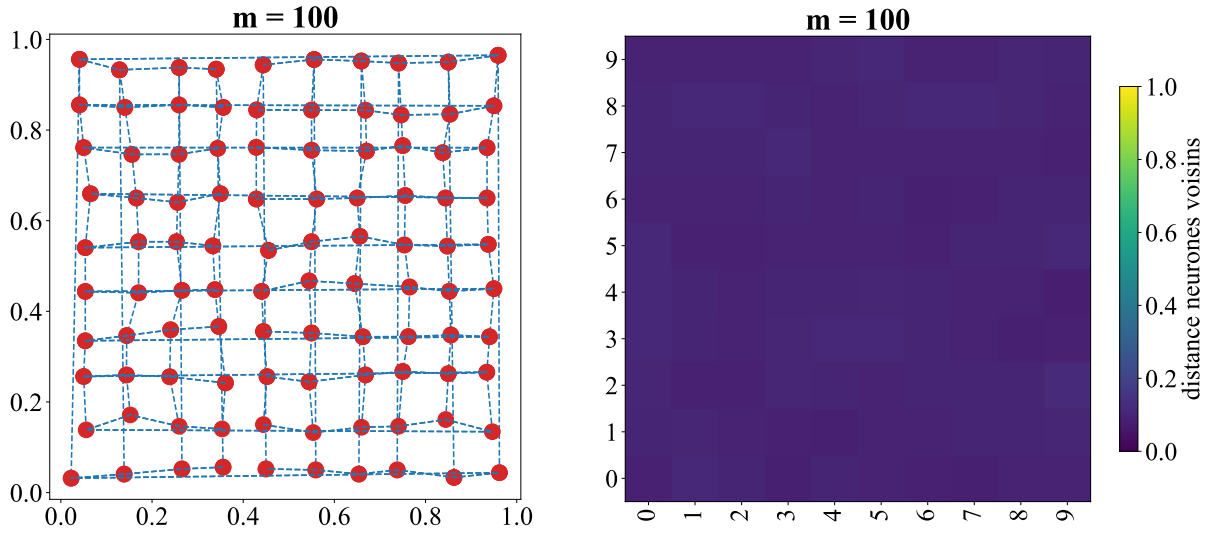

FIGURE 6.14 – Après la phase d'apprentissage : exemple de carte ordonnée. La figure à gauche illustre les vecteurs codes appris par les neurones, ainsi que les connexions des neurones à leurs voisins dans une topologie torique. La figure à droite illustre la distance moyenne d'un neurone à chacun de ses quatre voisins.

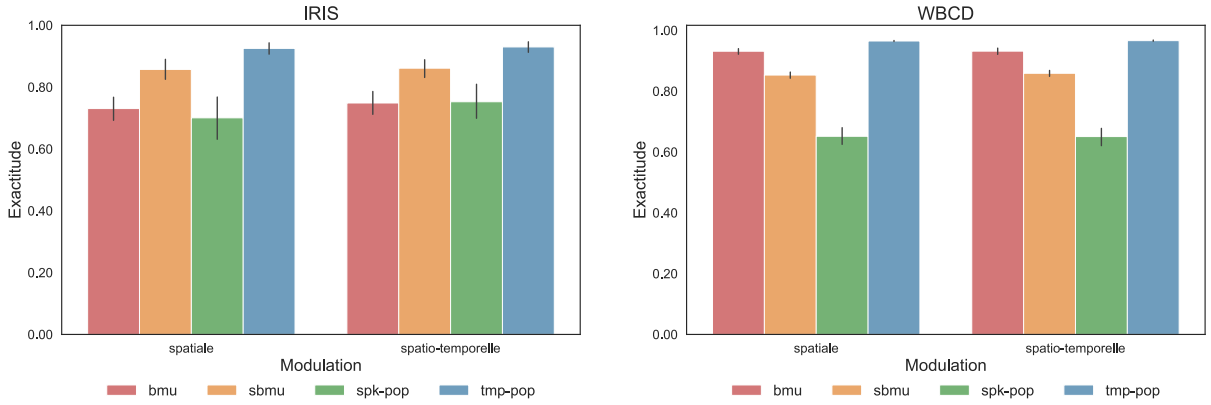

FIGURE 6.15 – Performances en termes d'exactitude pour une modulation spatiale et spatio-temporelle ainsi que pour différents schèmes de votes.

