Année 2025–2026

UNIVERSITÉ
DE GENÈVE

FACULTÉ DES SCIENCES
Département d'informatique

Projet de Bachelor d'informatique

SOM : Self-Organizing Maps

William Guillermo Calero Roncal

Année 2025–2026

Table des matières

Chapitre 1

Introduction

Durant les dernières décennies, plusieurs méthodes d'apprentissage automatique ont fait leur apparition. Cela s'est produit grâce à une croissance exponentielle des données récoltées avec lesquelles nous tentons de répondre à des questions dans plusieurs domaines. Plusieurs méthodes s'inspirent directement du cerveau humain et une d'entre elles se nomme les cartes auto-organisées (self organizing-maps : SOM). Cette méthode s'inspire des cartes cérébrales. Certaines cellules du cerveau répondent de manière sélective à un stimulus sensoriel spécifique et ces cellules ont tendance à s'assembler formant une topographie particulière en fonction du stimulus spécifique [10]. Les SOM permettent la projection des données de grande dimension dans une grille à petite dimension tout en essayant de préserver leur topologie locale en tenant compte de leur voisinage.

1.1 Contexte

Actuellement, les réseaux de neurones ont une place centrale dans le domaine de l'apprentissage automatique. Insipirée aussi des neurones biologiques, des cellules nerveuses dont la fonction est d'agréger et transmettre des informations via des signaux chimiques et électriques [17], ces modèles consistent à former des interconnexions entre couches de neurones artificiels.

Dans la littérature, nous observons une tendance à appeler les SOM des réseaux de neurones. Cependant, certaines différences remarquables entre les réseaux de neurones au sens du perceptron et les cartes auto-organisées existent. Afin de clairifier ces distinctions, nous allons rappeler la définition d'un neurone artificiel dans les réseaux de neurones classiques, puis nous comparerons ce modèle avec le "neurone" d'un SOM.

1.1.1 Neurone artificiel

Un neurone artificiel ou plus communément un neurone formel est une représentation d'une combinaison linéaire suivie d'une fonction d'activation [14]. Pour un neurone k, considérons $m+1$ entrées $x_0, \ldots, x_m$ et des poids associés $w_{k0}, \ldots, w_{km}$. Généralement, x_0 est une entrée de biais qui est fixée à 1. Son fonctionnement commence par une somme pondérée de ses entrées, l'activation :

$$a_k = \sum_{i=0}^{m} w_{ki} x_i$$

À cela, une fonction d'activation $\phi(.)$ [18] est appliqué pour avoir la sortie du neurone :

$$\hat{y}_k = \phi(a_k)$$

Cette valeur va devenir l'entrée de la couche suivante, sauf s'il s'agit de la dernière couche. Les réseaux de neurones, qui ont ce type de neurones et au moins une couche cachée, sont entraînés à travers une fonction de coût $\mathcal{L}$ (loss function) dont le but est de la minimiser. Cette correction est réalisée à travers un algorithme d'optimisation, tel que la descente de gradient (gradient descent), appliqué après le calcul du gradient par l'algorithme de rétropropagation (backpropagation) [14]. L'algorithme réduit les erreurs de prédiction entre $\hat{y}_k$ et y_k en ajustant les poids w_{ki}. Dans ce contexte, nous avons un apprentissage supervisé, car la fonction se repose sur la connaissance des labels y_k. Ainsi, un neurone artificiel est une analogie qui interprète mathématiquement le fonctionnement d'un neurone avec des méthodes d'apprentissage automatique (machine learning).

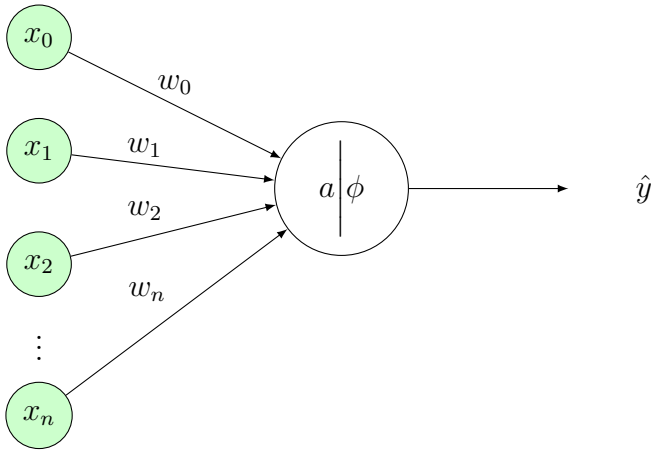

FIGURE 1.1 – Neurone formel

Dans ce contexte, le but du réseau de neurones est d'approximer une fonction de prédiction du type $y = \phi(x)$ par une fonction ϕ_θ, représentée par le réseau et paramétrée par $\theta = [w_{ij}^k]$ l'ensemble des poids.

Le théorème d'approximation universelle montre que cela est possible avec un certain niveau d'approximation [7].

1.1.2 Neurone du SOM

Par contraste, étant donné un ensemble de données $\mathcal{X} = \{x_i\}_{i=1}^N$ où $x_i \in \mathbb{R}^d$, une SOM cherche à établir une construction de l'ensemble des données ("vector quantization" cf. section 2.1.1) en associant à chaque position d'une topologie donnée (la carte) un sous ensemble de données de départ.

Dans la suite, nous aurons :

— $\mathcal{X} = \{x_1, x_2, \ldots, x_N\}$ avec $x_i \in \mathbb{R}^d$ sont l'ensemble de données d'entrée
— $\mathcal{S} = \{S_1, S_2, \ldots, S_M\}$ sont les noeuds de la carte
— Chaque noeud S_j est associé à un vecteur de poids $w_j \in \mathbb{R}^d$ (prototype)
— Chaque noeud a une position $p_j \in \mathbb{R}^D$
— Chaque noeud S_j a un voisinage N_j dans la topologie de la carte avec $S_j \in N_j$

Dans le texte, nous assimilerons le noeud a son vecteur de poids : $S_j \leftrightarrow w_j$.

Un "neurone" d'un SOM est un prototype représentant une région de l'espace des données. Il est associé à un vecteur de poids $w \in \mathbb{R}^d$, de même dimension que les données d'entrée $x \in \mathbb{R}^d$. Son vecteur de poids est mis à jour en fonction de la distance entre les données d'entrée et son prope vecteur de poids.

