

École doctorale n° 77 : IAEM

# THÈSE

pour obtenir le grade de docteur délivré par

## **I'Université de Lorraine Spécialité doctorale "Informatique"**

# **Calcul neuromorphique pour l'exploration et la catégorisation robuste d'environnement visuel et multimodal dans les systèmes embarqués.**

présentée et soutenue publiquement par

**Yann BERNARD**

le 7 Décembre 2021

Directeur de thèse : **Bernard GIRAU**  
Co-encadrant de thèse : **Nicolas HUEBER**  
Co-encadrant de thèse : **Pierre RAYMOND**

### **Jury**

<b>Mme. Marie Cotrell,</b>	Professeure	Rapporteure
<b>M. Michel Verleysen,</b>	Professeur	Rapporteur
<b>M. Michel Paindavoine,</b>	Professeur	Examinateur
<b>Mme. Isabelle Debled-Renaisson,</b>	Professeure	Examinateuse

# Résumé

Tandis que la quête pour des systèmes de calcul toujours plus puissants se confronte à des contraintes matérielles de plus en plus fortes, des avancées majeures en termes d'efficacité de calcul sont supposées bénéficier d'approches non conventionnelles et de nouveaux modèles de calcul tels que le calcul inspiré du cerveau. Le cerveau est une architecture de calcul massivement parallèle avec des interconnexions denses entre les unités de calcul. Les systèmes neurobiologiques sont donc une source d'inspiration naturelle pour la science et l'ingénierie informatiques. Les améliorations technologiques rapides des supports de calcul ont récemment renforcé cette tendance à travers deux conséquences complémentaires mais apparemment contradictoires : d'une part en offrant une énorme puissance de calcul, elles ont rendu possible la simulation de très grandes structures neuronales comme les réseaux profonds, et d'autre part en atteignant leurs limites technologiques et conceptuelles, elles ont motivé l'émergence de paradigmes informatiques alternatifs basés sur des concepts bio-inspirés. Parmi ceux-ci, les principes de l'apprentissage non supervisé retiennent de plus en plus l'attention.

Nous nous intéressons ici plus particulièrement à deux grandes familles de modèles neuronaux, les cartes auto-organisatrices et les champs neuronaux dynamiques. Inspirées de la modélisation de l'auto-organisation des colonnes corticales, les cartes auto-organisatrices ont montré leur capacité à représenter un stimulus complexe sous une forme simplifiée et interprétable, grâce à d'excellentes performances en quantification vectorielle et au respect des relations de proximité topologique présentes dans l'espace d'entrée. Davantage inspirés des mécanismes de compétition dans les macro-colonnes corticales, les champs neuronaux dynamiques autorisent l'émergence de comportements cognitifs simples et trouvent de plus en plus d'applications dans le domaine de la robotique autonome notamment.

Dans ce contexte, le premier objectif de cette thèse est de combiner cartes auto-organisatrices (SOM) et champs neuronaux dynamiques (DNF) pour l'exploration et la catégorisation d'environnements réels perçus au travers de capteurs visuels de différentes natures. Le second objectif est de préparer le portage de ce calcul de nature neuromorphe sur un substrat matériel numérique. Ces deux objectifs visent à définir un dispositif de calcul matériel qui pourra être couplé à différents capteurs de manière à permettre à un système autonome de construire sa propre représentation de l'environnement perceptif dans lequel il évolue. Nous avons ainsi proposé et évalué un modèle de détection de nouveauté à partir de SOM. Les considérations matérielles nous ont ensuite amené à des optimisations algorithmiques significatives dans le fonctionnement des SOM. Enfin, nous complémenté le modèle avec des DNF pour augmenter le niveau d'abstraction avec un mécanisme attentionnel de suivi de cible.

# Abstract

As the quest for ever more powerful computing systems faces ever-increasing material constraints, major advances in computing efficiency are expected to benefit from unconventional approaches and new computing models such as brain-inspired computing. The brain is a massively parallel computing architecture with dense interconnections between computing units. Neurobiological systems are therefore a natural source of inspiration for computer science and engineering. Rapid technological improvements in computing media have recently reinforced this trend through two complementary but seemingly contradictory consequences : on the one hand, by providing enormous computing power, they have made it possible to simulate very large neural structures such as deep networks, and on the other hand, by reaching their technological and conceptual limits, they have motivated the emergence of alternative computing paradigms based on bio-inspired concepts. Among these, the principles of unsupervised learning are receiving increasing attention.

We focus here on two main families of neural models, self-organizing maps and dynamic neural fields. Inspired by the modeling of the self-organization of cortical columns, self-organizing maps have shown their ability to represent a complex stimulus in a simplified and interpretable form, thanks to excellent performances in vector quantization and to the respect of topological proximity relationships present in the input space. More inspired by competition mechanisms in cortical macro-columns, dynamic neural fields allow the emergence of simple cognitive behaviours and find more and more applications in the field of autonomous robotics.

In this context, the first objective of this thesis is to combine self-organizing maps and dynamic neural fields for the exploration and categorisation of real environments perceived through visual sensors of different natures. The second objective is to prepare the porting of this neuromorphic computation on a digital hardware substrate. These two objectives aim to define a hardware computing device that can be coupled to different sensors in order to allow an autonomous system to construct its own representation of the perceptual environment in which it operates. Therefore, we proposed and evaluated a novelty detection model based on self-organising maps. Hardware considerations then led us to significant algorithmic optimisations SOM operations. Finally, we complemented the model with dynamic neural fields to increase the level of abstraction with an attentional target tracking mechanism.

# Table des matières

<b>Table des matières</b>	<b>iv</b>
<b>Liste des figures</b>	<b>v</b>
<b>Liste des tableaux</b>	<b>vii</b>
<b>1 Etat de l'art</b>	<b>1</b>
1.1 Contexte . . . . .	2
1.2 Réseaux neuronaux . . . . .	12
1.3 Références . . . . .	19
<b>2 Détection de Nouveauté</b>	<b>23</b>
2.1 Introduction . . . . .	24
2.2 Application aux images . . . . .	24
2.3 Détection de nouveauté par QVAT : quantification vectorielle et appren-tissage topologique . . . . .	29
2.4 Protocole expérimental . . . . .	34
2.5 Résultats expérimentaux . . . . .	49
2.6 Références . . . . .	60
<b>3 Calcul de Best Matching Unit accéléré avec la topologie</b>	<b>63</b>
3.1 Introduction . . . . .	64
3.2 Algorithme séquentiel . . . . .	66
3.3 Algorithme parallèle . . . . .	76
3.4 Conclusion . . . . .	79
3.5 Références . . . . .	79
<b>4 Vision événementielle et attention</b>	<b>81</b>
4.1 Introduction . . . . .	82
4.2 Détection de mouvements . . . . .	84
4.3 Combinaison avec la détection de nouveauté . . . . .	87
4.4 Mécanisme attentionnel . . . . .	89
4.5 Références . . . . .	92
<b>Conclusion</b>	<b>93</b>

# Liste des figures

1.1	Illustration de l'efficacité du traitement visuel humain . . . . .	9
1.2	Phonème SOM . . . . .	12
1.3	Apprentissage de SOM . . . . .	15
1.4	Apprentissage de GNG . . . . .	17
2.1	Lac de Nino . . . . .	25
2.2	Représentation d'une image . . . . .	26
2.3	Compression et décompression d'image . . . . .	27
2.4	Représentation d'une image . . . . .	29
2.5	Détection de nouveauté par quantification vectorielle . . . . .	30
2.6	Exemples de différence avec quantification vectorielle . . . . .	31
2.7	Détection de nouveauté avec topologie . . . . .	32
2.8	Exemples de différence avec distance neurale. . . . .	33
2.9	Catégorie Baseline . . . . .	35
2.10	Catégorie Bad Weather . . . . .	35
2.11	Catégorie Camera Jitter . . . . .	35
2.12	Catégorie Dynamic Background . . . . .	35
2.13	Catégorie Shadow . . . . .	36
2.14	Catégorie Night Videos . . . . .	36
2.15	Catégorie Thermal . . . . .	36
2.16	Catégorie Turbulence . . . . .	37
2.17	Catégorie Intermittent Object Motion - Reduced . . . . .	37
2.18	Catégorie Low Framerate - Reduced . . . . .	37
2.19	Différence entre nouveauté and changement . . . . .	38
2.20	Échantillonnage de l'évaluation . . . . .	42
2.21	Effet de l'aléatoire sur les métriques . . . . .	44
2.22	Optimisation de $\sigma$ . . . . .	46
2.23	Optimisation d' $\alpha$ . . . . .	47
2.24	Nombre d'époques . . . . .	48
2.25	Seuil de décision . . . . .	49
2.26	Seuil de décision en fonction de la séquence . . . . .	50
2.27	Nombre de neurones et de taille des imagettes, SOM/baseline . . . . .	51
2.28	Nombre de neurones et de taille des imagettes, GNG/baseline . . . . .	52
2.29	Nombre de neurones et de taille des imagettes, PSNR/SOM/baseline . . . . .	53
2.30	Nombre de neurones et de taille des imagettes, Global . . . . .	54
2.31	Visualisation résultats normaux . . . . .	58
2.32	Visualisation résultats dynamiques . . . . .	59
2.33	Visualisation des effets de la normalisation . . . . .	60

3.1	Visualisation de l'algorithme de particule . . . . .	68
3.2	Évaluation des gains de performance en fonction du nombre de neurones	74
3.3	Pourcentage d'évaluations par neurones pour FastBMU . . . . .	77
4.1	Architecture de la combinaison de modèles . . . . .	82
4.2	Montage des deux caméras . . . . .	83
4.3	Détection de mouvements avec DNF sur caméra événementielle . . . . .	84
4.4	Détection de mouvements multicibles avec DNF sur caméra événementielle . . . . .	85
4.5	Combinaison SOM et DNF par zone d'intérêt . . . . .	87
4.6	Exemple d'Attention avec DNF . . . . .	91

# Liste des tableaux

2.1	Estimations statistiques du nombre de graines requises . . . . .	44
2.2	Récapitulatif des paramètres SOM . . . . .	45
2.3	Paramètres GNG . . . . .	46
2.4	Résultats complets sur CDNET de notre détection de nouveauté . . . . .	55
2.5	Résultats complets sur CDNET de notre détection de nouveauté - Suite . . . . .	56
2.6	Comparatif avec d'autres modèles de détection de changement sur CD-NET . . . . .	57
3.1	Résultats de FastBMU sur différentes données . . . . .	72
3.2	Résultats de Fast-BMU en parallèle . . . . .	78
4.1	Paramètres DNF 1D . . . . .	86
4.2	Impacts du modèle étendu SOM-DNF . . . . .	88
4.3	Paramètres du DNF 2D attentionnel . . . . .	90



# Chapitre 1

## Etat de l'art

« *If I knew how I knew everything  
I knew, then I would only be able  
to know half has much, because  
it will all be clogged up with  
where I know it from.  
So I cannot always cite my  
sources, I'm sorry.* »

---

David Mitchell

### Sommaire

---

<b>1.1 Contexte</b> . . . . .	<b>2</b>
1.1.1 L'inspiration biologique . . . . .	3
1.1.2 L'émergence . . . . .	5
1.1.3 Particularités de la vision . . . . .	7
1.1.4 Vision humaine . . . . .	8
1.1.5 Détection de nouveauté . . . . .	10
<b>1.2 Réseaux neuronaux</b> . . . . .	<b>12</b>
1.2.1 Cartes auto organisatrices . . . . .	12
1.2.2 Principes de fonctionnement . . . . .	13
1.2.3 Gaz Neuronaux en Expansion . . . . .	15
1.2.4 DNF . . . . .	18
<b>1.3 Références</b> . . . . .	<b>19</b>

---

## 1.1 Contexte

Depuis les premiers pas de l'informatique, une question s'est posée : est-ce que les machines peuvent penser ? **TURING [1950]** Cette question, bien qu'abstraite dans sa formulation, se réfère à l'intelligence humaine. Il n'est en effet ici pas question de savoir si ce que les ordinateurs font est de la pensée, mais bien si l'on peut reproduire l'intelligence humaine dans un cerveau électronique. Cette question a été le fondement d'un nouveau domaine scientifique qui a été nommé, lors de la conférence de Dartmouth **McCARTHY et collab. [1955]** : l'intelligence artificielle (IA).

Les années qui suivirent n'amènerent pas de résultat au niveau des attentes des chercheurs. Le domaine était encore trop limité par la puissance de calcul des ordinateurs de cette époque, et les modèles primitifs n'arrivaient qu'à résoudre des tâches simples.

La première évolution majeure est arrivée dans les années 90, lorsque les processeurs ont dépassé le million de transistors par puce et lorsque les méthodes algorithmiques ont avancé, avec par exemple la recherche rapide dans une base de données. Ces changements ont permis aux modèles d'IA d'exploiter plus de données, plus rapidement et la programmation et les heuristiques étaient les facteurs les plus importants de réussite. Les succès les plus connus de cette époque comptent la victoire de Deep Blue sur Kasparov aux échecs **CAMPBELL et collab. [2002]**, et le programme Watson d'IBM **FERRUCCI [2012]**, qui pourrait être considéré comme le dernier grand projet d'IA de cette période. Il commençait déjà à utiliser quelque chose qui s'est développé en parallèle et qui allait constituer la nouvelle évolution de l'IA : l'apprentissage.

L'apprentissage automatique est une sous-discipline du domaine de l'IA qui s'est développée tout au long de son histoire et qui est passée sur le devant de la scène depuis les années 2010. C'est une combinaison de plusieurs facteurs qui l'ont amené à dépasser tous les records à ce moment-là : les réseaux de neurones multicouches étaient arrivés à maturation et permettaient l'apprentissage de données avec des représentations à haut niveau d'abstraction. Les dispositifs de calculs devinrent toujours plus puissants, avec notamment les cartes graphiques dédiées devenues programmables et qui permirent d'effectuer du calcul de nombres à virgule flottante en parallèle. Et en dernier, la présence de jeux de données massifs avec lesquels on pouvait entraîner des réseaux toujours plus grands et complexes, et en produisant des résultats toujours plus impressionnantes. Les exemples de révolutions applicatives sont légion. La compétition de reconnaissance d'image ImageNet est par exemple passée de taux d'erreurs de 25% avec des approches classiques à 15% en 2012 avec le réseau apprenant AlexNet **KRIZHEVSKY et collab. [2012]**, fondé sur des travaux antérieurs de Yann Le Cun **LECUN et collab. [1989]**. Les années suivantes ont vu l'explosion de ce type de réseaux, qui grâce à leur généricité, ont été appliqués à quasiment tous les domaines de l'informatique et au-delà. En 2017, 5 ans après AlexNet, ImageNet était résolu avec la majorité des participants atteignant des taux d'erreurs inférieurs à 5%. D'autres succès notables sont la victoire au Go par AlphaGo contre Lee Sedol **SILVER et collab. [2016]**, et la série de modèles de langage GPT (Generative Pre-trained Transformer) **BROWN et collab. [2020]**.

Malgré ces nombreux succès, des nuages ont commencé à apparaître dans le ciel des réseaux de neurones. GPT-3, le modèle le plus récent d'OpenAI, utilise 175

milliards de paramètres, nécessitant 350 gigaoctets de VRAM juste pour effectuer une inférence. Son coût d'apprentissage a été estimé entre 11 et 28 millions de dollars américains<sup>1</sup>. La consommation électrique elle, est estimée dans les environs de 190 MWh, correspondant à des émissions en gaz à effet de serre d'un aller-retour terre-lune en voiture<sup>2</sup>. Le corpus de textes utilisé était 150 fois plus gros que Wikipédia, lui-même inclus dedans. Les performances étaient quant à elles toujours inférieures à un humain, qui lui n'a accès qu'à un cerveau de 25 Watts KANDEL et collab. [2000] et un corpus d'apprentissage extrêmement petit en comparaison. On estime qu'un enfant dans une famille aisée entend environ 11,2 millions de mots par an HART et RISLEY [2003]. Extrapolé pour 20 ans, cela ferait 224 millions de mots pour avoir une bonne maîtrise de la langue, soit 0,05% des 500 milliards de mots sur lesquels GPT a été entraîné.

La même observation peut être faite sur tous les succès des réseaux neuronaux. AlphaGo par exemple a eu besoin de 1920 CPUs et 280 GPUs pour battre Lee Sedol, avec une puissance nécessaire estimée à 1 MW<sup>3</sup>, soit 100 fois plus que son opposant, Lee Sedol. Les performances surhumaines des réseaux de neurones actuels ne proviennent pas tant de l'intelligence dans les modèles déployés, que de leur capacité à mobiliser de grandes quantités de ressources pour résoudre un problème particulier. Les réseaux de neurones actuels montrant ainsi de plus en plus leurs limites, la question se pose de quelle sera la prochaine épine dorsale de l'IA, et quels seront les développements qui amèneront la prochaine évolution de la discipline.

Cette thèse s'inscrit dans un courant de pensée qui considère deux approches complémentaires comme étant les clés potentielles vers cette nouvelle évolution : l'accroissement des capacités de calculs par le neuromorphisme et l'augmentation de l'efficacité et de la puissance d'apprentissage par l'émergence et la complexité. Pour mieux situer notre approche, nous allons dans les sections suivantes préciser comment l'inspiration biologique est étroitement liée à la puissance de calcul (1.1.1), notamment au travers d'une forme de calcul émergent (1.1.2), qui trouve plus particulièrement un sens dans la vision (1.1.3), pour laquelle nos travaux mêlent des principes inspirés des neurosciences (1.1.4) et de la vision par ordinateur (1.1.5).

### 1.1.1 L'inspiration biologique

Depuis l'avènement des semi-conducteurs, la puissance computationnelle disponible n'a cessé de s'accroître exponentiellement. La célèbre prophétie auto-réalisatrice de Gordon Moore, que la densité de transistors dans les microprocesseurs double tous les deux ans, a fait progresser l'industrie pour lui faire atteindre le million de transistors par puce à l'aube des années 1990, et le milliard en 2006. Au moment de la rédaction de ce manuscrit, la plus grosse puce est le processeur A100 à 54 milliards de transistors<sup>4</sup>.

Cette progression cache cependant des difficultés pour utiliser efficacement tous ces milliards de transistors. L'architecture de Von Neumann VON NEUMANN [1945] qui

- 
1. <https://bdtechtalks.com/2020/09/21/gpt-3-economy-business-model>
  2. [https://www.theregister.com/2020/11/04/gpt3\\_carbon\\_footprint\\_estimate/](https://www.theregister.com/2020/11/04/gpt3_carbon_footprint_estimate/)
  3. <https://jacquesmattheij.com/another-way-of-looking-at-lee-sedol-vs-alphago/>
  4. [https://en.wikipedia.org/wiki/Ampere\\_\(microarchitecture\)](https://en.wikipedia.org/wiki/Ampere_(microarchitecture))

sert de modèle depuis les années 50 à quasiment tous les ordinateurs est le point bloquant de la puissance des processeurs. La séparation de la mémoire et du CPU place le bus de données entre les deux comme le centre névralgique de l'ordinateur et sa vitesse est limitée. Ce problème est compensé en partie dans les processeurs modernes avec l'utilisation de cache, mais ce n'est qu'une solution d'appoint qui ne tient pas la mise à l'échelle, car les tailles de cache resteront toujours très inférieures à la mémoire vive. Un second problème, plus important encore, est la limite des performances séquentielles des CPU.

L'informatique a, dès la machine de Turing, été principalement séquentielle et les algorithmes et processeurs se sont développés pour l'essentiel dans ce paradigme séquentiel. Cela n'a pas posé de problèmes tant que les fréquences augmentaient et que la finesse de gravure se réduisait, amenant toujours plus de performances. Mais un mur de fréquence a été atteint dans les années 2000, dû aux coûts énergétiques et à la chaleur engendrée par les fréquences trop élevées. Depuis les processeurs grand public ont rarement dépassé les 5 Ghz. Cette limite aux performances séquentielles des CPU combinée à la croissance exponentielle du nombre de transistors a naturellement amené vers le développement d'architectures multi-cœurs, qui ont nécessité un développement plus poussé vers le parallélisme tant au niveau matériel qu'algorithme. C'est ce parallélisme qui a été le catalyseur pour le développement des réseaux de neurones, particulièrement bien parallélisables. La limite de l'architecture de von Neumann elle, reste présente schématiquement. Lors de l'apprentissage d'un réseau de neurones avec de multiples GPU, la mémoire de ceux-ci doit être dupliquée dans chaque carte, et synchronisée régulièrement, si bien que la taille de ces réseaux de neurones est ultimement limitée par la mémoire vive du plus petit GPU. Mais l'architecture de von Neumann n'est pas une fatalité, et il existe de nombreuses idées pour pallier à ses problèmes. La piste que nous avons choisie ici est de s'inspirer de la machine la plus efficace pour traiter de l'information : le cerveau.

Un cerveau est une machinerie complexe, composé d'environ 170 milliards de cellules chez l'humain, dont la moitié environ sont des neurones, le reste des cellules gliales. C'est un ordre de grandeur similaire au nombre de transistors dans un circuit imprimé récent, mais il existe tout de même des différences importantes entre les deux. Un neurone est capable d'effectuer des opérations beaucoup plus complexes qu'un transistor. Il est également beaucoup plus grand, de l'ordre du micromètre, alors que les transistors ne font que quelques nanomètres. Ce qui explique en partie la différence de taille. Un cerveau humain occupe un volume d'environ  $1200 \text{ cm}^3$ , avec de grandes variations entre les genres et les individus. Les puces informatiques se contentent de deux dimensions et s'étalent sur une petite surface en comparaison, soit  $826\text{mm}^2$  pour la plus grande. En supposant une épaisseur de 5mm, on obtient un volume de  $4,13 \text{ cm}^3$ . Si l'on compare à volume égal, 1  $\text{cm}^3$  de cerveau contient environ 140 millions de cellules. Le même centimètre cube de puce informatique peut contenir 13 milliards de transistors, soit un ratio d'une cellule pour 92 transistors. On peut également noter qu'il est possible de réaliser des neurones suivant le modèle *Leaky Integrate and Fire* utilisant moins de 10 transistors [PARK et collab. \[2021\]](#). Les circuits électroniques ont aussi l'avantage de ne pas être limités en volume comme l'est le cerveau, et l'on peut utiliser une énergie et un volume plus grands de plusieurs ordres de grandeur pour ceux-ci. Ces comparaisons semblent donc indiquer que le nombre de transistors disponibles n'est pas un facteur limitant pour le développement de mo-

dèles informatiques plus proches des capacités humaines de raisonnement, puisque l'on pourrait raisonnablement disposer d'une puissance de calcul similaire.

La différence d'organisation entre un cerveau et un processeur est cependant majeure. Un cerveau fonctionne de façon analogique, évènementielle et asynchrone alors qu'un processeur actuel fonctionne en états discrets (généralement binaires) et synchronisés sur une horloge globale, propre à chaque processeur. Les informations dans le cerveau circulent de façon locale entre neurones voisins et avec très peu de connexions "longue distance". Un processeur est plutôt une chaîne d'assemblage, où toute l'information circule dans un seul sens. La "mémoire vive" dans le cerveau est gérée localement, en modifiant la "chimie neuronale" avec des neurotransmetteurs, en modifiant les connexions synaptiques voire même avec de la neurogenèse. Pour une machine de von Neumann, la mémoire et le calcul sont complètement distincts, et un circuit *Full Adder* fera aussi bien de la cryptographie que du jeu vidéo ou la compilation du code *LATEX*de ce manuscrit. Ces différences sont, de notre point de vue, fondamentales dans ce qui nous sépare des performances et de l'efficacité d'un cerveau humain.

Récemment, une nouvelle génération de puces appelées "neuromorphiques" a vu le jour chez les fabricants de processeurs. Notamment Loihi<sup>5</sup> d'Intel, ou TrueNorth<sup>6</sup> d'IBM. Ces circuits reprennent des propriétés cérébrales que nous avons évoquées, et comme attendu, la mise à l'échelle de ce genre d'architecture se fait aisément. Intel par exemple le démontre avec ses puces Loihi. Chacune d'entre elles comprend 131 072 neurones, et 10 fois plus de synapses. Une fois combinées par 32 sur une carte *Nahuku* on obtient un système neuromorphe à 4 millions de neurones. Combinez encore ces cartes dans un système plus grand et vous avez *Pohoiki Springs* et ses 100 millions de neurones distribués sur 768 puces. Une question reste cependant ouverte : comment utiliser efficacement toute cette puissance de calcul neuronal ?

### 1.1.2 L'émergence

La première idée pour exploiter efficacement des substrats de calcul neuromorphiques serait de les utiliser avec les outils que nous avons déjà. C'est-à-dire, combiner des algorithmes dans des programmes capables d'effectuer des tâches complexes comme jouer aux échecs, analyser une image ou piloter un avion. Cette approche fonctionne bien pour les ordinateurs classiques, car c'est relativement simple de combiner des algorithmes lorsque les opérations sont séquentielles et la mémoire un ensemble cohérent et interprétable. Il devient plus compliqué de programmer de cette façon des tâches parallèles, car il faut prendre en compte les temporalités de chaque tâche, ce qui ouvre la voie à de nombreux problèmes de blocages, d'incohérences à cause d'interdépendances entre les calculs et augmente significativement le nombre d'erreurs possibles. Le matériel neuromorphe lui est en comparaison proche d'être impossible à programmer car ce que fait chaque neurone dépend non seulement de son état interne, mais aussi du millier de ses neurones voisins et de la temporalité des informations qu'il reçoit. Les chaînes de causalité et de rétroaction sont si complexes et avec de si nombreux éléments que la programmation d'un système neuromorphe à 100 millions de neurones capable de piloter un avion semble hors de portée pour l'intelligence humaine.

---

5. <https://en.wikichip.org/wiki/intel/loihi>

6. <https://www.research.ibm.com/articles/brain-chip.shtml>

Le problème de la complexité n'est cependant pas unique au matériel neuromorphe. Un programmeur moderne n'a pas conscience de l'état de toutes les portes logiques dans un processeur à 50 milliards de transistors lorsqu'il écrit un programme. Mais il arrive à les coordonner pour leur faire simuler le comportement d'un joueur d'échecs ou de go. Cela est permis grâce à des langages de programmation de haut niveau d'abstraction, capables d'effectuer des tâches complexes en restant compréhensibles par un humain. Un tel langage n'existe pas encore pour les puces neuromorphiques. Copier les langages existants qui fonctionnent pour les ordinateurs classiques et les appliquer au neuromorphe copierait également leurs limites, et serait une utilisation inefficace de la puissance de calcul neuronal. Ainsi, il semble nécessaire de trouver des principes fondamentaux pouvant servir de base pour l'élaboration d'un langage de haut niveau qui permettrait d'organiser efficacement le calcul neuromorphe.

Nous pouvons de nouveau ici nous inspirer de ce que fait la biologie avec le principe d'émergence. L'émergence est un phénomène qui apparaît lorsqu'un ensemble d'éléments a des propriétés différentes de la somme de ses parties. L'émergence peut être observée à tous les niveaux dans la vie. Avec d'un côté les bancs de poissons ou nuées d'oiseaux qui sont le résultat d'un comportement simple au niveau de l'individu : rester proche des autres, aller dans la même direction, etc, et qui amène à une structure plus avancée au niveau du groupe : la nuée. Un phénomène similaire est à l'œuvre chez les fourmis, où par le dépôt de phéromones par chaque fourmi entre la fourmière et la nourriture, les fourmis empruntent automatiquement le plus court chemin entre les deux. Les phéromones s'évaporent avec le temps, le chemin le plus court sera celui où les phéromones auront le moins de temps pour s'évaporer, et sera de fait la route préférée des fourmis. L'émergence ne se limite cependant pas à l'échelle macroscopique entre individus, mais aussi dans le fonctionnement même d'un organisme.

Un être humain est composé en masse à 65% d'oxygène, 19% de carbone, 10% d'hydrogène, 3% d'azote, 2% de calcium, 1% de phosphore ainsi que d'autres éléments en plus petites quantités. Cependant, il ne suffit pas de mélanger tous ces éléments ensemble pour qu'un humain apparaisse. Même avec les bonnes molécules cela ne fonctionnerait pas. Pour avoir un humain, il faut que les molécules soient placées au bon endroit les unes par rapport aux autres. C'est à dire, qu'un humain est avant tout une certaine *organisation* de ces molécules. Cette description est valable pour l'ensemble du vivant, jusqu'aux plus petites bactéries et virus. L'étude de l'émergence est donc l'analyse de l'organisation de composants élémentaires pour leur faire adopter des comportements plus complexes. C'est un sujet de recherche pour de nombreux champs scientifiques en plus de la biologie, comme la physique, l'économie et l'informatique avec entre autres, les systèmes complexes.

Cette idée d'émergence est cruciale dans le fonctionnement du cerveau **TURKHEIMER et collab. [2019]**, et par conséquent dans l'utilisation efficiente de nos processeurs neuromorphiques. Il n'existe pour l'instant pas de méthode générale qui permette de transformer n'importe quel ensemble de neurones artificiels en système capable de tâches de haut niveau. Nous ne savons pas non plus si une telle méthode générale pourrait exister.

Il existe des modèles en informatique qui ont pour ambition de faire le lien entre

le niveau neuronal et des fonctions plus avancées avec leurs propriétés émergentes et auto-organisatrices. On pourrait considérer le premier pas comme étant l'apprentissage Hebbien. Présenté par Donald Hebb dans son livre *The organisation of behaviour* HEBB [1949] et qui introduit l'idée de l'apprentissage associatif, souvent résumé en « *Neurons that fire together, wire together* ». On peut également citer l'article de Dijkstra sur l'auto-stabilisation DIJKSTRA [1974]. Le sujet de cette publication est la tolérance aux fautes dans la programmation concurrente. Mais son concept est aussi applicable au domaine de l'émergence où l'on nomme l'auto-stabilisation l'homéostasie. Il s'agit de la capacité d'un système à se maintenir dans un état, comme par exemple la régulation de la température du corps humain. Ce principe d'auto-stabilisation est aussi très présent dans la théorie des champs neuronaux, introduite par AMARI [1977]. Les champs neuronaux essayent de définir les dynamiques d'activation dans des populations de neurones à un niveau mathématique. Selon la théorie, les champs neuronaux seraient l'élément fondamental pour produire des mécanismes cognitifs complexes SAN-DAMIRSKAYA [2014]. Enfin, nous avons les cartes auto-organisatrices KOHONEN [1982], qui sont un modèle de quantification vectorielle avec la particularité d'organiser automatiquement les neurones pour présenter une forme de continuité spatiale de l'information, de telle sorte que des neurones voisins réagiront à des stimulus similaires.

Nous avons dans cette thèse utilisé principalement les cartes auto organisatrices (SOM) et les champs neuronaux dynamiques (DNF). Nous les avons choisis parce qu'ils sont très utilisés dans la littérature et appliqués à de nombreux domaines. Leurs capacités à produire des comportements émergents nous intéressent particulièrement, ainsi que leur proximité avec le calcul neuronal qui en fait des modèles décentralisés efficaces pour les systèmes embarqués. Nous avons exploré leur utilisation dans le domaine de la vision et évalué leur adéquation au rôle de couche d'abstraction pour des approches neuromorphiques. Ils seront présentés plus en détail dans les sections suivantes 1.2.1 et 1.2.4.

### 1.1.3 Particularités de la vision

La vision fait partie des sens les plus développés dans la nature; presque l'intégralité des espèces vertébrées sont dotées d'yeux. Mais en plus d'être très utile, c'est également un sens très complexe. Fondamentalement, il consiste à capter des photons présents dans l'environnement et à en déduire des informations sur celui-ci. Chaque photon arrive dans l'œil ou le capteur avec deux informations : sa longueur d'onde et sa position. En accumulant assez de photons sur un capteur, on peut déduire deux modalités de l'environnement. La luminosité, qui est le nombre de photons provenant d'un endroit particulier et la couleur qui est une combinaison de différentes longueurs d'ondes dans un spectre.

La vision ne se limite cependant pas à ces propriétés physiques. La lumière, qui est un ensemble de photons, provient d'une source (une ampoule, ou le soleil), et arrive ensuite soit directement, soit après une ou plusieurs réflexions dans l'œil ou le capteur. Ce sont ces réflexions, généralement la première, qui donnent le plus d'informations sur l'environnement. En effet, lorsque de la lumière se reflète sur un objet, il y a une interaction entre les deux. L'objet absorbe une partie spécifique du spectre lumineux et reflète le reste. La lumière réfléchie contenant dorénavant de l'information sur la position de l'objet et sa couleur. Par conséquent, les informations auxquelles nous

avons accès grâce à la vision ne sont que partielles. L'objectif du traitement visuel est d'utiliser ces informations partielles pour créer une représentation de l'environnement cohérente.

Construire une représentation cohérente est cependant complexe. Le premier problème vient de la quantité de données à traiter. De la lumière arrive constamment à nos yeux ou capteurs photosensibles, provenant de l'intégralité du champ visuel. Ce flux continu de données est multiplié par le nombre de pixels ou photorécepteurs. Ceux-ci doivent être en très grand nombre pour couvrir un champ visuel assez large aussi bien horizontalement que verticalement, et pour lesquels une grande densité est nécessaire pour obtenir une précision visuelle suffisante. Par conséquent, le nombre d'informations à analyser par tout système de traitement visuel est conséquent.

Le second problème vient de la relation des différents pixels ou photorécepteurs entre eux. Un pixel pris isolément ne donne que peu d'informations. Un pixel bleu peut par exemple provenir de la mer, du ciel, d'une voiture, d'un vêtement ou autre. Il est impossible de le savoir juste avec ce pixel, et il faut intégrer les informations contextuelles de voisinages, c'est à dire connaître les valeurs des pixels voisins, voire de l'image entière, pour pouvoir le lier à un objet. Parfois la temporalité est aussi nécessaire pour construire un environnement cohérent. Mais surtout, il faut une connaissance de ce que l'on peut trouver dans des images. Il faut savoir à quoi un objet, un arbre par exemple, correspond matériellement, sa forme physique 3D par exemple, et la rattacher à l'information visuelle qu'elle renvoie, son image, pour construire une représentation cohérente de l'environnement. Cette forme, ou matérialité de l'objet est inaccessible à partir d'une seule image, qui rappelons-le, ne contient que des informations partielles sur la réalité. Une mémoire est donc nécessaire pour tout traitement visuel complexe.

Tous ces facteurs rendent le traitement visuel un problème particulièrement ardu, qui mobilise un grand nombre de neurones dans le cerveau humain et de gros efforts de recherches en informatique. On oublie facilement cette difficulté car notre système visuel est assez développé pour résoudre aisément ces problèmes de façon inconsciente. La complexité cependant refait surface rapidement lorsque l'on change du format habituel dans lequel sont présentées les images. Différencier des chats et des chiens dans des photographies est facile pour un humain, mais faire la même tâche sur les transformées de fourier de ces photographies est plus compliqué, comme illustré dans la figure 1.1.

### 1.1.4 Vision humaine

Cette section a pour but de présenter brièvement et très schématiquement les connaissances actuelles en neurosciences de la vision humaine et de faire le lien avec les modèles que nous avons choisis : les SOM et les DNF. Les informations présentées dans les paragraphes suivants proviennent majoritairement de [GILBERT et DAS \[2020\]](#).

Le système visuel humain commence à la rétine, point de départ du nerf optique pour traverser horizontalement le cerveau jusqu'aux Corps Géniculés Latéraux dans le Thalamus situés à peu près au centre du cerveau, pour finir dans le cortex visuel à l'arrière du cerveau. Les traitements visuels se font tout du long de cette chaîne, avec une gradation allant de traitements de bas niveau dans la rétine, jusqu'aux traite-

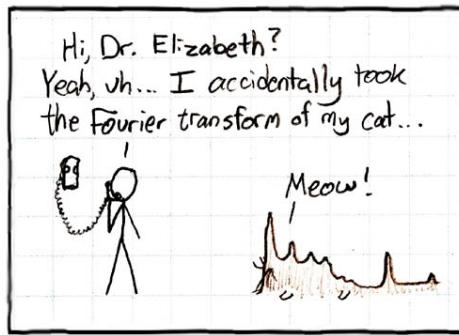


FIGURE 1.1 – Illustration des capacités du traitement visuel humain. Il est à l'aise pour reconnaître des signatures, même simplifiées, de son environnement visuel habituel mais incapable de reconnaître un chat représenté dans un espace, ici de fourrier, auquel il n'est pas habitué. Si le spectre affiché était effectivement la transformée de Fourier d'un chat.<sup>7</sup>

ments complexes de haut niveau tel que la reconnaissance de visage dans le cortex visuel. Chaque traitement est fortement parallélisé et se fait dans une zone spécifique du cerveau en première approximation, rendant le système visuel un amalgame de zones spécialisées. La stratégie "diviser pour conquérir" semble donc aussi utilisée dans le cerveau pour traiter des données complexes. Mais il existe aussi de nombreuses connexions entre les zones, chacune influençant des zones voisines et réciprocement.

Dans les premières couches du cortex visuel (V1, V2, V3 et un peu au delà), les neurones sont organisés en cartes rétinotopiques, c'est à dire en reproduisant l'arrangement spatial de la rétine. Des neurones voisins dans la carte traiteront des zones proches dans le champ visuel. Les DNF sont aussi rétinotopiques (plus de détails dans la section 1.2.4). Il y a également une organisation verticale en colonnes de neurones qui partagent des préférences proches comme la position ou l'orientation. C'est un concept qui se rapproche de l'idée qui sous-tend les cartes auto-organisatrices (section 1.2.1). Ainsi l'échange d'information dans le cortex se fait latéralement, verticalement et aussi parfois à longue distance vers une autre zone.

Un autre fait intéressant est que la distribution des neurones qui traitent chaque modalité telle que le contraste, l'orientation ou le mouvement est en proportion équivalente à la distribution de ces modalités dans l'environnement. L'implication est que le système visuel est dépendant de l'environnement. La part de génétique et d'apprentissage dans cette dépendance reste incertaine. Mais elle montre bien qu'une connaissance de l'environnement habituel est utile pour les tâches visuelles.

Le cortex visuel ne travaille pas de manière isolée du reste du cerveau, et il existe de nombreuses connexions entre celui-ci et des autres aires cérébrales comme le lobe pariétal, qui intègre toutes les informations sensorielles, ou le lobe ventral qui s'occupe de la mémoire et de la reconnaissance d'objets par exemple. Il est estimé qu'environ 10% de ces connexions entre le système visuel et les autres aires vont dans ce sens, et que les 90% restants sont dans le sens inverse. Les niveaux supérieurs du cerveau ont donc ainsi une influence importante sur comment les informations visuelles sont traitées à plus bas niveau. La raison de cette disparité est encore inconnue, mais on peut supposer, d'après les observations sur la complexité de la vision évoquées dans la sec-

7. Source de l'image : <http://xkcd.com/26> reproduit sous licence CC-BY-NC

tion 1.1.3, que l'information transmise en sens inverse est à la mesure de la connaissance préalable de l'environnement nécessaire pour compléter l'information partielle de la vision afin de construire une représentation cohérente de l'environnement.