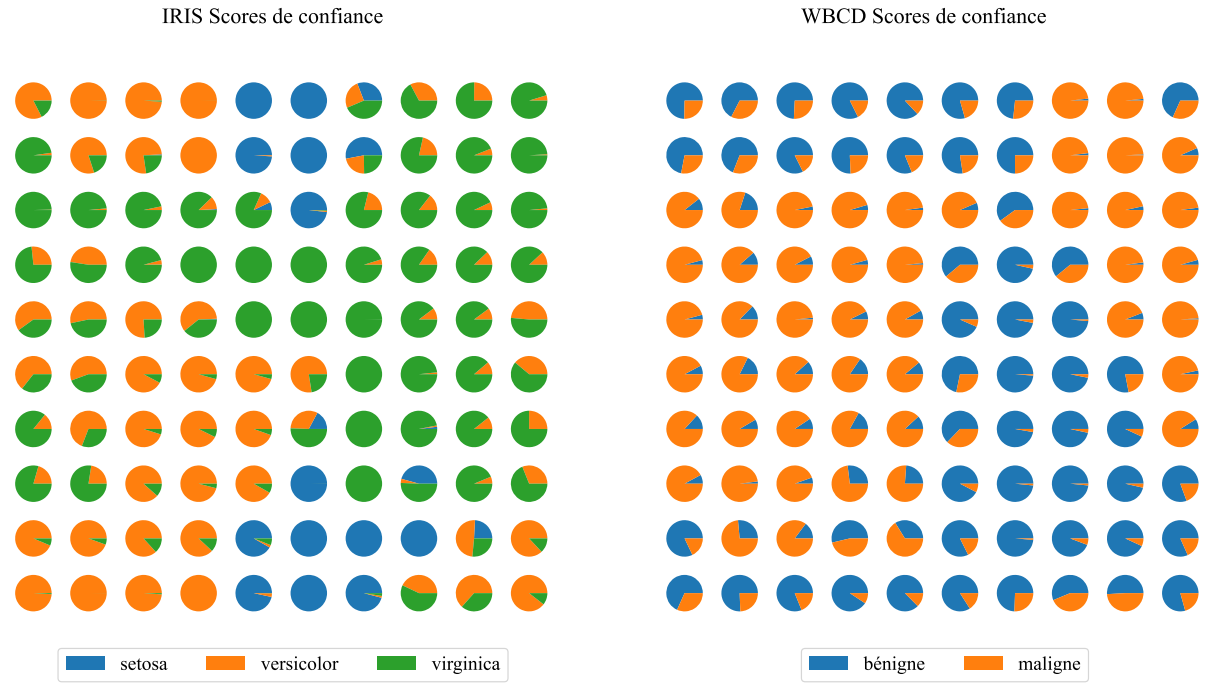

FIGURE 6.16 – Exemples de scores de confiances obtenus après la phase d'étiquetage, pour une carte de 10*10 neurones, une modulation spatio-temporelle et le schème de vote TMP-POP. Il s'agit ici d'une visualisation des scores de confiances $\text{conf}(i, h)$ générés par TMP-POP, avec i indiçant un neurone et h indiçant une valeur de catégorie. Ces scores de confiances ont été accumulés et normalisés pour les différentes catégories, i.e. la somme des $\text{conf}(i, h)$ donne 1.

Chapitre 7

Conclusion et perspectives

7.1 Conclusion

Le but de cette thèse était de développer des règles d'apprentissage événementielles, spatialement et temporellement locales, capables d'extraire efficacement des représentations de codes temporels. L'utilisation de codes temporels exploitant le temps pour représenter l'information est notamment motivée par la transmission d'une unique impulsion par neurone, tandis que la grande majorité des travaux connexes sont basés sur le nombre d'impulsions émis (taux de décharge). Alors que l'apprentissage de représentations est un domaine mature pour les ANN, atteindre de bonnes performances reste un défi pour les SNN. Nos travaux ont permis de significativement réduire cet écart, en particulier en termes d'erreur de reconstruction.

Après une étude comparative et analytique montrant que le codage temporel par population est le plus performant, e.g. exponentiellement plus efficace qu'un codage par taux de décharge, nous voulions instancier ce code.

Nous avons pour cela réalisé une analyse approfondie d'une méthode de codage temporel par population de neurones. Nous avons introduit une méthode de décodage basée sur la moyenne circulaire des directions préférentielles des neurones, pondérée par leurs latences relatives. L'utilisation de cette méthode de décodage a ensuite été étendue au décodage des représentations apprises par nos différents modèles dans les paramètres synaptiques. Nous avons ensuite analysé l'information délivrée par ce code neuronal grâce à l'entropie, et mis en évidence les relations liant le nombre de bits d'informations transmis avec le nombre de neurones ainsi que la largeur de la fenêtre temporelle. Ce code neuronal spatio-temporellement distribué présente ainsi l'avantage de pouvoir jouer sur deux paramètres indépendants pour augmenter la quantité d'information transmise : la fenêtre temporelle et/ou le nombre de neurones. Un autre avantage est que les premières impulsions transmettent la plus grande quantité d'information, permettant une prise de décision à la fois rapide et fiable pour les neurones recevant ces impulsions. Enfin nous avons analysé la structure de ce code. La structure du code révèle que l'activité neuronale est confinée à un sous-espace de l'espace d'états (*neural manifold*) prenant la forme d'une boucle courbée. Cette structure préserve les relations de proximité pour des données circulaires dans un espace spatio-temporel de haute dimension. Cette propriété est particulièrement pertinente pour toute forme d'apprentissage (e.g. *clustering*) qui pose une hypothèse essentielle : deux points proches ont une sémantique proche. La méthode d'encodage offre un compromis raisonnable entre structure (réduisant la quantité d'information en confinant à un sous-espace de l'espace d'états) et

diversité (augmentant la quantité d'informations en tendant vers une distribution uniforme) du code neuronal.

Nous avons ensuite proposé et analysé trois modèles de neurones impulsionnels pour l'apprentissage de représentations. Un premier modèle *Weight - Temporally Coded Representation Learning* (W-TCRL) basé sur de nouvelles règles événementielles et locales a d'abord été proposé. Ce modèle est capable d'extraire des représentations de codes temporels en stockant des centroïdes dans les poids synaptiques. La règle intègre un module clé de QV adapté au traitement de codes temporels et ciblant les poids synaptiques. La distribution des latences relatives est apprise dans la distribution des poids synaptiques, en maximisant le pouvoir causal des neurones afférents transmettant les premières impulsions, premières impulsions délivrant la plus grande quantité d'information. La nature du code et de la règle d'apprentissage a également pour effet de permettre aux neurones en aval de prendre une décision à la fois rapide et fiable, avant d'avoir intégré l'intégralité des impulsions dans leurs potentiels membranaires, accélérant ainsi la vitesse de traitement du SNN. WD-TCRL améliore significativement l'état de l'art en termes d'erreur de reconstruction, tout en opérant avec des codes temporels plutôt qu'avec des codes en taux de décharge. Il s'agit à notre connaissance du premier modèle événementiel et local capable d'extraire des représentations de codes temporels en visant une faible erreur de reconstruction.