Les données d'entrée sont visitées itérativement. À l'itération $t+1$ pour la donnée $x_i \in \mathbb{R}^d$, le neurone S_c le plus proche de la donnée d'entrée x_i est appelé *Best Matching Unit* (BMU) dont le vecteur de poids est w_c. Le vecteur de poids w_c, ainsi que les vecteurs de poids w_j des noeuds de son voisinage, sont mis à jour en fonction de la formule suivante :

$$w_j^{(t+1)} = w_j^{(t)} + h_{cj}(t)[x_i - w_j^{(t)}]$$

Où, $h_{cj}(t)$ est la fonction de voisinage qui diminue avec la distance entre le neurone S_j et le BMU S_c, ainsi que du temps t [10] (cf. section 2.1.2).

Dans ce contexte, nous avons un apprentissage dit non supervisé et compétitif, car les "neurones" sont en compétition pour représenter les données d'entrée, et sans labels pour guider l'apprentissage. Cela est une différence fondamentale avec les neurones formels, car les SOM sont utilisés pour la visualisation et la réduction de la dimensionnalité, tandis que les neurones formels sont utilisés plutôt pour la classification et la régression. De plus, les SOM préservent la topologie des données d'entrée, c'est-à-dire qu'ils conservent au maximum les relations de voisinage dans la projection de l'espace à dimension réduite, ce qui n'est pas le sujet pour les neurones formels. Nous remarquons également une différence dans la fonction de coût, elles n'ont pas la même finalité. Les SOM utilisent une fonction de coût basée sur la minimisation d'une mesure de distance entre les données d'entrée et les prototypes, tandis que les neurones formels ont une fonction de coût basée sur l'erreur de prédiction. Donc, la fonction de coût d'une carte auto-organisées ne produit pas de manière explicite de sortie ou de prédiction.

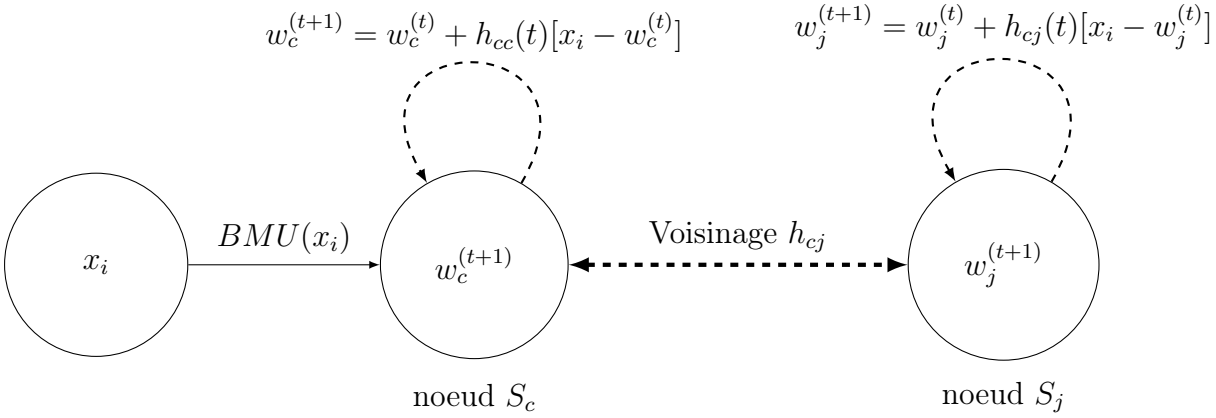

FIGURE 1.2 – Structure d'interaction de voisinage dans un SOM : mise à jour du BMU (S_c) et d'un voisin (S_j).

Même s'ils partagent une inspiration biologique commune, les SOM présentent des caractéristiques uniques qui les distinguent des réseaux de neurones traditionnels. Ainsi, même si historiquement les cartes auto-organisées sont classées comme des réseaux de neurones, par les différences que nous avons citées, il est raisonnable de les considérer comme des structures différentes.

Par la suite de ce rapport et pour éviter toutes confusions, nous allons utiliser "noeud" ou "prototypes" pour les unités de calcul d'un SOM.

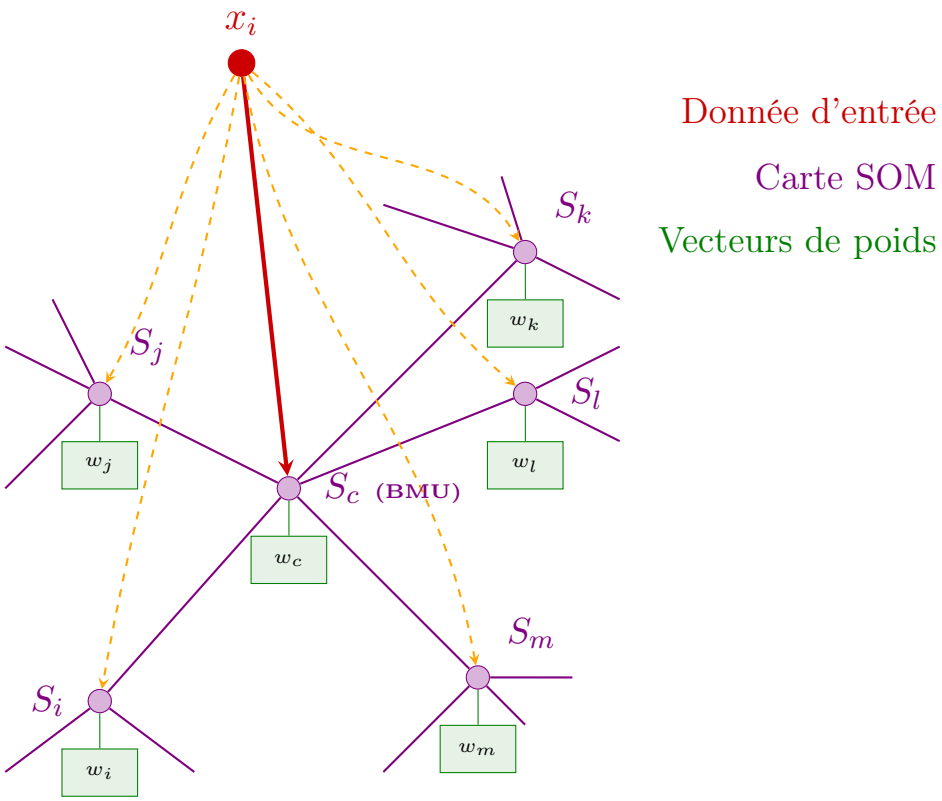

FIGURE 1.3 – Schéma illustrant la structure d'une carte auto-organisée (SOM) avec un point d'entrée x_i, les nœuds de la carte (dont le BMU S_c), les vecteurs de poids associés, et les interactions de voisinage.