Un dernier mécanisme essentiel de la vision humaine est l'attention. C'est un mécanisme qui mobilise plusieurs aires cérébrales de différentes zones du cerveau. Son intérêt est de concentrer les traitements complexes de la vision à une petite partie seulement du champ visuel. Cela permet d'utiliser moins de ressources pour les traitements complexes, en rendant les stimulus plus simples et en réduisant la surcharge d'informations [EVANS et collab. \[2011\]](#). C'est aussi ce mécanisme qui contrôle en partie les mouvements oculaires pour utiliser efficacement la vision foveale (une plus grande précision au centre qu'aux bords) des yeux. Nous essayerons de reproduire ce mécanisme avec des DNF dans le chapitre 4. Cependant l'attention humaine utilise une représentation complexe de l'environnement, et d'autres mécanismes cognitifs avancés, et son modèle est toujours un sujet d'études en neurosciences. Nous utiliserons donc une version simplifiée du mécanisme.

### 1.1.5 Détection de nouveauté

Il existe de nombreuses fonctions dans la vision et il serait trop ambitieux de vouloir toutes les implémenter dans cette thèse. Ainsi nous avons choisi la détection de nouveauté comme tâche à résoudre. Elle est assez complexe et haut niveau pour ne pas être triviale comme le serait la détection d'orientations (l'angle des lignes d'une image), et nécessite une compréhension de l'environnement pour séparer le fond de la nouveauté.

Les domaines de la détection de nouveauté et de la détection de changements sont l'objet de nombreuses recherches et il existe un grand nombre de modèles dans la littérature qui effectuent cette tâche. Nous présenterons les méthodes les plus courantes, celles qui ont les meilleurs résultats et celles qui sont intéressantes pour notre approche.

Les approches historiques classiques ont généralement consisté à modéliser localement les fonctions de densité de probabilité (*probability density functions* en anglais) pour classifier les pixels du fond et détecter ceux du premier plan. On peut citer par exemple les *Gaussian Mixture Models* (GMM) [ZIVKOVIC \[2004\]](#), ou plus récemment *ViBe* [BARNICH et VAN DROOGENBROECK \[2009\]](#). Certaines travaillent au niveau de groupes de pixels, comme les *Kernel Density Estimators* (KDE) [ELGAMMAL et collab. \[2000\]](#), ou combinent les deux approches [TOYAMA et collab. \[1999\]](#). Il existe aussi des méthodes différentes, par *Codebook* par exemple, comme dans [KIM et collab. \[2004\]](#). Ces approches classiques ont cependant toutes été supplantées récemment par des modèles fondés sur l'apprentissage.

Les méthodes les plus performantes actuellement utilisent des réseaux de neurones profonds [TEZCAN et collab. \[2021\]](#), [BOUWMANS et collab. \[2019\]](#). L'avantage de ceux-ci est la possibilité d'adapter pour la détection de changements des modèles ayant déjà fait leurs preuves sur d'autres problèmes. L'utilisation de réseaux pré-entraînés permet aussi de réduire le besoin de grands jeux de données d'apprentissages. Les réseaux de neurones profonds étant supervisés par nature, la généralisation requiert cependant un entraînement spécial dû à la variété d'environnements visuels et au peu

de données labellisées disponibles.

Une autre option est de combiner les approches classiques avec des réseaux de neurones, comme par exemple dans [BRAHAM et collab. \[2017\]](#) qui utilise un modèle neuronal de segmentation sémantique pour mieux représenter les objets détectés, réduisant les erreurs de l'algorithme de détection de changements. [ZENG et collab. \[2019\]](#) combine les résultats de plusieurs algorithmes de détection de changements en les faisant passer par un réseau convolutionnel.

Il est aussi intéressant de noter que les cartes auto-organisatrices ont déjà été employées pour faire de la détection de nouveauté. Le modèle le plus connu est SOBS (Self-Organized Background Subtraction) [MADDALENA et PETROSINO \[2008\]](#). Il utilise une petite SOM pour chaque pixel pour apprendre les variations de ce pixel particulier au cours du temps et ainsi décider si la variation fait partie du fond ou si c'est de la nouveauté. Ces petites SOM sont toutes reliées dans un ensemble rétinotopique plus grand représentant le fond complet de l'image et permettant un peu de prendre en compte le voisinage de chaque pixel pour la décision. D'autres versions ont été développées plus tard pour améliorer les performances et la robustesse de SOBS [GEMIGNANI et Rozza \[2016\]](#).

Comme la plupart des approches de détection de nouveauté, nous essayerons d'apprendre un modèle du fond à l'initialisation et de faire la détection de nouveauté en comparant le modèle de fond et l'image courante reçue par la caméra. Nous allons donc apprendre une image avec notre SOM, et utiliser ce que la SOM a appris pour trouver les nouveautés. Nous souhaitons explorer une approche différente de celle de SOBS, en ne travaillant pas au niveau des pixels individuels, mais de groupes de pixels. L'idée est de pouvoir intégrer plus d'informations de voisinage dans nos vecteurs d'entrée pour pouvoir effectuer un traitement de plus haut niveau avec les SOM. Nous présenterons ce concept dans le chapitre 2, consacré à la détection de nouveauté. Le chapitre 3 évoquera une amélioration de l'algorithme de la SOM pour en réduire le coût en calculs, car c'est particulièrement important pour une implantation matérielle dans un système embarqué. Le chapitre 4 adjoindra un mécanisme attentionnel avec des DNF à notre détection de nouveauté.

## 1.2 Réseaux neuronaux

Nous présenterons dans cette partie l'histoire et le fonctionnement des modèles neuronaux que nous avons utilisés. C'est à dire les cartes auto-organisatrices (1.2.1) et (1.2.2), les gaz neuronaux en expansion (1.2.3) et les champs neuronaux dynamiques (1.2.4).

### 1.2.1 Cartes auto organisatrices

Les cartes auto-organisatrices regroupent un ensemble de modèles proposé initialement par Teuvo Kohonen KOHONEN [1982]. Ces modèles sont caractérisés par leur capacité à projeter des données de façon ordonnée sur un espace d'une dimension plus faible (typiquement une ou deux dimensions). Cette réduction dimensionnelle donne ainsi une "carte" représentative des données qu'on lui a fournies, car les propriétés de voisinage sont conservées. Une des premières utilisations de ces cartes fut la représentation des phonèmes du finnois comme présenté dans la figure 1.2.

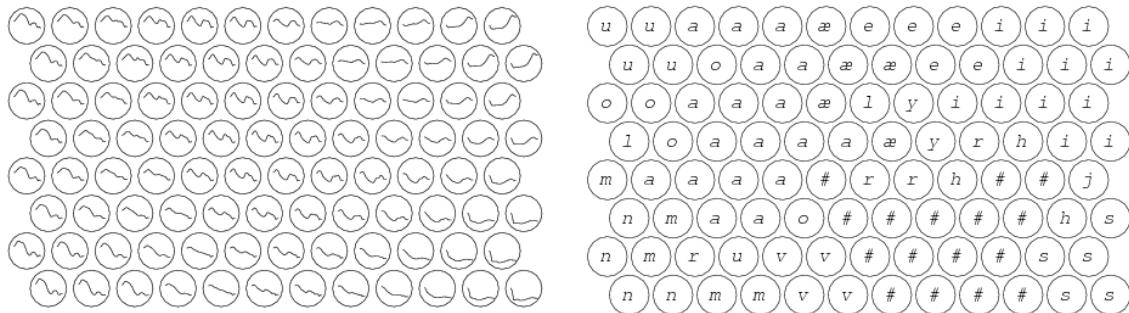


FIGURE 1.2 – Représentation des phonèmes du finnois par la première SOM. A gauche sont représentés les signaux sonores en haute dimension, et à droite leurs phonèmes correspondants. La réduction dimensionnelle provient de l'agencement de ces phonèmes sur la carte. Si ils sont proches entre eux dans leur espace d'entrée (signal), ils seront également proches dans la carte (neurones représentatifs)<sup>8</sup>.

Le but premier de Kohonen était de présenter un modèle capable de représenter informatiquement l'organisation spatiale des informations dans le cortex humain KOHONEN [2012]. Il s'inspira pour cela d'un concept du domaine des neurosciences, les colonnes corticales, qui sont des groupes de neurones arrangés verticalement et qui réagissent tous au même stimulus.

Il y a eu de nombreuses évolutions pendant près de 40 ans d'existence des SOM. En 2002, une bibliographie recensait 5384 articles scientifiques utilisant les SOM OJA et collab. [2003]. Ils étaient estimés à plus de 10000 en 2011 PÖLLÄ et collab. [2011]. Les domaines d'applications sont très variés, allant du traitement du signal de l'analyse d'image et de la parole, jusqu'à la médecine, l'économie, la finance, l'urbanisme et d'autres encore. Pour chacun de ces domaines il y a plusieurs types d'utilisations de la SOM. Elle peut par exemple être utilisée en tant que méthode de visualisation capable de rendre humainement interprétables des données de très grande dimension en les projetant sur des dimensions plus petites. Mais aussi pour faire des traitements sur

8. Source de l'image : [http://scholarpedia.org/article/Kohonen\\_network](http://scholarpedia.org/article/Kohonen_network) licence CC-BY-NC

des données, par exemple pour faire de la classification non supervisée de caractères, de chiffres ou de phonèmes, ou de la détection d'anomalies entre autres. COTTRELL et collab. [2018] présente une revue récente des différentes approches et applications des SOM.

## 1.2.2 Principes de fonctionnement

### Préparation des données

Nous présentons dans cette section le fonctionnement de l'algorithme de la SOM que nous avons utilisé. Notre version est tout à fait classique et correspond à ce qui est communément utilisé dans la littérature. De très nombreuses variantes existent néanmoins.

Les données présentées à la SOM doivent être numériques et sous forme de vecteurs. La taille des vecteurs peut être aussi grande que nécessaire, mais toutes les données de la base doivent avoir la même taille de vecteur. Nous n'avons utilisé que des données normalisées, c'est à dire, dont la valeur est comprise entre 0 et 1 inclus. Par exemple pour apprendre des couleurs avec une SOM, on pourra représenter chaque couleur par un vecteur de taille 3, un élément par composante R, G et B par exemple et normalisée pour être comprise entre 0 et 1. L'ordre de présentation des vecteurs lors de l'apprentissage est aléatoire.

### Paramètres

La forme de la SOM dépend de plusieurs paramètres. Le premier est la dimensionnalité. Les SOM peuvent aller d'une dimension de un à un nombre arbitrairement grand. Cependant, en pratique elles ne dépassent que rarement deux dimensions. La raison est que pour la visualisation de données par exemple, pouvoir afficher le résultat complet sur un écran est idéal. C'est aussi la bonne taille pour profiter de la réduction dimensionnelle sans pour autant augmenter de façon exponentielle les coûts en calculs. Une carte de 10 neurones de côté aura 100 neurones en deux dimensions et 1000 en 3 dimensions. Les coûts en calculs étant proportionnels au nombre de neurones, plus de dimensions, avec la même longueur de neurones  $n$  implique  $n$  fois plus de calculs, où à nombre de neurones équivalents, des cartes plus denses et donc avec moins de réduction dimensionnelle. Nous avons ainsi utilisé exclusivement des cartes bidimensionnelles dans nos expériences. Dans notre cas, nous avons également pris en compte la contrainte matérielle qui rend toutes les dimensions supérieures à deux difficiles à implémenter efficacement en raison des coûts en connexions "longue distance" accrus, car les circuits intégrés sont naturellement en deux dimensions.

Un second paramètre important est ce que nous appelons la topologie de la SOM. Par topologie, nous entendons la forme des connexions entre les neurones qui composent la SOM. Les deux topologies les plus communes pour les SOM sont en grille et hexagonale. Dans la topologie en grille chaque neurone a quatre voisins, un à chaque direction cardinale. En hexagone, chaque neurone a 6 voisins, formant un pavage hexagonal avec les neurones au centre des hexagones (la figure 1.2 est par exemple hexagonale). Ces topologies sont bi-dimensionnelles, mais il est possible de les rendre toriques ou sphériques. Nous n'explorerons pas cette possibilité dans cette

thèse, car cela apporte en général plus de contraintes topologiques, c'est plus difficile pour une sphère de bien modéliser l'espace d'entrée que pour une surface plane avec des degrés de libertés au extrémités. D'autres topologies plus exotiques existent et possèdent des propriétés intéressantes [BERNARD et collab. \[2018\]](#), cependant nous sommes limités aux topologies classiques, les différences entre les topologies des SOM n'étant pas notre objet d'étude ici.

Les autres hyper-paramètres de la SOM sont :

- La taille, communément notée  $n$ . Elle définit le nombre de neurones par côté de la SOM. Le nombre total de neurones  $N$  dans une SOM à deux dimensions est obtenu à partir du carré des côtés :  $n^2$ . Dans le cas d'une SOM non-carrée, on notera  $l$  et  $h$  respectivement sa largeur et sa hauteur.
- Le nombre d'époques, qui est un entier naturel définissant la durée de l'apprentissage (défini ci-dessous).
- Le coefficient d'apprentissage  $\alpha$  (alpha), défini dans  $[0, 1]$ . Il est décroissant linéairement tout au long de l'apprentissage. On notera dans cette thèse la valeur de départ et la valeur finale, toutes les valeurs intermédiaires seront extrapolées par la droite qui coupe ces deux points en fonction de l'époque courante de l'apprentissage.
- Le coefficient de voisinage  $\sigma$  (sigma), défini dans  $[0, 1]$ . Il sert à définir l'impact des neurones voisins sur les poids du neurone courant.  $\sigma$  est l'écart type du voisinage gaussien d'influence. Ainsi, plus il est élevé, plus la gaussienne sera étalée ce qui amènera à une contrainte topologique plus forte. Inversement, une valeur faible pour ce paramètre produit une gaussienne très centrée et réduit l'influence sur ses voisins de chaque neurone. Si placée à 0, elle enlève toute contrainte topologique et fait que la SOM se comporte comme un k-means. Il est important de noter aussi que nous utilisons une gaussienne normalisée. Cela implique que contrairement à une gaussienne standard, l'intégrale change en fonction de  $\sigma$ . Ainsi, plus  $\sigma$  sera grand, plus l'influence de chaque neurone sera plus grande sur tous les autres, et il n'y a pas de réduction d'influence du fait de l'étalement de la gaussienne. Comme pour le coefficient d'apprentissage, le coefficient de voisinage décroît linéairement pendant l'apprentissage et nous n'indiquerons que les valeurs de départ et de fin.

## Apprentissage

Au début de l'apprentissage, tous les poids des neurones sont initialisés aléatoirement entre 0 et 1. L'apprentissage dure un certain nombre d'époques définies avant lancement. Une époque contient le nombre d'itérations requises pour que chaque élément de la base d'apprentissage soit utilisé exactement une seule fois. Lors d'une itération, on sélectionne aléatoirement un vecteur d'apprentissage parmi la base d'apprentissage, qui n'a pas déjà été utilisé lors de cette époque.

Une itération se déroule en deux étapes :

- La phase de recherche, qui consiste à trouver la *Best Matching Unit* (BMU) parmi tous les neurones. Elle correspond au neurone qui a la plus petite distance  $L^2$  (distance euclidienne), entre ses poids et le vecteur d'apprentissage.

- La phase d’adaptation, qui modifie les poids des neurones selon l’équation suivante :

$$w_i(t+1) = w_i(t) + \alpha(t) \cdot \Theta(\sigma(t), d_{i,bmu}) \cdot (\nu - w_i(t)) \quad (1.1)$$

avec  $i$  le neurone courant,  $w_i$  les poids de ce neurone,  $t$  l’itération courante,  $\epsilon$  et  $\sigma$  des paramètres de la SOM définis dans la section 1.2.2.  $d_{i,bmu}$  est la distance  $L^1$  (distance de manhattan) normalisée entre le neurone  $i$  et la BMU pour une SOM en grille.  $\Theta$  est une fonction gaussienne centrée normalisée d’écart type  $\sigma$ .  $\nu$  est le vecteur d’apprentissage.

On répète ces deux étapes jusqu’à ce que l’on finisse la dernière époque, et l’apprentissage sera terminé. Un exemple d’apprentissage est illustré sur la figure 1.3

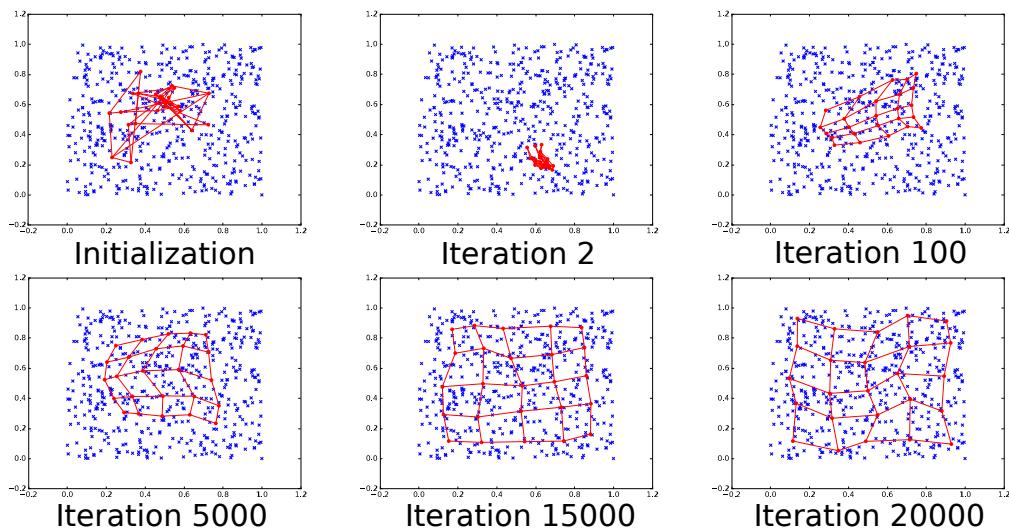


FIGURE 1.3 – Apprentissage d’une SOM. Les points bleus sont les vecteurs d’entrée en deux dimensions. Les point rouges les neurones de la SOM et les traits rouges représentent les connexions entre ceux-ci.

### Reconstruction

On appelle reconstruction le fait de remplacer un ensemble de vecteurs, de longueur similaire à ceux sur laquelle la SOM a été apprise, par les vecteurs prototypes des neurones les plus proches.

Cette reconstruction est par exemple utilisée pour de la compression de données ETTAOUIL et LAZAAR [2012], car à la place de mémoriser un ensemble de vecteurs, il n’y a besoin que de mémoriser les vecteurs prototypes de la SOM et l’indice de chaque neurone le plus proche de chaque vecteur d’entrée de l’ensemble. Cette compression est avec perte, du fait que les vecteurs prototypes des neurones ne sont en général pas exactement les mêmes que les vecteurs présentés, même lorsque ceux-ci faisaient partie de la base d’apprentissage.

### 1.2.3 Gaz Neuronaux en Expansion

Bien que nos travaux se soient focalisés sur les cartes auto-organisatrices, notre approche se veut généraliste et transposable à d’autres modèles de quantification vec-

torielle avec topologie. Nous avons souhaité valider expérimentalement cette transposition en utilisant les Gaz neuronaux en expansion. C'est un autre modèle similaire aux SOM mais avec une approche tout à fait différente sur la topologie, qui devient dynamique et induite par les données d'apprentissage à la place de fixe et prédéterminée comme dans une SOM.

## Développements

Une évolution majeure inspirée par les cartes auto-organisatrices a été le développement des différents types de gaz neuronaux (Neural Gases, NG), qui a commencé avec [MARTINETZ et collab. \[1991\]](#), en tant qu'alternative aux *k-means* car ils ne disposent pas de topologie non plus. Puis ce sont les Gaz Neuronaux en Expansion (Growing Neural Gases, GNG) [FRITZKE \[1995\]](#) qui ont sensiblement amélioré l'approche en ajoutant un mécanisme de croissance qui ajoute des neurones lors de l'apprentissage et une topologie avec des connexions qui se créent et qui disparaissent entre les neurones.

De nos jours, plusieurs variantes des GNG existent qui améliorent certains aspects de cet algorithme. Notablement, *Growing when required* [MARSLAND et collab. \[2002\]](#) adapte la croissance du nombre de neurones en fonction de ce que le réseau a déjà appris, produisant une forte neurogenèse au début de l'apprentissage et une stabilisation une fois que les données ont été suffisamment apprises. Une autre variante sont les *Incremental growing neural gases* [PRUDENT et ENNAJI \[2005\]](#). Ils permettent d'apprendre de nouvelles données sans oublier les anciennes en combinant plasticité et stabilité.

Pour la suite, nous avons décidé d'utiliser uniquement l'algorithme des Gaz neuronaux en expansion. C'est le plus utilisé et il est suffisant pour montrer que notre approche est généralisable.

## Fonctionnement

Nous ne présenterons le fonctionnement que des gaz neuronaux en expansion. L'algorithme ne se réduisant pas aisément en quelques formules, nous prendrons une approche itérative, similaire à celle que l'on peut trouver dans l'article original, pour expliquer les différents mécanismes à l'oeuvre. Il est également illustré dans la figure 1.4.

Une itération se décompose en 9 étapes :

1. Choisir au hasard un élément de la base d'apprentissage parmi ceux qui n'ont pas déjà été tirés lors de cette époque.
2. Trouver les deux neurones les plus proches :  $s_1$  et  $s_2$ .
3. Incrémenter l'âge des synapses de tous les voisins directs de  $s_1$ .
4. Ajouter la distance euclidienne au carré entre le vecteur d'apprentissage et  $s_1$  à la variable d'erreur de  $s_1$ .
5. Mettre à jour les poids de  $s_1$  et de tous ces voisins directs  $s_j$ . Les formules sont :

$$w_{s_1}(t+1) = w_{s_1}(t) + \epsilon_{bmu} \times (v - w_{s_1}(t)) \quad (1.2)$$

$$w_{s_j}(t+1) = w_{s_j}(t) + \epsilon_j \times (v - w_{s_j}(t)) \quad (1.3)$$

avec  $\epsilon_{bmu}$  et  $\epsilon_j$  des coefficients d'apprentissage.

6. Enlever toutes les synapses avec un âge supérieur à  $a_{max}$ . Si des neurones se retrouvent sans synapses, les enlever aussi.
7. Si l'itération courante est un multiple de  $\lambda$ , insérer un nouveau neurone comme suit :
  - Trouver le neurone avec l'erreur la plus élevée  $q$ .
  - Créer un nouveau neurone  $r$  à distance égale de  $q$  et de son voisin direct avec l'erreur la plus élevée  $f$ .

$$w_r(t+1) = \frac{w_q(t) + w_f(t)}{2} \quad (1.4)$$

- Ajouter des synapses d'âge 0 entre  $r$  et  $q$  ainsi qu'entre  $r$  et  $f$ . Enlever la synapse entre  $q$  et  $f$ .
  - Réduire l'erreur de  $q$  et  $f$  en la multipliant par une constante  $\alpha$ . Initialiser l'erreur de  $r$  avec la nouvelle valeur de l'erreur de  $q$ .
8. Réduire toutes les variables d'erreur en les multipliant par une constante  $d$ .

Les hyperparamètres sont donc  $\epsilon_{bmu}$  et  $\epsilon_j$  les coefficients d'apprentissage,  $a_{max}$ , l'âge maximum des synapses,  $d$  la réduction d'erreur globale et  $\alpha$  la réduction lors de l'ajout d'un nouveau neurone. Et enfin le nombre d'époques.

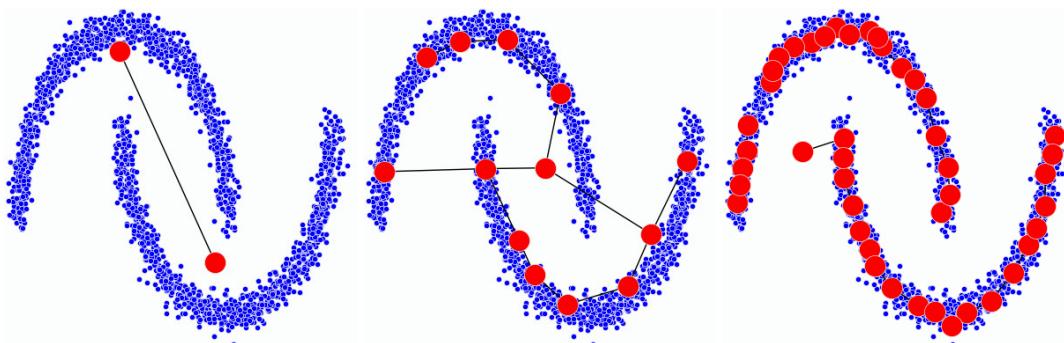


FIGURE 1.4 – Apprentissage d'un GNG de gauche à droite. Les neurones en rouge dans chaque croissant de lune sont connectés entre eux, mais il n'y a pas de connexions entre les neurones des deux croissants de lunes. La topologie du GNG représente la topologie présente dans les données.<sup>9</sup>

L'apprentissage s'arrête au bout d'un certain nombre prédéfini d'époques, comme pour la SOM. L'effet de chaque paramètre peut être compris aisément par le contexte dans lequel il est utilisé, ainsi nous n'irons pas dans le détail de chacun d'entre eux. Le paramètre  $\lambda$ , ajustant la vitesse de création de nouveaux neurones, sera fixé de telle sorte qu'à la fin de l'apprentissage il y ait le même nombre de neurones dans le GNG que dans la SOM à laquelle on souhaite se comparer. La reconstruction se passe de la même façon que pour la SOM. Les valeurs que l'on aura utilisées pour nos paramètres seront précisées dans la section expérimentale correspondante.

9. Images provenant de <https://github.com/AdrienGuille/GrowingNeuralGas>, licence MIT

## 1.2.4 DNF

Les champs neuronaux dynamiques (DNF), introduits par **AMARI [1977]** représentent mathématiquement l'évolution d'une population de neurones. Une dynamique « On center, Off surround » a été ajoutée par **ELLIAS et GROSSBERG [1975]**, qui ont employé des neurones inhibiteurs et excitateurs dans la forme typique de "chapeau mexicain" d'activations (un centre excitateur, un intermédiaire inhibiteur et des bords avec peu d'effet).

Les DNF ont été l'objets de nombreux travaux, notamment dans la vision avec **Fix et collab. [2011]** qui met en œuvre une recherche séquentielle dans un environnement (attention ouvert). Mais aussi pour du suivi de cibles comme par exemple dans **MARTEL et SANDAMIRSKAYA [2016]**.

Selon leur paramétrage, les DNF peuvent prendre de nombreux rôles. Ils peuvent faire de la sélection, filtrer du bruit ou mémoriser un stimulus grâce à leur adaptabilité et leur mécanisme de *soft winner-takes-all*. C'est pourquoi ils sont étudiés comme modèle de base pour construire des architectures multiDNF et faire émerger des mécanismes cognitifs complexes **SANDAMIRSKAYA [2014]**.

### Fonctionnement

Comme pour les SOM, les DNF peuvent être déclinés en plusieurs dimensions. Nous n'utiliserons que des DNF d'une ou deux dimensions dans cette thèse. Les neurones d'un DNF sont connectés de façon rétinotopique aux entrées, par conséquent, la taille et le nombre de neurones sont définis par la taille de la carte d'entrée. Chaque neurone est également connecté à tous les autres neurones. Enfin, tous les neurones possèdent un potentiel, qui est représenté par un nombre réel. Ce potentiel est noté  $u(x, t)$ ,  $x$  étant la position du neurone dans le champ et  $t$  un instant de la simulation. La valeur du potentiel d'un neurone intègre les influences des autres neurones, ce qui peut être modélisé sous une forme continue par l'équation différentielle suivante :

$$\tau \frac{\partial u(x, t)}{\partial t} = -u(x, t) + h + \int f(u(x', t)) \omega(x - x') \delta x' + S(x, t) g - gi \quad (1.5)$$

$$\omega(d) = c_{exc} \exp\left[-\frac{d^2}{2\sigma_{exc}^2}\right] - c_{inh} \exp\left[-\frac{d^2}{2\sigma_{inh}^2}\right] \quad (1.6)$$

$$f(x) = \frac{1}{1 + e^{-x}} \quad (1.7)$$

- $\tau$  est une constante qui règle la vitesse d'adaptation du DNF.
- $-u(x, t)$  est le terme de fuite. Son rôle est d'inhiber des neurones activés lorsqu'il n'y a plus d'activations par l'entrée ou par d'autres neurones.
- $h$  est la valeur de repos du neurone, qui doit être négative.
- $f(u(x, t))$  est une fonction logistique. Elle représente l'activation du neurone  $x$  au temps  $t$ . C'est cette valeur qui est généralement utilisée en tant que sortie du DNF.

- $\omega(x - x')$  est l'interaction latérale. Il représente l'effet d'autres neurones sur le potentiel de ce neurone. Nous utilisons une différence de gaussiennes. La gaussienne la plus pointue est excitatrice et la gaussienne plus allongée est inhibitrice. Cela a pour effet que les neurones proches sont excitateurs et les neurones lointains inhibiteurs. On appelle également les neurones ayant un impact excitateur ou inhibiteur des neurones afférents. La formule est détaillée dans l'équation 1.6, avec les quatre paramètres associés.
- $S(x, t)$  est la valeur au temps  $t$  de l'entrée à la position  $x$ .
- $g$  est le gain, et est une constante réelle utilisée pour ajuster l'effet de l'entrée.
- $gi$  est l'inhibition globale. C'est une constante divisée par le nombre de neurones présents dans la carte.

Par souci de simplicité et de calculabilité, nous avons implémenté une version spatialement et temporellement discrétisée de l'équation 1.5. Elle est obtenue en manipulant les potentiels d'un ensemble discret de neurones (carte neuronale au lieu de champ neuronal) et en utilisant une simple méthode d'Euler pour estimer l'état de  $u(x, t + \Delta t)$  en connaissant  $u(x, t)$  :

$$u(x, t + \Delta t) = u(x, t) + \frac{\Delta t}{\tau} (-u(x, t) + h + \sum f(u(x', t))\omega(x - x') + S(x, t + \Delta t)g - gi) \quad (1.8)$$

$\Delta t$  est le pas de temps entre deux estimations, il peut être le même pour tous les neurones (synchrone) ou différent à chaque fois (asynchrone).

Il est souvent ardu de comprendre le comportement d'un DNF à partir de sa formule mathématique uniquement, ainsi nous préciserons les paramètres et le comportement que nous avons choisis pour chaque DNF utilisé dans le Chapitre 4.

## 1.3 Références

- AMARI, S.-I. 1977, «Dynamics of pattern formation in lateral-inhibition type neural fields», *Biological cybernetics*, vol. 27, n° 2, p. 77–87. [7](#) [18](#)
- BARNICH, O. et M. VAN DROOGENBROECK. 2009, «Vibe : a powerful random technique to estimate the background in video sequences», dans *2009 IEEE international conference on acoustics, speech and signal processing*, IEEE, p. 945–948. [10](#)
- BERNARD, Y., E. BUOY, A. FOIS et B. GIRAU. 2018, «Np-som : network programmable self-organizing maps», dans *2018 IEEE 30th international conference on tools with artificial intelligence (ICTAI)*, IEEE, p. 908–915. [14](#)
- BOUWMANS, T., S. JAVED, M. SULTANA et S. K. JUNG. 2019, «Deep neural network concepts for background subtraction : A systematic review and comparative evaluation», *Neural Networks*, vol. 117, p. 8–66. [10](#)
- BRAHAM, M., S. PIERARD et M. VAN DROOGENBROECK. 2017, «Semantic background subtraction», dans *2017 IEEE International Conference on Image Processing (ICIP)*, IEEE, p. 4552–4556. [11](#)

- BROWN, T. B., B. MANN, N. RYDER, M. SUBBIAH, J. KAPLAN, P. DHARIWAL, A. NEELAKANTAN, P. SHYAM, G. SASTRY, A. ASKELL et collab.. 2020, «Language models are few-shot learners», *arXiv preprint arXiv :2005.14165*. [2](#)
- CAMPBELL, M., A. J. HOANE JR et F.-H. Hsu. 2002, «Deep blue», *Artificial intelligence*, vol. 134, n° 1-2, p. 57–83. [2](#)
- COTTRELL, M., M. OLTEANU, F. ROSSI et N. VILLA-VIALANEIX. 2018, «Self-organizing maps, theory and applications», *Revista de Investigacion Operacional*, vol. 39, n° 1, p. 1–22. [13](#)
- DIJKSTRA, E. W. 1974, «Self-stabilization in spite of distributed control», dans *Selected writings on computing : a personal perspective*, Springer, p. 41–46. [7](#)
- ELGAMMAL, A., D. HARWOOD et L. DAVIS. 2000, «Non-parametric model for background subtraction», dans *European conference on computer vision*, Springer, p. 751–767. [10](#)
- ELLIAS, S. A. et S. GROSSBERG. 1975, «Pattern formation, contrast control, and oscillations in the short term memory of shunting on-center off-surround networks», *Biological Cybernetics*, vol. 20, n° 2, p. 69–98. [18](#)
- ETTAOUIL, M. et M. LAZAR. 2012, «Improved self-organizing maps and speech compression», *International Journal of Computer Science Issues (IJCSI)*, vol. 9, n° 2, p. 197. [15](#)
- EVANS, K. K., T. S. HOROWITZ, P. HOWE, R. PEDERSINI, E. REIJNEN, Y. PINTO, Y. KUZMOVA et J. M. WOLFE. 2011, «Visual attention», *Wiley Interdisciplinary Reviews : Cognitive Science*, vol. 2, n° 5, p. 503–514. [10](#)
- FERRUCCI, D. A. 2012, «Introduction to “this is watson”», *IBM Journal of Research and Development*, vol. 56, n° 3.4, p. 1–1. [2](#)
- FIX, J., N. ROUGIER et F. ALEXANDRE. 2011, «A dynamic neural field approach to the covert and overt deployment of spatial attention», *Cognitive Computation*, vol. 3, n° 1, p. 279–293. [18](#)
- FRITZKE, B. 1995, «A growing neural gas network learns topologies», *Advances in neural information processing systems*, vol. 7, p. 625–632. [16](#)
- GEMIGNANI, G. et A. ROZZA. 2016, «A robust approach for the background subtraction based on multi-layered self-organizing maps», *IEEE transactions on image processing*, vol. 25, n° 11, p. 5239–5251. [11](#)
- GILBERT, C. et A. DAS. 2020, «The constructive nature of visual processing», *Principles of Neural Science*, p. 556–576. [8](#)
- HART, B. et T. R. RISLEY. 2003, «The early catastrophe.», *Education review*, vol. 17, n° 1. [3](#)
- HEBB, D. O. 1949, *The organisation of behaviour : a neuropsychological theory*, Science Editions New York. [7](#)

- KANDEL, E. R., J. H. SCHWARTZ, T. M. JESSELL, S. SIEGELBAUM, A. J. HUDSPETH et S. MACK. 2000, *Principles of neural science*, vol. 4, McGraw-hill New York. 3
- KIM, K., T. H. CHALIDABHONGSE, D. HARWOOD et L. DAVIS. 2004, «Background modeling and subtraction by codebook construction», dans *2004 International Conference on Image Processing, 2004. ICIP'04.*, vol. 5, IEEE, p. 3061–3064. 10
- KOHONEN, T. 1982, «Self-organized formation of topologically correct feature maps», *Biological cybernetics*, vol. 43, n° 1, p. 59–69. 7, 12
- KOHONEN, T. 2012, *Self-organization and associative memory*, vol. 8, Springer Science & Business Media. 12
- KRIZHEVSKY, A., I. SUTSKEVER et G. E. HINTON. 2012, «Imagenet classification with deep convolutional neural networks», *Advances in neural information processing systems*, vol. 25, p. 1097–1105. 2
- LECUN, Y., B. BOSER, J. S. DENKER, D. HENDERSON, R. E. HOWARD, W. HUBBARD et L. D. JACKEL. 1989, «Backpropagation applied to handwritten zip code recognition», *Neural computation*, vol. 1, n° 4, p. 541–551. 2
- MADDALENA, L. et A. PETROSINO. 2008, «A self-organizing approach to background subtraction for visual surveillance applications», *IEEE Transactions on Image Processing*, vol. 17, n° 7, p. 1168–1177. 11
- MARSLAND, S., J. SHAPIRO et U. NEHMZOW. 2002, «A self-organising network that grows when required», *Neural networks*, vol. 15, n° 8-9, p. 1041–1058. 16
- MARTEL, J. N. et Y. SANDAMIRSKAYA. 2016, «A neuromorphic approach for tracking using dynamic neural fields on a programmable vision-chip», dans *Proceedings of the 10th International Conference on Distributed Smart Camera*, p. 148–154. 18
- MARTINETZ, T., K. SCHULTEN et collab.. 1991, «A "neural-gas" network learns topologies», . 16
- MCCARTHY, J., M. MINSKY et N. ROCHESTER. 1955, «A proposal for the dartmouth summer research project on artificial intelligence», . 2
- OJA, M., S. KASKI et T. KOHONEN. 2003, «Bibliography of self-organizing map (som) papers : 1998–2001 addendum», *Neural computing surveys*, vol. 3, n° 1, p. 1–156. 12
- PARK, Y.-S., S. WOO, D. LIM, K. CHO et S. KIM. 2021, «Integrate-and-fire neuron circuit without external bias voltages», *Frontiers in neuroscience*, vol. 15, p. 309. 4
- PRUDENT, Y. et A. ENNAJI. 2005, «An incremental growing neural gas learns topologies», dans *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, vol. 2, IEEE, p. 1211–1216. 16
- PÖLLÄ, M., T. HONKELA et T. KOHONEN. 2011, «Bibliography of self-organizing maps», URL <http://www.cis.hut.fi/research/refs/>. 12
- SANDAMIRSKAYA, Y. 2014, «Dynamic neural fields as a step toward cognitive neuromorphic architectures», *Frontiers in neuroscience*, vol. 7, p. 276. 7, 18

- SILVER, D., A. HUANG, C. J. MADDISON, A. GUEZ, L. SIFRE, G. VAN DEN DRIESSCHE, J. SCHRITTWIESER, I. ANTONOGLOU, V. PANNEERSHELVAM, M. LANCTOT et collab.. 2016, «Mastering the game of go with deep neural networks and tree search», *nature*, vol. 529, n° 7587, p. 484–489. [2](#)
- TEZCAN, M. O., P. ISHWAR et J. KONRAD. 2021, «Bsuv-net 2.0 : Spatio-temporal data augmentations for video-agnostic supervised background subtraction», *IEEE Access*, vol. 9, p. 53 849–53 860. [10](#)
- TOYAMA, K., J. KRUMM, B. BRUMITT et B. MEYERS. 1999, «Wallflower : Principles and practice of background maintenance», dans *Proceedings of the seventh IEEE international conference on computer vision*, vol. 1, IEEE, p. 255–261. [10](#)
- TURING, A. M. 1950, «Computing machinery and intelligence», dans *Parsing the turing test*, Springer, p. 23–65. [2](#)
- TURKHEIMER, F. E., P. HELLYER, A. A. KEHAGIA, P. EXPERT, L.-D. LORD, J. VOHRYZEK, J. D. F. DAFFLON, M. BRAMMER et R. LEECH. 2019, «Conflicting emergences. weak vs. strong emergence for the modelling of brain function», *Neuroscience & Biobehavioral Reviews*, vol. 99, p. 3–10. [6](#)
- VON NEUMANN, J. 1945, «First draft of a report on the edvac», *IEEE Annals of the History of Computing*, vol. 15, n° 4, p. 27–75. [3](#)
- ZENG, D., M. ZHU et A. KUIJPER. 2019, «Combining background subtraction algorithms with convolutional neural network», *Journal of Electronic Imaging*, vol. 28, n° 1, p. 013 011. [11](#)
- ZIVKOVIC, Z. 2004, «Improved adaptive gaussian mixture model for background subtraction», dans *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, vol. 2, IEEE, p. 28–31. [10](#)

# Chapitre 2

## Détection de Nouveauté

« *D'abord, j'observe les êtres humains car je les aime bien. J'enregistre dans ma tête tout ce que j'ai remarqué, et ensuite, avec les souvenirs de ce que j'ai vu, je dessine.* »

---

Hayao Miyazaki

### Sommaire

---

<b>2.1</b>	<b>Introduction</b>	<b>24</b>
<b>2.2</b>	<b>Application aux images</b>	<b>24</b>
2.2.1	Apprentissage et reconstruction	26
2.2.2	La gestion des canaux (couleurs)	27
<b>2.3</b>	<b>Détection de nouveauté par QVAT : quantification vectorielle et apprentissage topologique</b>	<b>29</b>
2.3.1	Détection avec quantification vectorielle	29
2.3.2	Détection avec distance neurale	31
2.3.3	Considérations pour la combinaison	32
<b>2.4</b>	<b>Protocole expérimental</b>	<b>34</b>
2.4.1	Présentation de la base de données	34
2.4.2	Métriques utilisées	38
2.4.3	Préparation des données	41
2.4.4	Paramétrages des modèles	42
<b>2.5</b>	<b>Résultats expérimentaux</b>	<b>49</b>
2.5.1	Seuil de décision	49
2.5.2	Nombre de neurones et taille des imagettes	50
2.5.3	Résultats complets	54
2.5.4	Visualisation des résultats	57
2.5.5	Interprétations	58
<b>2.6</b>	<b>Références</b>	<b>60</b>

---

## 2.1 Introduction

Ce chapitre présente notre méthode pour réaliser une détection de nouveauté avec les cartes auto-organisatrices (SOM), ou plus généralement, avec des modèles de quantification vectorielle avec topologie.