Un deuxième modèle *Weight Delay - Temporally Coded Representation Learning* (WD-TCRL) a ensuite été présenté. Ce second modèle est basé sur deux nouvelles règles événementielles et locales de type STDP, capable d'extraire des représentations de codes temporels. La première règle STDP cible les délais synaptiques et apprend les temps relatifs des impulsions dans les délais de transmission, créant une correspondance entre la dimension physique des entrées (le temps) et des paramètres plastiques (les délais). Les représentations sont ainsi stockées non pas dans les poids synaptiques, mais dans les délais de transmission opérant intrinsèquement dans la dimension temporelle. Cette règle intègre un module de QV adapté aux codes temporels et ciblant les délais, ainsi qu'un module de régularisation favorisant les faibles valeurs de délais. Le module de QV minimise la distance temporelle entre les délais et l'intervalle de temps entre une impulsion pré et postsynaptique. Cela a pour effet de maximiser la synchronie au niveau du terminal postsynaptique et d'activer un mode opératoire clé des neurones impulsionnels : la détection de coïncidences temporelles. Une seconde règle STDP adapte les poids synaptiques en fonction de la variabilité temporelle des caractéristiques, estimée de façon événementielle, locale et en ligne. Les règles d'adaptation des poids et des délais opèrent de manière complémentaire. WD-TCRL délivre des performances similaires à W-TCRL en termes d'erreur de reconstruction. WD-TCRL est à notre connaissance le premier modèle événementiel et local capable d'extraire des représentations de codes temporels et de les stocker dans les délais plutôt que dans les poids synaptiques, en visant une faible erreur de reconstruction.

Un troisième et dernier modèle *Self-Organizing Temporally Coded Representation Learning* (SO-TCRL) étend le modèle WD-TCRL du chapitre 5 avec un nouveau neuromodulateur spatio-temporel, de sorte à générer des cartes topographiquement ordonnées, où des neurones spatialement proches partagent des représentations proches. La génération de cartes ordonnées étend ainsi le principe de localité au voisinage spatial des neurones. Le neuromodulateur spatio-temporel est le produit de deux facteurs : un facteur spatial et un facteur temporel. Le facteur spatial impose une contrainte topologique sur le voisinage de la SBMU dans le but de générer des cartes ordonnées, de façon similaire à la SOM de Kohonen (Kohonen, 2013). Le facteur temporel est une innovation spécifique aux SNN imposant une contrainte temporelle à partir du temps d'impulsion de la SBMU. Son rôle est de minimiser la distance intra-cluster en induisant

une grande valeur de modulation, et de maximiser la distance inter-cluster en induisant une faible valeur de modulation. Comparé à une pure modulation spatiale, augmenter le facteur spatial avec un facteur temporel permet d'améliorer sensiblement les performances obtenues, notamment en terme d'exactitude dans des tâches de catégorisation. SO-TCRL est à notre connaissance le premier modèle complet de *Self-Organizing Map* (SOM) impulsionnelle conjuguant 1) une capacité de quantification vectorielle, 2) de génération de cartes ordonnées et 3) de catégorisation pour des bases de données génériques. Les performances obtenues sur ces différentes fonctions sont compétitives vis-à-vis des travaux connexes implémentant au moins une de ces fonctions. En outre, la SO-TCRL est versatile : elle ne requiert pas de réglage fin et manuel d'hyperparamètres comme l'intervalle de variation des poids en fonction de la dimension des données d'entrées comme dans Rumbell et al., 2014.

En lien avec l'étude des capacités de catégorisation de SO-TCRL, une nouvelle méthode générique d'étiquetage et de vote *Temporal Population* (TMP-POP) a également été introduite pour des SNN traitant des codes temporels. TMP-POP est une méthode basée sur un softmax adapté au traitement de codes temporels. Elle est utilisée pour la catégorisation, assignant des scores de confiances aux neurones en fonction de leurs temps d'impulsions relatifs. Les premiers neurones qui déchargent se voient attribuer les plus hauts scores de confiance. Cette méthode extrait ainsi l'information contenue dans les motifs impulsionnels spatio-temporels de la couche de représentation pour des tâches de catégorisation. TMP-POP a été utilisée pour évaluer notre nouveau modèle de SOM impulsionnelle, la SO-TCRL, sur des tâches de catégorisation. Ce schème d'étiquetage et de vote spatio-temporellement distribué a systématiquement délivré les meilleures performances en termes d'exactitude comparativement aux autres schèmes testés, notamment des schèmes localistes.

7.2 Perspectives

7.2.1 Perspectives à court terme

Les modèles présentés dans ce manuscrit sont le résultat de tout un cheminement ayant exploré différentes pistes de recherche. D'autres pistes se sont présentées mais n'ont pas encore été explorées. Ainsi nous évoquons ici plusieurs perspectives immédiates d'amélioration des différents modèles.

La méthode d'encodage temporelle utilise une population de neurones pour représenter l'information dans des motifs spatio-temporels. Le ratio de quantité d'information délivrée par unité de temps pourrait être amélioré en optimisant les hyperparamètres de la couche d'encodage tels que le seuil de décharge des neurones, la constante de temps membranaire ou la période réfractaire des neurones. Cela permettrait d'augmenter la vitesse de transmission de l'information ou le nombre de bits transmis.

Les modèles W-TCRL et WD-TCRL utilisent des connexions latérales inhibitrices tous-à-tous coûteuses pour instancier un apprentissage compétitif dans la couche de représentation. Il serait avantageux d'utiliser un neurone inhibiteur (ou une couche inhibitrice) connecté à la couche de représentation. Nous passerions ainsi d'une complexité en $O(n^2)$ à $O(n)$. En outre, le mécanisme homéostatique augmentant le niveau de sparsité dans la couche de représentation pourrait être instancié au niveau du seuil de décharge du neurone inhibiteur. Ainsi la valeur du seuil de décharge serait initialement haute, de sorte qu'un grand pourcentage de neurones soient

initialement recrutés, puis diminuerait progressivement de sorte à recruter de moins en moins de neurones et spécialiser les vecteurs codes des neurones.

Actuellement l'initialisation aléatoire des délais de la SO-TCRL est cantonnée à un petit intervalle de variation de sorte à faciliter la génération de cartes ordonnées. Une étude approfondie de la sensibilité aux conditions initiales de la SO-TCRL serait souhaitable. La SO-TCRL permet théoriquement d'apprendre en continu, rejoignant les thématiques du *lifelong learning* (Parisi et al., 2019), en utilisant un rayon de voisinage et des taux d'apprentissage constants, contrairement à la SOM de Kohonen. Il serait désirable de tester cette capacité d'apprentissage en continu en évaluant la capacité de la SO-TCRL à s'adapter à des distributions de données non-stationnaires.