1.2 Objectifs du projet

L'objectif de ce projet consiste à comprendre les cartes auto-organisées, leurs limites ainsi que les extensions proposées dans la littérature pour y remédier. Dans la suite de ce rapport, un état de l'art sur les SOM sera présenté. Ensuite, nous ferons une étude empirique en modélisant et implémentant un SOM de zéro et en comparant avec l'extension DESOM. Enfin, nous analyserons et discuterons de leurs performances, afin de mettre en évidence leurs avantages et inconvénients.

Chapitre 2

État de l'art

2.1 Self-Organizing Maps (SOM)

Les cartes auto-organisées ont été introduites par Teuvo Kohonen en 1982 [6]. Ce modèle se base sur une approche d'apprentissage compétitive. Il s'agit d'une forme d'apprentissage non supervisé où chaque nœud concurrent représente une région de l'espace des données d'entrée. Ce type d'apprentissage est utilisé notamment pour le clustering sous le nom de quantification vectorielle.

2.1.1 Quantification vectorielle (VQ)

La quantification vectorielle (VQ) est une technologie utilisée surtout dans le traitement moderne de la numérisation des signaux [10]. Elle consite à partitionner l'espace d'origine en régions distinctes et contiguës dont chacune est représentée par un vecteur. Dans cette méthode, les données d'entrée fonctionnent comme des features qui sont représentées dans un espace fini par d'autres vecteurs de même dimension. Cet ensemble de vecteurs se nomme codebook. Une VQ s'interprète de la manière suivante : Soit $x \in \mathbb{R}^d$ une donnée d'entrée et soit le codebook avec w_j vecteurs de même dimension et $w_{c(x)}$, le vecteur gagnant (apprentissage compétitif), le plus proche de x [9].

$$c(x) = \arg\min_i ||x - w_i||^2$$

Admettons que $p(x)$ soit la densité de probabilité de x sur V, alors l'erreur moyenne de la quantification $\mathcal{E}$ est définie par :

$$\mathcal{E} = \int_V ||x - w_{c(x)}||^2 p(x) dV \tag{2.1}$$

où V est l'espace d'entrée et donc dV est le volume différentiel. La fonction $\mathcal{E}$ est une fonction d'energie qui peut être minimisée par l'algorithme de descente de gradient [10]. Par simplification, nous avons pris une distance euclidienne, mais d'autres mesures de distance peuvent être utilisées. La variante que nous avons présentée est connue sous le nom de "k-means" [2].

Dans la pratique, $p(x)$ est inconnue, mais nous avons un ensemble de données $\{x_i \in \mathbb{R}^d\}_{1 \leq i \leq n}$. Pour avoir une estimation de $\mathcal{E}$, il faut approximer $p(x)$ par une distribution empirique à partir de l'ensemble de données. Donc, nous avons :

$$p(x) \approx \frac{1}{n} \sum_{i=1}^{n} \delta(x - x_i) \tag{2.2}$$

Où la fonction δ est la fonction de Dirac concentrant la densité de probabilité en un point [19] :

$$\delta(x) = \begin{cases} \infty & \text{si } x = 0 \\ 0 & \text{sinon} \end{cases}$$

La particularité de cette fonction qui nous intéresse est la suivante [19] :

$$\int_{-\infty}^{\infty} f(x)\delta(x - a)dx = f(a)$$

Suivant l'equation (2.1) et (2.2), alors nous avons, avec $f(x) = ||x - w_{c(x)}||^2$:

$$\mathcal{E} \approx \int_V f(x) \frac{1}{n} \sum_{i=1}^{n} \delta(x - x_i)dV \Leftrightarrow$$

$$\mathcal{E} \approx \frac{1}{n} \sum_{i=1}^{n} \int_V f(x)\delta(x - x_i)dV = \frac{1}{n} \sum_{i=1}^{n} f(x_i) \Leftrightarrow$$

$$\mathcal{E} \approx \frac{1}{n} \sum_{i=1}^{n} ||x_i - w_{c(x_i)}||^2$$

Ainsi, la quantification vectorielle minimise l'énergie $\mathcal{E}$ en associant chaque donnée d'entrée a son prototype (BMU) le plus proche. Les SOM suivent cette même approche d'apprentissage compétitive. Cependant, contrairement à la VQ classique, les SOM imposent une topologie sur les prototypes (la carte), de manière à ce que les prototypes voisins dans la grille soient aussi proches dans l'espace des données d'entrée.

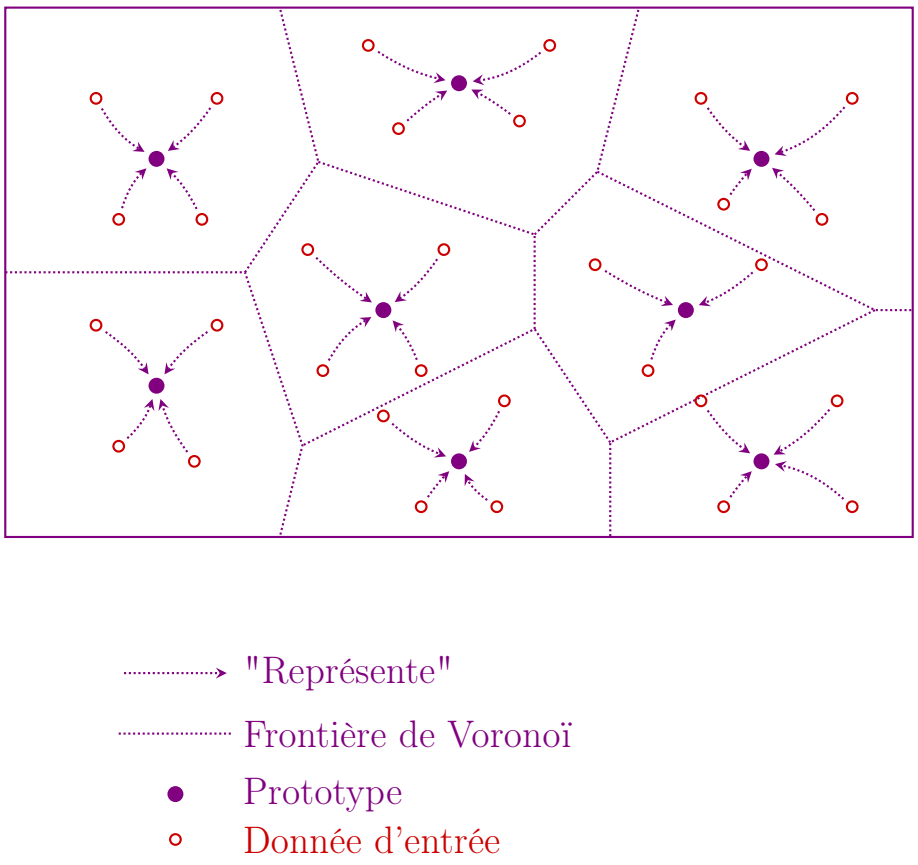

FIGURE 2.1 – Illustration schématique d'une quantification vectorielle sans contrainte (k-means) vue comme un diagramme de Voronoi : chaque prototype délimite une cellule, et chaque donnée d'entrée est associée au prototype le plus proche. Les frontières et les positions des points sont représentées à titre indicatif.