Nous commencerons par montrer différentes façons d'apprendre les images avec des SOM et celle que nous avons choisie. Puis nous introduirons notre méthode de détection de nouveauté en deux parties : d'abord avec quantification vectorielle, puis une seconde avec la topologie. Ensuite nous aborderons le protocole expérimental que nous avons mis en œuvre pour évaluer notre modèle. La dernière section présente nos résultats.

## 2.2 Application aux images

Il existe de différentes possibilités pour apprendre des images avec la quantification vectorielle, dû aux multiples façons de découper une image en vecteurs d'apprentissage. Pour présenter l'impact que peut avoir ce découpage, on peut considérer deux extrêmes : apprendre l'image en entier ou apprendre chaque pixel individuellement. Le premier cas peut sembler absurde dans le cas d'une seule image (une base de données d'un seul élément), mais peut présenter un intérêt lorsque l'on considère une suite d'images par exemple. Dans ce cas, l'environnement appris serait défini par l'entièreté de ce que voit le capteur et tous les changements, où qu'ils soient dans l'image, auraient de l'importance.

Le second extrême est l'apprentissage au niveau du pixel. Dans cette approche chaque pixel est un vecteur d'entrée, ce qui rendrait l'espace d'entrée unidimensionnel pour une image en niveau de gris (ou tridimensionnel si l'image est en couleur, plus de détails dans la section 2.2.2). Sur un plan conceptuel, ce choix considère qu'une image est définie uniquement par les couleurs ou luminosités présentes, peu importe leur positions dans celle-ci. C'est utilisé notamment dans SOBS (Self-organized background subtraction) [MADDALENA et PETROSINO \[2008\]](#), et toutes ses déclinaisons ultérieures. Il existe beaucoup d'autres découpages possibles entre ces deux extrêmes, chacun représentant une certaine façon de conceptualiser une image et définissant la notion de nouveauté. Le premier cas intègre la luminance, la couleur, et leur répartition spatiale dans le vecteur d'apprentissage alors que le deuxième ignore la relation spatiale entre les luminances/couleurs. Ainsi le premier modèle est spécifique à la situation apprise et sera très sensible aux mouvements des objets dans la scène. Le deuxième modèle est plus générique en réagissant aux objets dont la nature colorimétrique ou de luminance aura changé.

Il est important donc de considérer le contexte de nos travaux pour définir la façon de représenter une image, notre application étant la détection de nouveauté. Dans ce contexte, nous considérons qu'une image est une combinaison de nombreux éléments plus petits. Par exemple, une photographie d'un lac de montagne 2.1 peut être décrite comme étant la combinaison d'un élément de lac (avec sa couleur, bleu sombre et sa texture uniforme), d'un élément de plaine herbeuse (verte et uniforme), d'éléments rocheux qui sont gris et soit uniformes (dans le premier plan) soit plus contrastés en se combinant avec la verdure de la végétation (dans les bords de l'image),

et ainsi de suite. Ce découpage "sémantique" de l'image est ce qui permet la détection de nouveauté de fonctionner, car dans notre cas la nouveauté est par définition ce qui n'est pas déjà dans l'image et donc ne faisant pas partie de ces classes d'éléments. Pour regrouper les parties d'une image appartenant au même élément, il est nécessaire d'avoir une information mise en contexte (un pixel seul ne suffit généralement pas à savoir à quel élément il appartient dans l'image), ce qui implique que les pixels doivent être pris dans leurs environnements locaux pour conserver l'information de voisinage comme la texture par exemple. Nous avons donc choisi de découper l'image en plus petites images à la façon d'une mosaïque, que nous appellerons des imagettes. Ces imagettes (d'une taille arbitraire en hauteur et largeur) conservent l'environnement local tout en étant suffisamment petites pour être précises dans l'espace. Une imagette ne représentera qu'une partie d'un élément et non pas regroupant plusieurs éléments, ce qui réduirait sa capacité de généralisation. Mais elles permettent aussi d'avoir une base d'apprentissage suffisamment étendue pour tirer parti des propriétés de nos modèles de quantification vectorielle et de leur topologie. La partie pratique de l'apprentissage d'une image, sa représentation et sa reconstruction sont abordés dans la section suivante [2.2.1](#).

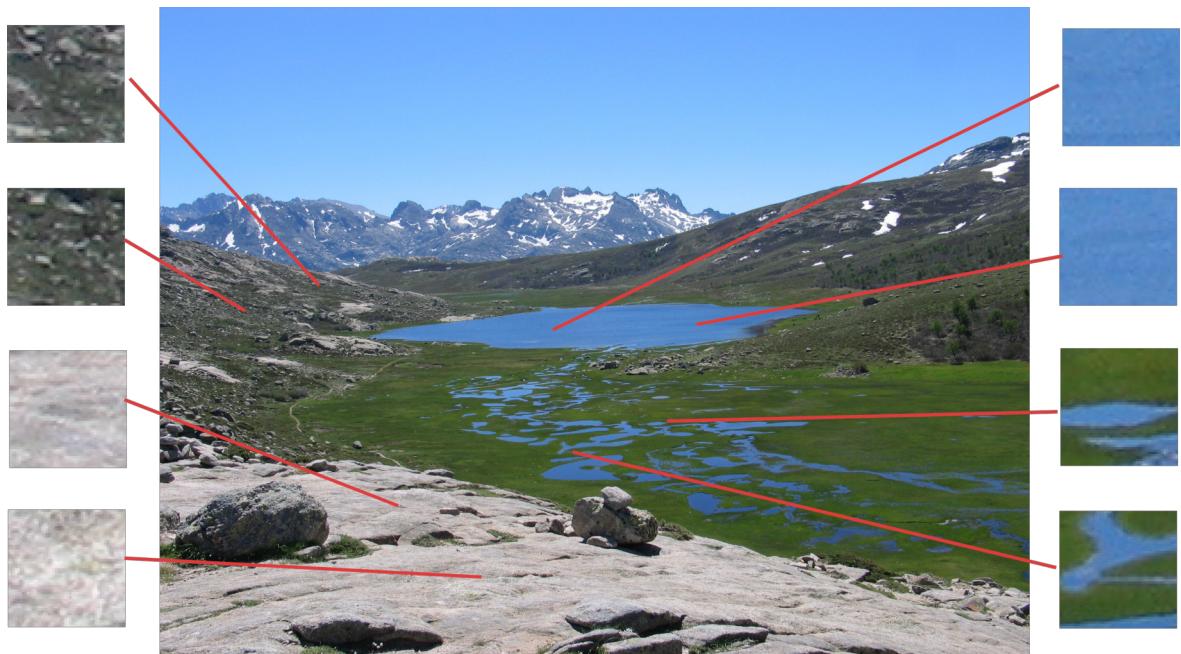


FIGURE 2.1 – Exemple d'image comportant plusieurs éléments notables tels qu'un lac (bleu sombre et uniforme), une plaine herbeuse (verte et uniforme), d'éléments rocheux qui sont gris et soit uniformes (dans le premier plan) soit plus contrastés en se combinant avec la verdure de la végétation (dans les bords de l'image), et ainsi de suite. Une image peut donc être vue comme un ensemble de zones différentes composées d'éléments proches entre eux.<sup>1</sup>

---

1. Image source : Lac de Nino, [https://fr.wikipedia.org/wiki/Lac\\_de\\_Nino#/media/Fichier:Lac\\_de\\_Nino.jpg](https://fr.wikipedia.org/wiki/Lac_de_Nino#/media/Fichier:Lac_de_Nino.jpg), CC BY-SA

## 2.2.1 Apprentissage et reconstruction

### Apprentissage

La première étape nécessaire à l'apprentissage est la constitution de la base d'apprentissage à partir de l'image. L'utilisation de modèles de quantification vectorielle établissent la première contrainte pour le découpage : les imagettes doivent être d'une taille fixe. En effet il est nécessaire pour les SOM comme pour les GNG et leurs variantes que tous les vecteurs d'entrées soient de la même longueur pour que le calcul de distance avec les neurones fonctionne, pour qu'ils représentent le plus fidèlement possible les entrées qui leurs sont attachées.

Nous avons également choisi de limiter nos tailles d'imagettes à des carrés, pour des raisons de simplicité. Il est possible que des imagettes plus larges que hautes ou plus hautes que larges représentent mieux les éléments de l'image que l'on apprend. Cependant ce serait une préférence spécifique à chaque image et peu généralisable, car si on effectue par exemple une rotation de 90° de l'image, la préférence s'inversera. L'inversement des tailles dans les imagettes carrées ne changeant rien, elles sont pour leur part insensibles aux rotations discrètes de l'image (par pas de 90°).

Dans la version classique [AMERIJCKX et collab. \[2003\]](#), le découpage de l'image se fait en mosaïque, sans superpositions entre les imagettes. C'est à dire que chaque pixel n'appartient qu'à une seule imagette. Le processus est montré sur la figure 2.2. Si les dimensions de l'images ne sont pas un multiple de la taille des imagettes, les pixels en trop sont rognés par la droite et par le bas.

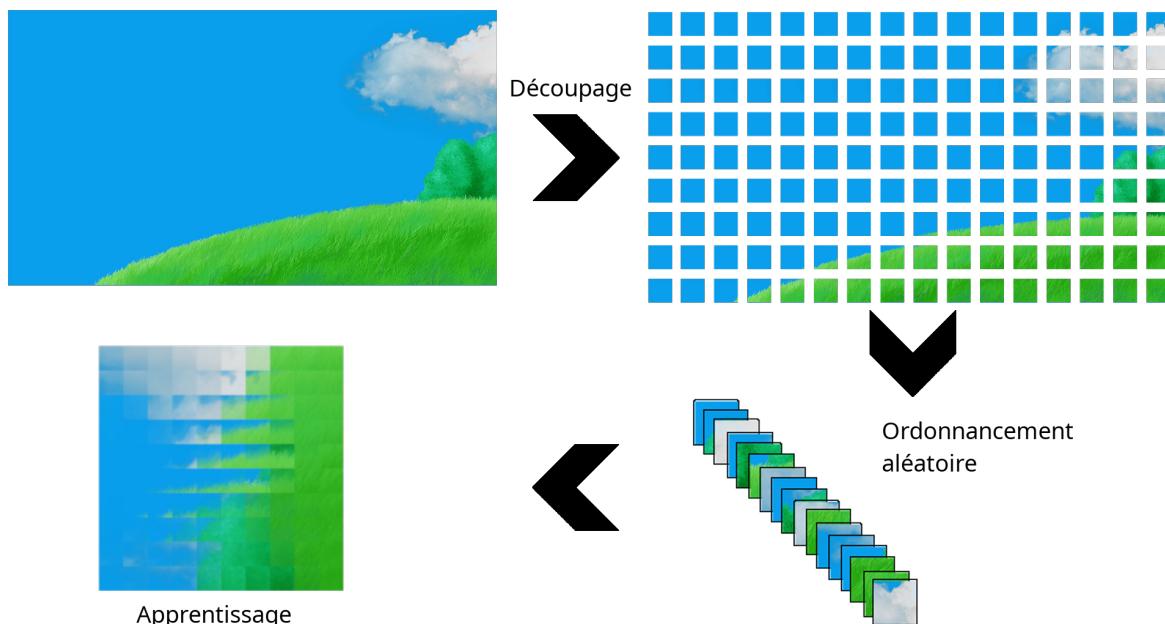


FIGURE 2.2 – Illustration du processus de représentation et d'apprentissage d'une image par une SOM.

## Reconstruction

Une fois l'apprentissage terminé, on obtient deux sorties. La première est le modèle entraîné avec les différents poids des neurones codant une imagette représentante du cluster d'imagettes associé à ce neurone. La seconde est la liste, pour chaque imagette de l'image, de l'indice du neurone le plus proche de celle-ci, qui pourra être utilisé pour la reconstruction de l'image d'apprentissage. Le procédé est montré sur la figure 2.3.

Pour reconstruire une image à partir de la liste des indices de BMU, il suffit de remplacer chaque indice par le vecteur prototype du neurone auquel il correspond. Ces vecteurs prototypes devront être réassemblés en imagettes (dans un tableau à 2 dimensions à la place d'un vecteur à une dimension), et seront placés à la bonne position pour reformer l'image.

Il est aussi possible de reconstruire une image qui n'a pas été apprise. Pour cela il faut créer la liste d'indices des BMU des imagettes de la nouvelle image, et de reconstruire ensuite l'image par le même procédé que montré précédemment. Il faut noter que l'image que l'on souhaite reconstituer doit être proche de l'image apprise, c'est à dire contenir des éléments similaires, pour obtenir un résultat correct.

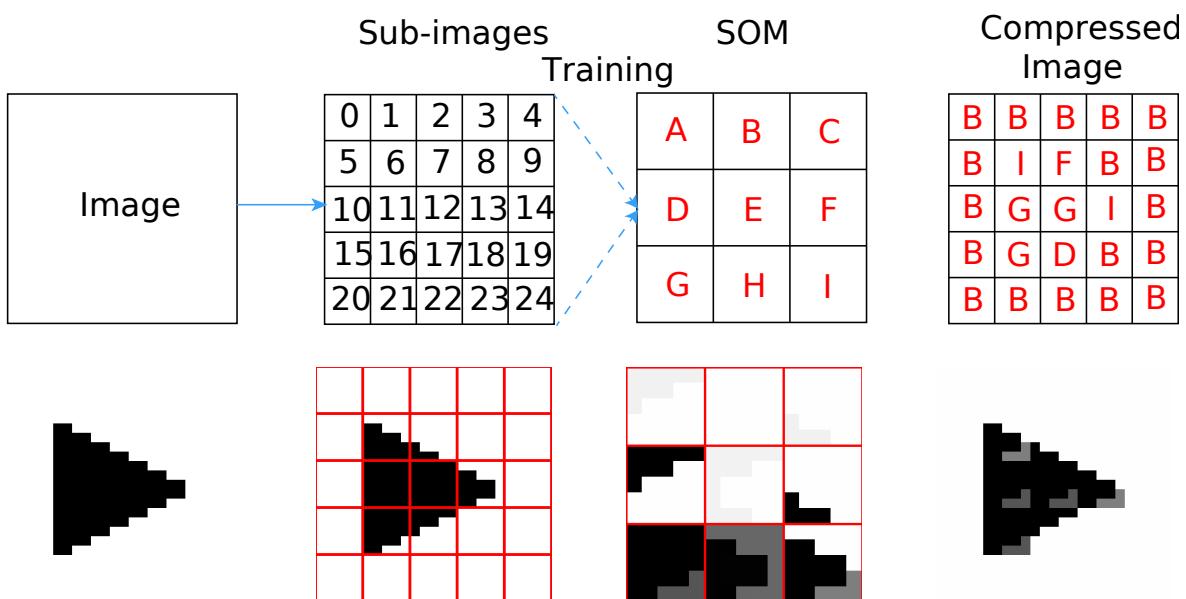


FIGURE 2.3 – Schéma simplifié du processus de compression et de reconstruction d'une image, avec ici seulement 9 neurones et 25 imagettes.

### 2.2.2 La gestion des canaux (couleurs)

Nous avons jusque là vu comment apprendre une image ayant une seule valeur par pixel (soit des images en nuances de gris). Cependant la majorité des dispositifs de capture actuels fournissent des images en couleurs, c'est à dire avec trois canaux. Nous allons voir dans cette section comment transposer l'apprentissage, la compression et la reconstruction à des images à un nombre arbitraire de canaux par pixels.

Une approche possible serait de séparer l'image par composante. Une image RVB par exemple donnerait trois images en niveau de gris, une R, une V et une B. Deux options s'offrent ensuite à nous. Soit utiliser une seule SOM pour apprendre toutes les images ainsi extraites en espérant que les différentes formes présentes dans chaque composante soient assez similaires entre elles. Cela augmente aussi les données que la SOM doit apprendre, car on vient de multiplier la taille de notre base d'apprentissage par le nombre de canaux. Ces données doivent être aussi cohérentes dans l'espace d'entrée pour que la réduction dimensionnelle se fasse correctement. Soit utiliser une SOM par composante pour apprendre chaque canal séparément, et de regrouper ensuite les différents canaux reconstitués en une image couleur.

Cependant ces deux approches ont un défaut majeur pour la compression d'images (ainsi que la détection de nouveauté en conséquence), c'est la création d'aberrations chromatiques dans l'image reconstituée. En effet, les canaux étant appris séparément avant d'être recombinés, la reconstruction peut donner pour certains pixels des teintes de couleurs qui n'existaient pas dans l'image de base, qui peuvent être très saillants visuellement. Par exemple un pixel blanc dans l'image d'entrée (avec une forte composante R, V et B), lorsque reconstitué par la SOM peut être bien reconstitué dans deux composantes (V et B par exemple), et mal reconstitué dans la troisième (R) avec une valeur beaucoup plus faible que dans l'image de base, car on minimise la distance entre les canaux séparément. Par conséquent ce pixel aura une couleur turquoise dans l'image reconstituée à la place de blanc. Cette erreur minimise bien la distance avec l'image de base, et ce n'est que visuellement que les changements de teintes sont apparents et dégradent proportionnellement plus la qualité de l'image que l'erreur mesurée.

Une meilleure approche consiste à inclure tous les canaux directement dans les imagettes. Chaque imagette devient donc une imagette en couleur, et sa taille augmente donc en conséquence. Une imagette de 10 par 10 pixels par exemple qui donnerait un vecteur prototype de taille 100, passe à 300 avec les trois couleurs. L'apprentissage et la reconstruction se déroulent de la même manière que dans la SOM classique. Il faut aussi noter que l'ordre n'a pas d'importance dans les vecteurs prototypes, on peut arranger les valeurs en RGBRGBRGB tout comme RRRGGGBBB sans que cela ne change le résultat, le calcul de distance euclidienne étant indépendant de l'ordre des coordonnées. La figure 2.4 montre une comparaison de reconstructions obtenues entre couleurs fusionnées et couleurs séparées.

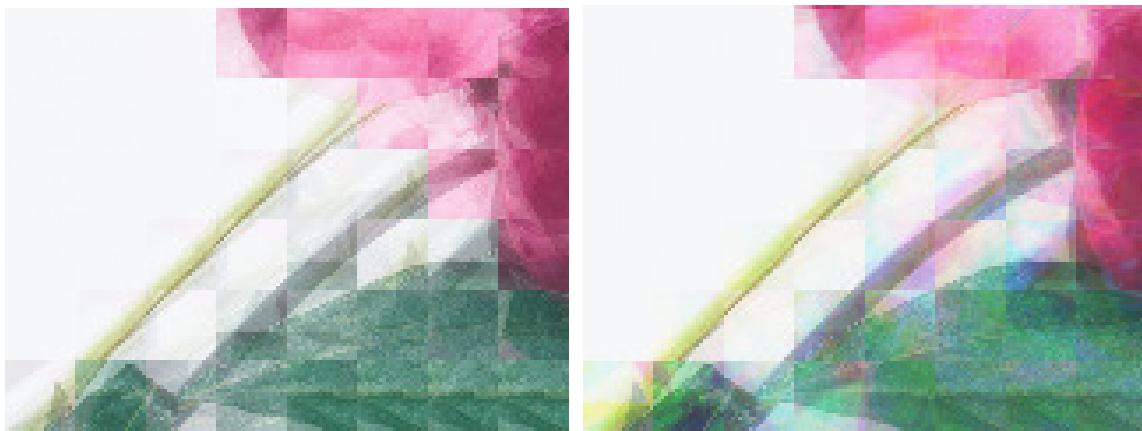


FIGURE 2.4 – Comparaison entre une image avec des couleurs fusionnées et la même image avec des couleurs séparées qui présente des artefacts visuels.<sup>2</sup>

## 2.3 Détection de nouveauté par QVAT : quantification vectorielle et apprentissage topologique

Nous avons à partir des différentes propriétés de nos modèles neuronaux et de l'apprentissage d'images, développé des processus de détection de nouveauté. Ces processus ne sont pas intrinsèques à nos modèles, c'est à dire que nos modèles neuronaux n'ont pas été définis dans le but d'effectuer une détection de nouveauté. Elle est une propriété émergente de ces modèles. Nous présentons ces deux méthodes dans cette section. La première est basée sur la propriété de quantification vectorielle et la seconde sur la topologie des SOM, avec une modulation par ce que nous appelons la distance neurale. La première a été introduite dans BERNARD et collab. [2019], et complémentée par la seconde dans BERNARD et collab. [2020].

### 2.3.1 Détection avec quantification vectorielle

La détection par quantification vectorielle (VQ) repose sur l'erreur de reconstruction de la nouvelle image provenant du capteur. Cette erreur est la différence entre la nouvelle image et celle reconstruite avec le modèle de VQ ayant effectué son apprentissage sur l'image de fond.

Il y a deux cas à considérer pour chaque imagette de la nouvelle image du capteur : soit la nouvelle imagette est similaire à une partie quelconque de l'arrière-plan appris, soit quelque chose de nouveau est présent dans l'imagette. Dans le premier cas, le neurone représentatif de cette imagette sera proche de celle-ci, c'est à dire que les poids du neurones et de l'imagette seront proches. L'erreur de reconstruction de cette imagette sera faible, surtout si la caméra est statique, mais avec une petite différence toujours présente en raison des pertes de la compression et des changements naturels de l'environnement visuel.

Cependant, dans le second cas, le neurone représentatif de l'imagette sera éloigné de celle-ci, dans le sens où ses poids seront très différents de ceux de l'imagette.

2. Image source : [https://unsplash.com/photos/9A\\_peGrSbZc](https://unsplash.com/photos/9A_peGrSbZc)

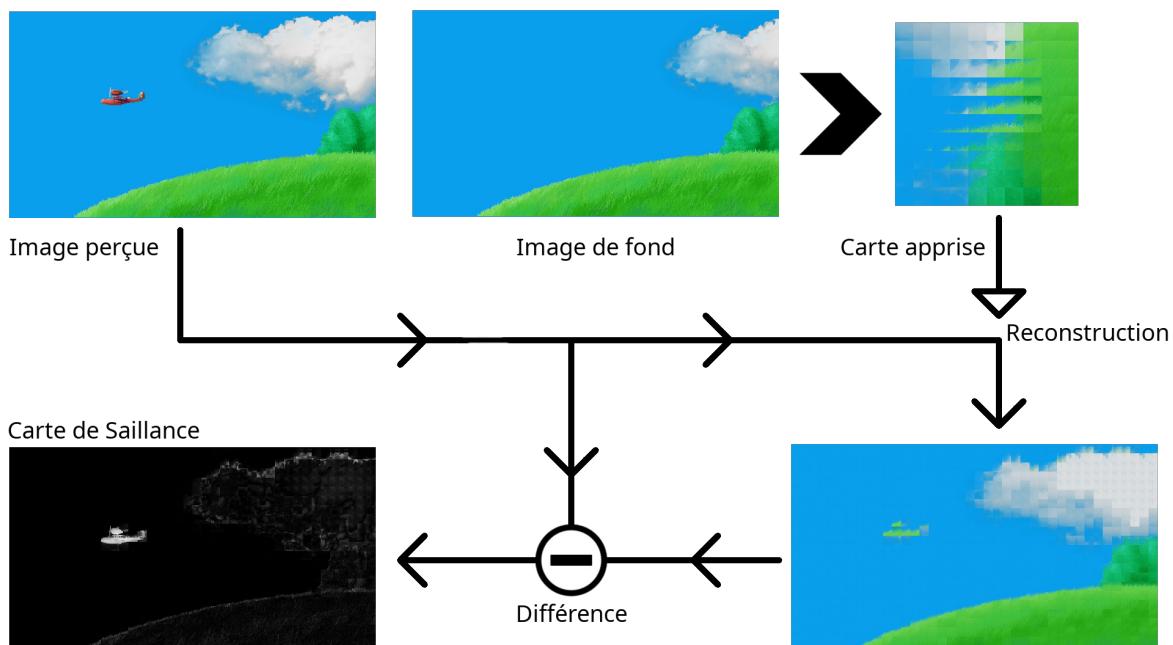


FIGURE 2.5 – On peut observer qu'il y a eu deux changements entre le fond et l'image perçue : un avion est apparu et les nuages ont bougé. Les nuages, déjà présents dans le fond sont bien reconstruits. L'avion cependant est nouveau, et n'est pas bien reconstruit. Ainsi la différence entre l'image perçue et la reconstruction rend plus saillant l'avion que les nuages. Contrairement à une simple différence entre le fond et l'image perçue, où les deux seraient saillants. Nous avons représenté le modèle appris comme étant une SOM sur cette figure, cependant il peut s'agir de n'importe quel modèle de quantification vectorielle.

Car la nouveauté est par définition quelque chose qui n'était pas présent dans l'arrière-plan et donc quelque chose que le modèle n'a pas appris. Cela entraînera une différence significative entre l'image perçue et sa reconstruction à l'endroit de la nouveauté. Ce processus est illustré dans la figure 2.5, et des exemples de résultats dans la figure 2.6.

Ce processus a l'avantage d'être précis au niveau du pixel pour la mise en évidence des changements, avec une certaine limite dûe à la taille des imagettes. Si les changements sont trop importants et que la nouvelle BMU est trop différente de celle de référence, alors il y a un risque de perte de précision au niveau de l'imagette. Il est aussi théoriquement insensible au déplacement d'objets du fond pour la détection de la nouveauté. Cependant, il peut être bruité en raison de l'apprentissage imparfait de la VQ et de la variabilité naturelle de l'environnement.

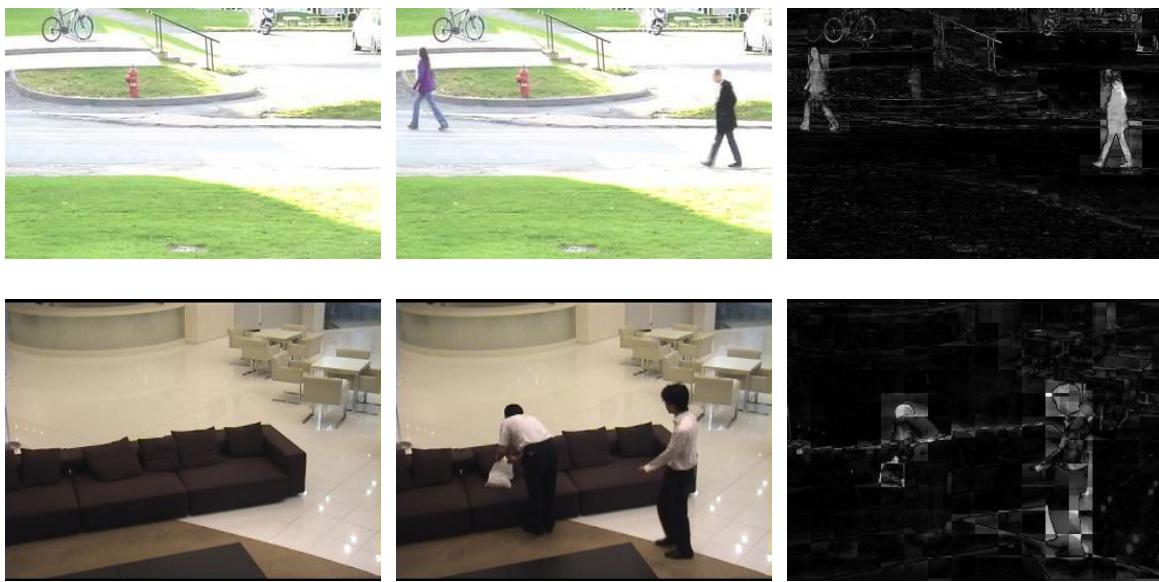


FIGURE 2.6 – Exemples de différence avec quantification vectorielle. On peut remarquer que la couleur est un facteur important pour savoir si un objet nouveau sera correctement détecté.

### 2.3.2 Détection avec distance neurale

La détection avec distance neurale se base sur les propriétés topologiques du modèle pour trouver la nouveauté dans une image. La topologie est ce qui relie les différents neurones de modèle par leur proximité : les neurones proches dans la topologie ont appris des poids similaires.

Après avoir effectué l'apprentissage du modèle, nous mémorisons la liste des positions dans la carte des BMU trouvés pour toutes les imagettes. Lorsqu'une nouvelle image est présentée au capteur, nous effectuons le processus de reconstruction comme dans la première méthode de détection de la nouveauté. Mais nous nous intéressons uniquement à la liste des positions dans la carte des BMU trouvées pour toutes les imagettes de la nouvelle image. En comparant les deux listes, nous pouvons trouver des changements dans les positions des BMU pour chaque emplacement d'imagette. Si la BMU est la même entre le fond et la nouvelle image à un endroit,

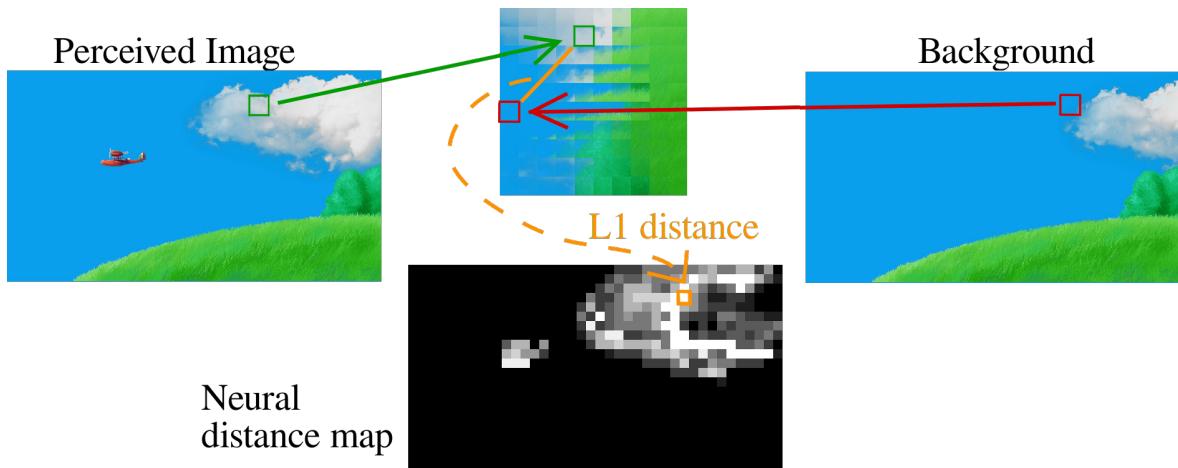


FIGURE 2.7 – Le processus présenté ici concerne une position dans l'image, et il est répété sur toute l'image pour obtenir la carte de distances neurales en bas. Nous avons représenté le modèle appris comme étant une SOM sur cette figure, cependant il peut s'agir de n'importe quel modèle avec une topologie regroupant les éléments proches.

alors il n'y a probablement pas de nouveauté à cet endroit. S'il y a une différence entre les deux BMU, alors il pourrait y avoir de la nouveauté à cet endroit. Pour quantifier cette différence, nous calculons la distance topologique qui les sépare sur la carte. La distance topologique est définie par le nombre de noeuds qui sont traversés par le chemin le plus court entre les deux BMU dans la topologie de la carte. Dans la SOM classique basée sur une grille, il s'agit simplement de la distance L1.

Grâce à ces distances, nous pouvons créer une carte de saillance où des distances élevées dans la topologie signifient des changements significatifs dans l'image, car la proximité dans la topologie signifie la proximité dans l'espace d'entrée. Cette méthode permet d'obtenir une carte de saillance plus robuste, avec moins de bruit que la version avec la VQ. La précision est limitée à la taille des imagettes et il n'y a pas d'inhibition pour les éléments déjà connus qui se sont déplacés dans le fond. Le processus est illustré dans la figure 2.7.

### 2.3.3 Considérations pour la combinaison

Une fois les deux cartes de saillance générées, il est nécessaire de les combiner pour n'avoir qu'un seul résultat qui représentera la sortie de notre système. Il existe un très grand nombre de façons de faire cette combinaison, et il existe dans la littérature des modèles qui se basent sur une bonne combinaison de différentes cartes de saillance pour obtenir de meilleurs résultats [BIANCO et collab. \[2017\]](#). Dans notre cas, nous avons préféré utiliser une combinaison simple de nos deux cartes de saillance. C'est à dire qu'elle n'utilise pas de paramètres, pour ne pas ajouter une variable de plus à optimiser. Nous souhaitons aussi bénéficier de la complémentarité des deux cartes de saillance. La solution la plus simple est de multiplier les deux cartes ensemble. Ainsi, sera considéré comme nouveauté dans la carte de sortie, ce qui apparaît comme nouveauté en même temps dans les deux cartes de saillance, sur le principe de  $\text{petit} \times \text{petit} = \text{petit}$ ,  $\text{grand} \times \text{petit} = \text{petit}$  et seulement  $\text{grand} \times \text{grand} = \text{grand}$ , à l'instar d'un "et" logique. Le bruit présent dans la carte résultant de la quantification vectorielle et qui n'est pas présent dans la carte topologique disparaît de la carte fi-

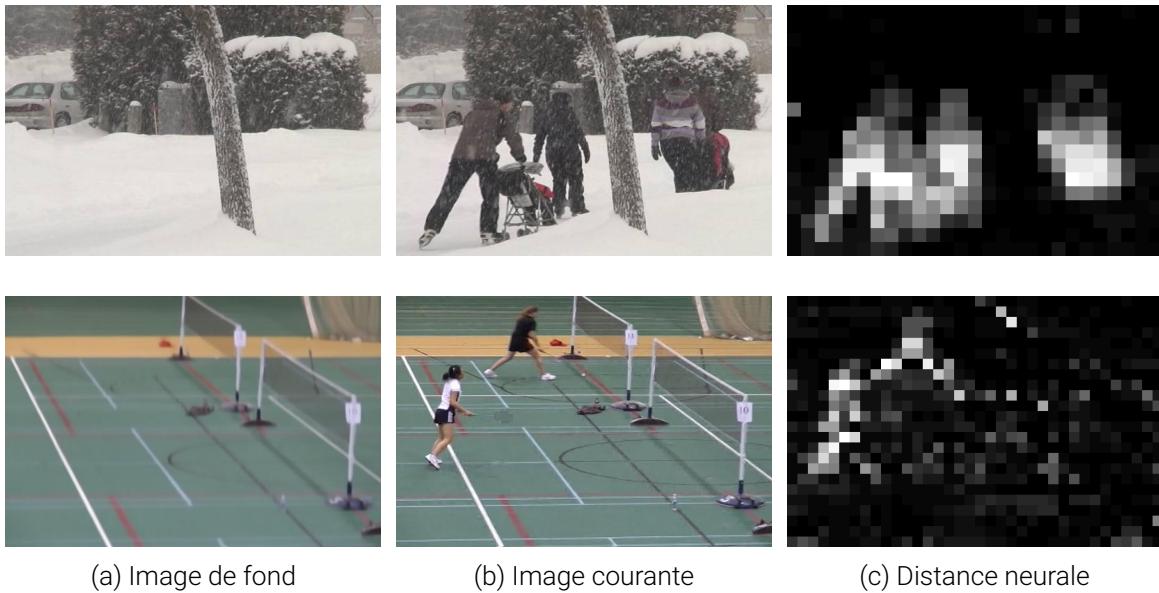


FIGURE 2.8 – Exemples de différence avec distance neurale.

nale. Il en va de même pour les mouvements qui ne sont pas des nouveautés qui apparaissent dans la carte topologique, mais pas dans la carte de quantification vectorielle.