7.2.2 Perspectives à long terme

Les modèles développés dans cette thèse offrent de nombreuses perspectives. Nous en suggérons plusieurs que nous catégorisons en trois axes, bien qu'ils se recoupent au moins partiellement : 1) calcul neuromorphique, 2) applications, 3) modèles.

Calcul neuromorphique

En termes de calcul neuromorphique, les différents modèles développés sont basés sur des règles événementielles spatialement et temporellement locales en bonne adéquation avec le mode opératoire des processeurs neuromorphiques. Un objectif serait donc d'implémenter ces modèles *in silico* pour bénéficier de l'accélération matérielle ainsi que de l'efficacité énergétique induite, fortement décuplée par l'utilisation de codes temporels. Si nous considérons le cas du processeur neuromorphique Loihi (Davies et al., 2018), les cœurs neurosynaptiques sont limités à opérer avec des règles d'apprentissage basées sur des sommes de produits. Cela semble à première vue fortement restreindre l'expressivité du calcul et ne pas permettre l'implémentation des règles des modèles WD-TCRL et SO-TCRL utilisant une opération de logarithme népérien. Cependant nous pouvons aisément approximer ces opérations avec des polynômes pouvant être exprimés comme des sommes de produits. Par exemple en utilisant l'algorithme d'approximation minimax, nous pouvons obtenir une erreur maximale d'approximation du logarithme népérien de l'ordre de 5 % pour un polynôme de degré 2. Cela permet d'envisager l'implémentation de règles d'apprentissage relativement sophistiquées en tenant compte des contraintes matérielles.

Les processeurs neuromorphiques font partie d'un écosystème plus large, dont font aussi partie les capteurs neuromorphiques comme les caméras événementielles. Les caméras événementielles comme la DVS (*Dynamic Vision Sensor*) fonctionnent de manière analogue à la rétine en transmettant l'information sous forme d'impulsion uniquement lorsqu'un changement local de luminosité - au niveau du pixel - est détecté. Ce traitement asynchrone de l'information visuelle apporte de grands avantages (Gallego et al., 2022) : 1) une vitesse d'échantillonnage près d'un million de fois supérieure à celle des caméras standard, 2) une latence d'une microseconde et 3) une plage dynamique de 130 décibels (les caméras standards n'ont que 60 dB). Le tout pour une consommation énergétique significativement inférieure à celle des caméras standards. Il serait ainsi fortement désirable d'interfacer nos modèles avec des flux de données dynamiques provenant de caméras événementielles, nos modèles ayant jusqu'à présent été évalués sur des bases de données statiques. Cela permettrait d'obtenir une solution neuromorphique de bout en bout à ultra-basse consommation énergétique, haute résolution et capacité de traitement en temps réel : les données événementielles transmises par le capteur neuromorphique seraient traitées par des

règles événementielles implémentées sur un processeur neuromorphique.

Applications

Le modèle WD-TCRL permet d'envisager une application d'estimation du flux optique potentiellement très performante à partir de données issues de caméras événementielles. Rappelons que WD-TCRL stocke des centroïdes dans les délais de transmission. Les délais pourraient ainsi encoder à la fois la direction (gradient dans la distribution des délais) et la vitesse (magnitude des délais) du flux optique. Un travail connexe (Paredes-Vallés et al., 2020) utilise un SNN avec des délais fixes, sélectionnés par une règle STDP adaptant les poids synaptiques pour l'estimation du flux optique. Comme nous l'avons évoqué dans le chapitre 5, cela introduit de coûteuses connexions multisynaptiques et injecte un fort biais dans le modèle. Plutôt que de sélectionner les délais, l'apprentissage des délais réalisé par notre modèle pourrait ainsi offrir une solution plus performante et implémentable *in silico*. La SO-TCRL, qui étends le modèle WD-TCRL, est également un modèle indiqué pour l'estimation du flux optique, où des neurones spatialement proches partageraient des représentations de mouvements proches.

Une autre application possible serait de reproduire l'architecture HOTS (Lagorce et al., 2017) en se basant sur WD-TCRL. Cette architecture extrait des centroïdes de données issues de caméras événementielles à l'aide de trois couches empilées, opérant sur des échelles temporelles croissantes. La sortie de la dernière couche est donnée en entrée à un classifieur pour reconnaître des caractères et des cartes. HOTS est basé sur l'extraction de centroïdes à partir de motifs temporels. Cependant l'architecture n'utilise pas de neurones impulsionnels et de règles locales. Nous pouvons noter que HOTS est basé sur l'extraction de centroïdes de surfaces temporelles, pouvant être efficacement implémentées par des traces synaptiques. Cela rejoint le comportement d'extraction de centroïdes des valeurs des traces synaptiques par nos règles STDP. Ainsi en construisant une architecture hiérarchique à trois couches basée sur le modèle WD-TCRL, il serait possible d'envisager une solution complètement neuromorphique reproduisant le comportement de HOTS. Il serait également pertinent de tester le nouveau schème TMP-POP pour la catégorisation sur de telles architectures hiérarchiques.

Modèles

Il serait également intéressant de construire des architectures hiérarchiques à partir du modèle SO-TCRL. Une architecture hiérarchique serait particulièrement adaptée aux applications nécessitant de regrouper des *clusters* de données de façon hiérarchique. Chaque couche pourrait ainsi produire des motifs temporels pouvant être appris par les neurones de la couche supérieure grâce à nos règles événementielles. Notons que cela n'est pas réalisable dans la plupart des travaux connexes de SOM impulsionnelle. Par exemple comme l'indique Rumbell et al., 2014, leur modèle nécessite que les neurones de la couche de représentation déchargent en très forte proximité temporelle à cause de la nature de leur règle STDP, pour que l'auto-organisation puisse opérer, i.e. que des neurones spatialement proches se mettent à partager des représentations proches. Ces fortes contraintes ne se retrouvent pas dans nos règles d'apprentissage, ouvrant ainsi la voie à des architectures hiérarchiques.