2.1.2 SOM de base

Les SOM créent une structure qui préserve la topologie locale dans un voisinage défini. Ce "calibrage" ou mise à jour des noeuds voisins dépend du Best Matching Unit(BMU), le noeud gagnant, et du temps t [9]. Les cartes auto-organisées ne dérivent pas de manière explicite d'une fonction de coût globale, mais d'une règle de mise à jour locale qui est appliquée à chaque noeud en fonction de sa proximité avec le BMU.

Soit un ensemble de données d'entrée $\mathcal{X} = \{x_i \in \mathbb{R}^d\}_{1 \leq i \leq n}$ et un codebook $\{w_j \in \mathbb{R}^d\}_{1 \leq j \leq m}$. Le BMU suit le même principe que celui de la VQ, c'est à dire le noeud le plus proche d'un certain x :

$$c(x) = \arg\min_j ||x - w_j||^2$$

$$BMU(x) = w_{c(x)}$$

L'entier $c(x)$ représente l'indice du noeud gagnant pour la donnée x. En partant de la quantification vectorielle, l'équation de mise à jour d'un noeud w_j est la suivante [9] :

$$w_j^{(t+1)} = \begin{cases} w_j^{(t)} + \alpha(t)[x - w_j^{(t)}] & \text{si } j = c(x) \\ w_j^{(t)} & \text{sinon} \end{cases}$$

Où $\alpha(t)$ est le taux d'apprentissage qui diminue avec le temps t. Nous observons que dans cette formule, seul le noeud gagnant est mis à jour, il est "attiré" vers la donnée d'entrée. Dans le VQ classique, seulement le noeud gagnant est mis à jour et sans interaction avec ses voisins. Cependant, pour imposer la topologie de la carte dans les SOM, les noeuds voisins du BMU sont aussi mis à jour.

Admettons l'ensemble de noeuds voisins du BMU $N_c(t)$ qui décroit avec le temps, alors la formule de mise à jour devient [9] (rappelons que $w_{c(x)} \in N_c(t)$) :

$$w_j^{(t+1)} = \begin{cases} w_j^{(t)} + \alpha(t)[x - w_j^{(t)}] & \text{si } w_j \in N_c(t) \\ w_j^{(t)} & \text{sinon} \end{cases}$$

Avec cette nouvelle approche, les noeuds voisins du BMU sont aussi mis à jour. Cependant, le taux de mise à jour pour les noeuds voisins est la même que celui du BMU, ce qui peut poser des problèmes de convergence, des transitions plus brusques qui peuvent provoquer des cartes moins lisses et organisées. Donc nous dégradons la propriété la plus importante des SOM qui est la préservation de la topologie locale.

Un choix plus standard et efficace est de prendre la fonction gaussienne de voisinage $h_{cj}(t)$ qui diminue avec la distance entre le noeud w_j et le BMU $w_{c(x)}$ ainsi que du temps t [10] :

$$h_{cj}(t) = \alpha(t) e^{-\frac{||p_{c(x)} - p_j||^2}{2\sigma(t)^2}}$$

Où $p_{c(x)}$ et p_j sont les positions du BMU et du noeud w_j dans la grille. Lors de la mise à jour de $w_{c(x)}$, nous avons pour tout t :

$$h_{cc}(t) = \alpha(t)$$

La fonction $\sigma(t)$ est le calcul du rayon de voisinage qui diminue avec t et qui contrôle l'étendue de la mise à jour des noeuds voisins. Elle peut être définie de la manière suivante, avec T le nombre total d'itérations [6] :

$$\sigma(t) = \sigma_0 e^{-\frac{t}{T}}$$

Donc, la distance entre les noeuds dans la grille influence la mise à jour, plus un noeud est éloigné du BMU, moins il sera mis à jour. Cette approche permet d'avoir des cartes plus lisses et organisées avec des transitions plus douces.

Cela nous mène à la formule de mise à jour finale d'un noeud w_j dans un SOM :

$$w_j^{(t+1)} = w_j^{(t)} + h_{cj}(t)[x - w_j^{(t)}]$$

Nous pouvons aussi interpréter la mise à jour en deux termes de variation de poids. Pour le noeud gagnant (BMU) :

$$\Delta w_{c(x)}^{(t)} = h_{cc}(t)\,[x - w_{c(x)}^{(t)}] = \alpha(t)\,[x - w_{c(x)}^{(t)}]$$

Pour un noeud voisin w_j :

$$\Delta w_j^{(t)} = h_{cj}(t)\,[x - w_j^{(t)}]$$

Ainsi, la mise à jour des poids peut s'écrire de manière équivalente :

$$w_j^{(t+1)} = w_j^{(t)} + \Delta w_j^{(t)}$$

De manière plus schématique, nous avons la figure 2.2 qui illustre la mise à jour des noeuds en fonction du BMU $w_{c(x)}$ et de la distance entre la donnée x et les noeuds voisins de $w_{c(x)}$.