Un problème qui peut apparaître avec cette méthode est la trop petite valeur du résultat et le déséquilibre d'impact de nos deux cartes, car nos cartes de saillance sont toutes les deux définies entre 0 et 1. Il est possible qu'il existe des situations où les deux cartes ont un impact disproportionnel sur le résultat. Par exemple si la saillance a une valeur de 0.2 sur une carte et 0.8 sur l'autre, alors la seconde aura plus d'impact sur les valeurs de la sortie finale. De même, en multipliant deux nombres compris entre 0 et 1, le résultat sera forcément inférieur à chacun des deux nombres. Cela a un effet réducteur sur toutes les valeurs de la carte de saillance finale. La solution que nous avons choisi à ces deux problèmes, est de re-normaliser les deux cartes de saillances avant de les multiplier. Cela a pour effet d'éviter une trop grande disproportion d'impact entre les deux cartes.

Cependant, cela vient aussi avec des désavantages, comme par exemple le fait que si il n'y a pas de signal dans l'entrée, le maximum des cartes de saillance sera quand même 1. On pourrait observer des signaux positifs dans la sortie alors que l'entrée et les cartes de saillances n'en montrent pas. En pratique, cela est peu fréquent car la valeur maximum dans un cas où il n'y a pas de signal en entrée vient du bruit de reconstruction, et est donc décorrélée entre les deux cartes car la distance neurale n'y est pas sujette, et disparaîtra lors de la multiplication.

Une autre option aurait été de pondérer les deux cartes avant de les multiplier. Nous ne l'avons pas choisie car cela aurait eu pour effet de rendre la valeur de la nouveauté dans la carte de saillance très variable : différentes nouveautés pouvant générer des saillances différentes, par forcément représentatives de leur qualité de nouveauté. Or, nous ne nous intéressons pas à la valeur absolue de la nouveauté, qui change en fonction de la séquence évaluée, mais à la différence relative entre la nouveauté et le fond. Ce qui ne peut pas être obtenu avec une pondération.

## 2.4 Protocole expérimental

Cette section regroupe l'ensemble des considérations pratiques pour la réalisation de nos expériences. Nous présenterons la base de donnée utilisée, comment ces données ont été préparées, les différentes métriques que nous avons utilisées et les paramétrages de nos modèles.

### 2.4.1 Présentation de la base de données

Nous avons choisi d'utiliser la base de données ChangeDetection.Net (CDnet) [WANG et collab. \[2014\]](#) pour réaliser nos expériences. C'est un jeu de données très utilisé dans le domaine de la détection de changements dans le domaine visible et infrarouge avec une caméra statique. Ce sont des données tirées de situations réelles par le biais de caméras de surveillance par exemple, et dédiées à la détection de changements.

Nous avons dans cette thèse avant tout parlé de détection de nouveauté, et il existe une différence conceptuelle entre la nouveauté et le changement. La détection de changement se concentre sur le mouvement pour séparer le fond des objets intéressants dans une image. La détection de nouveauté quant à elle se réfère à une représentation apprise de l'environnement. Dans les captures vidéos réelles, les deux sont généralement équivalentes dû au fait que lorsqu'une nouveauté apparaît, elle le fait généralement en se déplaçant. En pratique cela veut dire que la majorité, mais pas la totalité, de CDnet peut être utilisée pour de la détection de nouveauté. Nous présenterons les catégories et vidéos que l'on a utilisées, et des exemples de vidéos qui n'ont pas été retenues avec des explications dans la suite.

CDnet regroupe 53 séquences vidéos d'origines variées. Elles proviennent principalement de caméras de surveillance ou de captures effectuées par des chercheurs pour leur propres besoins. Il y a des captures réelles, sans images synthétiques, de scènes intérieures et extérieures. Les vidéos sont toutes en couleur, sauf pour deux catégories *thermal* et *turbulences*, et de résolution assez faible, allant de  $320 \times 240$  à  $720 \times 486$  pixels. Elles sont groupées en 11 catégories de 4 à 6 vidéos représentant une variété de difficultés rencontrées sur le terrain. Il existe cependant un certain biais dans CDnet, car il est fortement orienté vers de la détection de personnes et de véhicules. Ces catégories sont présentées dans les figures suivantes.

Parmi les 11 catégories de CDnet, nous avons décidé d'en utiliser 8 complètement, d'utiliser une version réduite pour 2 d'entre elles et d'en enlever une. Les deux catégories réduites sont *Intermittent Object Motion* et *Low framerate*. La réduction consiste à enlever une partie des vidéos qui ne correspondaient pas à notre application visée de détection de nouveauté et de conserver les autres. Une illustration de la différence entre la détection de changements et la détection de nouveauté, qui est la raison pour laquelle nous avons écartées ces vidéos, est montrée sur la figure 2.19. La catégorie que nous avons décidé de ne pas du tout utiliser car elle ne correspondait pas à notre scénario est *Pan tilt zoom*. C'est une catégorie un peu spéciale car elle change une partie fondamentale du scénario. La caméra n'est plus statique mais effectue des rotations et des zooms ce qui change significativement son environnement visuel.



FIGURE 2.9 – *Baseline* : La catégorie de base qui comprend des scénarios typiques de détection de changement (trafic, piétons) sans difficultés particulières.



FIGURE 2.10 – *Bad Weather* : Cette catégorie comprend des variations du scénario de base avec une météo dégradée. La difficulté principale vient de la neige qui tombe, et du changement de l'environnement avec les traces de pneus sur la neige par exemple.

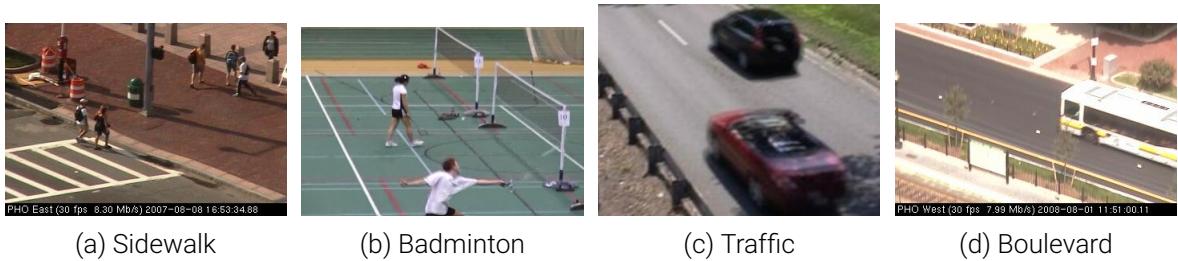


FIGURE 2.11 – *Camera Jitter* : Ces vidéos proviennent de caméras instables à cause de vent fort ou d'autres raisons. Elles ont de façon irrégulière des translations verticales et horizontales rapides et de petite amplitude.



FIGURE 2.12 – *Dynamic Background* : La difficulté se porte sur le contenu du fond qui est changeant. Il peut s'agir d'eau ou d'arbres qui bougent dans le vent.



FIGURE 2.13 – *Shadow* : Catégorie de vidéos qui présente plus d'ombres que la moyenne.



FIGURE 2.14 – *Night Videos* : Vidéos de nuit avec un contraste fort entre l'obscurité ambiante et les lumières artificielles de l'éclairage public et des phares de voitures.

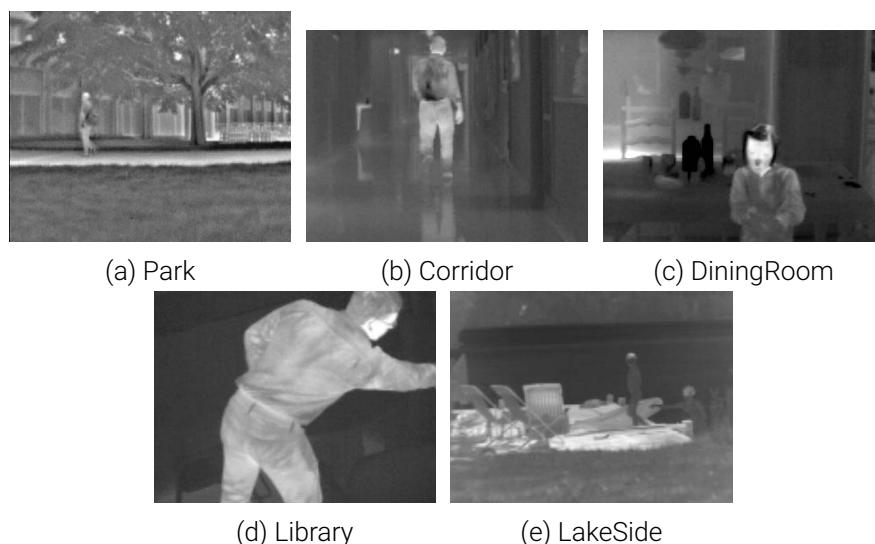


FIGURE 2.15 – *Thermal* : Ces vidéos ont été prises par une caméra infrarouge et sont en niveau de gris.

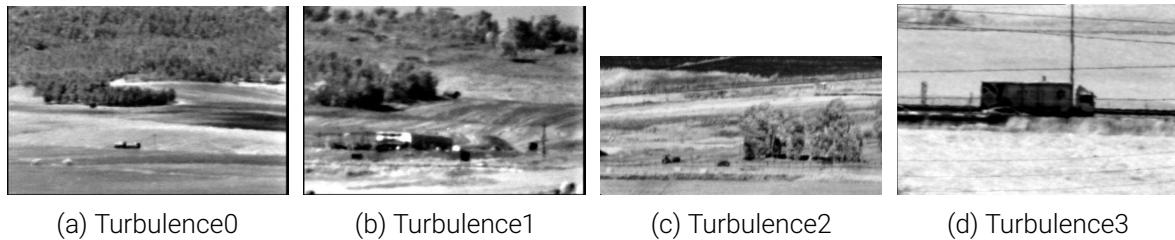


FIGURE 2.16 – *Turbulence* : Catégorie qui regroupe des vidéos provenant d'une même caméra infrarouge. Les captures ont été faites avec un objectif longue distance filmant des scènes éloignées de 5 à 15 km de l'objectif. Elle présente de nombreuses distorsions et turbulences atmosphériques dues à la chaleur et à la distance.



FIGURE 2.17 – *Intermittent Object Motion Reduced* : Cette catégorie comprend des scénarios particuliers dans lesquels le changement est intermittent (c'est à dire qu'un objet passe d'une situation statique à une situation en mouvement et inversement). Dans cette catégorie trois vidéos sur six on été conservées.



FIGURE 2.18 – *Low Framerate Reduced* : Cette catégorie regroupe des vidéos avec beaucoup de temps entre les images (entre 1 seconde et 6 secondes entre chaque image). Cela a pour but de pénaliser les approches à partir de flot optique, cependant notre approche n'est pas concernée. Trois vidéos sur les quatre ont été conservées. Seule une vidéo d'une marina a été retirée car la nouveauté (des bateaux) était trop similaire au fond, qui consiste en un grand nombre de bateaux similaires amarrés. Notre approche considérant la nouveauté au sens de l'inattendu. Un objet qui apparaît similaire à des objets déjà présents dans le fond ne rentre pas dans cette interprétation.

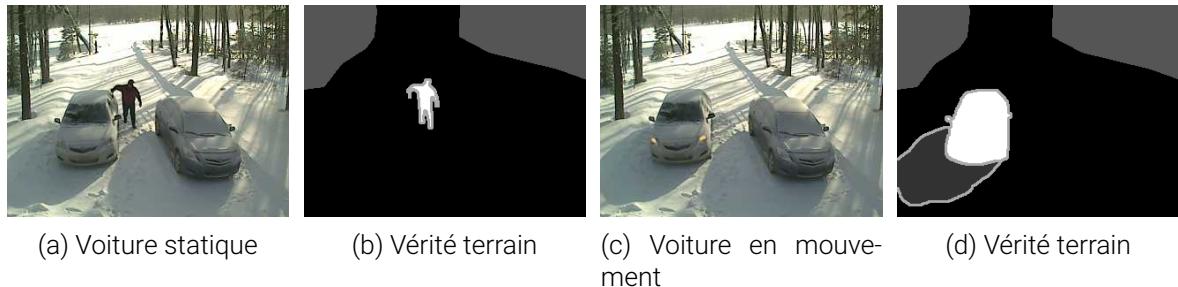


FIGURE 2.19 – Ces images sont extraites d'une vidéo de la catégorie *Intermittent Object Motion* et illustrent la différence entre détection de changement et détection de nouveauté. Pour le changement la voiture fait partie du fond pendant une partie de la vidéo car elle est statique. Elle devient objet à détecter à partir du moment où elle commence à se déplacer. Pour la nouveauté, une telle distinction n'est pas possible. Soit elle fait partie du fond, et dans ce cas, même en mouvement elle ne devrait pas être considérée comme nouveauté. Soit elle ne fait pas partie du fond, et dans ce cas elle sera tout le temps considérée comme nouveauté, même lors de la séquence statique.

## 2.4.2 Métriques utilisées

Nous présenterons dans cette section les différentes métriques que l'on a utilisées pour évaluer nos modèles. Elles peuvent être regroupées en deux grandes catégories, la première est proche des modèles évalués (la SOM et les GNG), et essaie de mesurer la qualité d'apprentissage de ceux-ci. La seconde est plus orientée vers la tâche de détection de nouveauté. Cette deuxième catégorie utilise des métriques définies par CDnet pour comparer les résultats avec d'autres modèles. Celles-ci étant en grand nombre, nous nous sommes limités aux trois plus pertinentes qui sont la précision, le rappel et la F-mesure.

### MQE : Erreur de Quantification Moyenne

L'erreur de quantification moyenne (Mean Quantization Error) mesure la qualité de l'apprentissage ou de la reconstruction d'un algorithme de quantification vectorielle. C'est la somme des différences entre tous les vecteurs et leurs représentants, divisée par la dimension et le nombre de vecteurs pour obtenir l'erreur moyenne des composantes.

$$\text{MQE} = \frac{1}{dn} \sum_{i=0}^{n-1} |\nu_i - u_i| \quad (2.1)$$

Avec  $n$  le nombre de vecteurs d'entrée,  $d$  la dimension de ces vecteurs,  $\nu_i$  le  $i^{\text{ème}}$  vecteur d'entrée et  $u_i$  le prototype du neurone représentant de ce vecteur d'entrée.

Il s'agit d'une mesure simple à calculer et à comprendre. Elle présente néanmoins une mauvaise pondération des différences en pénalisant les *outliers* de la même manière que des différences diffuses. Par exemple un pixel qui aura un changement maximal (qui passe de noir à blanc par exemple) aura le même impact sur l'erreur que cent pixels qui passent de 0 à 0,01. Pour des images, le premier changement sera visible pour notre œil, mais pas le second. Nous avons tout de même choisi d'utiliser cette

mesure dans nos expériences car elle représente le mieux la différence numérique des vecteurs à leurs représentants avant notre interprétation subjective de ces valeurs en images.

### PSNR : Peak Signal to Noise Ratio

Le PSNR est une mesure très présente dans le domaine de la compression d'images [HUYNH-THU et GHANBARI \[2008\]](#); [KORHONEN et You \[2012\]](#). Elle est similaire à la MQE, à la différence qu'il y a un carré à la place de la valeur absolue. D'où le nom de Mean Squared Quantization Error (MSQE) à calculer pour pouvoir obtenir le PSNR.

$$\text{MSQE} = \frac{1}{dn} \sum_{i=0}^{n-1} (v_i - u_i)^2 \quad (2.2)$$

$$\text{PSNR} = 10 \times \log_{10} \left( \frac{1}{\text{MSQE}} \right) \quad (2.3)$$

Avec  $n$  le nombre de vecteurs d'entrée,  $d$  la dimension de ces vecteurs,  $v_i$  le  $i^{\text{ème}}$  vecteur d'entrée et  $u_i$  le prototype du neurone représentant de ce vecteur d'entrée.

Le PSNR est inspiré du domaine du traitement du signal. L'idée est de mesurer le ratio de bruit introduit par la compression (Noise), avec l'intensité maximale du signal (Peak Signal), qui est la valeur maximum d'un pixel (1 dans notre cas). On peut noter que le PSNR fait passer d'un objectif de minimisation à une maximisation. L'ordre des valeurs reste inchangé par rapport au MSQE et le logarithme ajoute un effet de rendement décroissant. Si on a par exemple trois valeurs de MSQE  $m_1$ ,  $m_2$  et  $m_3$  avec  $m_1 < m_2 < m_3$ .  $m_1$  sera la meilleure (la plus petite) et  $m_3$  la moins bonne (la plus grande). Une fois converties en PSNR, les valeurs seront dans l'ordre suivant :  $p_1 > p_2 > p_3$ , avec  $p_1$  étant toujours la meilleure et  $p_3$  toujours la moins bonne, mais pour les raisons inverses cette fois : plus un PSNR est grand, mieux c'est. Un autre changement sera que si les intervalles étaient les même entre les trois valeurs, c'est à dire si  $m_2 - m_1 = m_3 - m_2$  alors on aura  $p_1 - p_2 > p_2 - p_3$ , à cause de l'inversion de MSQE.

Nous avons choisi d'utiliser le PSNR à la place de la MSQE car il est beaucoup plus facilement interprétable ; sa valeur typique étant comprise entre 0 et 100 dB. L'utilisation du carré dans la MSQE entraîne une plus grande pénalisation des *outliers* en comparaison à MQE. Ce n'est malgré tout pas une mesure complète de la qualité d'une image, car elle ne prend pas en compte certains paramètres comme le voisinage par exemple, qui sont importants dans la perception visuelle [1.1.3](#). Puisque l'on utilise la distance quadratique dans nos algorithmes de quantification vectorielle, le PSNR est une métrique intéressante car elle intègre ce que notre algorithme d'apprentissage tente de minimiser.

### Précision

La précision mesure la proportion de pixels correctement labellisés en tant que nouveauté parmi tous les pixels que le modèle a étiquetés comme nouveauté. La précision est comprise entre 0 et 1 et s'exprime souvent en pourcentage.

$$\text{Précision} = \frac{\text{Vrais Positifs}}{\text{Vrais Positifs} + \text{Faux Positifs}} \quad (2.4)$$

## Rappel

Le rappel mesure la proportion de pixels correctement labellisés en tant que nouveauté parmi tous les pixels avec de la nouveauté dans la vérité terrain. Le rappel est compris entre 0 et 1 et s'exprime souvent en pourcentage.

$$\text{Rappel} = \frac{\text{Vrais Positifs}}{\text{Vrais Positifs} + \text{Faux Négatifs}} \quad (2.5)$$

## F-mesure

La précision et le rappel sont deux mesures qui, prises séparément, peuvent être facilement maximisées. Pour avoir une très bonne précision, il suffit de n'inclure que les pixels positifs dont le modèle est sûr dans la carte de saillance, pour réduire la proportion de faux négatifs et ainsi améliorer la précision. Cela entraînera cependant un rappel faible, car le nombre total de vrais positifs diminue. Pour maximiser le rappel, il suffit de faire l'inverse, c'est à dire de catégoriser le plus possible de pixels en positifs dans la carte de saillance et ainsi réduire le nombre de faux négatifs. Cela se fait au détriment de la précision cependant, car le nombre de faux positifs sera en augmentation. La F-mesure [HRIPCSAK et ROTHSCHILD \[2005\]](#) tente d'être une solution à ce problème en combinant la précision et le rappel en un seul nombre à maximiser. Ce n'est cependant pas une mesure sans défauts [POWERS \[2011\]](#).

$$\text{F-mesure} = 2 \times \frac{\text{Précision} \times \text{Rappel}}{\text{Précision} + \text{Rappel}} \quad (2.6)$$

La formule de la F-mesure est assez simple, mais elle cache un comportement plus complexe. La précision et le rappel étant tous les deux entre 0 et 1, la F-mesure ne peut aussi prendre des valeurs que dans cet intervalle. Car lorsque les deux sont égaux à 1, la formule donne aussi 1. De par les propriétés de la multiplication entre deux nombres entre 0 et 1, la F-mesure favorise les précisions et rappels proches entre elles, et pénalise lorsque les deux valeurs sont éloignées. Ainsi, pour maximiser la F-mesure, augmenter la valeur la plus basse entre la précision et le rappel aura le plus d'effet.

Une propriété notable dans le calcul de la F-mesure est l'absence de distributivité. En pratique, cela implique que la moyenne des F-mesures n'est pas égal à la F-mesure des moyennes de précision et rappel. Cela peut poser problème lorsque l'on essaye d'agréger des valeurs sur plusieurs images par exemple. Il est donc nécessaire de calculer les F-mesures pour chaque image séparément pour ensuite en faire la moyenne. On peut également observer cette propriété dans les résultats présentés dans la section [2.5.3](#), où la valeur de la F-mesure ne suit pas la formule lorsqu'elle est appliquée aux moyennes des précisions et rappels, car nous avons bien calculé la moyenne des F-mesures.

### L'œil humain

Il n'existe pas de mesure objective pour déterminer la qualité de l'apprentissage d'une quantification vectorielle ou de détection de nouveauté, il n'y a que des estimations. Pour la quantification vectorielle par exemple, le calcul de l'erreur semble naturel, mais il ne capture pas toutes les spécificités des modèles. Notamment lorsque l'on travaille avec des images, le problème déjà évoqué est que certaines erreurs sont visuellement plus perceptibles que d'autres, alors qu'elles ont la même valeur numérique. Mais il y a aussi des propriétés de certains modèles de VQ qui ne peuvent être simplement quantifiées, comme par exemple la gestion des *outliers*. Un algorithme peut par exemple préférer représenter le mieux possible la majorité de la base de données en laissant de côté les *outliers*, ou au contraire faire en sorte que toutes les données soient bien représentées, mais en sacrifiant une meilleure précision sur les données les plus nombreuses.

Le même problème de subjectivité se présente pour la détection de nouveauté, les mesures quantitatives que nous avons sélectionnées ne représentent pas forcément l'aspect qualitatif de la détection de nouveauté. Par exemple si on a un algorithme qui a pour résultat le contour des objets nouveaux dans une image, la tâche sera bien remplie, mais le score sur les métriques sera mauvais car il sera attendu de l'algorithme qu'il marque de façon positive tous les pixels de l'objet nouveau et non pas seulement le contour. Ce genre de problème peut parfois être résolu par l'utilisation de post processing, le remplissage à partir des contours ici par exemple. Mais ce n'est pas toujours le cas.

Il est donc nécessaire d'adoindre à ces métriques objectives, une mesure qualitative du comportement des modèles évalués. Pour les images, nous avons la chance d'être naturellement dotés de très bons capteurs et d'un réseau neuronal biologique performant très entraîné sur des données visuelles. Par conséquent nos interprétations s'appuieront sur le résultat visuel de nos modèles en complément des métriques.

### 2.4.3 Préparation des données

#### Image de fond

Notre modèle a besoin pour l'apprentissage d'un fond sans nouveauté pour fonctionner. Il nous faut donc une image du fond pour chaque séquence vidéo. Il arrive cependant que des vidéos présentent pendant toute la séquence de la nouveauté, et qu'une image sans perturbation n'existe pas dans la séquence. Pour contrer ce problème, nous avons calculé une image médiane de la séquence pour obtenir une image de fond sans perturbations.

Pour une mise en pratique potentielle, nous supposons qu'il serait facile d'avoir une image du fond sans perturbations, ou qu'il serait aisément d'appliquer la même méthode en effectuant une capture vidéo et en calculant la médiane lors de l'initialisation.

#### Échantillonage de l'évaluation

Le calcul des métriques sur les séquences de CDnet se font sur de nombreuses images. Pour la catégorie *baseline* par exemple, il y a en moyenne 1100 images par sé-

quence. Chaque image nécessitant quelques secondes de temps de calculs, utiliser l'intégralité de chaque séquence serait beaucoup trop long pour pouvoir étudier en détail les nombreux paramètres de notre modèle. En partant du principe que des images proches dans le temps sont aussi généralement similaires dans leur contenu, il serait possible d'évaluer une séquence en n'en évaluant qu'un sous-échantillon afin d'obtenir des mesures représentatives du résultat complet. Les images d'une séquence étant indépendantes temporellement les unes des autres dans notre modèle, nous avons choisi de prendre un sous-échantillonnage régulier de la séquence.

Nous avons mené une étude pour déterminer la taille de sous-échantillonnage optimal, que nous présentons dans la figure 2.20. Nous avons choisi d'évaluer 105 images par séquences. Cela représente une évaluation entre 7 et 14 fois plus rapide que si on évaluait l'intégralité de la séquence, la vitesse dépendant du nombre d'images total de la séquence complète. La précision des métriques en est impactée, mais la variation ne dépasse pas 0.5% du résultat.

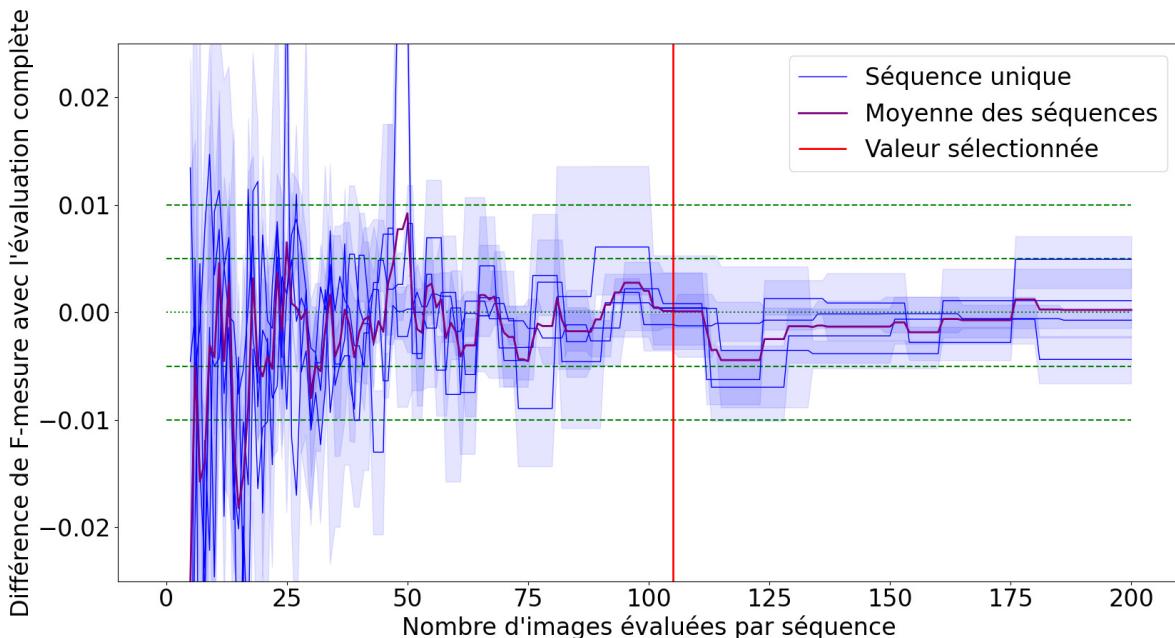


FIGURE 2.20 – Variation de la F-mesure en fonction du nombre d'images analysées. La variation atteint un plateau à partir de 105, qui a été choisi car c'est le plus petit échantillon proche de la moyenne et ne dépassant quasiment pas les 0.5% de différence. Les sections droites des lignes, surtout vers des grands échantillons, sont à cause d'arrondis sur la taille du pas entier utilisé. Ainsi les échantillons ne sont pas de la taille exacte comme l'indique l'abscisse, mais au moins de la taille indiquée, souvent plus grands. La variabilité qui augmente après 105 n'est pas nécessairement un problème, car l'on ne mesure pas un procédé stochastique, mais la représentativité des images sélectionnées. Ce qui veut dire que la variation due à l'échantillonnage devrait en normalement rester similaire même avec des paramètres différents pour la SOM.

#### 2.4.4 Paramétrages des modèles

Notre modèle comporte de nombreux paramètres pour lesquels l'impact sur les performances n'est pas trivial. C'est à dire que changer un paramètre dans un sens pourrait amener à une augmentation des performances à une certaine valeur, et à

une diminution des performances à une autre valeur. Il existe également des interdépendances entre les paramètres, ce qui signifie que le paramètre  $a$  optimal ne sera pas le même pour deux valeurs du paramètre  $b$  par exemple.

En général, dans ces cas de figure, on effectue une optimisation globale de tous les paramètres en même temps, pour prendre en compte l'interaction entre les paramètres. Cependant, dans notre cas, l'espace de recherche serait très grand (8 dimensions pour 8 paramètres) et nécessiterait un très grand nombre d'exécutions pour le couvrir entièrement. Chacune de nos exécutions durera en moyenne quelques minutes, il n'est pas souhaitable d'en effectuer un trop grand nombre. De plus, il est difficile avec ce genre d'approche de la paramétrisation de comprendre l'impact de chaque paramètre sur le résultat.

Nous avons ainsi choisi de faire une étude paramétrique en séparant les paramètres le plus possible. Cela nous permet d'analyser chaque paramètre, ou groupe de paramètres en détail pour mieux comprendre leur effet sur le comportement de notre modèle. Cela nous amènera également à pouvoir prédire un comportement dans des espaces paramétriques que nous n'avons pas explorés; vers quelle valeur les performances convergent-elles si on fait tendre un paramètre vers l'infini par exemple. Cette étude pourra être faite avec un nombre raisonnable d'exécutions avec les moyens matériels que nous avons à disposition. Le défaut sera que l'interaction entre certains paramètres sera difficile à évaluer. Nous optimiserons nos paramètres pour maximiser la  $F$ -mesure, car c'est la métrique la plus proche de la tâche de détection de nouveauté. La section se conclura par un tableau récapitulatif des paramètres que nous aurons utilisés pour générer nos résultats.

### Variation aléatoire

Les SOM que nous utilisons ne sont pas déterministes, et plusieurs facteurs aléatoires peuvent influencer le résultat d'un apprentissage. Ces deux facteurs sont l'initialisation des poids des neurones, que nous faisons débuter à des valeurs aléatoires entre 0 et 1 pour chaque composante. Mais aussi l'ordre de présentation de la base d'apprentissage qui est aléatoire, et donc varie avec la graine du générateur d'aléatoire paramétrée avant l'apprentissage. Cette dépendance à l'aléatoire implique que les métriques que l'on aura calculées peuvent varier d'un apprentissage à l'autre. Nous avons étudié cette variabilité pour connaître quel serait le plus petit nombre d'exécutions nécessaire pour donner une estimation fiable de la moyenne des résultats pour un ensemble de paramètres.

D'après la figure 2.21, le comportement aléatoire de notre modèle peut-être assimilé à une loi normale. Cela nous permet, en utilisant la règle empirique, de donner un intervalle de confiance lorsque l'on estimera la moyenne pour nos mesures. Nous supposerons que les variances de la *baseline* sur laquelle nous avons mesuré ces valeurs est du même ordre que sur toutes les vidéos du jeu de données CDNET. Nous supposerons aussi que toutes les distributions suivent une loi normale.

D'après le tableau 2.1, nous pouvons observer que l'écart type est très variable en fonction de la vidéo que l'on traite. Chaque exécution pouvant durer plusieurs minutes, il est aussi difficilement concevable de faire nos optimisations en visant un résultat à  $\delta = 0.5 / 3\sigma$ , car cela nécessiterait dans certains cas presque 100 exécutions pour

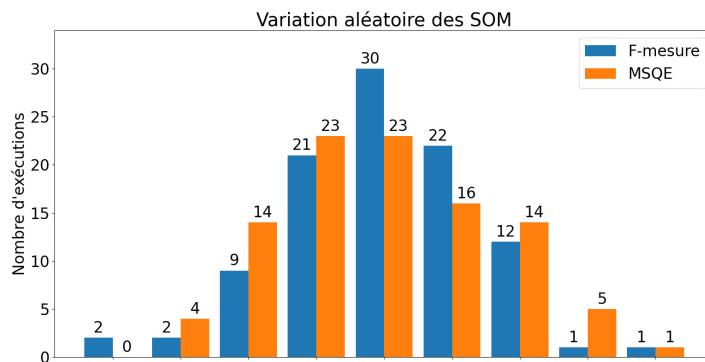


FIGURE 2.21 – Distribution de la F'mesure et du PSNR pour un ensemble de paramètres donnés pour une vidéo. On a découpé l'intervalle de résultats en 9 sections égales. La section numéro 5 a la moyenne en son centre. La largeur de chaque région a été ajustée pour que le maximum soit à la limite haute de la section 9 ou le minimum à la limite basse de la section 1, en choisissant celui qui donnerait les plus grandes sections. L'axe des ordonnées quant à lui donne le nombre d'exécutions incluses dans chaque catégorie, sur 100 exécutions au total. Nous pouvons observer que les distributions suivent une loi normale. Il semblerait que la variabilité de la F-mesure est inférieure à celle de la MSQE.

Video	Écart-type $\sigma$	$\delta = 1\% / 2\sigma$	$\delta = 1\% / 3\sigma$	$\delta = 0.5\% / 2\sigma$	$\delta = 0.5\% / 3\sigma$
highway	$5.68 \times 10^{-3}$	1.3	2.9	5.2	11.6
office	$9.9 \times 10^{-3}$	4.0	8.9	15.8	35.6
pedestrians	$1.52 \times 10^{-2}$	9.2	20.8	36.9	83.1
PETS2006	$1.08 \times 10^{-2}$	4.7	10.5	18.6	41.9

TABLEAU 2.1 – Nombre d'exécutions avec graines aléatoires différentes requises pour que la moyenne des F-mesures de l'échantillon soit au moins d'une distance  $\delta$  de la vraie moyenne, avec une probabilité de 95% pour  $2\sigma$  et 99,7% pour  $3\sigma$ . L'écart type à partir duquel on déduit ces valeurs, a été calculé sur un échantillon de 100 exécutions pour *highway*, et 50 échantillons pour les autres.

chaque ensemble de paramètres. Nous avons ainsi choisi de nous limiter à 8 exécutions avec des graines aléatoires différentes, car les ordinateurs sur lesquels nous expérimentons possèdent 8 coeurs, et que cela nous permet de dépasser le premier seuil de  $\delta = 1\% / 2\sigma$  pour la plupart des vidéos.

### Optimisation d' $\alpha$ et de $\sigma$

Nous avons présenté dans la section 1.2.2 ces deux paramètres et leurs effets. Ces paramètres sont très dépendants entre eux, car ils pondèrent tous les deux la formule de modification des poids de l'apprentissage. Nous les avons donc optimisés ensemble. La recherche a été faite par un *Tree-structured Parzen Estimator* BERGSTRA et collab. [2011], sur des vidéos de la *baseline*. Nous avons fait plus de 1000 mesures par vidéo, avec pour chaque mesure, la moyenne de 8 exécutions, deux pour chaque vidéo, pour réduire le bruit statistique. La figure 2.22 explique le choix de  $\sigma$  que nous avons fait commencer à 0.7 et finir à 0.015. Puis nous avons relancé une optimisation pour les valeurs d'alpha uniquement pour mieux observer les tendances avec les valeurs fixées de  $\sigma$ . La figure 2.23 présente ces résultats. Nous avons choisi pour  $\alpha$  un départ à 0.2 et de finir à 0.05.

Nous pouvons aussi noter l'ordre d'importance des paramètres par leur impact sur le résultat : *Sigma end > Sigma start > Alpha end > Alpha start*.

### Durée de l'apprentissage

Le dernier paramètre des SOM qu'il reste à optimiser est le nombre d'époques. Une époque étant un parcours complet de la base d'apprentissage, ce paramètre doit être assez élevé pour que la SOM converge et assez bas pour que l'apprentissage ne soit pas trop long, la durée de celui-ci étant directement proportionnelle au nombre d'époques. Le résultat de nos tests est montré figure 2.24. Nous avons choisi 120 époques d'apprentissage pour les SOM.

### Récapitulatif

TABLEAU 2.2 – Récapitulatif des paramètres SOM

$\alpha$ -start	$\alpha$ -end	$\sigma$ -start	$\sigma$ -end	Époques	Images par séquence
0.2	0.05	0.7	0.015	120	105

### Paramètres des GNG

Nous avons choisi d'utiliser des GNG pour montrer que notre méthode est généralisable à d'autres modèles avec quantification vectorielle et topologie. Ainsi, nous nous sommes contentés pour la paramétrisation d'utiliser de nouveau un *Tree-structured Parzen Estimator* pour optimiser l'intégralité des paramètres en une fois. Leurs valeurs sont présentées dans le tableau 2.3.

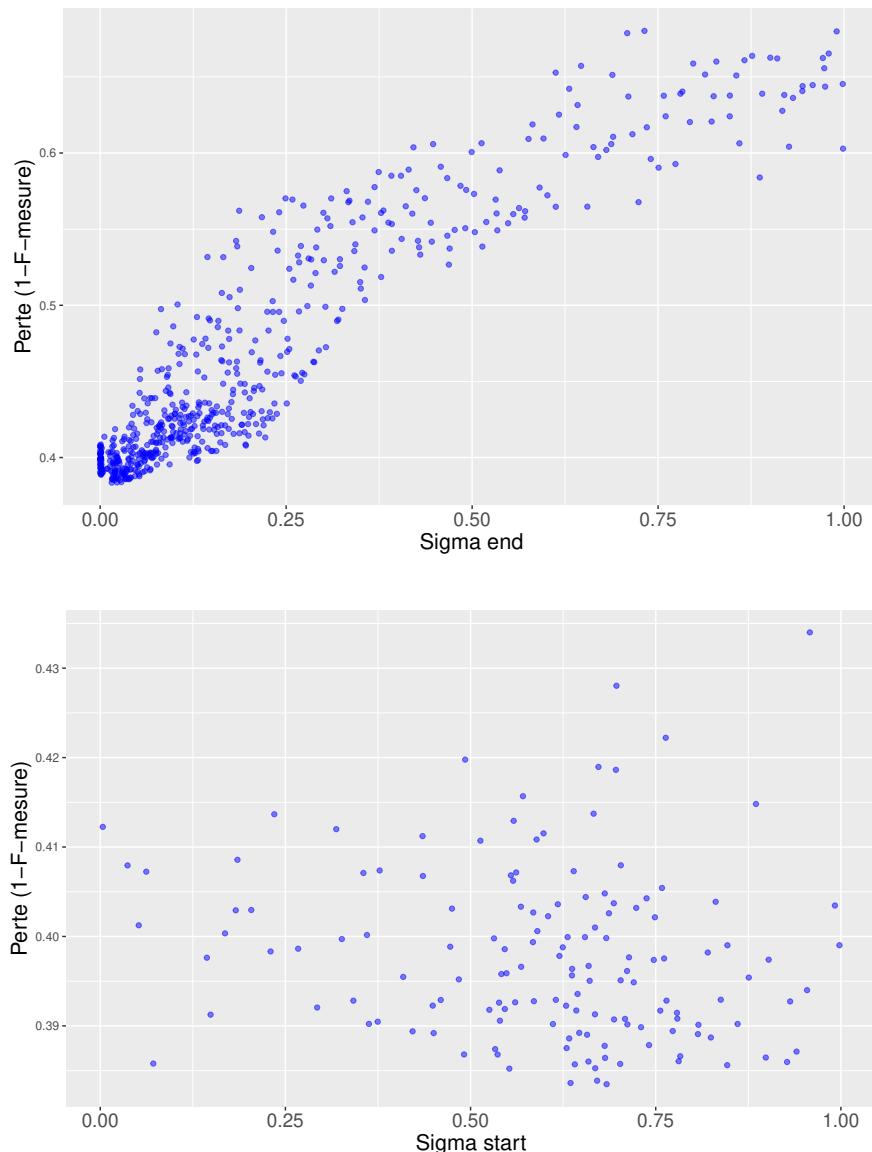


FIGURE 2.22 – Le premier paramètre qui a une valeur optimale évidente est *Sigma end*, où une petite valeur proche de zéro semble idéale. Cela s’explique facilement du fait qu’une valeur faible de *Sigma end* amène à ce que lors des dernières époques la SOM va se focaliser sur l’optimisation de la quantification vectorielle au dépend de la topologie, pour obtenir des neurones les plus proches possibles des imagettes qu’ils représentent. Nous l’avons choisi à 0.015.