En évaluant l'importance des hyperparamètres des différents modèles en termes d'impact sur les performances à l'aide de l'ANOVA fonctionnelle, on constate que la constante de temps membranaire τ_m et le seuil de décharge V_θ des neurones de la couche de représentation sont ab-

solument déterminants. Il serait ainsi souhaitable de soumettre ces paramètres à l'apprentissage, plutôt que de les fixer (Fang et al., 2021). En outre, utiliser un seuil de décharge et une constante de temps fixée implique de ne pas pouvoir faire du *subspace clustering* : un neurone sélectif à un faible nombre de dimensions avec une majorité de poids synaptiques afférents déprimés, devient une unité morte qui ne décharge jamais, car insuffisamment activé pour pouvoir atteindre son seuil V_θ. Nous pouvons interpréter V_θ comme un paramètre fixant le nombre de dimensions d'entrées minimales pour lesquelles le neurone produit une réponse, i.e. une impulsion. La constante de temps membranaire τ_m peut être interprétée comme un paramètre réglant la sélectivité du neurone dans un domaine temporel. Ainsi apprendre V_θ et τ_m de façon non-supervisée serait l'objet d'un dernier modèle, qui permettrait notamment de réaliser du *subspace clustering* avec WD-TCRL ou la SO-TCRL.

Bibliographie

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* [6, 13, 37, 124]

Bahdanau, Dzmitry, Kyunghyun Cho et Yoshua Bengio (2015). "Neural Machine Translation by Jointly Learning to Align and Translate". In : *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings* [4]

Bakó, László (2010). "Real-Time Classification of Datasets with Hardware Embedded Neuromorphic Neural Networks". In : *Briefings Bioinform.* [148]

Bengio, Yoshua (2009). "Learning Deep Architectures for AI". In : *Found. Trends Mach. Learn.* [128]

Bengio, Yoshua, Aaron C. Courville et Pascal Vincent (2013). "Representation Learning : A Review and New Perspectives". In : *IEEE Trans. Pattern Anal. Mach. Intell.* [29, 68, 102, 124, 128, 147]

Bergstra, James et al. (2011). "Algorithms for Hyper-Parameter Optimization". In : *Advances in Neural Information Processing Systems 24 : 25th Annual Conference on Neural Information Processing Systems 2011. Proceedings of a Meeting Held 12-14 December 2011, Granada, Spain* [72, 105]

Bialek, William S. (2012). *Biophysics : Searching for Principles*. 640 p. [3]

Billings, Guy et Mark C. W. van Rossum (2009). "Memory Retention and Spike-Timing-Dependent Plasticity". In : *Journal of Neurophysiology* [28, 97]

Bohté, Sander M., Han La Poutré et Joost N. Kok (2002). "Unsupervised Clustering with Spiking Neurons by Sparse Temporal Coding and Multilayer RBF Networks". In : *IEEE Trans. Neural Networks* [39, 40, 148]

Brette, Romain et Emmanuel Guigon (2003). "Reliability of Spike Timing Is a General Property of Spiking Model Neurons". In : *Neural Computation* [13]

Burbank, Kendra S. (2015). "Mirrored STDP Implements Autoencoder Learning in a Network of Spiking Neurons". In : *Plos Computational Biology* [30, 62, 69, 76]

Burgess, Neil (2014). "The 2014 Nobel Prize in Physiology or Medicine : A Spatial Model for Cognitive Neuroscience". In : *Neuron* [48]

Buzsáki, György et Kenji Mizuseki (2014). "The Log-Dynamic Brain : How Skewed Distributions Affect Network Operations". In : *Nature Reviews Neuroscience* (4) [12]

Cessac, Bruno, Hélène Paugam-Moisy et Thierry Viéville (2010). "Overview of Facts and Issues about Neural Coding by Spikes". In : *Journal of Physiology-Paris* [52]

Chase, Steven M. et Eric D. Young (2007). "First-Spike Latency Information in Single Neurons Increases When Referenced to Population Onset". In : *Proceedings of the National Academy of Sciences* [13]

Chaudhuri, Rishidev et al. (2019). "The Intrinsic Attractor Manifold and Population Dynamics of a Canonical Cognitive Circuit across Waking and Sleep". In : *Nature Neuroscience* [57]

Chéreau, Ronan et al. (2017). "Superresolution Imaging Reveals Activity-Dependent Plasticity of Axon Morphology Linked to Changes in Action Potential Conduction Velocity". In : *Proceedings of the National Academy of Sciences of the United States of America* [35]

Chklovskii, Dmitri B. et Alexei A. Koulakov (2004). "Maps in the Brain : What Can We Learn from Them ?" In : *Annual Review of Neuroscience* [124]

Coates, Adam, Andrew Y. Ng et Honglak Lee (2011). "An Analysis of Single-Layer Networks in Unsupervised Feature Learning". In : *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2011, Fort Lauderdale, USA, April 11-13, 2011* [134]

Davies, Mike et al. (2018). "Loihi : A Neuromorphic Manycore Processor with On-Chip Learning". In : *IEEE Micro* [6, 13, 31, 37, 51, 94, 96, 113, 124, 127, 166]

Deng, Li (2012). "The MNIST Database of Handwritten Digit Images for Machine Learning Research [Best of the Web]". In : *IEEE Signal Process. Mag.* [68]

Diard, Julien, Pierre Bessière et Alain Berthoz (2013). "Spatial Memory of Paths Using Circular Probability Distributions : Theoretical Properties, Navigation Strategies and Orientation Cue Combination". In : *Spatial Cogn. Comput.* [49]

Diehl, Peter U. et Matthew Cook (2015). "Unsupervised Learning of Digit Recognition Using Spike-Timing-Dependent Plasticity". In : *Frontiers Comput. Neurosci.* [10, 26, 61, 134]

Domeniconi, Carlotta et al. (2007). "Locally Adaptive Metrics for Clustering High Dimensional Data". In : *Data Mining and Knowledge Discovery* [103]