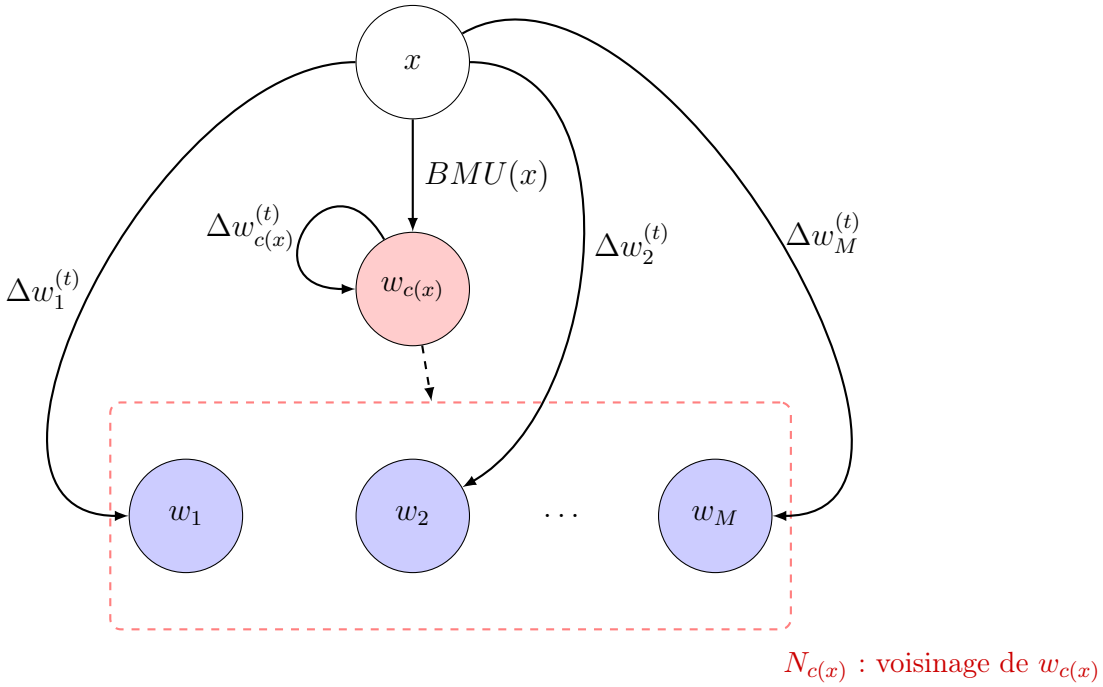

FIGURE 2.2 – Mise à jour des noeuds voisins $w_1, w_2, \ldots, w_M$ en fonction du BMU $w_{c(x)}$ et de la distance dans la grille

Cela nous donne une première idée d'algorithme d'apprentissage :

Algorithm 1 Algorithme d'apprentissage d'un SOM (stepwise)

Initialiser les poids w_j de manière aléatoire ou avec PCA
Initialiser le taux d'apprentissage $\alpha(0)$ et la largeur de voisinage $\sigma(0)$
Définir le nombre d'itérations maximum T
for $t = 1$ **à** T **do**
    Sélectionner un échantillon d'entrée x
    $c = \arg\min_j ||x - w_j^{(t)}||^2$ ▷ indice du BMU
    $\alpha(t) = \alpha(0) \cdot e^{-t/T}$ ▷ décroissance du taux d'apprentissage
    $\sigma(t) = \sigma(0) \cdot e^{-t/T}$ ▷ décroissance de la largeur de voisinage
    for tous les noeuds j **do**
        $h_{cj}(t) = \alpha(t) \cdot e^{\left(-\frac{\|c-j\|^2}{2\sigma(t)^2}\right)}$ ▷ calcul de la fonction de voisinage
        $w_j^{(t+1)} = w_j^{(t)} + h_{cj}(t) \cdot [x - w_j^{(t)}]$ ▷ mise à jour du poids du noeud j
    end for
end for

Ainsi, nous avons présenté les étapes de l'algorithme d'apprentissage d'un SOM de manière détaillée. Neanmoins, il existe un autre variante plus efficace en pratique appelée **batch SOM** [9].

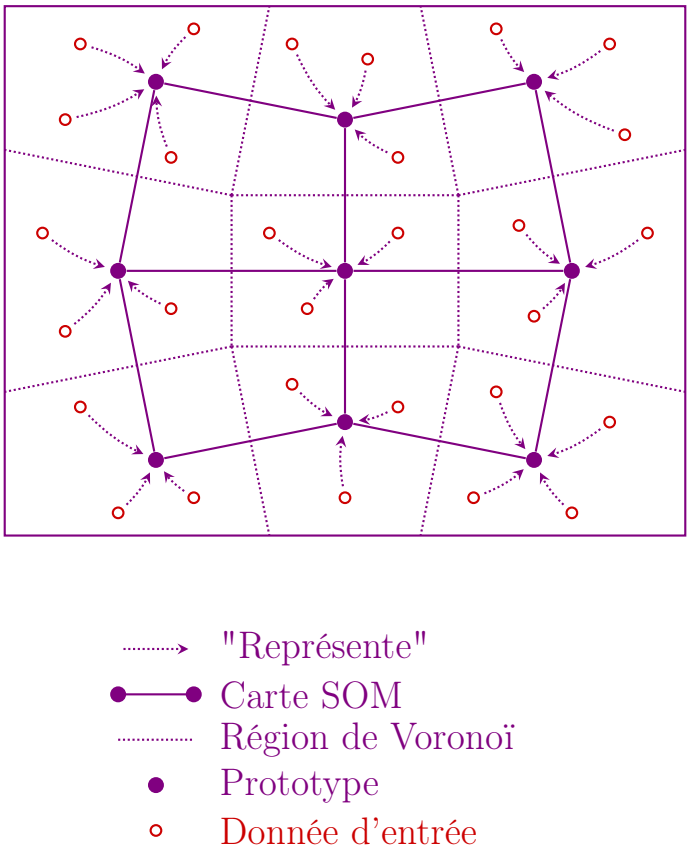

FIGURE 2.3 – Quantification Vectorielle (VQ) par SOM avec contrainte topologique. Les nœuds de la carte SOM (cercles violets) représentent des prototypes, et les données d'entrée (cercles rouges) sont associées à leur BMU. Les lignes pointillées indiquent les frontières de Voronoï entre les prototypes, tandis que les lignes continues représentent les connexions entre les nœuds de la grille SOM.

2.1.3 Batch SOM

Batch SOM est une variante du SOM classique qui diffère de l'approche stepwise présentée précédemment. Au lieu de mettre à jour les poids à partir d'un seul échantillon à chaque itération, elle utilise l'ensemble des données d'entrée. Cette approche permet la parallélisation des calculs, ce qui peut accélérer la convergence, notamment pour les grands ensembles de données [10].