Le second graphique représente un sous échantillon d’expériences sélectionnés avec une valeur de *Sigma end* inférieure à 0.05. En affichant les résultats en fonction de *Sigma start*, on observe une légère préférence pour des valeurs de *Sigma start* assez élevées, c’est à dire dans les alentours de 0.7.

TABLEAU 2.3 – Paramètres des GNG

$\epsilon_{bmu}$	$\epsilon_j$	$a_{max}$	$\alpha$	$d$ (erreur globale)	Époques
0.4	0.01	85	0.6	0.3	110

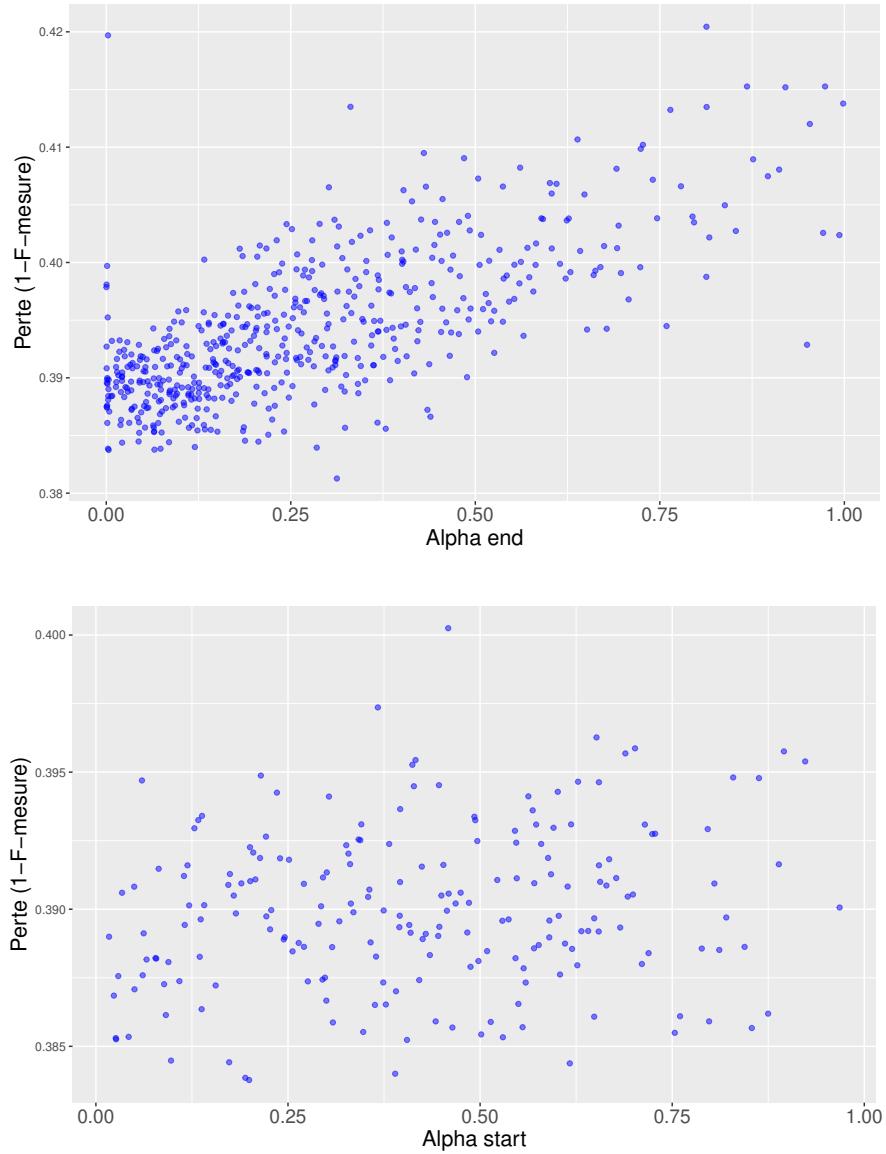


FIGURE 2.23 –  $\text{Alpha end}$  semble fonctionner avec une préférence similaire à  $\text{Sigma end}$ . Une faible valeur d' $\text{Alpha end}$  non nulle semble optimale. En pratique cela se traduit par un ajustement fin des poids des neurones à la fin de l'apprentissage. Nous l'avons mis à 0.05

Pour  $\text{Alpha start}$  nous avons à nouveau réduit les expériences à un échantillon dont la valeur d' $\text{Alpha end}$  est inférieure à 0.2. Il ne semble pas y avoir de préférence forte pour  $\text{Alpha start}$ , nous prendrons donc le minimum à 0.2.

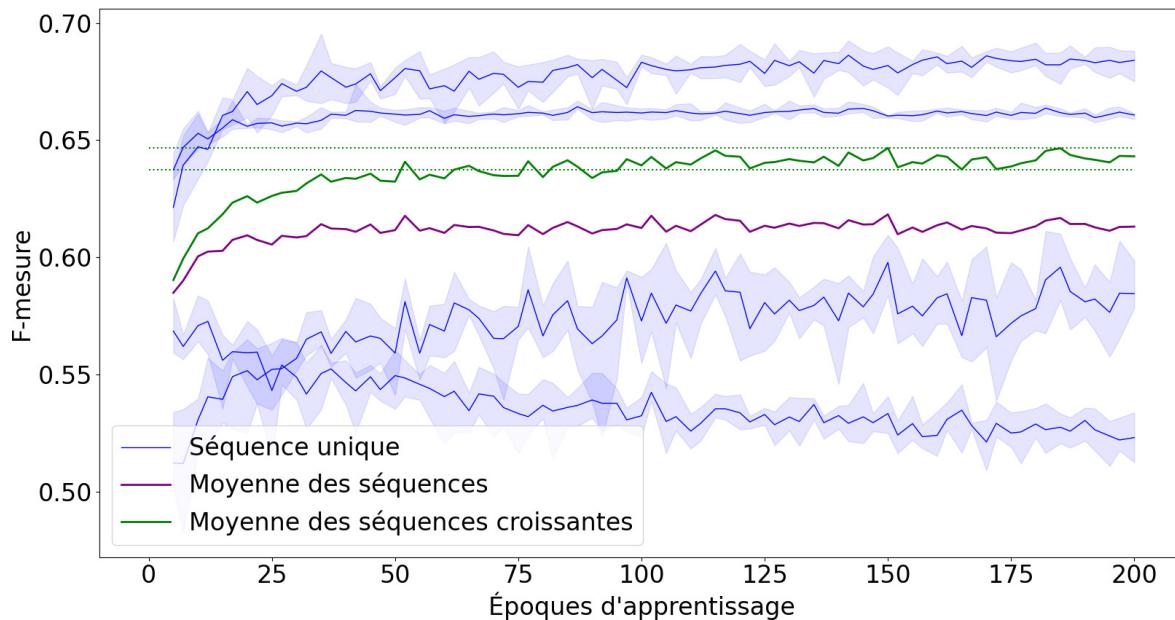


FIGURE 2.24 – F-mesure en fonction du nombre d'époques sur la *baseline*. Une chose étrange que l'on peut remarquer est qu'une des séquences (PETS2006) a des performances décroissantes avec l'apprentissage. Cela pourrait s'expliquer par un seuil trop haut pour cette séquence, car plus l'apprentissage avance, plus les erreurs sont petites. De plus petites erreurs auront plus de chances d'être en dessous du seuil de décision, et ainsi diminuer le rappel, qui diminue ensuite aussi la F-mesure. Nous avons utilisé la moyenne des séquences croissantes uniquement pour déterminer le moment de convergence de l'apprentissage. Après environ 100 époques, les performances semblent stables et restent dans un intervalle restreint de f-mesure. Nous avons donc choisi 120 époques d'apprentissage pour que la convergence soit atteinte sans trop coûter en calculs.

## 2.5 Résultats expérimentaux

Nous présentons dans cette section tous les résultats quantitatifs et visuels que nous avons obtenus avec notre méthode sur CDnet. Cette section est découpée en quatre parties. La première présente l'impact du seuil, du nombre de neurones et de la taille des imagettes sur les performances et les différences entre les vidéos lorsque l'on fait varier ces paramètres. La seconde présente les résultats complets sur CDNET et les comparera avec l'état de l'art. La troisième présente visuellement les images que produit notre modèle. La dernière est consacrée aux discussions et perspectives.

Le seuil, la taille des imagettes et le nombre de neurones auraient pu faire partie de la section des paramètres. Cependant compte tenu de l'importance qu'ils ont sur le résultat et des informations et perspectives qu'ils nous donnent, nous les avons inclus dans la partie résultats.

Cette section est basée sur un ensemble d'exécutions sur l'intégralité de CDnet. Les SOM sont exécutées les paramètres définis dans la section précédente [2.4.4](#). Au total, cette expérience contient 2520 exécutions pour la SOM, et le même nombre pour les GNG.

### 2.5.1 Seuil de décision

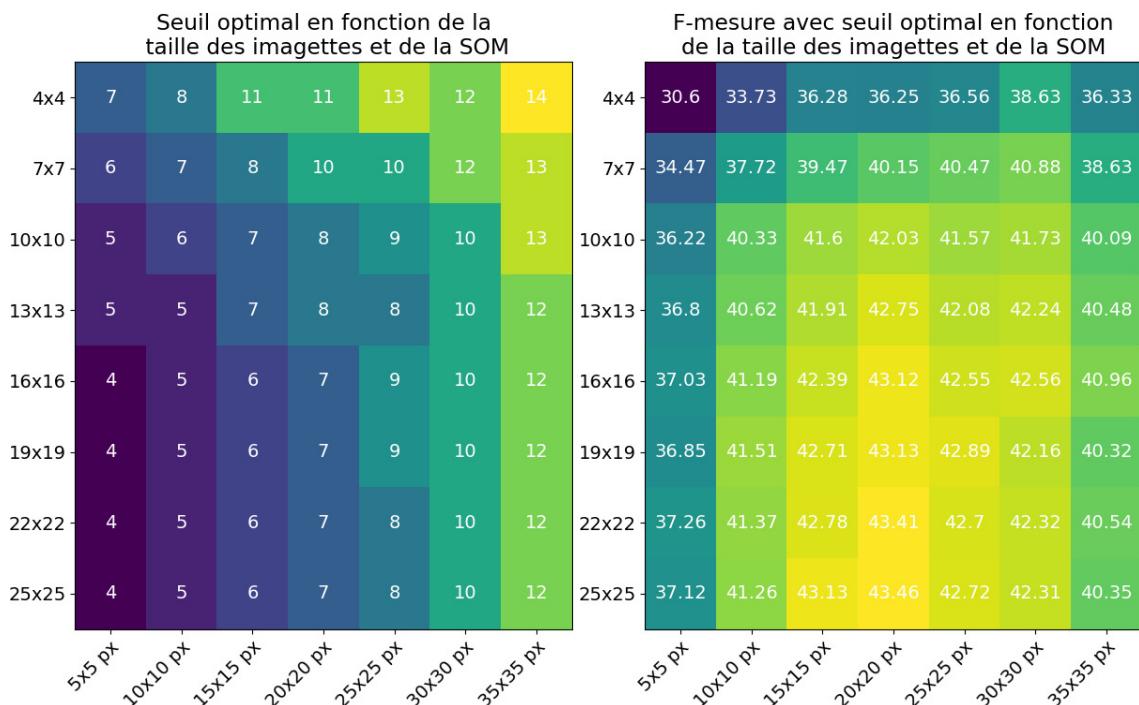


FIGURE 2.25 – La figure de gauche indique le seuil optimal pour chaque couple de taille de SOM et de taille d'imagettes que l'on a testé. On peut remarquer que le seuil maximisant la F-mesure diminue plus la taille de la carte est grande, car il y a moins d'erreurs, et qu'il augmente avec la taille des imagettes, car cela augmente les pertes dues à la compression.

La figure de droite montre la F-mesure complète (sur l'ensemble de CDnet) pour ce seuil optimal. Le maximum est obtenu pour 25 × 25 neurones, 20 × 20 pixels et un seuil de 7.

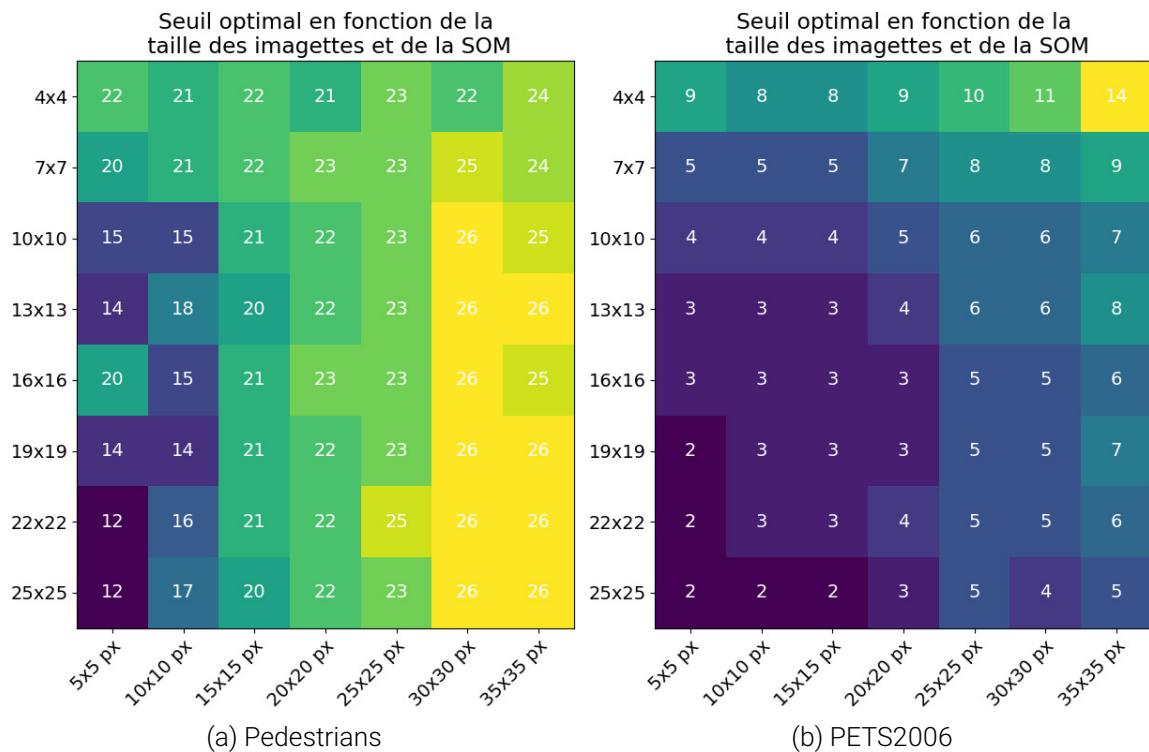


FIGURE 2.26 – Les deux figures présentent les seuils optimaux pour deux séquences différentes, qui montrent la forte variabilité de ce paramètre.

Le seuil de décision est ce qui permet de passer de notre carte de saillance qui contient des valeurs entre 0 et 255 à la carte de détection binaire. Toutes les valeurs en dessous ou égales au seuil seront considérées comme fond, et les valeurs au dessus du seuil comme nouveauté. La figure 2.25 présente nos résultats pour le seuil global, et la figure 2.26 montre que l'optimal est très variable en fonction de la séquence choisie.

## 2.5.2 Nombre de neurones et taille des imagettes

### Baseline

La figure 2.27 montre une différence significative de comportement entre les séquences vidéo. La taille des imagettes semble être le facteur de variations le plus important. Pour *Pedestrians* par exemple, une taille d'imagette petite de  $10 \times 10$  donnera les meilleures performances, alors que pour *PETS2006*, ce sont les grosses imagettes de  $25 \times 25$  qui sont optimales. La variabilité est aussi très différente en fonction de la séquence considérée. *Highway* et *Office* sont relativement stables et peu sensibles aux changements de taille d'imagettes et de neurones, tant que ceux-ci ne sont pas trop petits. Au contraire de *PETS2006* pour qui les performances peuvent aller du simple au double en fonction de la taille choisie.

Pour le nombre de neurones, l'optimal semble toujours être du côté où les cartes sont les plus grandes. Cependant les gains apportés par une carte plus grande sont très variables. Généralement importants au début pour une carte petite, les valeurs atteignent rapidement un plateau et n'améliorent que peu la détection de nouveauté. C'est une information importante pour une détection rapide et efficace, car le nombre de neurones dans la carte est un facteur important du coût en calcul de la SOM.

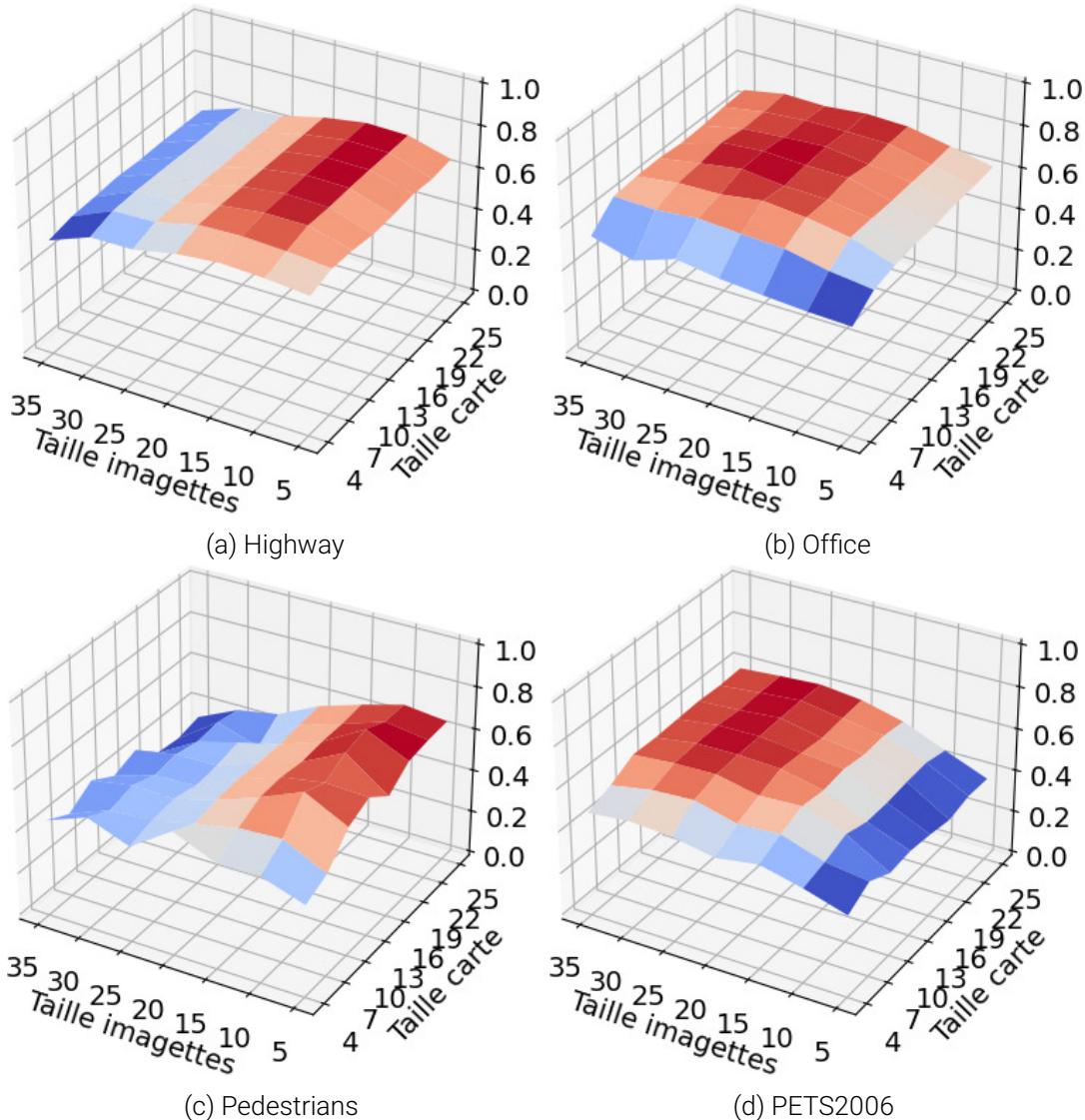


FIGURE 2.27 – F-mesure en fonction du nombre de neurones et de la taille des imagettes pour les séquences de la *baseline* avec une SOM.

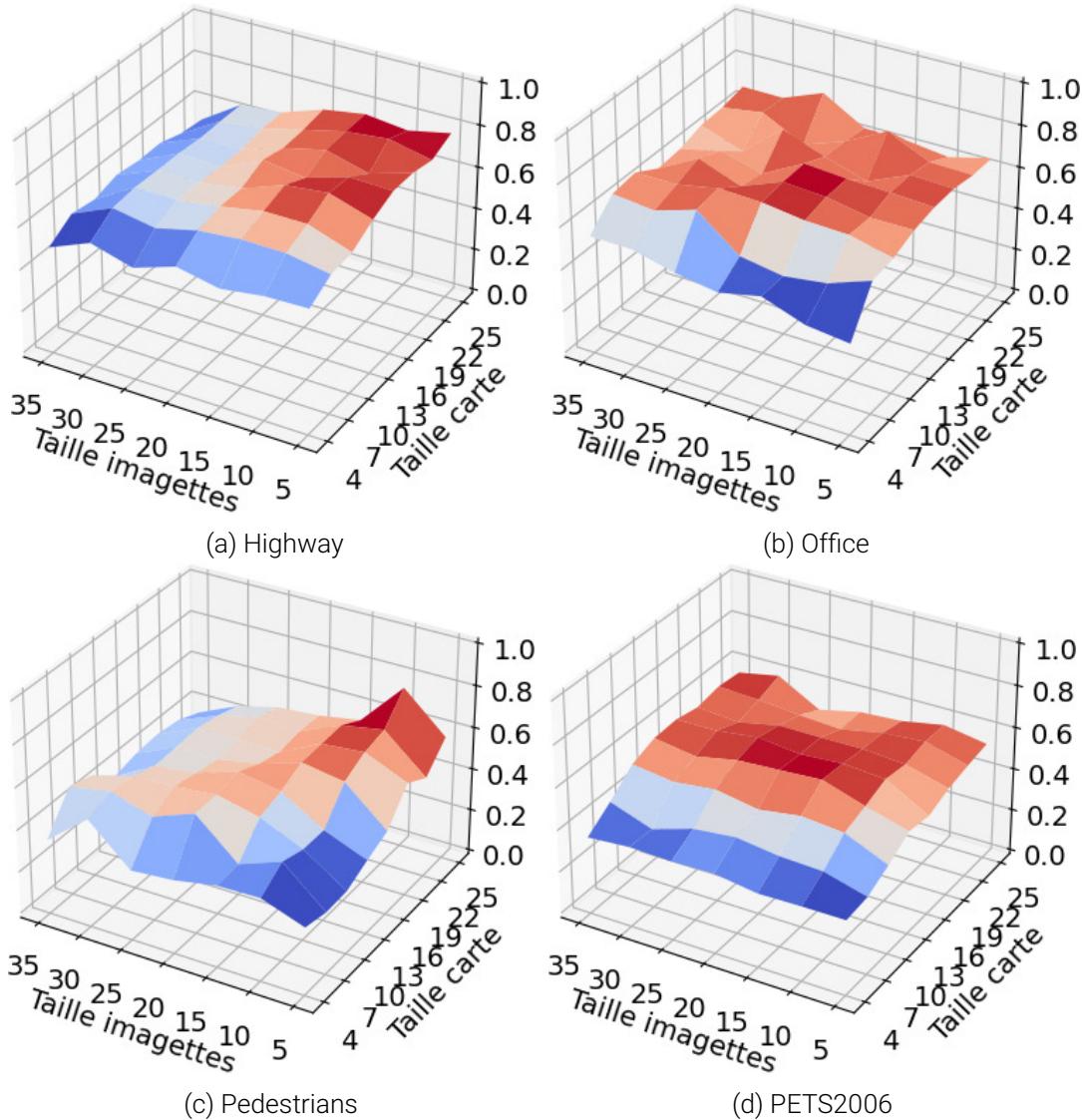


FIGURE 2.28 – F-mesure en fonction du nombre de neurones et de la taille des imagettes pour les séquences de la *baseline* avec un GNG. Les GNG n'ayant pas une topologie carrée comme les SOM, il suffit de mettre au carré la taille de la carte pour obtenir le nombre de neurones utilisés par le GNG.

Les résultats pour les GNG de la figure 2.28 montrent un comportement plus chaotique. Les variations sont particulièrement grandes entre plusieurs exécutions, mais on peut tout de même discerner les mêmes optimums de tailles d'imagettes que pour les SOM. *Pedestrians* préfère encore des imagettes petites (10 par 10), et *PETS2006* des imagettes grandes (25 par 25). Cela semble indiquer que ces optimums ne sont pas dépendantes des modèles (SOM, GNG) mais bien seulement de la vidéo ou de la nouveauté présente dans la vidéo.

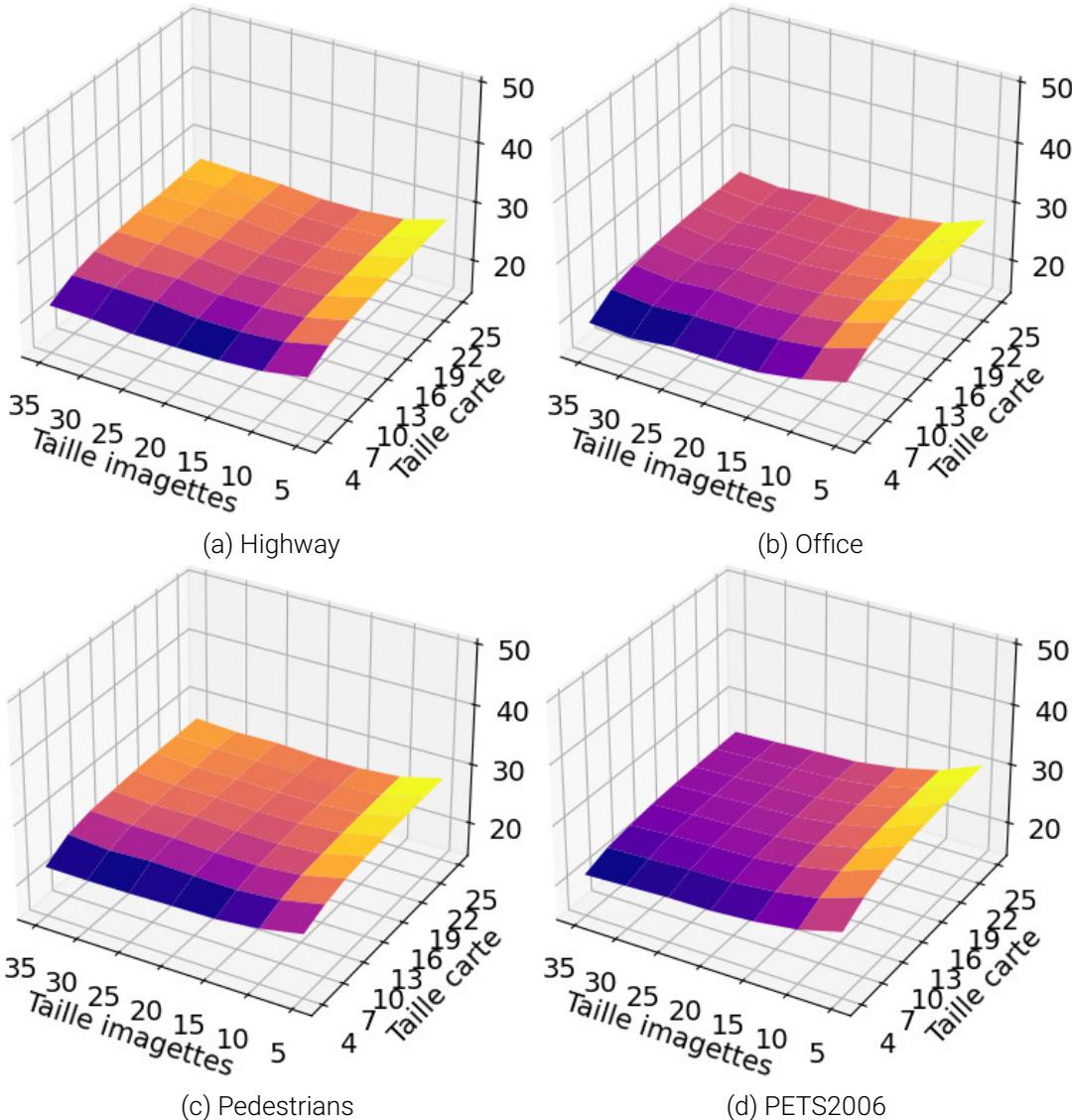


FIGURE 2.29 – PSNR en fonction du nombre de neurones et de la taille des imagettes pour les séquences de la baseline avec une SOM.

La variation du PSNR est montrée dans la figure 2.29. On peut y observer que les optimums pour la compression sont les mêmes pour toutes les vidéos, et donc différentes d'optimums pour les F-mesures. La compression donne le meilleur résultat pour des petites imagettes de  $5 \times 5$  et s'améliore aussi légèrement avec le nombre de neurones. La conclusion est que la qualité de compression et la détection de nouveauté ne sont pas corrélées. Améliorer le résultat de la compression n'amènera ainsi pas forcément des gains au niveau de la détection de nouveauté.

Ce résultat peut s'expliquer par le fait que les petites imagettes reconstruisent bien le fond, mais aussi la nouveauté. Une imagette de 20 par 20 pixels avec de la nouveauté aura par exemple une distance plus grande avec un seul neurone et son vecteur prototype de 20 par 20 pixels que de quatre neurones de 5 par 5 pixels, chacun d'entre eux étant la BMU de leur portion de l'imagette. Ainsi, la saillance de la nouveauté est réduite en même temps que l'erreur de reconstruction avec des imagettes plus petites, ce qui peut amener à une F-mesure réduite.

### Global

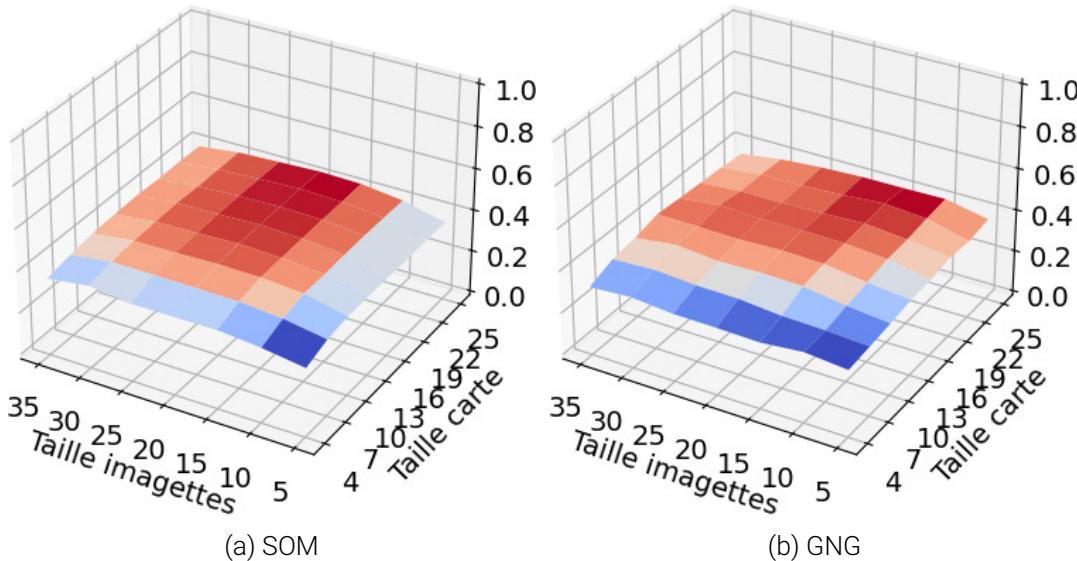


FIGURE 2.30 – Moyenne des F-mesure en fonction du nombre de neurones et de la taille des imagettes pour l'ensemble des vidéos de notre jeu de données.

Lorsque sont agrégés toutes les séquences vidéos, les graphes de la figure 2.30 montrent les tendances globales sur notre modèle. La détection de nouveauté avec la SOM fonctionne le mieux avec des tailles d'imagettes proches de  $20 \times 20$ . Augmenter le nombre de neurones est quant à lui toujours bénéfique, mais amène des gains décroissants. Les résultats pour les GNG sont plus étranges, car ils sont à leur maximum pour des valeurs intermédiaires de neurones et d'imagettes, plus petites que celles de la SOM. Il s'agit peut être uniquement d'un résultat de la forte variabilité observée dans les vidéos individuelles qui se répercute dans la moyenne globale.

### 2.5.3 Résultats complets

Les résultats complets sont présentés dans les tableaux 2.4 et 2.5. Les versions *Global* indiquent que l'on a choisi les valeurs de seuil, de taille de carte et d'imagettes qui étaient optimales sur l'ensemble de CDNET, c'est à dire  $25 \times 25$  pour les tailles de carte et  $20 \times 20$  d'imagettes et un seuil à 7 pour la SOM. Des imagettes de  $15 \times 15$  pixels,  $20 \times 20$  neurones et un seuil de 5 pour les GNG. Les versions *Max* sont celles où on a pris la valeur optimale pour chaque vidéo.

La première chose que l'on peut remarquer, c'est que pour 8 catégories sur 10, le rappel est toujours meilleur que la précision. Pour seulement 13 vidéos sur 45, la

	SOM			SOM Max	GNG	GNG Max
<b>Categories/vidéos</b>	Précision	Rappel	F-mesure	F-mesure	F-mesure	F-mesure
<b>Baseline</b>	59.2%	72.9%	63.9%	73.4%	<b>67.0%</b>	<b>76.4%</b>
Highway	63.9%	75.7%	69.3%	78.9%	75.9%	83.2%
Office	71.2%	71.0%	71.1%	74.4%	75.9%	77.8%
Pedestrians	41.4%	87.0%	56.1%	76.7%	60.0%	78.4%
PETS2006	60.1%	57.9%	59.0%	63.7%	56.2%	66.4%
<b>Bad weather</b>	32.6%	57.8%	<b>38.9%</b>	48.0%	34.8%	<b>54.5%</b>
Blizzard	19.4%	48.9%	27.8%	37.3%	14.4%	47.8%
Skating	72.5%	68.3%	70.3%	72.7%	74.3%	76.1%
SnowFall	17.1%	57.6%	26.4%	37.9%	24.9%	45.9%
WetSnow	21.5%	56.4%	31.1%	44.2%	25.7%	48.1%
<b>Camera jitters</b>	37.6%	65.2%	<b>46.5%</b>	51.7%	36.6%	<b>52.3%</b>
Badminton	49.2%	81.3%	61.3%	68.6%	62.1%	68.9%
Boulevard	47.9%	53.1%	50.4%	51.8%	34.7%	51.6%
Sidewalk	14.7%	45.4%	22.2%	27.4%	14.5%	30.9%
Traffic	38.5%	80.9%	52.1%	59.0%	35.1%	57.9%
<b>Dynamic bkg</b>	8.3%	67.1%	<b>14.1%</b>	30.1%	12.4%	<b>40.0%</b>
Boats	2.7%	79.8%	5.2%	34.2%	4.2%	62.8%
Canoe	16.7%	92.4%	28.2%	56.3%	26.0%	74.2%
Fall	8.8%	54.1%	15.2%	21.2%	8.5%	18.0%
Fountain01	1.1%	43.5%	2.1%	3.1%	1.9%	9.8%
Fountain02	3.1%	55.3%	5.9%	21.2%	5.0%	19.5%
Overpass	17.2%	77.8%	28.2%	44.6%	28.7%	55.7%
<b>Night videos</b>	27.8%	47.1%	<b>32.1%</b>	<b>38.8%</b>	32.0%	37.1%
BridgeEntry	10.8%	33.9%	16.4%	26.3%	12.9%	21.5%
BusyBoulvard	40.5%	28.9%	33.8%	39.5%	31.5%	33.1%
FluidHighway	19.1%	55.7%	28.5%	31.2%	29.6%	30.6%
StreetCornerAtNight	21.8%	59.7%	31.9%	39.7%	29.6%	35.7%
TramStation	47.9%	41.1%	44.2%	56.4%	49.5%	62.1%
WinterStreet	26.7%	63.4%	37.6%	39.8%	38.6%	39.7%
<b>Shadow</b>	48.6%	64.2%	<b>52.8%</b>	59.8%	49.7%	<b>62.0%</b>
Backdoor	27.1%	76.2%	40.0%	49.4%	36.6%	48.7%
Bungalows	54.9%	49.8%	52.2%	72.1%	63.0%	72.2%
BusStation	64.6%	67.3%	66.0%	66.1%	54.4%	66.3%
CopyMachine	77.7%	67.1%	72.0%	79.2%	58.9%	78.0%
Cubicle	17.0%	42.9%	24.3%	26.9%	19.5%	39.5%
PeopleInShade	50.0%	81.7%	62.1%	65.0%	65.7%	67.3%
<b>Thermal</b>	52.0%	66.2%	<b>54.0%</b>	65.5%	51.4%	<b>66.5%</b>
Corridor	32.4%	87.3%	47.2%	75.4%	26.4%	72.8%
DiningRoom	67.4%	55.7%	61.0%	70.8%	65.6%	72.5%
LakeSide	13.8%	53.0%	21.9%	33.4%	14.7%	30.4%
Library	85.9%	88.4%	87.1%	91.5%	87.8%	94.2%
Park	60.7%	46.4%	52.6%	56.4%	62.6%	62.8%
<b>Global</b>	41.3%	62.3%	<b>43.5%</b>	54.6%	41.9%	<b>57.0%</b>

TABLEAU 2.4 – Résultats complets sur CDNET de notre détection de nouveauté

<b>Catégories</b> /vidéos	SOM Global			SOM Max	GNG Global	GNG Max
	Précision	Rappel	F-mesure	F-mesure	F-mesure	F-mesure
<b>Turbulence</b>	8.7%	69.6%	12.4%	35.1%	<b>14.9%</b>	<b>36.9%</b>
Turbulence0	1.5%	82.7%	3.0%	28.0%	2.9%	24.3%
Turbulence1	6.0%	68.2%	11.0%	34.1%	13.5%	34.6%
Turbulence2	1.3%	81.8%	2.6%	33.1%	0.9%	35.4%
Turbulence3	25.9%	45.6%	33.1%	45.1%	42.3%	53.1%
<b>Inter. motion [r]</b>	68.0%	46.7%	52.0%	<b>69.0%</b>	<b>57.7%</b>	68.5%
Sofa	61.1%	49.0%	54.4%	67.7%	58.4%	72.2%
StreetLight	64.6%	68.4%	66.4%	85.1%	77.4%	83.3%
Tramstop	78.4%	22.7%	35.3%	54.3%	37.1%	49.9%
<b>Low framerate [r]</b>	70.5%	66.4%	<b>68.0%</b>	74.3%	62.7%	<b>75.4%</b>
TramCrossroad	61.4%	72.9%	66.6%	68.7%	67.5%	69.3%
TunnelExit	77.3%	63.7%	69.9%	75.4%	48.1%	74.5%
Turnpike	73.0%	62.7%	67.4%	78.7%	72.5%	82.4%
<b>Global</b>	41.3%	62.3%	<b>43.5%</b>	54.6%	41.9%	<b>57.0%</b>

TABLEAU 2.5 – Résultats complets sur CDNET de notre détection de nouveauté - Suite

précision est meilleure. Notre modèle se révèle globalement plus sensible que précis, même si les résultats dépendent des séquences évaluées. Comme par exemple avec *Tramstop* qui a une précision trois fois supérieure à son rappel.