Dosselmann, Richard et Xue Dong Yang (2011). "A Comprehensive Assessment of the Structural Similarity Index". In : *Signal Image Video Process.* [69]

Doya, Kenji (1999). "What Are the Computations of the Cerebellum, the Basal Ganglia and the Cerebral Cortex ?" In : *Neural Networks* [20]

— (2000). "Complementary Roles of Basal Ganglia and Cerebellum in Learning and Motor Control". In : *Current Opinion in Neurobiology* [20, 21]

Dubuis, Julien O. et al. (2013). "Positional Information, in Bits". In : *Proceedings of the National Academy of Sciences* [3]

Durbin, Richard et Graeme Mitchison (1990). "A Dimension Reduction Framework for Understanding Cortical Maps". In : *Nature* (6259) [21, 124]

Ebitz, R. Becket et Benjamin Y. Hayden (2021). "The Population Doctrine in Cognitive Neuroscience". In : *Neuron* [11]

Eurich, Christian W. et al. (1999). "Dynamics of Self-Organized Delay Adaptation". In : *Physical Review Letters* [93]

Falez, Pierre et al. (2019). "Unsupervised Visual Feature Learning with Spike-Timing-Dependent Plasticity : How Far Are We from Traditional Feature Learning Approaches ?" In : *Pattern Recognition* [29, 68, 102, 127]

Fang, Wei et al. (2021). "Incorporating Learnable Membrane Time Constant to Enhance Learning of Spiking Neural Networks". In : *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021* [168]

Fields, R. Douglas (2015). "A New Mechanism of Nervous System Plasticity : Activity-Dependent Myelination". In : *Nature Reviews Neuroscience* (12) [35]

Fois, Adrien et Bernard Girau (2020). "A Spiking Neural Architecture for Vector Quantization and Clustering". In : *Neural Information Processing* [78]

Földiák, P. (1990). "Forming Sparse Representations by Local Anti-Hebbian Learning". In : *Biological Cybernetics* [29]

Frémaux, Nicolas et Wulfram Gerstner (2016). "Neuromodulated Spike-Timing-Dependent Plasticity, and Theory of Three-Factor Learning Rules". In : *Frontiers in Neural Circuits* [127]

Froemke, Robert C. et al. (2006). "Contribution of Individual Spikes in Burst-Induced Long-Term Synaptic Modification". In : *Journal of Neurophysiology* [27]

Furber, Steve (2016). "Large-Scale Neuromorphic Computing Systems". In : *Journal of Neural Engineering* [37, 61]

Gallego, Guillermo et al. (2022). "Event-Based Vision : A Survey". In : *IEEE Trans. Pattern Anal. Mach. Intell.* [166]

Gardner, Richard J. et al. (2022). "Toroidal Topology of Population Activity in Grid Cells". In : *Nature* [57]

Georgopoulos, A. P., J. F. Kalaska et al. (1982). "On the Relations between the Direction of Two-Dimensional Arm Movements and Cell Discharge in Primate Motor Cortex". In : *The Journal of Neuroscience : The Official Journal of the Society for Neuroscience* [10]

Georgopoulos, A. P., A. B. Schwartz et R. E. Kettner (1986). "Neuronal Population Coding of Movement Direction". In : *Science (New York, N.Y.)* [49]

Gerstner, W. et al. (1996). "A Neuronal Learning Rule for Sub-Millisecond Temporal Coding". In : *Nature* [93]

Gollisch, Tim et Markus Meister (2008). "Rapid Neural Coding in the Retina with Relative Spike Latencies". In : *Science (New York, N.Y.)* [15]

Goodhill, Geoffrey J. et Terrence J. Sejnowski (1997). "A Unifying Objective Function for Topographic Mappings". In : *Neural Computation* [140]

Graziano, Michael S. A. et Tyson N. Aflalo (2007). "Mapping Behavioral Repertoire onto the Cortex". In : *Neuron* [21, 124]

Gueorguieva, Natacha, Iren Valova et George Georgiev (2006). "Learning and Data Clustering with an RBF-based Spiking Neuron Network". In : *Journal of Experimental & Theoretical Artificial Intelligence* [148]

Haykin, Simon S. et Simon S. Haykin (2009). *Neural Networks and Learning Machines*. 3rd ed. 906 p. [25]

Hazan, Hananel et al. (2020). "Lattice Map Spiking Neural Networks (LM-SNNs) for Clustering and Classifying Image Data". In : *Annals of Mathematics and Artificial Intelligence* [126, 134, 149]

Hebb, D. O. (1949). *The Organization of Behavior ; a Neuropsychological Theory*. xix, 335 [25]

Horé, Alain et Djemel Ziou (2010). "Image Quality Metrics : PSNR vs. SSIM". In : *20th International Conference on Pattern Recognition, ICPR 2010, Istanbul, Turkey, 23-26 August 2010* [69]

Hubel, D. H. et T. N. Wiesel (1959). "Receptive Fields of Single Neurones in the Cat's Striate Cortex". In : *The Journal of Physiology* [4]

Hussain, Irshed et Dalton Meitei Thounaojam (2020). "SpiFoG : An Efficient Supervised Learning Algorithm for the Network of Spiking Neurons". In : *Scientific Reports* (1) [40]

Izhikevich, Eugene M. (2004). "Which Model to Use for Cortical Spiking Neurons ?" In : *IEEE transactions on neural networks* [31, 32]

— (2006). "Polychronization : Computation with Spikes". In : *Neural Computation* [35, 93]

Jenkin, Michael R. M., Allen D. Jepson et John K. Tsotsos (1991). "Techniques for Disparity Measurement". In : *CVGIP : Image Understanding* [69]

Kandel, Eric R, James H Schwartz et Thomas M Jessell (2000). *Principles of Neural Science* [21, 124]

KASABOV, Nikola et al. (2013). "Dynamic Evolving Spiking Neural Networks for On-Line Spatio- and Spectro-Temporal Pattern Recognition". In : *Neural Networks : The Official Journal of the International Neural Network Society* [40]

KHANMOHAMMADI, Sina, Naiier ADIBEIG et Samaneh SHANEHBANDY (2017). "An Improved Overlapping K-Means Clustering Method for Medical Applications". In : *Expert Systems with Applications* [128]