Nous partons de la supposition que la convergence vers un état stable d'un SOM existe [10]. Nous avons quelques exigences pour que cela soit possible :

— lorsque $t \to \infty$, alors l'espérance de $w_j^{(t)}$ est égale à l'espérance de $w_j^{(t+1)}$
— $h_{cj} \neq 0$
— c est l'indice du BMU pour x

Cela peut être vu comme la recherche d'un point fixe des poids w_j, où les mise à jour sont en moyenne nulles. Nous pouvons l'écrire avec $\mathbb{E}_t$ l'opérateur d'espérance sur t [10] :

$$\forall j, \quad \mathbb{E}_t(h_{cj}[x - w_j^{(t)}]) = 0$$

Remarquons que $w_j^{(t)}$ est indépendant de t lorsque $t \to \infty$, nous allons le noter w_j^*. Si

l'espérance $\mathbb{E}_t$ est calculée à partir de la distribution empirique des données d'entrée, alors nous avons (pour un horizon T) :

$$\mathbb{E}_t(h_{cj}(t)[x - w_j^*]) = \frac{1}{T}\sum_{t=1}^{T} h_{cj}(t)[x - w_j^*] = 0$$

En développant cette expression, nous obtenons :

$$\frac{1}{T}\sum_{t=1}^{T} h_{cj}(t)[x - w_j^*] = 0 \Leftrightarrow$$

$$\sum_{t=1}^{T} h_{cj}(t)x - w_j^* \sum_{t=1}^{T} h_{cj}(t) = 0 \Leftrightarrow$$

$$w_j^* = \frac{\sum_{t=1}^{T} h_{cj}(t)x}{\sum_{t=1}^{T} h_{cj}(t)}$$

L'expression actuelle peut être écrite en regroupant leur BMU. Soit B_k l'ensemble des données dont le BMU est w_k, n_k le nombre de points dans ce groupe, et $x_{w,k}$ la moyenne de ces points. Alors, la mise à jour peut s'écrire [10] :

$$w_j^* = \frac{\sum_k h_{kj}\, n_k\, x_{w,k}}{\sum_k h_{kj}\, n_k}$$

Avec cette nouvelle interprétation, chaque noeud w_j est mise à jour comme une moyenne pondérée des données qui lui sont associées. Nous avons donc une première approche de comment batch SOM fonctionne. Comme déjà mentionné, cette variante traite l'ensemble des données d'entrée à chaque itération et pour cela les noeuds du SOM sont associés à une liste des copies des x dont ils sont le BMU [10]. Cette approche s'apparente à l'algorithme des k-means, mais avec des contraintes topologiques (la carte) entre les noeuds (équivalents aux centroïdes dans k-means). Schématiquement, la figure 2.4 illustre la mise à jour des noeuds dans le batch SOM.

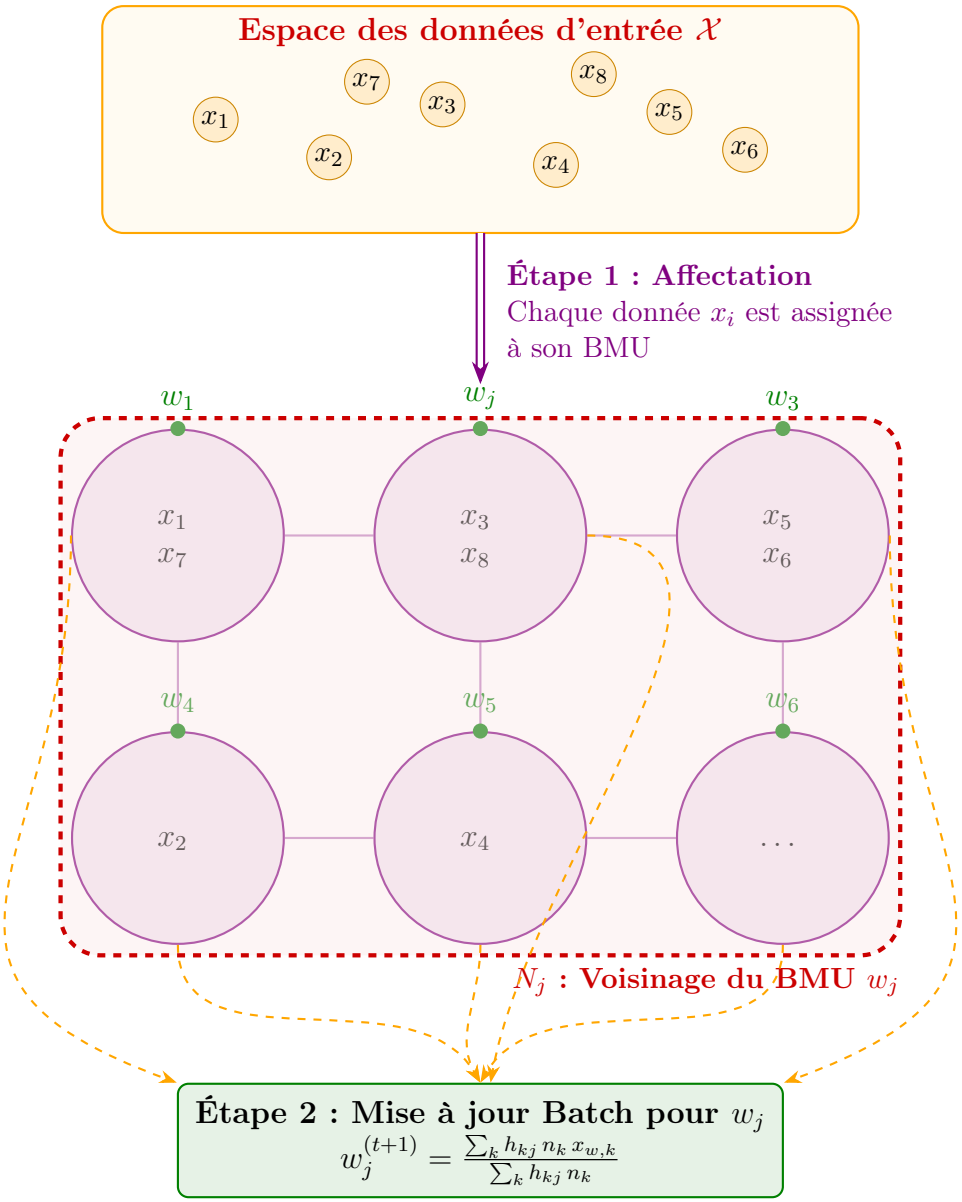

FIGURE 2.4 – Illustration du fonctionnement du **Batch SOM**. **Étape 1 :** L'ensemble des données d'entrée est réparti dans les nœuds correspondants. **Étape 2 :** Les vecteurs de poids (ex : w_j) sont mis à jour simultanément comme la moyenne pondérée de toutes les données tombant dans leur voisinage temporel N_j.