On peut également remarquer que notre modèle a beaucoup de mal dans les catégories avec un fond dynamique, c'est à dire avec des feuillages bruissants au vent ou des reflets dans l'eau, comme *Dynamic background* et *Turbulence* avec des précisions ne dépassant pas les 10%. Mais des légers mouvements de caméras ne pénalisent que peu la F-mesure de notre modèle dans *Camera jitters*. Les autres catégories sont aux alentours de la moyenne globale de la SOM.

La version Max de la SOM présente des pistes intéressantes. Elle gagne 10% de performance supplémentaire au global. Cela montre que des gains importants peuvent être faits en ajustant automatiquement le nombre de neurones, la taille des imagettes et le seuil par vidéo. Les gains les plus importants se trouvent dans les catégories dynamiques, avec *Dynamic background* où la F-mesure a été doublée et *Turbulence* où elle a été triplée. On suppose que le seuil est le facteur le plus influent dans ces cas là, car il permet de supprimer une grande partie du bruit environnemental et ainsi d'améliorer la précision pour ces vidéos là.

Les GNG ont des performances similaires à la SOM, et réussissent et échouent dans les mêmes catégories. Au global, ils sont légèrement moins bons que les SOM avec 41,9% de F-mesure contre 43,5%. La version Max cependant arrive à 57%, soit au dessus de la version SOM Max à 54,6%. Cela provient peut-être de la plus forte variabilité observée dans les résultats des GNG, que l'on a par conséquent favorisée en choisissant le maximum pour chaque vidéo.

### Comparaison avec l'état de l'art

CDNET étant un jeu de données très utilisé, il existe de nombreux algorithmes de détection de changement qui ont été testés sur celui-ci. Leur site web<sup>3</sup> référence les résultats obtenus dans la littérature. Ceux-ci sont généralement obtenus en utilisant du post processing pour maximiser les performances. Vu que nous n'en utilisons pas, nous avons choisi de se comparer à [Xu et collab. \[2016\]](#), qui est une review qui reproduit les résultats de méthodes communes de détection de changement, sans pré ni post processing. Nous avons reproduit leurs résultats dans le tableau 2.6.

Modèle	Précision	Rappel	F-mesure
SACON	46.3%	38.2%	24.0%
KDE	37.6%	<b>68.7%</b>	40.8%
CodeBook	61.2%	38.6%	41.1%
Vibe	65.3%	46.5%	47.2%
GMM	49.9%	62.3%	47.6%
AGMM	63.4%	56.0%	53.9%
PBAS	<b>70.9%</b>	50.8%	55.1%
SOBS	66.7%	60.1%	<b>56.4%</b>
GNG Global	39.6%	66.1%	41.9%
SOM Global	41.3%	62.3%	43.5%

TABLEAU 2.6 – Comparatif avec d'autres modèles de détection de changement sur CDNET

Notre méthode est avant dernière en précision, 2<sup>ème</sup> ex-æquo en rappel et 6<sup>ème</sup> sur 9 en F-mesure. Les chiffres indiqués ne sont partiellement comparables, car nos valeurs ont été calculées sur une partie de CDNET, mais l'ordre de grandeur devrait environ être le même. En comparaison des autres modèles, la précision est le plus gros facteur limitant de notre approche. Il faut aussi remarquer que les modèles utilisés dans cette review sont tous "classiques", et le meilleur algorithme actuel BSUV-Net 2 de [TEZCAN et collab. \[2021\]](#), est un réseau de neurones qui obtient 83.9% en F-mesure sur CDNET, en utilisant de l'augmentation de données.

#### 2.5.4 Visualisation des résultats

Dans cette section nous avons regroupé les résultats de détection de notre modèle. La figure 2.31 montre les résultats qui ont obtenu un bon score de détection. On peut y voir que le découpage en bloc de la SOM est présent dans le résultat de la détection. Cela peut expliquer en partie le rappel élevé et la précision plus faible car notre modèle aura tendance à déborder au delà de la stricte nouveauté. On remarquera aussi qu'il n'y a pas de bruit dans nos détections. Le découpage entre zone de nouveauté et zone de fond est très net, même si il présente quelques erreurs, comme une zone de nouveauté qui n'apparaît pas, ou une zone de fond qui apparaît comme nouveauté.

La figure 2.32 nous présente des cas de fond dynamiques où notre modèle a eu particulièrement du mal, notamment avec une précision très faible. On peut observer

---

3. <http://changedetection.net/>

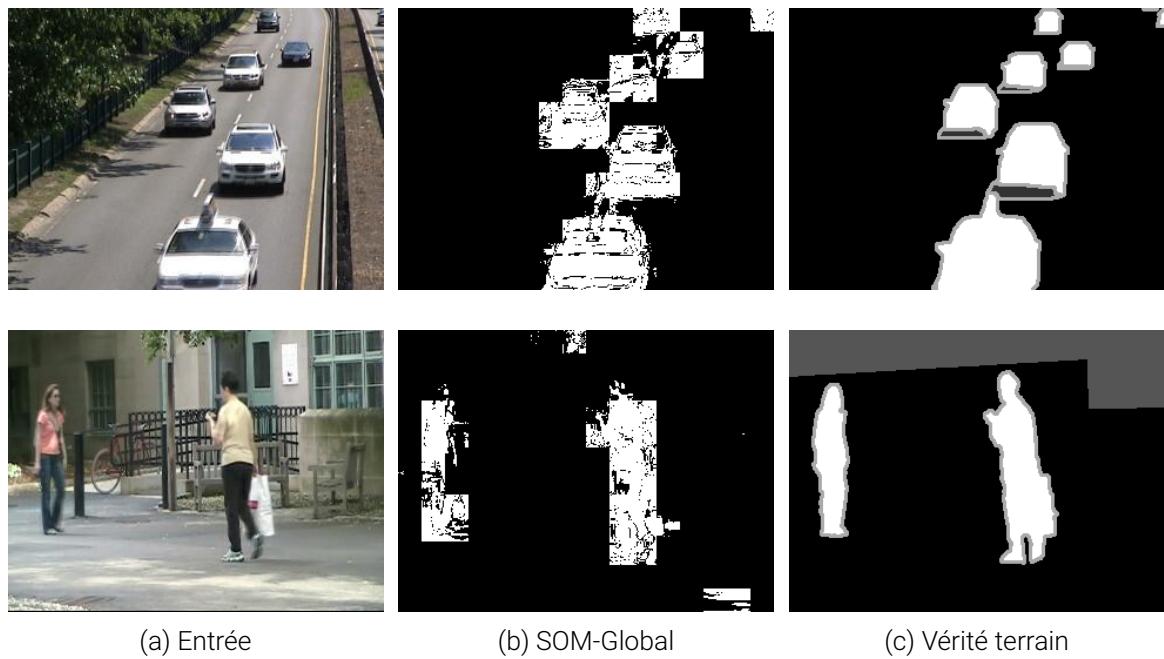


FIGURE 2.31 – Visualisations de résultats normaux de notre modèle. On peut y voir le découpage en bloc dû à l'utilisation d'imagettes et l'absence de bruit autour des nouveautés à déetecter, mais avec quelques erreurs tout de même.

que le mouvement présent dans le fond est confondu avec de la nouveauté. Le mouvement étant permanent, cela cause un nombre très important de faux positifs, ce qui réduit significativement la précision.

La figure 2.33 présente un des défauts de la normalisation de la carte de saillance, ce qui se traduit par un fort taux de bruit en l'absence de nouveauté, comme expliqué dans la section 2.3.3.

## 2.5.5 Interprétations

Les éléments que nous avons présentés montrent que la détection de nouveauté en utilisant la quantification vectorielle et la topologie d'une SOM fonctionne. Quantitativement parlant la précision de notre détection de nouveauté est loin derrière l'état de l'art, que ce soit des modèles classiques programmés ou des réseaux de neurones apprenants. Il faut cependant remarquer que la comparaison pixel à pixel entre la vérité terrain et l'image de détection est particulièrement désavantageuse pour notre modèle, du fait d'une recherche de nouveauté au niveau des imagettes plutôt que du pixel. Si l'on ne compare que de façon binaire si un objet a été détecté ou non, nos résultats pourraient être plus proches des résultats autres méthodes.

Notre modèle n'étant particulièrement pas adapté pour les fonds dynamiques, l'hypothèse que nous avons faite dans la section 2.2 ne s'est donc pas concrétisée. Cette hypothèse supposait que la SOM serait capable de généraliser un environnement présent dans l'image de fond, et que tout mouvement dans cet environnement ne serait pas saillant du fait qu'il ait été appris et généralisé par la SOM. En pratique, un mouvement dans une zone de fond entraîne des changements très importants sur tous les pixels d'une imagette, ce qui se traduit par une grande différence au niveau du

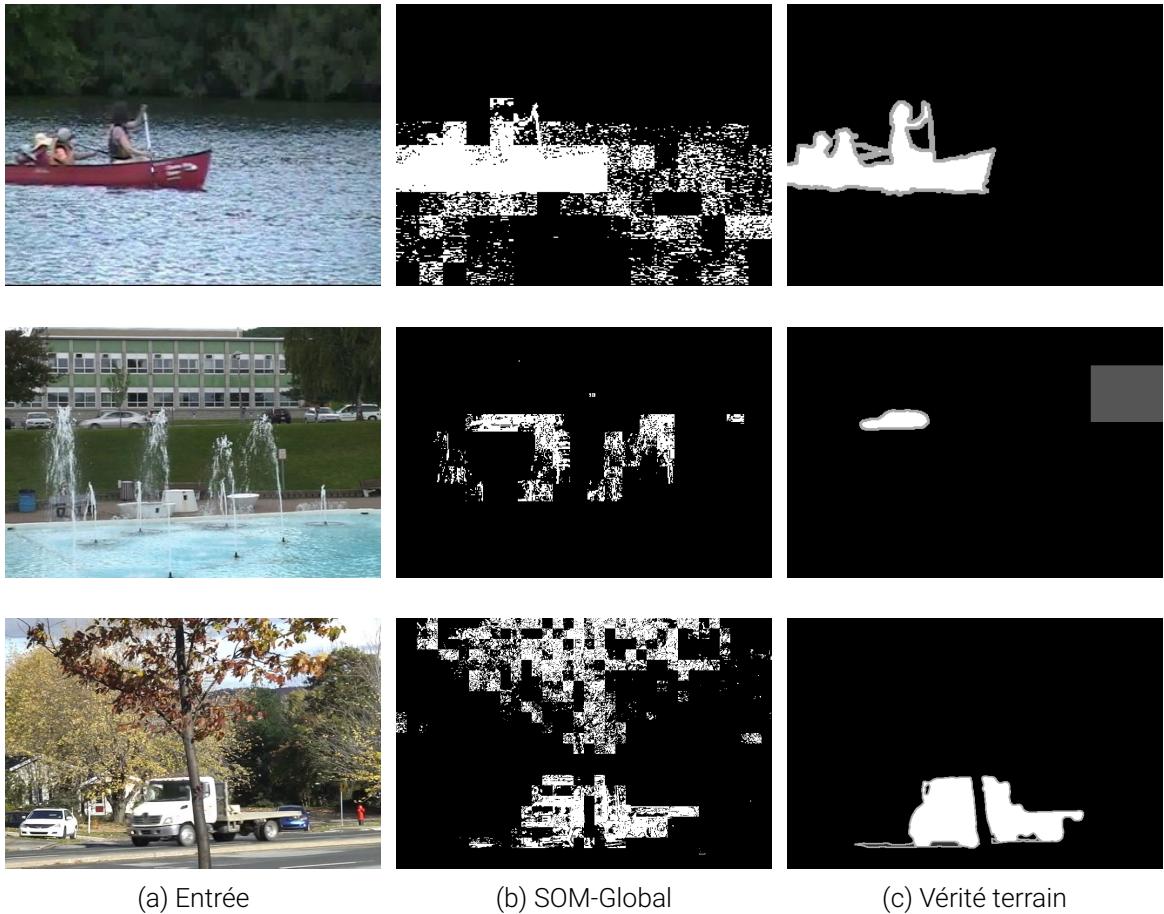


FIGURE 2.32 – Résultats sur des séquences avec un fond dynamique. Les changements trop importants du fond sont constamment considérés en nouveauté et ainsi réduisent drastiquement la précision.

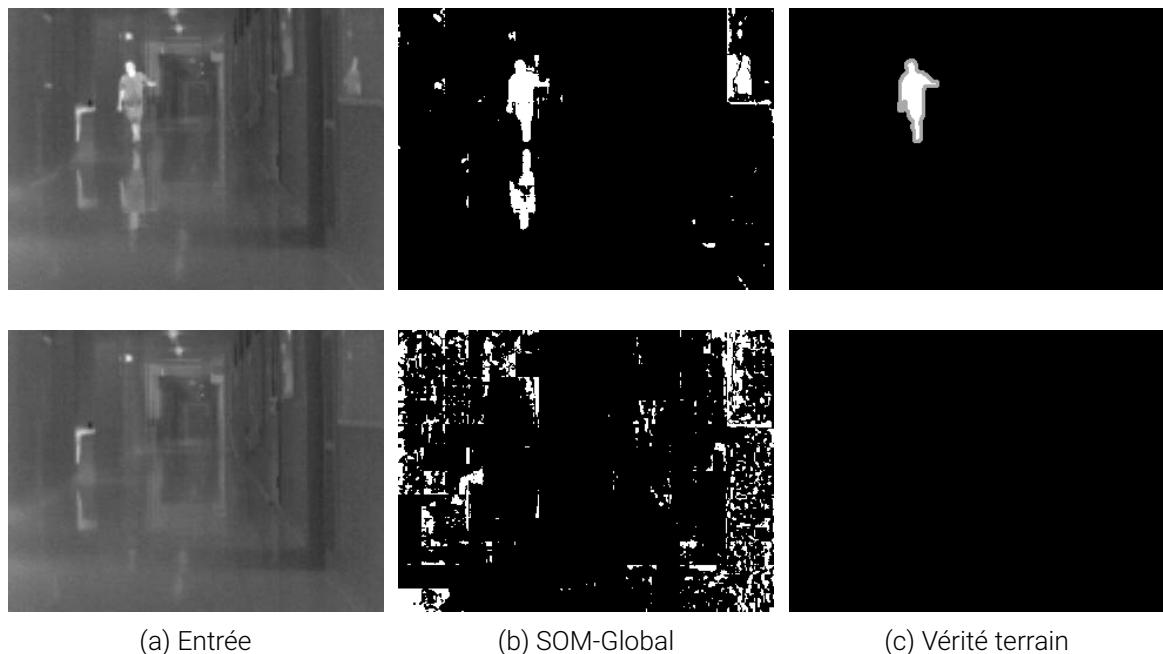


FIGURE 2.33 – Exemple d'un cas où le fait de normaliser la carte de saillance pose problème. Dans la première ligne, l'objet est correctement détecté avec les reflets absents de la vérité terrain, et avec très peu de bruit. La seconde ligne sans objet à détecter fait fortement ressortir le bruit à la place.

calcul de la distance avec les neurones, même si visuellement les deux imagettes sont similaires. Par conséquent, les différences seront élevées lorsqu'un fond dynamique est présent, contrairement à l'hypothèse formulée initialement.

La piste que nous considérons la plus prometteuse pour résoudre ce problème serait de modifier le calcul de distances dans la SOM pour mieux refléter les différences dans les images. Une mesure comme la Structural Similarity (SSIM) [WANG et collab. \[2004\]](#), qui est un calcul de différences mieux adapté pour les images, même si elle ne suit pas les propriétés d'une distance. Une autre option plus neuronale serait d'essayer d'apprendre les types des différences qui peuvent se produire au cours du temps, comme les remous de l'eau par exemple, et ainsi ajouter une notion temporelle à notre modèle, qui n'en possède pas encore.

Nous avons aussi montré avec les GNG que notre approche est facilement généralisable à d'autres modèles de quantification vectorielle avec topologie. Des variantes de SOM ou de GNG pourraient ainsi être essayées pour cette tâche pour potentiellement amener à de meilleurs résultats que la SOM classique sur cette application.

## 2.6 Références

AMERIJCKX, C., J.-D. LEGAT et M. VERLEYSEN. 2003, «Image compression using self-organizing maps», *Systems analysis modelling simulation*, vol. 43, n° 11, p. 1529–1543. [26](#)

BERGSTRA, J., R. BARDET, Y. BENGIO et B. KÉGL. 2011, «Algorithms for hyper-parameter optimization», *Advances in neural information processing systems*, vol. 24. [45](#)

- BERNARD, Y., N. HUEBER et B. GIRAU. 2019, «Novelty detection with self-organizing maps for autonomous extraction of salient tracking features», dans *International Workshop on Self-Organizing Maps*, Springer, p. 100–109. [29](#)
- BERNARD, Y., N. HUEBER et B. GIRAU. 2020, «Novelty detection in images using vector quantization with topological learning», dans *2020 27th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, IEEE, p. 1–4. [29](#)
- BIANCO, S., G. CIOCCA et R. SCHETTINI. 2017, «Combination of video change detection algorithms by genetic programming», *IEEE Transactions on Evolutionary Computation*, vol. 21, n° 6, p. 914–928. [32](#)
- HRIPCSAK, G. et A. S. ROTHSCHILD. 2005, «Agreement, the f-measure, and reliability in information retrieval», *Journal of the American medical informatics association*, vol. 12, n° 3, p. 296–298. [40](#)
- HUYNH-THU, Q. et M. GHANBARI. 2008, «Scope of validity of psnr in image/video quality assessment», *Electronics letters*, vol. 44, n° 13, p. 800–801. [39](#)
- KORHONEN, J. et J. You. 2012, «Peak signal-to-noise ratio revisited : Is simple beautiful?», dans *2012 Fourth International Workshop on Quality of Multimedia Experience*, IEEE, p. 37–38. [39](#)
- MADDALENA, L. et A. PETROSINO. 2008, «A self-organizing approach to background subtraction for visual surveillance applications», *IEEE Transactions on Image Processing*, vol. 17, n° 7, p. 1168–1177. [24](#)
- POWERS, D. M. 2011, «Evaluation : from precision, recall and f-measure to roc, informedness, markedness and correlation», *arXiv preprint arXiv:2010.16061*. [40](#)
- TEZCAN, M. O., P. ISHWAR et J. KONRAD. 2021, «Bsuv-net 2.0 : Spatio-temporal data augmentations for video-agnostic supervised background subtraction», *IEEE Access*, vol. 9, p. 53 849–53 860. [57](#)
- WANG, Y., P.-M. JODOIN, F. PORIKLI, J. KONRAD, Y. BENEZETH et P. ISHWAR. 2014, «Cdnet 2014 : An expanded change detection benchmark dataset», dans *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, p. 387–394. [34](#)
- WANG, Z., A. C. BOVIK, H. R. SHEIKH et E. P. SIMONCELLI. 2004, «Image quality assessment : from error visibility to structural similarity», *IEEE transactions on image processing*, vol. 13, n° 4, p. 600–612. [60](#)
- Xu, Y., J. DONG, B. ZHANG et D. Xu. 2016, «Background modeling methods in video analysis : A review and comparative evaluation», *CAAI Transactions on Intelligence Technology*, vol. 1, n° 1, p. 43–60. [57](#)



# Chapitre 3

## Calcul de Best Matching Unit accéléré avec la topologie

« Il semble que la perfection soit atteinte non quand il n'y a plus rien à ajouter, mais quand il n'y a plus rien à retrancher. »

---

Antoine de Saint-Exupéry

### Sommaire

---

<b>3.1</b>	<b>Introduction</b>	<b>64</b>
<b>3.2</b>	<b>Algorithme séquentiel</b>	<b>66</b>
3.2.1	Analyse de la complexité	68
3.2.2	Protocole expérimental	70
3.2.3	Résultats	71
<b>3.3</b>	<b>Algorithme parallèle</b>	<b>76</b>
3.3.1	Adaptation pour le parallélisme	76
3.3.2	Evaluation	77
<b>3.4</b>	<b>Conclusion</b>	<b>79</b>
<b>3.5</b>	<b>Références</b>	<b>79</b>

---

## 3.1 Introduction

Nous avons dans cette thèse développé des modèles en gardant toujours à l'esprit les contraintes d'une implantation matérielle. Celles-ci impliquent une puissance de calcul limitée et la nécessité d'être réactif en temps réel. Cependant, la quantité de calculs requis par les SOMs est conséquente, que ce soit lors de l'apprentissage ou de la reconstruction. Elle augmente linéairement avec le nombre de neurones, le nombre d'éléments dans le jeu de données et la dimensionnalité de l'entrée. Par conséquent, l'application des SOM à des ensembles de données comportant un grand nombre d'éléments et un nombre élevé de neurones pour représenter précisément l'entrée comme dans notre détection de nouveauté, induit un coût de calcul important qui peut dépasser certaines contraintes telles que le calcul en temps réel ou la faible consommation en énergie. L'optimisation est donc cruciale pour notre approche.

Des variantes de l'algorithme classique de la SOM ont été développées pour réduire le temps de calcul requis par l'apprentissage. La variante la plus connue est le *batch learning*, comme expliqué dans [COTTRELL et collab. \[2018\]](#). Contrairement à l'apprentissage *online* classique, le *batch learning* calcule la moyenne des modifications sur plusieurs vecteurs d'apprentissage avant de mettre à jour les poids des neurones. Des efforts similaires ont été faits dans [FIANNACA et collab. \[2013\]](#) ou dans [OYANA et collab. \[2012\]](#). Cependant, toutes ces variantes se concentrent uniquement sur la réduction du temps de convergence de l'apprentissage. À notre connaissance, aucun travail n'a été effectué pour réduire le temps nécessaire à chaque itération. Cela peut s'expliquer en partie par la nature hautement parallèle des calculs de distances nécessaires pour chaque itération, si bien que lorsqu'une implantation matérielle rapide est envisagée, la solution est généralement de paralléliser tous ces calculs sur un substrat FPGA avec un circuit dédié pour chaque neurone, comme dans [ABADI et collab. \[2018\]](#) ou dans [HUANG et collab. \[2017\]](#). Cependant, l'existence de solutions parallèles ne doit pas conduire à un manque d'effort dans l'optimisation des algorithmes, car la majorité des applications de la SOM sont sur CPU, sans parallélisation, et que les implantations des SOM dans les systèmes embarqués sont souvent limitées par la place disponible sur le FPGA, la consommation électrique ou la vitesse de traitement, qui peuvent tous être améliorés par l'utilisation d'algorithmes d'apprentissage plus efficaces.

Une itération comprend deux phases, la recherche de la BMU et la modification des poids. Nous nous intéresserons ici à la recherche de BMU uniquement, car utile aussi après l'apprentissage. Traditionnellement, la BMU est déterminée par une recherche exhaustive en comparant toutes les distances entre le vecteur d'entrée et les prototypes des neurones. Cette méthode, bien qu'elle permette de toujours trouver le bon minimum dans la carte, est inefficace en calculs, car contrairement aux k-means par exemple, les neurones d'une SOM ne sont pas indépendants entre eux, et l'information présente dans la topologie de la SOM peut être exploitée.

Comme pour la détection de nouveauté, nous avons de nouveau utilisé la propriété de réduction dimensionnelle des SOM pour cette fois-ci accélérer la recherche de la BMU. Pour rappel, la réduction dimensionnelle signifie que les neurones proches dans la topologie de la SOM représentent des vecteurs proches dans l'espace d'entrée. Par conséquent, lorsque l'on calcule les distances de tous les neurones avec un vecteur d'entrée, un gradient pseudo-continu apparaît dans la SOM dont le minimum est situé à la best matching unit. En utilisant une approche à base de descente de

gradient discréte, on peut alors s'attendre à pouvoir atteindre la BMU de la carte, en ne devant calculer qu'une partie seulement des distances entre le vecteur d'entrée et les vecteurs prototypes des neurones. Ce principe est la base de notre algorithme de recherche de BMU que nous appellerons ici FastBMU. Avec notre méthode, nous explorons une nouvelle approche pour l'optimisation des SOM qui peut être combinée avec d'autres méthodes d'optimisation couramment utilisées dans ces modèles pour un calcul encore plus rapide dans les phases d'apprentissage et de rappel.

Ce chapitre est découpé en deux sections. La première présente l'algorithme et son fonctionnement détaillé dans le cas séquentiel. Cette section inclut également une analyse de la complexité et une évaluation des performances et des gains en calculs obtenus. La seconde partie présente l'adaptation de cet algorithme dans le cas parallèle, pour les implantations matérielles.

## 3.2 Algorithme séquentiel

Pour bien comprendre le fonctionnement de notre algorithme Fast-BMU, nous allons le présenter partie par partie. Nous commencerons par l'idée générale de la descente du gradient de la SOM. Puis nous aborderons les cas spéciaux en expliquant pourquoi ils existent, les conséquences possibles et comment Fast-BMU les résout.

---

### Algorithme 1 : Particule

---

**Entrées :**

vecteur : vecteur d'entrée

pos : position de la particule

**Données :**

mémoire\_dist : historique des distances déjà évaluées

**Sorties :**

bmu : index de la Best Matching Unit potentielle

**début**

```
// On commence par regarder les voisins directs pour trouver si l'un
// d'entre eux est plus proche du vecteur d'entrée.
bmu_voisin ← RechercheVoisin(vecteur, pos, 1)
si mémoire_dist [bmu_voisin] < mémoire_dist [pos] alors
| retourner Particule(vecteur, bmu_voisin)
fin
// Si il n'y en a pas, alors on se trouve dans un minimum local.
// On effectue alors une nouvelle recherche dans le voisinage de 2.
bmu_voisin ← RechercheVoisin(vecteur, pos, 2)
si mémoire_dist [bmu_voisin] < mémoire_dist [pos] alors
| retourner Particule(vecteur, bmu_voisin)
fin
// Si on ne trouve toujours rien, alors on peut considérer que l'on a atteint
// la fin de la descente de gradient.
retourner pos
fin
```

---

Le cœur de Fast-BMU est la descente de gradient discrétisée sur les neurones qui minimise la distance entre ceux-ci et le vecteur d'entrée. Le pseudo-code de cette descente de gradient, appelée *Particule*, est présenté dans les algorithmes 1 et 2. Une *Particule* commence à une position choisie arbitrairement dans la carte. Pour cet exemple, illustré dans la figure 3.1, considérons qu'elle débute aux coordonnées (0,0) (coin supérieur gauche). Ce neurone a deux voisins, un à l'est (1,0) et un au sud (0,1). Nous évaluons les distances au vecteur d'entrée de ces trois neurones pour trouver la prochaine étape de la descente du gradient. Si l'un des neurones voisins donne la plus petite distance, alors la descente du gradient va dans la direction de ce neurone particulier, qui devient la prochaine position sélectionnée à partir de laquelle nous répétons ce processus. Si la plus petite distance est trouvée là où nous sommes actuellement positionnés, alors nous considérons que nous nous trouvons dans un minimum local. Dans notre exemple, la plus petite distance est mesurée pour le neurone situé à l'est en (1,0). Le processus de recherche se déplace donc d'un pas vers l'est. Ce neurone

---

**Algorithme 2 :** RechercheVoisin

---

**Entrées :**

vecteur : vecteur d'entrée  
pos : position de la particule  
x : distance topologique

**Données :**

resultats : dictionnaire vide  
prototypes : valeurs des vecteurs prototypes de la SOM  
memoire\_dist : historique des distances déjà évaluées

**Sorties :**

bmu : index de la Best Matching Unit potentielle

**début**

```
pour voisin dans le voisinage à distance topologique x de pos faire
    si memoire_dist[voisin] est vide alors
        memoire_dist[voisin] ← distance(prototypes[voisin],
                                         vecteur)
        resultats[voisin] ← memoire_dist[voisin]
    fin
fin
bmu_voisin ← minimum de resultats
retourner bmu_voisin
fin
```

---

a trois voisins, un au sud (1,1), un à l'est (2,0) et un à l'ouest dont nous venons et que nous allons donc ignorer. Nous comparons à nouveau les trois distances et nous nous déplaçons vers la distance la plus faible (le Sud). Ce processus itère jusqu'à atteindre un minimum local à la position (6,5), où tous les neurones voisins ont une distance au vecteur d'entrée plus élevée que le neurone sélectionné.

Trouver un neurone sans meilleurs voisins ne garantit cependant pas que l'on a terminé la descente de gradient. On peut observer par exemple sur la figure 3.1 que les neurones au sud et à l'ouest du premier minimum local ont un gradient qui pointent vers un autre neurone que le minimum local actuel. Cela s'explique par le fait que, surtout pour les topologies en grille, le gradient soit localement orienté en diagonale, et que par conséquent il faille deux étapes (sud puis ouest ou ouest puis sud) pour continuer la descente. Pour compenser ce problème, et afin de s'assurer que le minimum local qui a été trouvé est bien le meilleur dans le voisinage local, nous effectuons une recherche de minimum dans tous les voisins à voisinage topologique de 2. Si un nouveau minimum a été trouvé, nous continuons la descente de gradient à partir de celui-ci. Si il n'y en a pas, alors nous considérons que la descente de gradient est terminée, et que la position actuelle est le minimum trouvé. Car tous ses voisins sont moins bons, et que le gradient de tous ses voisins amènent à lui.

Un autre problème qui peut survenir avec une unique *Particule* est dû aux bords. Lors de la projection de données à haute dimension sur une SOM 2D, la carte peut parfois devenir tordue sans gradient continu entre le neurone supérieur gauche et le neurone inférieur droit. Imaginez par exemple un filet que l'on jette sur une sphère et

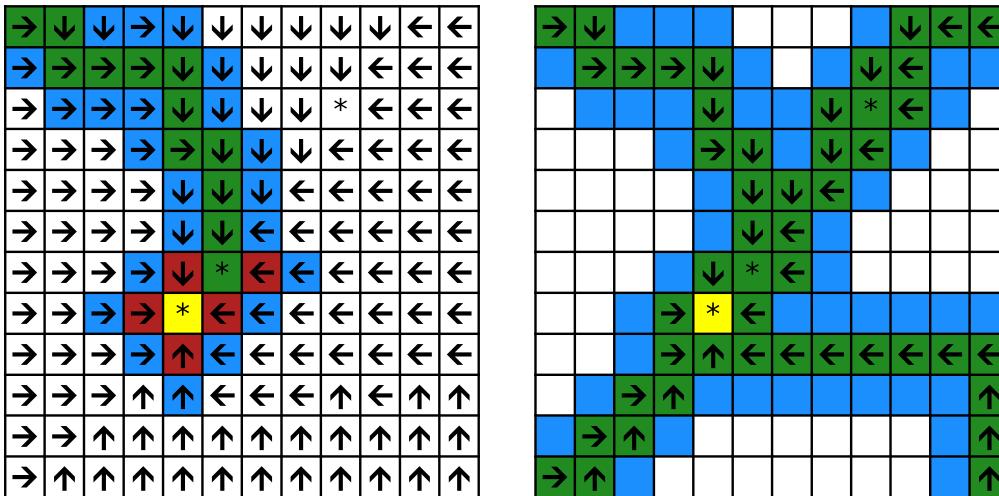


FIGURE 3.1 – À Gauche : exemple d'exécution d'une particule. Chaque cellule représente un neurone. La flèche dans chaque cellule pointe vers le meilleur voisin (qui a une distance inférieure). Une étoile représente un minimum local. Les cellules vertes sont les neurones qui ont été explorés pendant la recherche, et les cellules bleues sont les neurones dont la distance au vecteur d'entrée a été calculée mais qui n'ont pas été sélectionnés par l'algorithme. Après avoir trouvé un minimum local, de nouvelles particules sont créées (en rouge) et continuent la descente du gradient en explorant le voisinage local. La cellule en jaune est la BMU, et le minimum global de la carte. À droite : exécution des quatre particules.

qui l'enveloppe presque complètement : le chemin le plus court entre un coin du filet et le coin opposé du filet ne suivra pas les mailles du filet. Si, au cours de la phase d'apprentissage, la particule se retrouve dans le mauvais coin de la SOM, elle trouvera une très mauvaise BMU et la mise à jour des vecteurs prototypes des neurones brisera localement la continuité du voisinage, et créera une boucle de rétroaction négative qui détruira toutes les propriétés de réduction dimensionnelles de la SOM. Il existe un moyen facile d'éviter cela, en démarrant 4 *Particules*, une dans chaque coin de la SOM, et en sélectionnant la plus petite BMU trouvée parmi les 4. Cette technique préserve la continuité du SOM et rend l'algorithme Fast-BMU plus robuste aux discontinuités locales du gradient, notamment au début de l'apprentissage, lorsque la carte n'est pas encore bien déployée. Fast-BMU complet est défini dans l'algorithme 3, et le résultat d'une exécution complète est illustré dans la figure 3.1 à droite.

Il faut aussi remarquer que dans le cas où il n'y a pas de gradient clair dans la SOM (typiquement lorsque la SOM est initialisée de façon aléatoire avant l'apprentissage), Fast-BMU ne trouvera pas la BMU correcte. Toutefois, ce n'est pas un problème pour l'apprentissage, car la fonction de voisinage créera un gradient en dépliant la SOM, quel que soit le neurone initialement sélectionné, et une fois le gradient créé, Fast-BMU trouvera les bonnes BMU.

### 3.2.1 Analyse de la complexité

Le temps d'exécution d'une recherche de BMU dans la SOM classique dépend de deux facteurs : le nombre de neurones multiplié par la durée d'une comparaison entre le vecteur d'entrée et les poids d'un neurone. L'idée de Fast-BMU étant de réduire le nombre de ces comparaisons, la durée d'une comparaison ne sera pas prise en

---

**Algorithme 3 :** FastBMU

---

**Entrées :**

vecteur : vecteur d'entrée

l, h : largeur et hauteur de la SOM

**Données :**

positions : positions de départ des particules

mémoire\_dist : historique des distances déjà évaluées

resultats : dictionnaire vide

**Sorties :**

bm̄u : index de la Best Matching Unit

**début**

```
    positions ← [(0, 0), (l - 1, 0), (0, h - 1), (l - 1, h - 1)]
    pour pos dans positions faire
        bm̄u_particule = Particule(vector, pos)
        resultats[bm̄u_particule] ← mémoire_dist[bm̄u_particule]
    fin
    bm̄u = minimum de resultats
    retourner bm̄u
fin
```

---

compte dans cette analyse. C'est normalement une valeur qui ne dépend que de la dimensionnalité des données. Le nombre de comparaisons pour Fast-BMU est quant à lui dépendant de la taille de la carte, c'est à dire hauteur et largeur, et de la topologie, c'est à dire du nombre de voisins par neurone.

Nous ferons dans cette analyse l'hypothèse qu'il y a une certaine continuité dans la carte et qu'en partant de n'importe quel point de la carte, on atteigne la BMU en suivant le gradient sans revenir en arrière, c'est à dire que la distance parcourue en suivant le gradient entre les deux soit égale à la distance de manhattan. C'est une hypothèse forte, mais qui doit être vraie pour que notre algorithme de Fast-BMU fonctionne. En pratique, Fast-BMU est plus robuste que cela, et n'a pas besoin que tous les gradients de la carte mènent à la BMU, il en faut juste assez pour qu'une *particule* la trouve. De plus, les problèmes les plus communs dans le gradient de la carte sont des minimums locaux, qui réduisent le nombre de *pas* en arrêtant la *particule* avant la fin. Pour augmenter le nombre de *pas* effectués par une *particule*, il faudrait un gradient qui prenne une direction à l'opposé de ce qu'il avait précédemment. Qu'il fasse des méandres sur son trajet par exemple, ce qui est inhabituel dans des SOM en général. Ainsi, la complexité que nous allons calculer sera probablement une limite haute à la vraie complexité de l'algorithme Fast-BMU.