KHERADPISHEH, Saeed Reza, Mohammad GANJTABESH et al. (2018). "STDP-based Spiking Deep Convolutional Neural Networks for Object Recognition". In : *Neural Networks* [10, 61]

KHERADPISHEH, Saeed Reza et Timothée MASQUELIER (2020). "Temporal Backpropagation for Spiking Neural Networks with One Spike per Neuron". In : *International Journal of Neural Systems* [10, 61]

KING, Paul D., Joel ZYLBERBERG et Michael R. DEWEESE (2013). "Inhibitory Interneurons Decorrelate Excitatory Cells to Drive Sparse Code Formation in a Spiking Model of V1". In : *The Journal of Neuroscience : The Official Journal of the Society for Neuroscience* [30, 62, 76]

KOHONEN, Teuvo (1990). "The Self-Organizing Map". In : *Proceedings of the IEEE* [20]

— (2013). "Essentials of the Self-Organizing Map". In : *Neural Networks* [20, 24, 124, 127, 129, 148, 164]

KÖNIG, Peter, Andreas K. ENGEL et Wolf SINGER (1996). "Integrator or Coincidence Detector ? The Role of the Cortical Neuron Revisited". In : *Trends in Neurosciences* [32, 92]

LAGORCE, Xavier et al. (2017). "HOTS : A Hierarchy of Event-Based Time-Surfaces for Pattern Recognition". In : *IEEE Transactions on Pattern Analysis and Machine Intelligence* [98, 167]

LECUN, Yann et al. (1999). "Object Recognition with Gradient-Based Learning". In : *Shape, Contour and Grouping in Computer Vision* [4]

LEE, Chankyu et al. (2020). "Enabling Spike-Based Backpropagation for Training Deep Neural Network Architectures". In : *Frontiers in Neuroscience* [10, 61]

LEE RODGERS, Joseph et W. Alan NICEWANDER (1988). "Thirteen Ways to Look at the Correlation Coefficient". In : *The American Statistician* [69]

LUTTRELL, Stephen P. (1994). "A Bayesian Analysis of Self-Organizing Maps". In : *Neural Computation* [24]

MAASS, Wolfgang (1997). "Networks of Spiking Neurons : The Third Generation of Neural Network Models". In : *Neural Networks* [6, 31]

MAASS, Wolfgang et Michael SCHMITT (1999). "On the Complexity of Learning for Spiking Neurons with Temporal Coding". In : *Information and Computation* [93]

MARTINETZ, Thomas (1993). "Competitive Hebbian Learning Rule Forms Perfectly Topology Preserving Maps". In : *Icann '93* [20]

MARTINETZ, Thomas, Stanislav G. BERKOVICH et Klaus SCHULTEN (1993). "'Neural-gas' Network for Vector Quantization and Its Application to Time-Series Prediction". In : *IEEE Trans. Neural Networks* [24]

MASQUELIER, Timothée, Rudy GUYONNEAU et Simon J. THORPE (2008). "Spike Timing Dependent Plasticity Finds the Start of Repeating Patterns in Continuous Spike Trains". In : *Plos One* [92]

MASQUELIER, Timothée et Saeed Reza KHERADPISHEH (2018). "Optimal Localist and Distributed Coding of Spatiotemporal Spike Patterns Through STDP and Coincidence Detection". In : *Frontiers in Computational Neuroscience* [128]

MATSUBARA, Takashi (2017). "Conduction Delay Learning Model for Unsupervised and Supervised Classification of Spatio-Temporal Spike Patterns". In : *Frontiers in Computational Neuroscience* [93]

MERZENICH, M. M., P. L. KNIGHT et G. L. ROTH (1975). "Representation of Cochlea within Primary Auditory Cortex in the Cat". In : *Journal of Neurophysiology* [21]

Moser, Edvard I. et al. (2014). "Grid Cells and Cortical Representation". In : *Nature Reviews Neuroscience* [21, 124]

Mozafari, Milad et al. (2018). "First-Spike-Based Visual Categorization Using Reward-Modulated STDP". In : *IEEE transactions on neural networks and learning systems* [10, 61]

Nadafian, Alireza et Mohammad Ganjtabesh (2020). "Bio-plausible Unsupervised Delay Learning for Extracting Temporal Features in Spiking Neural Networks". In : *ArXiv* abs/2011.09380 [94]

Ohki, Kenichi et al. (2006). "Highly Ordered Arrangement of Single Neurons in Orientation Pinwheels". In : *Nature* [21, 124]

Olshausen, B. A. et D. J. Field (1996). "Emergence of Simple-Cell Receptive Field Properties by Learning a Sparse Code for Natural Images". In : *Nature* [68]

Painkras, Eustace et al. (2013). "SpiNNaker : A 1-W 18-Core System-on-Chip for Massively-Parallel Neural Network Simulation". In : *IEEE Journal of Solid-State Circuits* [6, 37]

Paredes-Vallés, Federico, Kirk Y. W. Scheper et Guido C. H. E. de Croon (2020). "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* [167]

Parisi, German Ignacio et al. (2019). "Continual Lifelong Learning with Neural Networks : A Review". In : *Neural Networks* [166]

Patel, Devdhar et al. (2019). "Improved Robustness of Reinforcement Learning Policies upon Conversion to Spiking Neuronal Network Platforms Applied to Atari Breakout Game". In : *Neural Networks* [10, 61]

Paugam-Moisy, Hélène et Sander Bohte (2012). "Computing with Spiking Neuron Networks". In : *Handbook of Natural Computing* [33]

Paugam-Moisy, Hélène, Régis Martinez et Samy Bengio (2008). "Delay Learning and Polychronization for Reservoir Computing". In : *Neurocomputing* [93]

Pawlak, Verena et al. (2010). "Timing Is Not Everything : Neuromodulation Opens the STDP Gate". In : *Frontiers in Synaptic Neuroscience* [127]