L'algorithme suit l'idée présentée dans la figure 2.4. Nous avons une notion de regroupement de données pour chaque noeud. L'implémentation est faite à l'aide d'un dictionnaire qui associe au noeud w_j l'ensemble $B_j = \{x \in \mathbb{R}^d : BMU(x) = w_j\}$. L'algorithme d'apprentissage du batch SOM est présenté dans l'algorithme 2, où la fonction de voisinage est définie par :

$$h_{ji}(t) = \exp\left(-\frac{\|p_j - p_i\|^2}{2\sigma(t)^2}\right)$$

Algorithm 2 Algorithme d'apprentissage d'un batch SOM

Initialiser les poids w_i de manière aléatoire ou avec PCA
Initialiser $\sigma(0)$
Définir T le nombre d'itérations maximum
for $t = 1$ **à** T **do**
    nodes = {} ▷ Dictionnaire pour stocker les données associées à chaque noeud
    $\sigma(t) = \sigma(0)e^{-t/T}$ ▷ Calcul de la largeur de voisinage
    for chaque x_i dans les données **do**
        $c(x_i) = \arg\min_j \|x_i - w_j\|^2$
        nodes[$c(x_i)$].append(x_i) ▷ Ajout de x_i à la liste de $w_{c(x_i)}$
    end for
    for chaque noeud j **do**
        $x_{w,j} = mean(nodes[j])$ ▷ Moyenne des données dans B_j
        $n_j = len(nodes[j])$ ▷ Nombre de données dans B_j
    end for
    for chaque noeud i **do**
        calculer $h_{ji}(t)$ pour tout j
        $w_i^{(t+1)} = \frac{\sum_j h_{ji}(t)\, n_j\, x_{w,j}}{\sum_j h_{ji}(t)\, n_j}$ ▷ Mise à jour du poids du noeud i
    end for
end for

Ainsi, cette variante a pu être construite à partir de l'hypothèse de convergence d'un SOM vers un état stable. Elle permet une convergence plus rapide, surtout pour les grands ensembles de données, et se prête naturellement à la parallélisation. Dans la littérature, Kohonen souligne également que le batch SOM peut aussi être généralisée pour des données d'entrée non vectorielles, à condition de disposer d'une mesure de distance entre les données d'entrée et les noeuds du SOM [10].

Enfin, nous avons présenté deux variantes d'apprentissage pour un SOM classique : la version stepwise et la version batch. Cependant, malgré ses avantages, le batch SOM présente certaines limites, notamment en termes de complexité computationnelle et de mémoire requise [6].

2.2 Limites

Les cartes auto-organisées possèdent la capacité de réaliser simultanément une quantification vectorielle et une réduction de dimension. Cependant, certains travaux ont montré que, du point de vue de l'erreur de la quantification ou de la préservation des distances, les SOM peuvent être moins performants que la combinaison de deux méthodes spécialisées [4]. En particulier, l'association d'une méthode de quantification vectorielle telle que le k-means, suivie d'une technique de réduction de dimension comme le Sammon mapping [13], permet d'optimiser séparément ces deux objectifs.

Cette différence s'explique par le fait que les méthodes utilisées dans une approche séquentielle sont chacune conçues pour minimiser explicitement une fonction de coût spécifique, tandis que les SOM réalisent simultanément ces tâches à travers un mécanisme unique d'apprentissage. Ce couplage induit un compromis entre la qualité de la quantification et celle de la projection, pouvant conduire à des performances inférieures sur chacun de ces aspects pris individuellement.

Dans le cas de la quantification vectorielle, la méthode k-means a tendance à produire des performances meilleures que les SOM. En effet, k-means est spécifiquement conçu pour minimiser la distance intra-cluster, tandis que les SOM doivent également satisfaire une contrainte d'organisation topologique. Cela limite la capacité de répresentation des structures de données complexes, en particulier lorsque nous avons un nombre plus élevé de clusters [4].

Concernant la réduction de dimension, les techniques traditionnelles, notamment celles issues du multidimensional scaling (MDS), ont tendance à mieux préserver les distances entre les points de données [3], tandis que les SOM peuvent introduire des distorsions dans la représentation des distances à cause de leur structure en grille et de la manière dont ils mettent à jour les poids. Cela s'accentue particulièrement avec l'augmentation de dimensionnalité des données d'entrée.

2.2.1 Quantification vectorielle et réduction de dimension

Une des méthodes les plus courantes pour la quantification vectorielle est le k-means, qui est un algorithme de clustering qui partitionne les données en k clusters en minimisant la somme des distances au carré entre les points de données et les centres de cluster. Cette méthode a été déjà mentionnée dans la section 2.1.1. Contrairement au k-means, qui attribue simplement chaque observation à un centroïde, le SOM organise ses prototypes sur une grille. Cette structure permet de projeter les données sur une carte de prototypes, où la proximité entre nœuds cherche à refléter les similarités entre les données. Donc, pour les SOM, le codebook est organisé sur une grille régulière cela induit une contrainte sur la manière dont les poids sont mis à jour. Il est également important de savoir que lorsque la fonction de voisinage $h_{ci} \to 0$, l'algorithme SOM se comportent de manière similaire au k-means, car seul le BMU est mis à jour[10].

Une réduction de dimension, pour être plus précis, est une transformation $\Phi : \mathbb{R}^D \to \mathbb{R}^d$ qui préserve les similarités ou distances dans l'espace de départ lors de la projection dans $\mathbb{R}^d$ [3]. Il existe plusieurs techniques de réduction de dimension, chacune avec ses propres avantages et inconvénients. Par exemple, le Sammon mapping [13], le Umap[12], t-SNE[15] ou encore les cartes auto-organisées. Le sammon mapping, en particulier, permet la réduction de dimension en minimisant la fonction de coût qui pondère les différences de distances entre les points de données et leurs projections. Cela est optimisé par la méthode de la decente de gradient [4]. Soit $\hat{\mathcal{X}} = \{\Phi(x_i)\}_{i=0}^{n-1}$ l'ensemble de projections, d_{ij} la distance entre les points x_i et x_j dans l'espace d'origine, et $\hat{d}_{ij}$ la distance entre leurs projections $\Phi(x_i)$ et $\Phi(x_j)$ dans l'espace réduit. La fonction de coût s'écrit :

$$\mathcal{L}(\hat{\mathcal{X}}) = \frac{1}{\sum_{i=0}^{n-1}\sum_{j<i} d_{ij}} \sum_{i=0}^{n-1}\sum_{j<i} \frac{(d_{ij} - \hat{d}_{ij})^2}{d_{ij}}$$

Enfin, il est important de remarquer que le codebook d'un SOM est limité en nombre de noeuds, cela entraine une discrétisation de l'espace d'entrée, ce qui peut conduire à une perte d'information si le nombre de noeuds est insuffisant et à une représentation moins précise des données [4].