Pour estimer la complexité de notre algorithme, c'est à dire le nombre de comparaisons dans notre cas, nous devons estimer le nombre de *pas* de chaque *particule* pour trouver la BMU potentielle. En considérant que la BMU est localisée à la position  $x, y$  de la SOM, on a :

- La *particule* en haut à gauche commençant aux coordonnées  $0, 0$  devra faire  $x$  *pas* horizontaux et  $y$  *pas* verticaux pour atteindre la BMU.
- De la même façon, la *particule* du haut à droite avec les coordonnées  $(w, 0)$  devra

faire  $w - x$  pas horizontaux et  $y$  pas verticaux.

- Pour la *particule* en bas à gauche  $(0, h)$ , ce sera  $x$  et  $h - y$  pas.
- Pour la *particule* en bas à droite  $(w, h)$ , ce sera  $w - x$  et  $h - y$  pas.

Pour obtenir le nombre total de pas pour une itération, nous additionnons les nombres de pas de toutes les particules ensemble comme montré dans l'équation 3.1.

$$\begin{aligned} \text{NbrPas} &= (x + y) + (w - x + y) + (x + h - y) + (w - x + h - y) \\ &= 2x - 2x + 2y - 2y + 2w + 2h \end{aligned} \quad (3.1)$$

Les  $x$  et  $y$  s'annulant, le nombre total de pas ne dépend donc que de la largeur et la hauteur de la carte, soit  $(2w + 2h)$ . On peut donc en déduire la Complexité Estimée  $\mathcal{C}$  avec l'équation 3.2

$$\mathcal{C}(w, h) = 2 \times (w + h) \times \text{NbrEvalParPas} \quad (3.2)$$

avec  $w, h$  la largeur et la hauteur de la SOM respectivement et  $\text{NbrEvalParPas}$  le nombre de nouvelles distances à calculer pour chaque *pas* d'une particule. Sa valeur dépend de la topologie. Elle est au maximum de 3 avec une grille (4 voisins moins le neurone de l'étape précédente) et également de 3 avec une topologie hexagonale (6 voisins moins le neurone de l'étape précédente et 2 neurones qui étaient voisins du neurone précédent et qui ont donc déjà été évalués).

D'un point de vue analytique, nous pouvons estimer que notre algorithme Fast-BMU ( $\mathcal{C}(w, h) = 6(w + h)$  dans le pire des cas dans une configuration de grille standard) est significativement plus rapide que l'algorithme actuel de recherche exhaustive ( $O(w, h) = wh$ ) lorsque le nombre de neurones dans la carte est important. Par exemple, il est théoriquement deux fois plus rapide avec des SOM de 24 par 24, et 10 fois plus rapide avec des SOM de 120 par 120. Une évaluation expérimentale de la différence de vitesse est présentée dans la section 3.2.3.

### 3.2.2 Protocole expérimental

En pratique, il arrive parfois malgré tout que FastBMU se trompe de BMU. Nous allons donc expérimentalement mesurer l'impact de ces erreurs sur l'apprentissage et la reconstruction. Pour évaluer la robustesse de notre algorithme avec différents types de données, nous avons sélectionné 6 jeux de données représentatifs du type de données sur lesquelles les SOM sont habituellement entraînés. Pour le cas 2D, nous avons généré des données avec différentes propriétés (une distribution uniforme et une forme hautement non convexe), pour les données 3D nous utilisons une forme cubique uniformément distribuée et les valeurs de couleur des pixels d'une image. Pour les hautes dimensions, nous considérons l'apprentissage par imagettes d'une image tel qu'il a déjà été présenté, avec des sous-images de 10 par 10 comme vecteurs d'entraînement (100 pixels avec 3 couleurs chacun, ce qui donne 300 dimensions), ainsi que le Free Spoken Digits Dataset JACKSON et collab. [2018] qui utilise des ondes

sonores que nous avons réduites à 1000 dimensions. Nous employerons des SOM à 2 dimensions avec des topologies en grille et hexagonales, car ce sont les plus utilisées.

Pour ces tests, nous avons fait débuter  $\alpha$  à 0.6 pour finir à 0.05.  $\sigma$  commence à 0.5 et termine à 0.001. La SOM est de taille  $32 \times 32$  (1024 neurones) et apprend pendant 10 époques.

Afin d'évaluer les différences entre tous les modèles testés, nous avons utilisé deux métriques. La première est la *Mean Squared Quantization Error* (MSQE) que nous avons déjà utilisée dans le chapitre 2. MSQE mesure la qualité de quantification vectorielle de l'algorithme testé. La seconde métrique est la *Mean Squared Distance to Neurons* (MSDtN) qui calcule la distance moyenne au carré entre les prototypes des neurones et ceux de leurs voisins directs. Plus cette valeur est faible, plus les neurones voisins sont proches dans l'espace d'entrée et plus la propriété de réduction dimensionnelle est bonne. De nombreuses métriques existent dans la littérature sur les SOM pour la qualité de la réduction dimensionnelle d'une SOM PÖLZLBAUER [2004], mais MSDtN a l'avantage d'être facile à calculer, sans paramètres et de ne dépendre que des poids des neurones. Pour les deux métriques retenues, une valeur plus faible est meilleure.

$$\text{MSQE} = \frac{1}{dn} \sum_{i=0}^{n-1} (\nu_i - w_{\text{bm}(i)})^2 \quad (3.3)$$

$$\text{MSDtN} = \frac{1}{dN} \sum_{i=1}^N \sum_{j=1}^N \begin{cases} (w_i - w_j)^2, & \text{Si } \text{dist}(i, j) = 1 \\ 0, & \text{Sinon} \end{cases} \quad (3.4)$$

Avec  $n$  étant le nombre de vecteurs dans le jeu de données,  $d$  leur dimensionnalité,  $\nu_i$  le  $i^{\text{ème}}$  vecteur d'entrée, et  $\text{bm}(i)$  l'indice du neurone qui est la *best matching unit* de  $\nu_i$ . De même,  $N$  est le nombre de neurones de la SOM et  $w_i$  les poids du  $i^{\text{ème}}$  neurone, tandis que  $\text{dist}(i, j)$  est le nombre de connexions dans le chemin le plus court entre les neurones  $i$  et  $j$  dans la carte neuronale.

### 3.2.3 Résultats

#### Performances

Dans cette section, nous examinerons les différences de qualité d'apprentissage et de reconstruction entre la version standard et notre version FastBMU pour le calcul de la BMU. Nous regarderons également les différences pratiques dans le nombre de calculs requis pour les deux versions et nous les comparerons avec l'analyse de complexité présentée précédemment.

Pour chaque combinaison de jeu de données et de modèle (choix de la topologie et de l'algorithme BMU), nous avons effectué 50 exécutions avec différentes graines aléatoires qui affectent les jeux de données générés, l'initialisation des poids des neurones (qui sont tous initialisés aléatoirement sans gradient préexistant) et l'ordre dans lequel les vecteurs d'apprentissage sont présentés. Les résultats sont présentés dans le tableau 3.1.

La colonne apprentissage du tableau indique la combinaison de l'algorithme de recherche de BMU et de la topologie qui a été utilisée pour l'entraînement. La MSDtN (voir section 3.2.2) est calculée sur les poids des neurones entraînés. La MSQE\_S (Standard) est l'erreur de quantification dans la phase de reconstruction (après apprentissage) en utilisant la recherche de BMU standard. La comparaison des différentes valeurs de MSQE\_S pour des SOM ayant été apprises par une version standard et une version rapide de la recherche de BMU donne une indication de l'influence de l'algorithme Fast-BMU sur la qualité de l'apprentissage uniquement, car il sélectionne toujours la vraie BMU dans la phase de reconstruction. La métrique MSQE\_F (Fast) mesure l'erreur de quantification vectorielle avec la sélection de BMU effectuée par l'algorithme Fast-BMU. Si l'entraînement a été effectué sur la SOM standard, il donne une indication de l'influence de l'algorithme Fast-BMU sur la précision du reconstruction uniquement; s'il a été entraîné sur une version Fast, il représente le résultat MSQE d'une SOM qui utilise uniquement l'algorithme Fast-BMU. La colonne *Mismatch* donne la proportion de BMU qui sont sélectionnées différemment par les deux algorithmes.

TABLEAU 3.1 – Résultats avec une SOM de  $32 \times 32$  neurones, avec 50 executions par ligne. La colonne Apprentissage précise avec quel algorithme et quelle topologie la SOM a été entraînée. MSQE\_S est le MSQE calculé après apprentissage avec l'algorithme BMU standard (exhaustif) tandis que MSQE\_F utilise la version Fast-BMU. Nous combinons les deux apprentissages avec les deux reconstructions possibles (Standard et Fast). Les différences de MSQE\_S entre les différents algorithmes reflètent ainsi la qualité de la phase d'apprentissage. Le *Mismatch* est la proportion de BMU qui sont sélectionnés différemment par les deux algorithmes.

Données	Apprentissage	MSDtN	MSQE_S	MSQE_F	Mismatch
Carré (2D)	Grille	1,94e-4	2,22e-4	2,22e-4	0%
	Fast-Grille	<b>1,93e-4</b>	2,23e-4	2,23e-4	0%
	Hexagonal	2,39e-4	<b>2,12e-4</b>	<b>2,12e-4</b>	0%
	Fast-Hexa	2,38e-4	2,15e-4	2,15e-4	0%
Forme (2D)	Grille	<b>1,38e-4</b>	1,40e-4	1,40e-4	$\approx$ 0%
	Fast-Grille	<b>1,38e-4</b>	1,40e-4	1,40e-4	$\approx$ 0%
	Hexagonal	1,65e-4	<b>1,31e-4</b>	<b>1,31e-4</b>	$\approx$ 0%
	Fast-Hex	1,65e-4	<b>1,31e-4</b>	<b>1,31e-4</b>	$\approx$ 0%
Cube (3D)	Grille	<b>4,48e-4</b>	2,21e-3	2,50e-3	4.8%
	Fast-Grille	4,61e-4	2,25e-3	3,21e-3	9.8%
	Hexagonal	5,29e-4	<b>2,09e-3</b>	<b>2,34e-3</b>	<b>3.1%</b>
	Fast-Hex	5,38e-4	2,11e-3	2,79e-3	7.6%
Couleurs (3D)	Grille	<b>1,15e-4</b>	8,64e-5	8,80e-5	4.4%
	Fast-Grille	1,19e-4	<b>8,91e-5</b>	9,08e-5	5.4%
	Hexagonal	1,33e-4	8,29e-5	8,30e-5	<b>0.4%</b>
	Fast-Hex	1,35e-4	<b>8,26e-5</b>	<b>8,29e-5</b>	0.7%
Image (300D)	Grille	<b>1,64e-4</b>	1,80e-3	1,83e-3	4.2%
	Fast-Grille	1,65e-4	1,82e-3	1,85e-3	4.4%
	Hexagonal	1,97e-4	<b>1,75e-3</b>	1,77e-3	<b>1.2%</b>
	Fast-Hex	1,99e-4	<b>1,75e-3</b>	<b>1,76e-3</b>	<b>1.2%</b>
Sons (1000D)	Grille	2,02e-4	1,42e-2	1,49e-2	31.3%
	Fast-Grille	<b>1,93e-4</b>	1,44e-2	1,51e-2	32.2%
	Hexagonal	2,29e-4	<b>1,41e-2</b>	<b>1,45e-2</b>	19.8%
	Fast-Hex	2,25e-4	1,42e-2	<b>1,45e-2</b>	<b>13.3%</b>

Nous observons d'abord que la distance entre les neurones après apprentissage (MSDtN) est plus faible avec la topologie en grille qu'avec la topologie hexagonale, mais cette différence pourrait être attribuée au nombre différent de voisins entre les deux topologies et ne devrait donc être utilisée que pour comparer les différences entre les algorithmes standard et Fast-BMU, et pas les topologies. Nous remarquons également que l'algorithme Fast ne présente pas d'écart significatif sur les jeux de données Square et Shape, et donc que MSQE\_S et MSQE\_F ont des résultats de reconstruction similaires sur ces jeux de données.

Les pourcentages de *mismatch* varient considérablement d'un ensemble de données à l'autre. De 0 à 6% pour les jeux de données Images et Couleurs, de 3 à 10% pour le Cube et de 13 à 33% pour les données sonores. Ces différences peuvent s'expliquer par la distribution des données d'entrée, car les jeux de données Images et Couleurs contiennent des éléments qui sont étroitement liés entre eux. Dans les images, par exemple, il y a généralement quelques couleurs dominantes avec beaucoup de valeurs intermédiaires qui rendent les continuités de la distribution plus faciles à réduire dimensionnellement pour une SOM, améliorant ainsi les performances de notre algorithme Fast-BMU. En revanche, le jeu de données *Spoken Digits* présente des valeurs de *mismatch* élevées, ce qui semble indiquer qu'une forte continuité dans les poids des neurones n'est pas présente après l'apprentissage. La topologie hexagonale est également plus performante avec l'algorithme Fast-BMU que la topologie en grille, car les *mismatch* sont nettement plus faibles avec elle. Enfin, la dimensionnalité d'un jeu de données ne semble pas jouer un rôle clé ici, puisque le jeu de données Image (300 dimensions) présente moins de *mismatch* que le jeu de données Cube (3 dimensions).

Pour la partie quantification vectorielle, la topologie hexagonale donne à nouveau les valeurs d'erreur les plus faibles. Ce qui est plus surprenant, c'est que la version Fast-BMU présente des résultats de quantification très similaires à ceux de la version standard. Même avec des *mismatch* élevés (30% avec Digits utilisant un SOM basé sur une grille), le MSQE n'est supérieur que d'environ 5%, et encore moins lorsque seul l'apprentissage est comparé. Les seules valeurs de MSQE significativement plus élevées avec Fast-BMU concernent le jeu de données Cube, où l'algorithme sélectionne de très mauvais choix de BMU dans la phase de reconstruction tout en étant capable de former correctement la SOM. Dans la plupart des cas, une différence dans la topologie de la SOM a plus d'impact sur la MSQE que l'utilisation de l'algorithme Fast-BMU.

## Gains computationnels

Pour évaluer les gains de calcul induits par l'utilisation de l'algorithme Fast-BMU indépendamment des techniques d'implémentation, nous avons comparé le pourcentage de neurones qui doivent être évalués afin de trouver le BMU. Les résultats sont présentés dans la figure 3.2. La SOM standard évalue par définition tous les neurones, le pourcentage est donc toujours de 100%. La courbe de complexité de Fast-BMU représente la fonction définie dans la section 3.2.1. Pour obtenir la courbe mesurée par Fast, nous avons effectué des tests avec des SOM de taille  $n \times n$ , où  $n$  est tout nombre pair compris entre 10 et 50 (21 tests au total). Chaque test comprenait tous les ensembles de données et toutes les topologies (soit 12 exécutions par test).

Notre algorithme est deux fois plus rapide avec des SOM de  $16 \times 16$ , quatre fois plus rapide avec 1000 neurones ( $32 \times 32$ ). La SOM  $50 \times 50$  évalue environ 375 neurones

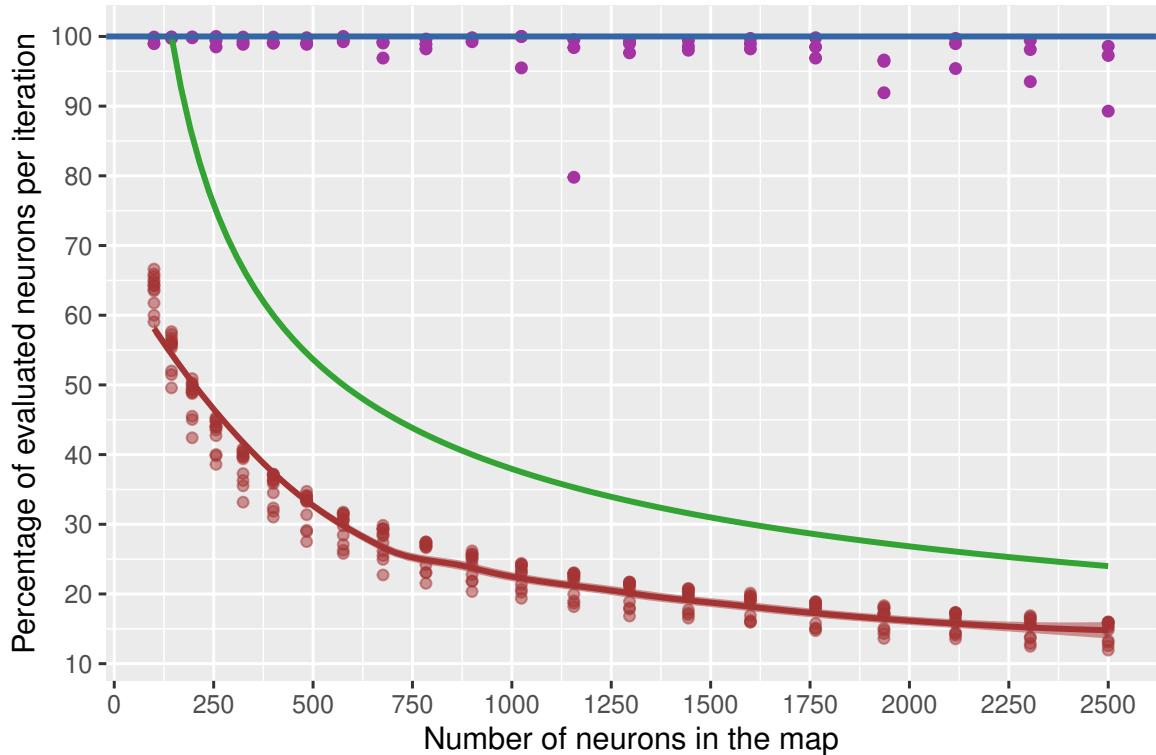


FIGURE 3.2 – Évaluation des gains de quantité de calculs en fonction du nombre de neurones. Le gain en performance est exprimé en pourcentage de neurones qui ont dû être évalués pendant la recherche de BMU. Tous les résultats ont été calculés sur des cartes carrées. La ligne bleue représente la SOM standard, en vert la valeur analytique calculée et en rouge la valeur mesurée. En outre, les points violetts représentent le pourcentage de BMU correctes dans une exécution sur le jeu de données Image.

par itération, ce qui est similaire aux 400 neurones qu'une SOM standard de  $20 \times 20$  doit évaluer. Nous pouvons également observer que la courbe de complexité suit une forme similaire à la courbe mesurée, tout en surestimant le nombre requis de neurones évalués d'environ 75%.

## 3.3 Algorithme parallèle

Une implémentation matérielle sur FPGA de notre détection de nouveauté nécessiterait du parallélisme pour utiliser efficacement les ressources à disposition. Car implémenter une SOM complète sur FPGA pour n'avoir qu'une dizaine de neurones actifs en même temps serait sous optimal. FastBMU est cependant un algorithme fondamentalement séquentiel. Chaque étape de celui-ci dépend fortement de l'étape précédente pour suivre le gradient. Par conséquent en faire une version parallèle nécessiterait de changer fondamentalement son fonctionnement. Une idée plus simple pour réaliser ce parallélisme serait de le placer non pas au niveau de la recherche de BMU, mais au niveau des itérations. En effet, si il est possible de paralléliser les instructions d'une façon qui permette d'utiliser quasiment tous les neurones simultanément avec FastBMU sans compromettre la qualité des BMU trouvées, alors nous aurons un moyen d'être plus performant que les versions purement parallèles de SOM qui calculent les distances de tous les neurones en une seul étape. Si nous atteignons un pourcentage d'utilisation par neurone proche de 100%, en utilisant FastBMU qui fait beaucoup moins d'évaluations que la version standard, alors nous calculerons plus de BMU par étape que la version standard parallèle.

### 3.3.1 Adaptation pour le parallélisme

Une des propriétés de FastBMU, est qu'il part des quatre coins et qu'il converge vers le centre pendant son exécution. Ce qui veut dire que l'utilisation des neurones se fait par vague. Ainsi on si on lance une itération de FastBMU par étape, on utilisera de façon plus efficiente les neurones. Lancer une itération par étape donnera une vitesse équivalente pour FastBMU en parallèle à la version standard en parallèle. Mais avec FastBMU, les poids des neurones pourront changer pendant la descente de gradient, puisque ce sont les itérations qui sont dorénavant parallélisées.

L'intérêt de FastBMU étant de réduire le nombre d'évaluations effectuées par recherche de BMU, envoyer une itération par étape ne suffira pas à saturer les neurones, et il existera toujours des neurones inactifs grâce aux évaluations évitées. Mais on ne peut pas simplement envoyer plus d'itérations par étape, car nous savons que les neurones dans les coins par exemple sont utilisés à 100%, car ce sont les points de départ des particules. Nous avons donc calculé le pourcentage d'utilisation de tous les neurones dans la figure 3.3. Les valeurs de la figure proviennent d'un exemple particulier d'une carte de 20 par 20 neurones, mais les catégories sont généralisables pour tout types de données et pour toutes les tailles de carte assez grandes, tant qu'un gradient est présent pour que FastBMU fonctionne.

D'après la figure, on peut diviser les neurones en trois catégories, les coins, les bords et le centre. Les coins étant utilisés 100% du temps, les bords entre 50 et 60% du temps et moins de 25% du temps pour les neurones centraux. Ainsi, pour augmenter l'utilisation des neurones proche des 100%, il faut multiplier l'utilisation des neurones centraux par 4. Pour ce faire il est nécessaire d'augmenter le nombre d'itérations commencées par étape à 4. Cependant cela augmentera l'utilisation des neurones des bords et des coins au delà de 100%. Pour résoudre ce problème, il est possible d'augmenter matériellement le nombre de comparaisons que peuvent effectuer ces neurones. Si l'on multiplie par 4 le nombre de comparaisons simultanées que l'on peut

100	100	65	64	63	63	62	59	59	60	60	60	59	60	61	62	63	64	100	100	
100	100	65	65	64	64	63	61	61	61	61	61	61	61	63	64	65	65	100	100	
56	58	17	18	20	21	20	19	20	20	20	19	18	19	19	19	19	19	61	60	
56	58	17	18	20	21	21	20	20	20	19	18	18	19	20	21	19	17	59	57	
56	58	17	18	21	22	22	22	21	21	21	20	19	21	21	21	19	18	59	57	
56	58	18	19	21	23	23	23	22	22	21	21	21	21	22	22	21	19	19	59	58
57	59	18	20	22	23	23	24	24	24	23	23	22	23	23	22	20	20	60	58	
57	60	19	20	21	23	24	24	24	24	24	24	23	24	22	22	20	19	60	58	
58	61	21	20	21	22	23	25	25	25	25	25	24	24	23	21	20	20	60	57	
59	61	21	21	22	22	23	24	25	25	25	25	25	24	24	23	20	19	60	58	
60	61	19	21	23	23	24	24	24	24	25	25	26	25	24	23	22	20	61	59	
60	61	19	19	21	22	24	24	25	25	24	25	25	24	24	23	22	21	62	61	
59	61	18	19	21	22	23	24	24	24	25	25	25	24	24	24	23	22	63	61	
60	61	19	19	20	20	22	22	23	23	24	24	23	23	23	23	22	22	65	63	
62	64	19	19	21	20	21	21	23	23	22	22	22	22	22	21	20	66	64		
62	64	19	18	20	19	19	20	21	21	20	20	20	20	19	18	19	18	66	64	
64	65	19	19	18	17	18	20	20	20	19	18	18	18	18	17	17	16	66	64	
65	66	17	17	18	17	18	19	20	20	19	19	18	18	17	16	16	16	66	64	
100	100	57	57	57	56	57	58	58	58	57	56	56	56	56	56	56	56	100	100	
100	100	55	55	55	54	56	56	56	55	54	54	54	54	54	54	55	55	100	100	

FIGURE 3.3 – Pourcentage d'évaluations par neurones pour FastBMU. On peut diviser ceux-ci en trois catégories. La première en rouge regroupe tous les coins qui sont évalués à chaque itération, puisque point de départ des particules. La seconde sont les bords en jaune qui sont évalués un peu plus de la moitié du temps. La dernière est le centre en vert, qui sont évalués moins d'un quart du temps.

faire pour les neurones de chaque coin par exemple, ils pourront traiter 4 itérations simultanément, et seront capables de gérer la surcharge nécessaire pour augmenter l'efficacité des neurones centraux. Le même principe peut être appliqué pour les neurones des bords, en leur permettant de faire deux ou trois comparaisons simultanées.

Ces valeurs pourraient changer avec la taille de la carte. Une carte plus grande utilisera moins d'évaluations par recherche de BMU, et les pourcentages d'utilisation des neurones centraux diminueront en conséquence. Les neurones des bords et des coins resteront eux à des pourcentages similaires. Des cartes plus grandes nécessiteront donc de faire plus d'itérations par étape si l'on souhaite garder des neurones centraux proche des 100% d'utilisation. Le nombre d'évaluations parallèles possibles par les neurones des bords et des coins devront aussi être augmentés.

En lançant 4 nouvelles itérations par étape, nous devenons plus performants que la version classique de la SOM parallèle, qui ne peut faire qu'une itération par étape. Mais il reste la question de l'impact qu'a la modification des poids des neurones pendant que des descentes de gradient sont en cours. Nous l'évaluerons dans la section suivante.

### 3.3.2 Evaluation

Nous avons évalué les performances de notre algorithme parallèle de la même manière que la version séquentielle, avec quelques données en moins. Les résultats sont présentés dans le tableau 3.2. Ils ne sont cependant pas directement comparables avec ceux montrés précédemment dans le tableau 3.1, car ils ont été obtenus avec une nouvelle expérience et avec des paramètres légèrement différents de la pre-

mière. Les différences relatives entre les algorithmes sont néanmoins similaires entre les deux expériences.

La version *Parallel* est une version de FastBMU qui lance 4 itérations par étape, c'est à dire qu'il peut y avoir plusieurs dizaines de descentes de gradient en cours sur la carte à tout moment de l'apprentissage. Nous ne sommes cependant pas rentrés dans les détails de l'implémentation matérielle, car le but est ici de mesurer l'impact de la simultanéité des itérations sur l'apprentissage de la SOM. Ainsi la modification des poids se fait instantanément après que la dernière particule d'une recherche de BMU se termine.

TABLEAU 3.2 – Résultats avec une SOM de  $32 \times 32$  neurones, les métriques de chaque ligne sont calculées sur une moyenne de 10 exécutions. La colonne Apprentissage précise avec quel algorithme et quelle topologie la SOM a été entraînée. MSQE\_S est le MSQE calculé après apprentissage avec l'algorithme BMU standard (exhaustif) tandis que MSQE\_F utilise la version Fast-BMU. Nous combinons les trois apprentissages (Standard, Fast et Parallel) avec les deux reconstructions possibles (Standard et Fast, sans Parallel puisqu'il est équivalent à Fast pour la reconstruction). Les différences de MSQE\_S entre les différents algorithmes reflètent ainsi la qualité de la phase d'apprentissage. Le *Mismatch* est la proportion de BMU qui sont sélectionnés différemment par les deux algorithmes.

Données	Apprentissage	MSDtN	MSQE_S	MSQE_F	Mismatch
Carré (2D)	Grille	<b>2,44e-4</b>	2,28e-4	2,28e-4	0.0%
	Grille Fast	<b>2,44e-4</b>	2,28e-4	2,28e-4	0.0%
	Grille Parallel	2,49e-4	<b>2,02e-4</b>	<b>2,02e-4</b>	0.0%
	Hexagonal	<b>2,80e-4</b>	2,20e-4	2,20e-4	0.0%
	Hexa Fast	<b>2,80e-4</b>	2,20e-4	2,20e-4	0.0%
	Hexa Parallel	2,84e-4	<b>2,02e-4</b>	<b>2,02e-4</b>	0.0%
Couleurs (3D)	Grille	<b>1,47e-4</b>	8,49e-5	8,71e-5	5.3%
	Grille Fast	1,53e-4	8,63e-5	8,78e-5	5.4%
	Grille Parallel	1,88e-4	<b>8,06e-5</b>	<b>8,44e-5</b>	7.6%
	Hexagonal	<b>1,62e-4</b>	8,05e-5	8,10e-5	1.0%
	Hexa Fast	1,62e-4	7,96e-5	7,98e-5	<b>0.5%</b>
	Hexa Parallel	1,83e-4	<b>7,60e-5</b>	<b>7,68e-5</b>	1.5%
Cube (3D)	Grille	5,63e-4	2,22e-3	<b>2,68e-3</b>	6.0%
	Grille Fast	5,80e-4	<b>2,20e-3</b>	3,12e-3	7.5%
	Grille Parallel	<b>5,59e-4</b>	2,34e-3	3,77e-3	12.8%
	Hexagonal	<b>6,09e-4</b>	<b>2,12e-3</b>	<b>2,34e-3</b>	<b>2.6%</b>
	Hexa Fast	6,31e-4	<b>2,12e-3</b>	2,51e-3	5.5%
	Hexa Parallel	6,19e-4	2,18e-3	2,46e-3	3.2%
Image (300D)	Grille	<b>2,05e-4</b>	<b>1,80e-3</b>	1,84e-3	3.5%
	Grille Fast	2,05e-4	1,81e-3	<b>1,83e-3</b>	4.5%
	Grille Parallel	2,23e-4	1,82e-3	1,88e-3	4.7%
	Hexagonal	<b>2,34e-4</b>	1,76e-3	1,78e-3	1.4%
	Hexa Fast	2,34e-4	1,75e-3	<b>1,77e-3</b>	<b>1.2%</b>
	Hexa Parallel	2,49e-4	<b>1,75e-3</b>	1,77e-3	<b>1.2%</b>

Le premier point intéressant dans le tableau se trouve pour les données Carré. En effet, tous les algorithmes arrivent à 0% de *mismatch*, mais on peut voir des différences entre les versions Standard et Fast et la version Parallel. Cette différence ne

pouvant pas être expliquée par la sélection de BMU différentes, elle provient donc probablement de la simultanéité des recherches de BMU dans la version Parallel. Cette recherche en simultané ajoute du bruit dans l'apprentissage, ce qui a l'air d'améliorer légèrement les performances de la SOM en quantification vectorielle, réduisant très légèrement sa continuité topologique. La même chose semble se produire pour *Couleurs*.

*Cube* quand à lui est encore plus mauvais avec Parallel. *Image* a des résultats de MSQE très similaire entre les trois algorithmes. Parallel étant juste légèrement moins bon en continuité topologique. Il semblerait donc que sur les données où FastBMU séquentiel fonctionne, la version parallèle fonctionne tout aussi bien, et que sur des données où la version séquentielle a des problèmes, ils empirent avec la version parallèle. Pour la catégorie de données qui nous intéresse (*Images*), FastBMU en parallèle constitue donc une variante capable de réduire significativement le coût en calcul de la SOM lorsqu'implémentée matériellement pour notre détection de nouveauté par exemple.

## 3.4 Conclusion

Nous avons présenté un nouvel algorithme pour calculer la BMU plus rapidement, en s'appuyant sur la propriété de réduction dimensionnelle des SOM. Nous avons montré une version séquentielle, utilisable dans des applications sur ordinateur et une version parallèle pour des systèmes embarqués sur FPGA par exemple. FastBMU est significativement plus rapide que la version exhaustive, permettant même de passer de la classe de complexité  $O(n^2)$  à  $O(n)$ , tout en ayant des performances en quantification vectorielle très proches de l'approche standard. Cette amélioration en complexité permet aussi d'imaginer l'utilisation de SOM plus grandes que les tailles généralement utilisées. Nous pensons également que notre algorithme, ou une autre approche qui exploite le même principe, peut encore être amélioré car il existe encore de nombreuses redondances dans les calculs que l'on effectue.

FastBMU suit également un paradigme cellulaire, qui consiste à effectuer uniquement des calculs localisés entre neurones voisins, réduisant le besoin de connexions à longue distance dans des implantations matérielles, et ainsi simplifier la mise à l'échelle. Nous travaillons également à l'implantation matérielle de Parallel-FastBMU pour en montrer la faisabilité et pour mesurer les gains apportés par cette approche en conditions réelles.

## 3.5 Références

ABADI, M., S. JOVANOVIC, K. B. KHALIFA, S. WEBER et M. H. BEDOUI. 2018, «A scalable and adaptable hardware noc-based self organizing map», *Microprocessors and Microsystems*, vol. 57, p. 1–14. [64](#)

COTTRELL, M., M. OLTEANU, F. ROSSI et N. VILLA-VIALANEIX. 2018, «Self-organizing maps, theory and applications», *Revista de Investigacion Operacional*, vol. 39, n° 1, p. 1–22. [64](#)

FIANNACA, A., G. DI FATTA, R. RIZZO, A. URSO et S. GAGLIO. 2013, «Simulated annealing technique for fast learning of som networks», *Neural Computing and Applications*, vol. 22, n° 5, p. 889–899. [64](#)

HUANG, Z., X. ZHANG, L. CHEN, Y. ZHU, F. AN, H. WANG et S. FENG. 2017, «A hardware-efficient vector quantizer based on self-organizing map for high-speed image compression», *Applied Sciences*, vol. 7, n° 11, p. 1106. [64](#)

JACKSON, Z., C. SOUZA, J. FLAKS, Y. PAN, H. NICOLAS et A. THITE. 2018, «Jakobovski/free-spoken-digit-dataset : v1.0.8», . [70](#)

OYANA, T. J., L. E. ACHENIE et J. HEO. 2012, «The new and computationally efficient mil-som algorithm : potential benefits for visualization and analysis of a large-scale high-dimensional clinically acquired geographic data», *Computational and mathematical methods in medicine*. [64](#)

PÖLZLBAUER, G. 2004, *Survey and comparison of quality measures for self-organizing maps*, na. [71](#)

# Chapitre 4

## Vision événementielle et attention

« *On ne voit que ce que l'on regarde.* »

---

Maurice Merleau-Ponty

### Sommaire

---

<b>4.1</b>	<b>Introduction</b>	<b>82</b>
<b>4.2</b>	<b>Détection de mouvements</b>	<b>84</b>
4.2.1	Multi cibles	85
4.2.2	Paramètres	86
<b>4.3</b>	<b>Combinaison avec la détection de nouveauté</b>	<b>87</b>
<b>4.4</b>	<b>Mécanisme attentionnel</b>	<b>89</b>
4.4.1	DNF pour l'attention	89
4.4.2	Proposition d'une séquence de détection	90
4.4.3	Évolutions possibles	92
<b>4.5</b>	<b>Références</b>	<b>92</b>

---

## 4.1 Introduction

L'objectif de ce chapitre est d'augmenter le niveau d'abstraction de notre modèle en combinant un mécanisme attentionnel afin de réduire les coûts calculatoires. Ce chapitre regroupe les travaux que nous avons effectués sur l'utilisation de caméras événementielles, expliquées dans le prochain paragraphe, en combinaison avec des champs neuronaux dynamiques (DNF) que nous avons présentés dans le premier chapitre (1.2.4). Nous présenterons les différents mécanismes que nous avons développés et mis en œuvre jusqu'à notre système attentionnel. Celui-ci combine les modèles présentés dans ce chapitre et la détection de nouveauté du chapitre 2.

A la différence des caméras classiques qui acquièrent des images à une certaine cadence proche de la milliseconde, les capteurs événementiels, ou DVS (*Dynamic vision sensor*) détectent au niveau des pixels tout changement de luminosité de façon asynchrone à une résolution temporelle avoisinant la microseconde. L'information transmise n'est pas une image, mais uniquement des évènements asynchrones qui contiennent la position du pixel, l'horodatage et la polarité du changement (+1 si la luminosité augmente; -1 si elle diminue). Les DNF que nous avons utilisés ayant eux un taux de rafraîchissement fixe, nous avons intégré ces évènements sur ce pas de temps et générant une image intégrant tous les changements qu'il y a eu dans cet intervalle. C'est un capteur qui est plus proche conceptuellement de la perception humaine que les caméras classiques. Il a aussi l'avantage d'avoir une très grande plage dynamique et possède un temps de latence très faible. [GALLEGOT et collab. \[2019\]](#)

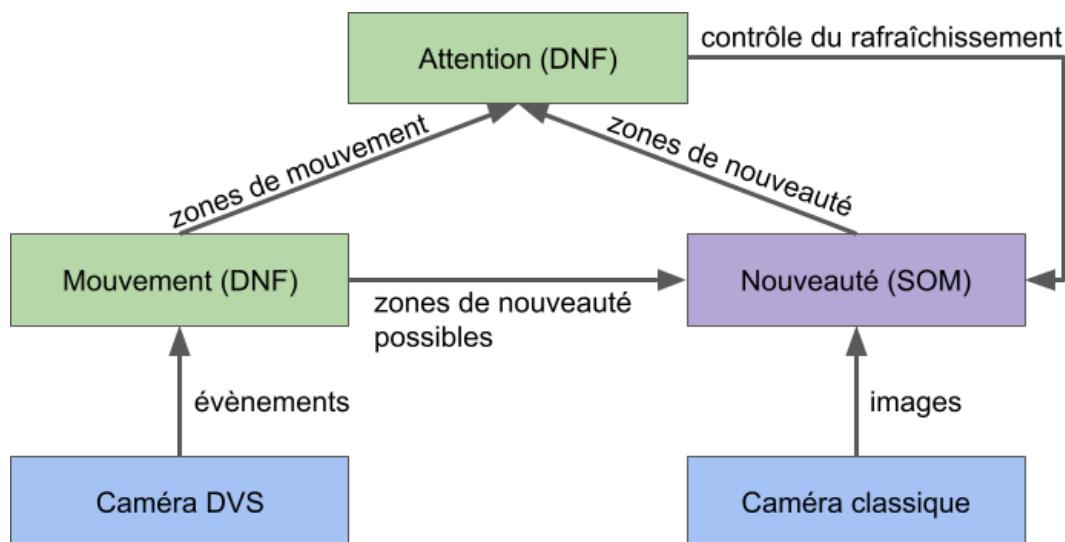


FIGURE 4.1 – Architecture de la combinaison de modèles. Ce chapitre décrit les modèles de détection de mouvement et d'attention (ici représentés en vert), ainsi que leurs interactions de nouveauté.

Les jeux de données bi-modaux de caméras DVS et à trame classique sont encore très rares. De plus, il n'y a pas à notre connaissance, de jeux de données adaptés à notre scénario de détection de nouveauté. Il est possible de recréer des évènements d'après des images [Hu et collab. \[2021\]](#), mais il est nécessaire de disposer d'une source vidéo acquise à une cadence minimale de 100 images/s. Comme nous

disposons d'une caméra événementielle, nous avons combiné celle-ci avec une caméra classique en les synchronisant logiciellement pour réaliser des acquisitions en extérieur. Les sections suivantes utilisent des images et évènements provenant de ces captures. Elles nous permettent d'évaluer la faisabilité du couplage de nos modèles sur des images standard ainsi que sur les signaux événementiels.