Petersen, R. S., S. Panzeri et M. E. Diamond (2001). "Population Coding of Stimulus Location in Rat Somatosensory Cortex". In : *Neuron* [13]

Pfister, Jean-Pascal et Wulfram Gerstner (2006). "Triplets of Spikes in a Model of Spike Timing-Dependent Plasticity". In : *Journal of Neuroscience* [96]

Pham, D.T., M.S. Packianather et E.Y.A. Charles (2007). "A Self-Organising Spiking Neural Network Trained Using Delay Adaptation". In : *2007 IEEE International Symposium on Industrial Electronics*. 2007 IEEE International Symposium on Industrial Electronics [126]

Pouille, Frédéric et Massimo Scanziani (2004). "Routing of Spike Series by Dynamic Circuits in the Hippocampus". In : *Nature* [33]

Pugh Jr., Edward N. (2018). "The Discovery of the Ability of Rod Photoreceptors to Signal Single Photons". In : *Journal of General Physiology* [3]

Rossum, M. C. W. van, G. Q. Bi et G. G. Turrigiano (2000). "Stable Hebbian Learning from Spike Timing-Dependent Plasticity". In : *Journal of Neuroscience* [28, 97]

Rougier, Nicolas et Yann Boniface (2011). "Dynamic Self-Organising Map". In : *Neurocomputing* [127, 130, 149]

Ruf, Berthold et Michael Schmitt (1998). "Self-Organization of Spiking Neurons Using Action Potential Timing". In : *IEEE Trans. Neural Networks* [125, 126, 149]

Rumbell, T., S. L. Denham et T. Wennekers (2014). "A Spiking Self-Organizing Map Combining STDP, Oscillations, and Continuous Learning". In : *IEEE Transactions on Neural Networks and Learning Systems* [39, 40, 126, 128, 144-149, 165, 167]

Sabatini, B. L. et W. G. Regehr (1999). "Timing of Synaptic Transmission". In : *Annual Review of Physiology* [35]

Sjöström, Jesper et Wulfram Gerstner (2010). "Spike-Timing Dependent Plasticity". In : *Scholarpedia* [65]

Song, Sen, Kenneth D. Miller et L. F. Abbott (2000). "Competitive Hebbian Learning through Spike-Timing-Dependent Synaptic Plasticity". In : *Nature Neuroscience* [27, 28]

Steuber, Volker et David Willshaw (2004). "A Biophysical Model of Synaptic Delay Learning and Temporal Pattern Recognition in a Cerebellar Purkinje Cell". In : *Journal of Computational Neuroscience* [35]

Stimberg, Marcel, Romain Brette et Dan FM Goodman (2019). "Brian 2, an Intuitive and Efficient Neural Simulator". In : *eLife* [63]

Subbulakshmi Radhakrishnan, Shiva et al. (2021). "A Biomimetic Neural Encoder for Spiking Neural Network". In : *Nature Communications* (1) [38]

Swadlow, H. A. (1985). "Physiological Properties of Individual Cerebral Axons Studied in Vivo for as Long as One Year". In : *Journal of Neurophysiology* [35]

Taherkhani, Aboozar et al. (2015). "DL-ReSuMe : A Delay Learning-Based Remote Supervised Method for Spiking Neurons". In : *IEEE Transactions on Neural Networks and Learning Systems* [93]

Tavanaei, Amirhossein et Anthony Maida (2019). "BP-STDP : Approximating Backpropagation Using Spike Timing Dependent Plasticity". In : *Neurocomputing* [26]

Tavanaei, Amirhossein, Timothée Masquelier et Anthony Maida (2018). "Representation Learning Using Event-Based STDP". In : *Neural Networks* [29, 62, 69, 74-78, 109, 110, 112, 113]

Thorpe, Simon J. et Jacques Gautrais (1996). "Rapid Visual Processing Using Spike Asynchrony". In : *Advances in Neural Information Processing Systems 9, NIPS, Denver, CO, USA, December 2-5, 1996* [12]

— (1998). "Rank Order Coding". In : *Computational Neuroscience* [14, 135]

Veredas, Francisco J., Héctor Mesa et Luis A. Martínez (2008). "Imprecise Correlated Activity in Self-Organizing Maps of Spiking Neurons". In : *Neural Networks* [125]

Wade, John J. et al. (2010). "SWAT : A Spiking Neural Network Training Algorithm for Classification Problems". In : *IEEE transactions on neural networks* [148]

Wang, Xiangwen, Xianghong Lin et Xiaochao Dang (2019). "A Delay Learning Algorithm Based on Spike Train Kernels for Spiking Neurons". In : *Frontiers in Neuroscience* [93]

Wickens, Andrew (2009). *Introduction to Biopsychology*. 3rd edition. 626 p. [20]

Wright, Paul W et Janet Wiles (2012). "Learning Transmission Delays in Spiking Neural Networks : A Novel Approach to Sequence Learning Based on Spike Delay Variance". In : *The 2012 International Joint Conference on Neural Networks (IJCNN)*. 2012 International Joint Conference on Neural Networks (IJCNN 2012 - Brisbane) [94]

Wu, Qingxiang et al. (2006). "Learning under Weight Constraints in Networks of Temporal Encoding Spiking Neurons". In : *Neurocomputing* [148]

Yen, Eugene K et Roger G Johnston (1996). "The Ineffectiveness of the Correlation Coefficient for Image Comparisons". In : *Technical Report LA- UR-96-2474, Los Alamos* [69]

Yin, Hujun (2008). "The Self-Organizing Maps : Background, Theories, Extensions and Applications". In : *Computational Intelligence : A Compendium* [24]

Yoon, Song-Yee et Soo-Young Lee (1999). "Training Algorithm with Incomplete Data for Feed-Forward Neural Networks". In : *Neural Processing Letters* [148]

Yu, Qiang et al. (2014). "A Brain-Inspired Spiking Neural Network Model with Temporal Encoding and Learning". In : *Neurocomputing* [40, 127]

ZHANG, Malu et al. (2020). "Supervised Learning in Spiking Neural Networks with Synaptic Delay-Weight Plasticity". In : *Neurocomputing* [93]