FIGURE 4.2 – Montage des deux caméras pour filmer la même scène. La DVS est en bas. La synchronisation temporelle a été faite logiciellement.

## 4.2 Détection de mouvements

Le rôle de la première couche pour traiter les événements de la caméra DVS est de transformer les informations ponctuelles dans le temps et l'espace en zones d'activités continues dans l'espace et le temps, pour faciliter le travail des couches supérieures. Pour ce faire, nous utilisons des DNF, particulièrement appropriés par leur capacité à intégrer de l'information dans le temps, et de leur robustesse au bruit très présent dans ce type de caméra, dû aux nombreuses variations locales de luminosité dans l'environnement. Habituellement, on utilise des DNF à deux dimensions pour gérer des images. Cependant dans le cas d'un modèle rétinotopique avec notre caméra DVS en  $640 \times 480$  pixels, le DNF nécessiterait 307 200 neurones, tous connectés dépendants les uns des autres, et devant être recalculés à chaque itération. Afin de réduire la complexité et le coût calculatoire, nous avons choisi d'utiliser deux DNF à une dimension, un pour chaque axe.

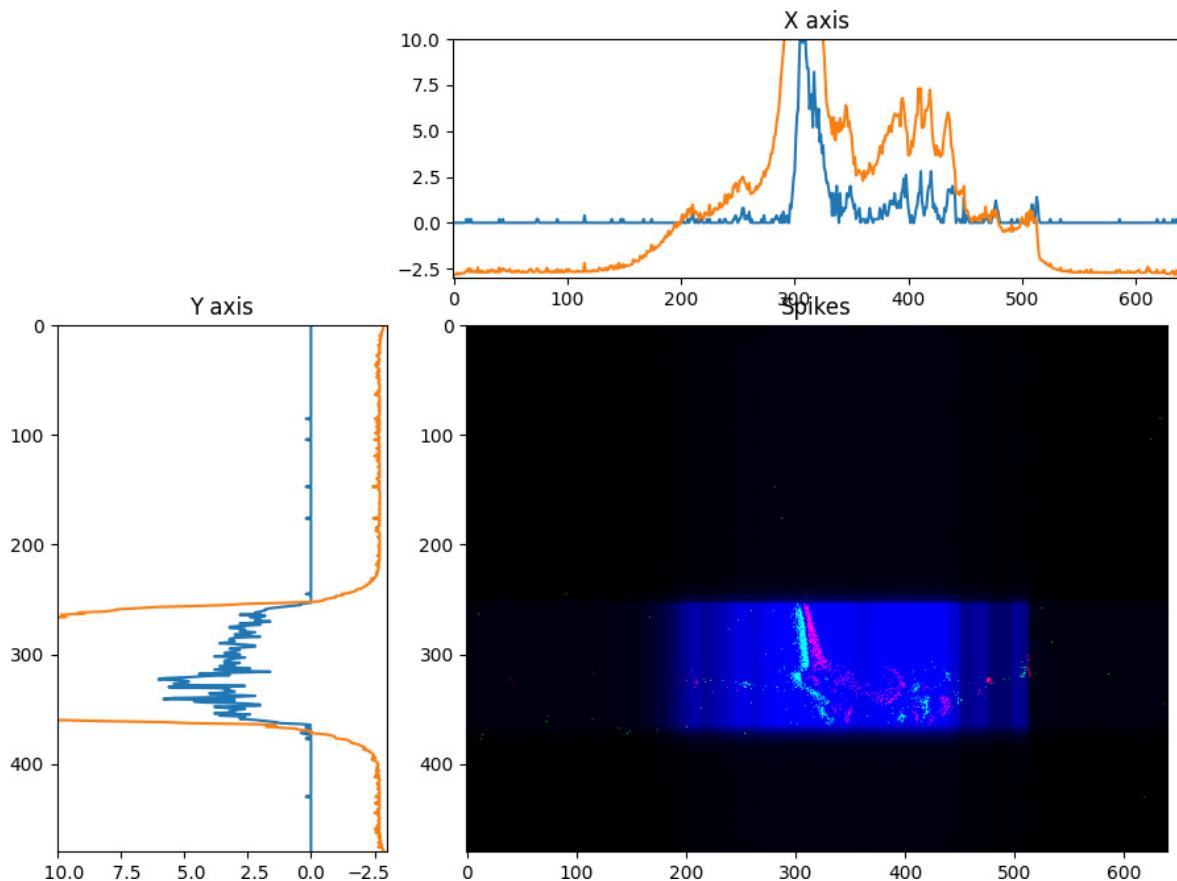


FIGURE 4.3 – Détection de mouvements avec DNF sur caméra événementielle. Les points verts et rouges dans l'image correspondent respectivement aux augmentations et réductions de luminosités détectées par la caméra. La somme des événements de chaque axe est affichée dans les courbes bleues qui sont en entrée des deux DNF. En orange sont les potentiels des DNF. On calcule le produit dyadique de la sortie des DNF, c'est à dire après la sigmoïde, pour obtenir la zone de détection de mouvement en bleu sur l'image.

Le DNF que nous avons choisi utilisant un pas de temps discrétilisé (1.2.4), il nous a donc fallu intégrer les évènements avec un pas de temps que nous avons mis à 10 ms. Il est aussi possible de le placer à 1 ms ou en dessous, si l'on souhaite détecter

des mouvements très rapides. Il y aura juste plus de mises à jour par seconde des potentiels du DNF, et donc plus de calculs. Nous n'avons pas utilisé la polarité des événements, uniquement si un pixel détectait un changement ou pas.

Les événements que nous recevons de la caméras sont très bruités. Il existe notamment certains pixels défectueux qui envoient en permanence des événements. Il s'agit de bruit artificiel dû à la technologie et non présent dans la nature. Il n'est donc pas particulièrement intéressant pour nous de l'inclure dans notre étude. Nous avons donc désactivé logiciellement ces pixels défectueux. Le bruit présent dans la nature et plus aléatoire, a lui été conservé. Le résultat est présenté sur la figure 4.3.

#### 4.2.1 Multi cibles

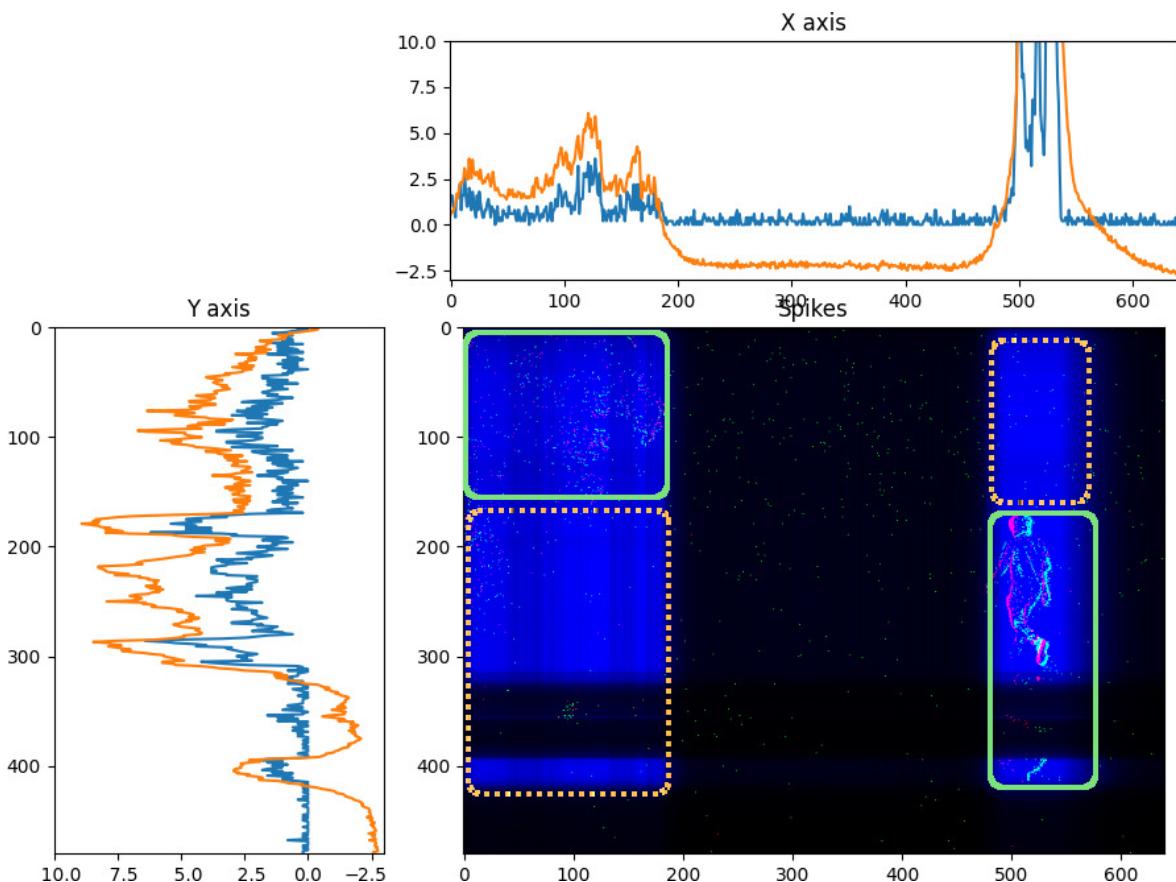


FIGURE 4.4 – Détection multicybes. Il y a deux stimulus dans la scène, entourés par des lignes vertes. La personne qui se déplace en bas à droite et un mouvement dans un arbre en haut à gauche. Les activations parasites, dû à la projection de deux détections en 1D vers du 2D sont entourés par des pointillés orange.

Notre système est également capable de suivre de multiples cibles, mais au prix de détections parasites. En effet, passer de 2 dimensions à 2 fois 1 dimension pour ensuite projeter à nouveau en 2 dimensions est efficace en calculs, mais il y a une perte d'informations dans le processus. Lorsque l'on passe en 1 dimension, on ne connaît que les valeurs sur chaque axe des deux stimulus, sans pouvoir les associer entre elles. Un exemple est présenté sur la figure 4.4.

## 4.2.2 Paramètres

Le paramétrage de DNF est une tâche délicate. Nous avons ajusté à la main les valeurs de chaque paramètre. Bien que nous n'ayons pas beaucoup modifiés les paramètres entre les deux exemples que nous avons présentés, il est possible qu'une séquence nouvelle ne fonctionne pas avec les paramètres que nous avons sélectionnés, et qu'il faille reparamétriser les DNF pour que cela fonctionne comme attendu. Une étude paramétrique sera nécessaire pour approfondir et améliorer leur généralité d'emploi. Les deux DNF de chaque axe utilisent les mêmes paramètres. Chaque événement compte pour 0.2 dans l'entrée des DNF pour une meilleure visualisation sur les figures, car mettre cette valeur à 1 et diviser le gain par 5 aurait eu le même effet sur le DNF.

TABLEAU 4.1 – Paramètres DNF 1D

Paramètre	Figure 4.3	Figure 4.4
$\delta t$	0.01	0.01
$\tau$	0.05	0.05
Coef. excitateur	4.5	4.5
$\sigma$ excitateur	0.01	0.01
Gain	4	1.5
Repos $h$	-3	-3

### 4.3 Combinaison avec la détection de nouveauté

La détection de mouvement présentée dans la section précédente a des points communs avec la détection de nouveauté de la SOM. En effet, chaque cible détectée par la détection de nouveauté sera également détectée par la détection de mouvement. Car la nouveauté ne peut apparaître qu'avec du mouvement. L'inverse n'est cependant pas vrai (il peut y avoir du mouvement sans nouveauté), et ce principe ne compte que lors de l'apparition de nouveauté. Si de la nouveauté s'arrête de se déplacer, elle n'apparaîtra plus comme mouvement, mais sera toujours de la nouveauté.

La détection de mouvement étant beaucoup plus rapide que la détection de nouveauté, nous pouvons utiliser celle-ci comme déclencheur de la détection de nouveauté dans une zone particulière de l'image. Ainsi, la détection de nouveauté ne sera appelée que lorsque du mouvement apparaîtra dans l'image, et limitera le calcul de la SOM à une zone d'intérêt dans l'image. Un exemple est montré sur la figure 4.5.

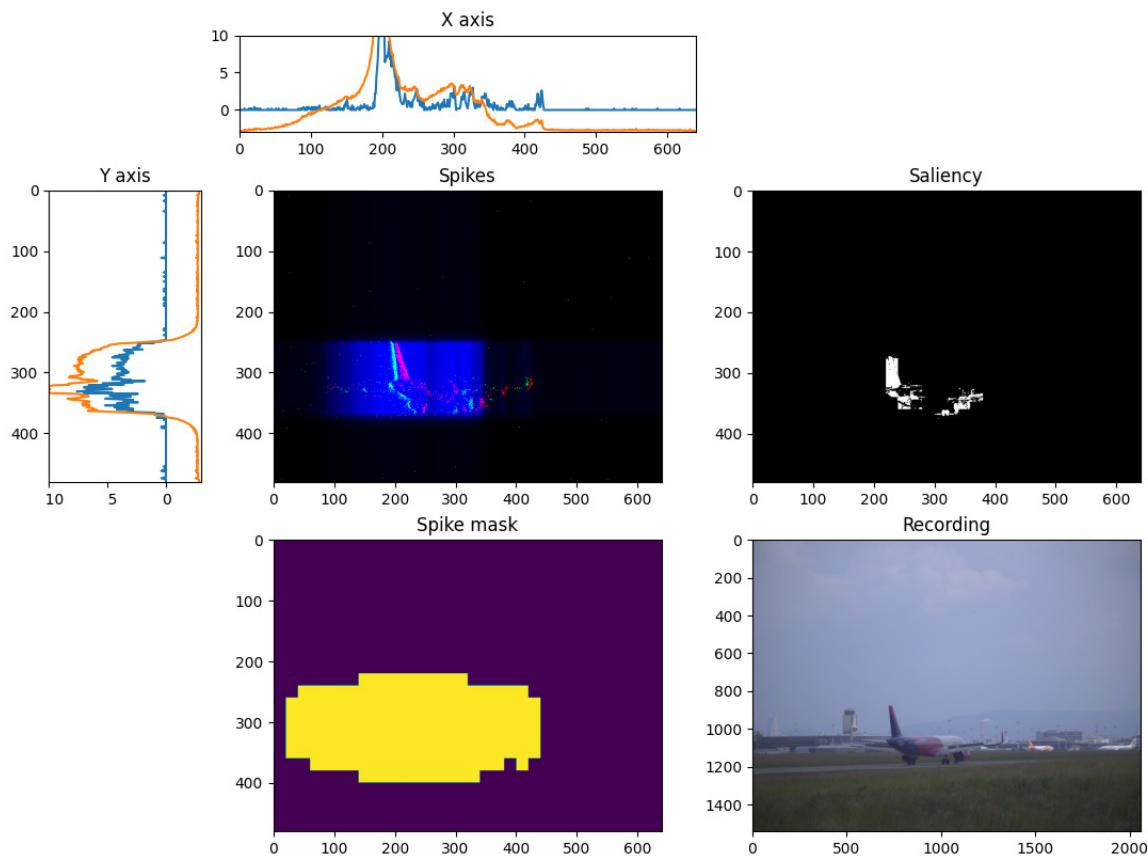


FIGURE 4.5 – Un exemple de combinaison possible entre une caméra événementielle et la détection de nouveauté avec une SOM. Les DNF travaillant sur les entrées événementielles permettent la création d'une zone d'intérêt (image en bas à gauche). Cette zone d'intérêt sert à concentrer les efforts de la SOM là où de la nouveauté a des chances d'apparaître.

Le masque est obtenu en appliquant un seuil sur la combinaison des deux DNF par le produit dyadique. La valeur du seuil a été choisie pour englober la zone d'intérêt le plus possible sans dépasser au delà. Un seuil trop bas a pour conséquence qu'une activation d'un seul DNF suffira à dépasser le seuil, et ainsi la zone d'intérêt formera une "croix" avec pour centre la "vraie" zone d'intérêt.

TABLEAU 4.2 – Impacts du modèle étendu SOM-DNF par rapport au modèle SOM sur des séquences que nous avons acquises en extérieur : faible impact sur le nombre de détections mais réduction importante du nombre d'imagettes traitées. Il n'y a pas beaucoup de nouveauté qui est perdue (83% de la sortie de la SOM conservée au minimum), et les pertes sont en majorité du bruit. Les réductions de coûts de calculs quand à eux peuvent aller de 2 fois plus rapide à 10 fois, en fonction de la taille du signal, et du bruit ambiant.

Séquence	Taux de positifs conservés	Taux d'imagettes évaluées
Plane	98,9%	10%
Walk	83,6%	43%
Highway	90,8%	18,3%

Le tableau 4.2 présente quelques métriques pour estimer les gains potentiels de cette méthode sur des séquences que nous avons enregistrées. Les gains dépendent fortement de la taille des objets en mouvements (plus ils sont gros, plus la SOM sera sollicitée), et du bruit ambiant. En effet, des mouvements sans nouveauté dans le fond d'une image solliciteront la SOM pour les classer en nouveauté ou non continuellement. Mais si il n'y a pas de signal et que les DNF ne s'activent pas à cause du bruit, la SOM peut être complètement ignorée et la caméra non événementielle éteinte. Cela permet donc un mode d'observation passif consommant peu d'énergie (uniquement avec caméra événementielle et DNF), qui peut automatiquement devenir actif en fonction de la scène observée.

## 4.4 Mécanisme attentionnel

Cette section présente notre modèle attentionnel avec un DNF en combinant la détection de nouveauté par images et la détection de mouvement par évènements. Cette implémentation de l'attention a pour objectif de se focaliser sur un unique stimulus à la fois dans la scène. Elle vise également à poursuivre en continu la cible sélectionnée dans le champ visuel.

### 4.4.1 DNF pour l'attention

Des modèles attentionnels ont déjà été employés à partir d'entrée événementielles directement [EVANUSA et collab. \[2019\]](#). Nous souhaitons plutôt placer notre attention au niveau d'abstraction le plus élevé de notre modèle, c'est à dire après la détection de mouvement des DNF et la détection de nouveauté de la SOM.

Pour produire l'attention, nous avons cette fois-ci choisi d'utiliser un DNF à deux dimensions. En effet utiliser deux DNF à une dimension dans ce cas pourrait introduire des problèmes de localisation spatiales. Si il y a plusieurs cibles dans la scène visuelle, il est possible que les deux DNF ne sélectionnent pas la même. Par exemple si il y en a une étalée horizontalement et une autre étalée verticalement et toutes les deux ont une aire similaire, le DNF de l'axe  $x$  se focalisera préférentiellement sur la cible verticale et le DNF de l'axe  $y$  sur la cible horizontale, car les points seront plus denses lorsque projetés sur ces axes, et les neurones correspondants seront donc plus activés que ceux de l'autre cible et auront plus de chance de générer une bulle d'activité. Si les deux DNF ne produisent pas la bulle d'activité pour la même cible, alors il est possible que la combinaison des deux déclenche une attention dans une zone vide de la scène.

Utiliser un DNF à deux dimensions permet d'éviter cette situation. Mais le nombre de neurones augmente en conséquence, ainsi que les coûts en calculs avec. Nous avons donc choisi de mettre un seul neurone pour couvrir une zone de 20 par 20 pixels. Pour notre entrée en 640 par 480, cela donne un DNF de 32 par 24 neurones, soit 768 neurones, le même ordre de grandeur que deux DNF à une dimension auraient donné. La perte de précision ainsi introduite n'est pas gênante, car l'objectif ici n'est que de trouver une zone d'attention, pas d'entourer la forme de la nouveauté avec précision. L'entrée de notre DNF consiste en une somme de la carte seuillée de la détection de nouveauté et du produit dyadique des deux DNF 1D de la détection de mouvement. Nous avons paramétré notre DNF de telle sorte qu'il faille un stimulus à la fois dans la détection de nouveauté et dans la détection de mouvement pour créer un blob attentionnel, et que celle-ci puisse se maintenir avec une seule des deux modalités. Si par exemple l'objet arrête de se déplacer par exemple, alors la bulle attentionnelle restera fixée sur lui grâce à la détection de nouveauté uniquement.

La paramétrisation d'un DNF pour la sélection est différente de la version multicibles des DNF 1D. Les paramètres sont montrés dans le tableau 4.3. Nous avons réduit la valeur de  $\tau$  pour rendre le DNF plus réactif aux changements. Nous avons ajouté de l'inhibition pour permettre la formation d'un blob de taille à peu près fixe autour du stimulus au centre de l'attention et ajusté la valeur de l'excitation en conséquence. La valeur de repos a été augmenté à -1 pour permettre à un blob d'activité d'apparaître plus facilement, et nous avons activé le mécanisme d'inhibition globale

qui limite le DNF à l'apparition d'une seule bulle.

TABLEAU 4.3 – Paramètres du DNF 2D attentionnel

Paramètre	Valeur
$\delta t$	0.01
$\tau$	0.02
Coef. excitateur	8
$\sigma$ excitateur	0.01
Coef. inhibiteur	5
$\sigma$ inhibiteur	0.1
Gain	1.5
Repos $h$	-1
Inhibition globale $gi$	400

La figure 4.6 présente un exemple d'attention sur une séquence capturée au-dessus d'une autoroute.

#### 4.4.2 Proposition d'une séquence de détection

Avec notre modèle attentionnel actuel, nous pouvons organiser la dynamique d'un mécanisme de détection embarqué. La séquence de détection pourrait se dérouler de la façon suivante :

- Le système est en "veille active". Seule la caméra évènementielle basse consommation est active, ainsi que la détection de mouvement par DNF.
- Un nouvel objet apparaît produisant un grand nombre d'évènements détectés par les DNF. Ils activent la caméra standard qui prend une photographie de son champ visuel. À partir de cette photo la détection de nouveauté est réalisée à l'endroit où se trouve l'objet'.
- Si aucune nouveauté n'est trouvée à cet endroit, alors l'attention ne s'active pas. La détection de mouvement continue de suivre la cible, et une détection de nouveauté est re-exécutée à une fréquence réduite.
- Si une nouveauté a été trouvée, alors la combinaison de mouvement et de nouveauté active le mécanisme attentionnel qui se focalise sur cette cible et suit et enregistre son déplacement. Une détection de nouveauté est réalisée à une fréquence rapide, et des mécanismes cognitifs plus avancés et coûteux en calculs peuvent être lancés sur la zone attentive.
- Lorsque l'objet disparaît, l'attention s'estompe également et le système retourne en "veille active".

Un tel fonctionnement dans un système embarqué permet des économies considérables en temps de calculs et en énergie et offre une plus grande autonomie du système.

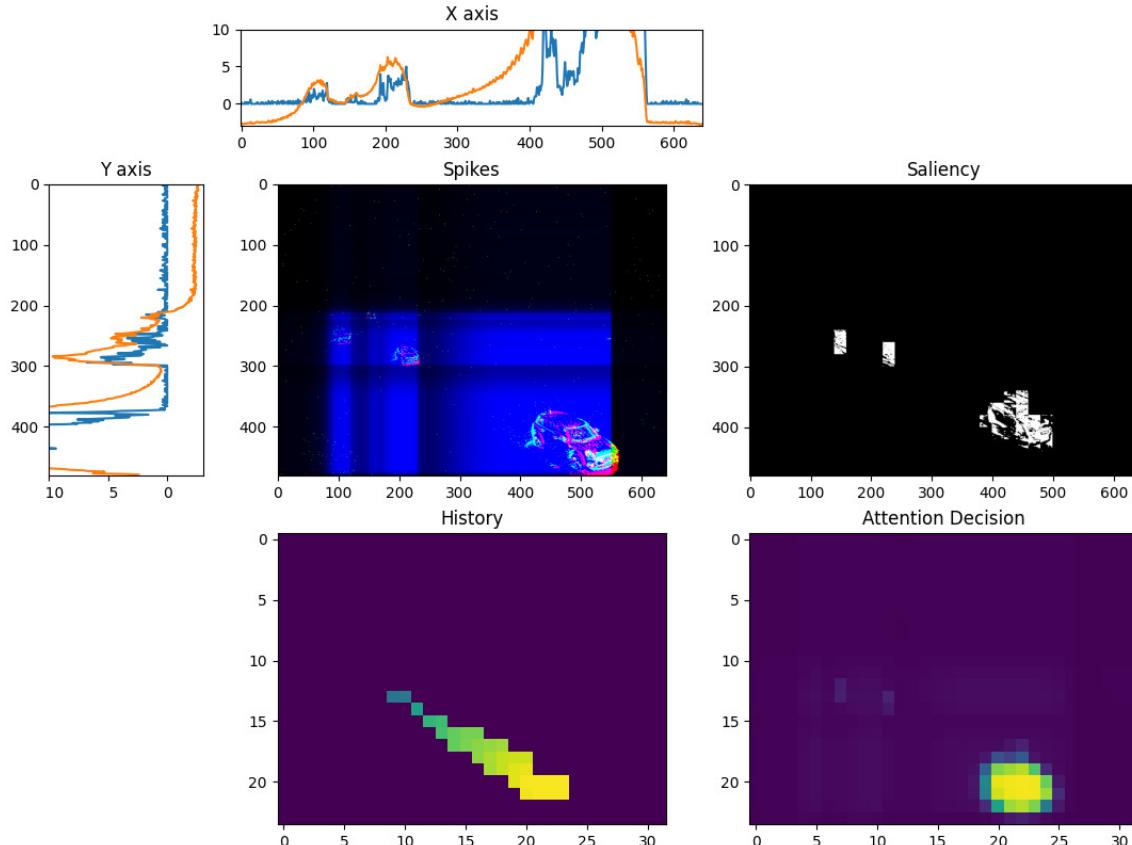


FIGURE 4.6 – Un exemple d’attention avec un DNF. Le DNF attentionnel choisi parmi les 3 voitures dans l’entrée celle qui produit le plus de saillance, c’est à dire la plus visible dans la détection de nouveauté et qui produit le plus d’événements. Ici, il s’agit de la voiture la plus proche. La zone d’attention est montrée dans la figure en bas à droite. Le suivi de la trajectoire, depuis la création du blob attentionnel jusqu’à sa position actuelle est présentée dans la graphe en bas à gauche. On peut aussi remarquer que la détection de nouveauté et la caméra évènementielle placent la voiture à deux endroits légèrement différents. Cela est dû au fait que la caméra évènementielle est plus rapide et elle a détecté la progression du véhicule depuis la dernière image capturée par la caméra standard. Les données provenant de deux caméras différentes, les variations dans l’angle de vue et dans l’objectif amènent également à des différences dans la position des objets.

### 4.4.3 Évolutions possibles

Nous présentons ici quelques pistes d'améliorations que l'on pourrait apporter à l'attention.

- La capacité de suivre plusieurs cibles. Cela peut-être réalisé en utilisant deux DNF attentionnels identiques, avec une connexion inhibitrice entre eux. Ainsi lorsque l'un d'eux choisit une cible. L'inhibition empêche l'autre DNF de choisir la même. Il se tournera donc vers la seconde cible la plus proéminente. Utiliser deux mécanismes d'attention en simultané nous éloigne de l'inspiration biologique (les humains n'ont qu'une seule attention), mais cela peut être utile pour certaines applications.
- L'attention peut être, comme chez l'homme, la base de processus cognitifs plus complexes. Dans notre exemple, on pourrait imaginer un mécanisme capable d'identifier les voitures qui serait activé que sur les objets sur lesquels le système est attentif. Mais aussi des choses plus simples comme compter le nombre de voitures rouges empruntant cette autoroute. Le tout uniquement de façon neuromorphique.
- Une autre inspiration biologique serait d'ajouter une mémoire à court terme inhibitrice de l'attention. Cela permettrait d'effectuer une exploration séquentielle des différentes cibles présentes dans la scène, comme par exemple dans [FIX et collab. \[2011\]](#).

## 4.5 Références

EVANUSA, M., Y. SANDAMIRSKAYA et collab.. 2019, «Event-based attention and tracking on neuromorphic hardware», dans *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, p. 0–0. [89](#)

Fix, J., N. ROUGIER et F. ALEXANDRE. 2011, «A dynamic neural field approach to the covert and overt deployment of spatial attention», *Cognitive Computation*, vol. 3, n° 1, p. 279–293. [92](#)

GALLEGÓ, G., T. DELBRUCK, G. ORCHARD, C. BARTOLOZZI, B. TABA, A. CENSI, S. LEUTENEGGER, A. DAVISON, J. CONRADT, K. DANILIDIS et collab.. 2019, «Event-based vision : A survey», *arXiv preprint arXiv:1904.08405*. [82](#)

Hu, Y., S.-C. LIU et T. DELBRUCK. 2021, «V2e : From video frames to realistic dvs events», dans *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, p. 1312–1321. [82](#)

# Conclusion

Cette thèse avait pour objectif l'utilisation de modèles neuronaux d'apprentissage non supervisés, notamment les cartes auto-organisatrices (SOM) et les champs neuronaux dynamiques (DNF), pour explorer des scènes visuelles depuis un système de vision embarqué. Nous y avons répondu en découplant l'objectif en trois sous problématiques : comment utiliser les propriétés de chaque modèle pour des tâches visuelles, comment optimiser leur implémentation matérielle dans un système embarqué et comment combiner ces modèles pour augmenter leur capacités

La première problématique m'a amené à montrer qu'il était possible d'utiliser, en plus de la quantification vectorielle, les propriétés d'émergence des cartes auto organisatrices. Nous avons ainsi développé un nouveau modèle de détection de nouveauté dans des images combinant deux méthodes, une utilisant la quantification vectorielle des SOM, et une seconde qui utilise la continuité topologique. Ce modèle a été évalué sur le jeu de données CDNET, qui est un standard dans le domaine de la détection de changements. Bien que notre modèle ne soit pas encore en mesure de concourir efficacement avec l'état de l'art, que ce soit les méthodes algorithmiques classiques ou l'apprentissage profond, il a mis en avant les avantages à utiliser l'émergence dans les modèles neuronaux.

En effet, ce qui différencie les k-means des cartes auto-organisatrices, c'est la topologie, qui est la propriété émergente des SOM. La continuité topologique des SOM nous a ainsi permis de créer la détection de nouveauté par distance topologique et de réduire significativement les coûts en calculs des SOM en développant un algorithme de recherche de la BMU plus rapide grâce à cette information de continuité topologique. L'émergence a ainsi pu apporter dans notre cas des gains en performance et en coût calculatoire, et nous pensons que généraliser l'utilisation de celle-ci pourrait également amener à des bénéfices similaires pour d'autres problèmes, de la vision ou d'analyse de signaux, et à d'autres modèles avec des propriétés émergentes, comme les *Growing neural cellular automata* MORDVINTSEV et collab. [2020] par exemple.

Pour ce qui est de la détection de nouveauté, il existe de nombreuses pistes d'améliorations. La piste la plus prometteuse pour nous serait d'ajouter une capacité de représentation plus complexe de l'environnement, pour permettre de mieux généraliser le fond appris. Le principal problème actuellement réside dans l'utilisation de la distance euclidienne pour l'apprentissage de la SOM. Celle-ci ne prend pas en compte les relations entre les pixels, et nous pensons que pour avoir des distances représentatives de la différence visuelle entre deux images, il sera nécessaire de les apprendre avec un autre réseau de neurones, qui serait utilisé comme évaluation de la distance entre les entrées et les prototypes de la SOM. Une autre piste d'amélioration est rem-

placer la SOM par un autre modèle de quantification vectorielle avec une topologie émergente. Nous avons montré que notre approche était généralisable en l'appliquant à des gaz neuronaux en expansion (GNG), il est donc possible que parmi tous les variants des SOM et des GNG certains puissent être meilleurs pour cette tâche, de par leur topologie, ou leur apprentissage particulier. On peut par exemple imaginer utiliser des *Incremental neural gases* PRUDENT et ENNAJI [2005], qui est capable d'apprendre des nouvelles données sans oublier les précédentes, pour produire un système qui apprend tout au long de sa vie. Il pourrait effectuer un réapprentissage à un intervalle régulier, ou lorsqu'il détecte un changement global dans le fond, pouvant être dû à un changement de météo par exemple.

Nous avons présenté pour la seconde problématique des méthodes pour réduire le coût en calculs de la SOM et de notre modèle pour leur utilisation dans les systèmes embarqués.

La première est FastBMU, un algorithme pour calculer la BMU plus rapidement en s'appuyant sur la propriété de réduction dimensionnelle des SOM. Nous avons montré une version séquentielle, utilisable dans des applications sur ordinateur et une version parallèle pour des systèmes embarqués sur FPGA par exemple. FastBMU est significativement plus rapide que la version exhaustive (jusqu'à 80% de calculs en moins pour des cartes de 1250 neurones par exemple), en réduisant sa classe de complexité, tout en ayant des performances en quantification vectorielle très proches de l'approche standard, parfois même améliorant le résultat. Nous travaillons désormais sur son implantation matérielle pour en montrer la faisabilité et en mesurer les performances en conditions réelles. Cet algorithme est encore améliorable car il existe toujours des redondances dans le calcul du gradient dans FastBMU. Une première piste d'amélioration pourrait utiliser un pas de plusieurs neurones pour effectuer la descente de gradient pourrait amener au même BMU en ayant moins de neurones à évaluer. Une seconde piste serait abandonner l'approche cellulaire, et essayer de trouver le gradient en évaluant au hasard des neurones dans la SOM et en recombinant leurs gradients locaux respectifs.

La seconde avec l'emploi de DNF et d'une caméra impulsionale pour localiser les zones de l'image où la détection de nouveauté est possible et ainsi, réduire le nombre de reconstructions nécessaires.

Enfin, pour la troisième problématique de réalisation d'un modèle étendu, nous sommes partis de notre détection de nouveauté travaillant sur des images, et nous lui avons adjoint une détection de mouvement par Champ neuronaux dynamiques, travaillant sur des événements d'une caméra DVS (*Dynamic vision sensor*). Les jeux de données contenant à la fois des événements et des images étant rares, nous avons effectué nos propres captures, et avons pu montrer que l'on pouvait focaliser la détection de nouveauté uniquement sur une zone réduite de l'image sans perte de qualité grâce à la détection de mouvements et à la caméra évènementielle. Nous avons complémenté ce modèle par un mécanisme attentionnel simple, s'appuyant à la fois sur la nouveauté et le mouvement. Cela lui permet de suivre en continu la nouveauté en déplacement rapide grâce à la détection de mouvement de faible latence, et de rester actif sur de la nouveauté qui s'est arrêtée et qui ne génère plus d'événements dans la caméra évènementielle.

Nous avons observé que la combinaison de modèles a été un moyen efficace

pour augmenter les capacités de notre système neuromorphique. La SOM en isolation n'a pas de concepts de temporalité et les DNF ne peuvent pas apprendre des stimulus aussi complexes que des images. Mais lorsque l'on combine les deux, le système global compense les défauts de chaque modèle. On obtient une détection de nouveauté qui peut être appelée seulement dans les moments où cela est nécessaire. Mais aussi un DNF prenant en compte de la nouveauté dans une image dans son mécanisme attentionnel, ce qu'il serait incapable de faire en isolation, car il ne pourrait dépendre que du mouvement ou d'une modalité comme une couleur spécifique choisie lors de sa programmation. La première piste d'évolution de cet assemblage de modèles est l'amélioration de l'attention. En effet, il est possible de la modifier pour effectuer une recherche visuelle parmi plusieurs stimulus **Fix et collab.** [2011] par exemple. Une seconde piste d'amélioration est d'augmenter le niveau d'abstraction. On peut par exemple ajouter un mécanisme de catégorisation des stimulus observés capable de reconnaître des objets déjà vus par exemple. En restant dans le domaine du non-supervisé, il serait possible d'ajouter un réseau *Incremental neural gas* après l'attention, et qui, contrairement à la SOM de la détection de nouveauté, recevrait en entrée uniquement les objets nouveaux. La catégorisation non-supervisée a déjà été faite pour des SOM par exemple **KHACEF et collab.** [2019], et pourrait être transposable aux autres modèles de quantification vectorielle topologiques.

## Références

- Fix, J., N. ROUGIER et F. ALEXANDRE. 2011, «A dynamic neural field approach to the covert and overt deployment of spatial attention», *Cognitive Computation*, vol. 3, n° 1, p. 279–293. [95](#)
- KHACEF, L., B. MIRAMOND, D. BARRIENTOS et A. UPEGUI. 2019, «Self-organizing neurons : toward brain-inspired unsupervised learning», dans *2019 International Joint Conference on Neural Networks (IJCNN)*, IEEE, p. 1–9. [95](#)
- MORDVINTSEV, A., E. RANDAZZO, E. NIKLASSON et M. LEVIN. 2020, «Growing neural cellular automata», *Distill*, doi :10.23915/distill.00023. [Https://distill.pub/2020/growing-ca](https://distill.pub/2020/growing-ca). [93](#)
- PRUDENT, Y. et A. ENNAJI. 2005, «An incremental growing neural gas learns topologies», dans *Proceedings. 2005 IEEE International Joint Conference on Neural Networks*, 2005., vol. 2, IEEE, p. 1211–1216. [94](#)

# Publications

BERNARD, Y., E. BUOY, A. FOIS et B. GIRAU. 2018, «NP-SOM : network programmable self-organizing maps», dans *2018 IEEE 30th International Conference on Tools with Artificial Intelligence (ICTAI)*, IEEE, Volos, Greece, p. 908–915, doi :10.1109/ICTAI.2018.00141. URL <https://hal.inria.fr/hal-02058458>.

BERNARD, Y., N. HUEBER et B. GIRAU. 2019, «Novelty detection with self-organizing maps for autonomous extraction of salient tracking features», dans *13th International Workshop on Self-Organizing Maps and Learning Vector Quantization, Clustering and Data Visualization*, Barcelona, Spain, doi :10.1007/978-3-030-19642-4\\_10. URL <https://hal.archives-ouvertes.fr/hal-02156627>.

BERNARD, Y., N. HUEBER et B. GIRAU. 2020a, «A fast algorithm to find Best Matching Units in Self-Organizing Maps», dans *ICANN 2020, 29th International Conference on Artificial Neural Networks*, Bratislava, Slovakia. URL <https://hal.univ-lorraine.fr/hal-02984424>.

BERNARD, Y., N. HUEBER et B. GIRAU. 2020b, «Novelty Detection in Images Using Vector Quantization with Topological Learning», dans *ICECS 2020, 27th IEEE International Conference on Electronics Circuits and Systems*, Glasgow/Virtual, United Kingdom. URL <https://hal.univ-lorraine.fr/hal-02984427>.