Dans [4], nous trouvons une approche qui consiste à faire d'abord une quantification vectorielle avec le k-means, puis à appliquer une réduction de dimension avec le Sammon mapping. Les performances de cette approche sont comparées à celles des SOM pour différentes configurations, comme illustré dans le tableau 2.1.

algorithm	no. clusters	dimension	msqe ↓	Rand ↑	corr.↑
SOM	4	4	0.53	1.00	0.64
		6	1.53	0.91	0.72
		8	1.15	0.99	0.74
	9	4	0.33	0.97	0.48
		6	0.54	0.97	0.66
		8	0.81	0.97	0.84
		mean SOM	**0.81**	**0.97**	**0.67**
oKMC+	4	4	0.53	0.99	0.87
		6	1.06	0.99	0.87
		8	1.17	1.00	0.91
	9	4	0.29	0.99	0.89
		6	0.47	0.99	0.87
		8	0.56	0.98	0.86
		mean oKMC+	**0.68**	**0.99**	**0.88**

TABLE 2.1 – tableau de comparaison des performances entre les SOM et la combinaison k-means + Sammon mapping (oKMC+) pour différentes configurations de nombre de clusters et de dimensions d'entrée.

À partir des résultats, nous observons que la combinaison k-means + Sammon mapping (oKMC+) a ,en moyenne, des meilleures performances que les SOM :

— une plus faible erreur quadratique moyenne (msqe) pour la quantification vectorielle,
— un indice de Rand plus élevé (qui mesure la similarité entre les partitions) pour la qualité du clustering,
— une corrélation plus forte pour la mesure de la qualité de préservation de la topologie.

Cela montre une limite importante des SOM, leur performance est inférieure par rapport à des méthodes spécialisées pour la quantification vectorielle et la réduction de dimension. Les résultats suggèrent que les SOM peuvent ne pas être la meilleure option pour des tâches de clustering ou de réduction de dimensionnalité, surtout lorsque la qualité de la quantification ou de la réduction de dimension est cruciale.

2.2.2 Autres limitations

Les SOM présentent d'autres contraintes, notamment l'augmentation de la compléxité computationnelle avec la dimensionnalité des données d'entrée [6]. Lorsque la dimension de les données sont élevées, le nombre de noeuds dans la grille et la dimensionalité doivent être augmentées pour capturer la complexité des données, ce qui peut entraîner une augmentation significative du temps de calcul et de la mémoire requise pour l'apprentissage du SOM.

En effet, la complexité de l'algorithme d'apprentissage est généralement de l'ordre de $O(N \cdot M \cdot T)$, où N est le nombre d'échantillons d'entrée et M est le nombre de noeuds dans la grille du SOM (en général $M \ll N$) et T le nombre d'itérations d'apprentissage. Cela le rend moins adapté pour des jeux de données de grande taille ou de forte dimensionnalité [6].

Une autre limitation des SOM est leur incapacité à traiter les données de nature séquentielle ou temporelle. Les SOM classiques ne sont pas conçus pour traiter des données temporelles, ce qui peut limiter leur application dans des domaines tels que la reconnaissance de la parole ou l'analyse de séries temporelles [6].

Ainsi, ces différentes limitations montrent qu'ils peuvent être moins adaptés à certaines applications modernes, notamment dans le domaine de l'apprentissage automatique et de l'analyse de données. C'est pour cela que de nombreuses extensions ont été proposées pour remédier à ces limitations et améliorer les performances des SOM dans différents contextes.

2.3 Extensions

Dans cette section, nous allons parcourir certaines extension qui permettent une amélioration considérable pour ainsi s'adapter à des contextes plus modernes et complexes. Nous allons présenter les extensions suivantes : K-SOM, flow-SOM, hybride SOM (MLP + SOM et CNN + SOM).

2.3.1 Accelerating Stochastique K-SOM

Cette extension, présentée dans [11], propose une approche qui reduit le temps de calcul d'une autre extension nommée Kernel self-organizing maps (K-SOM). K-SOM est une extension des SOM qui utilise des fonctions de noyau pour permettre la modélisation de relations non linéaires entre les données d'entrée et les prototypes du SOM. Cependant, l'apprentissage d'un K-SOM peut être plus coûteux en termes de temps de calcul que celui d'un SOM classique, en particulier pour les grands ensembles de données. Pour avoir une convergence correcte, l'algorithme d'apprentissage aura besoin de βn itérations, où β est un facteur de proportionnalité pour le nombre d'itérations qui dépend de la complexité du modèle et de la nature des données d'entrée. Cela nous donne une complexité de $O(\beta n^3 \cdot m)$ [11], ce qui peut être prohibitif pour les grands ensembles de données.

L'approche proposée dans [11] consiste à faire une reformulation des calculs effectués lors des étapes d'affectation et de mise à jour. Elle introduit deux structures intermédiaires : un vecteur A^t et une matrice B^t, qui permettent de stocker des quantités liées aux produits scalaires dans l'espace de Hilbert induit par le noyau. Cela permet d'éviter le recalcul complet des distances entre chaque donnée et les prototypes à chaque itération.

Ainsi la complexité de l'apprentissage est réduit à $O(\beta n^2 \cdot m)$, ce qui est une amélioration significative par rapport à la complexité de l'apprentissage d'un K-SOM classique lors de grands ensembles de données. Cependant, cette extension a besoin d'une mémoire additionnelle pour stocker les structures A^t et B^t. Le tableau 2.2 présente une comparaison des performances entre le K-SOM classique et l'approche accélérée proposée dans [11].

	numeric SOM	rSOM	acc. rSOM
"polblogs"	*NA*	1112.34 (165.86)	**36.99** (4.49)
"cowrie"	*NA*	1715.40 (249.60)	**42.52** (2.32)
"Wine"	**16.40** (2.13)	8527.16 (874.58)	206.32 (38.24)

TABLE 2.2 – Comparaison des temps d'exécution (en secondes) entre le SOM classique, l'approche k-SOM classique (rSOM) et l'approche accélérée (acc. rSOM) proposée dans [11]. Les résultats sont présentés sous la forme de moyenne (écart-type) sur plusieurs exécutions.

Ce tableau montre que l'approche accélérée (acc. rSOM) a des temps d'exécution significativement plus faibles que le K-SOM classique (rSOM) avec un gain de plusieurs ordres de grandeur pour les jeux de données "polblogs" et "cowrie". Cependant, pour les données numériques "Wine", le SOM classique a des temps d'exécution plus faibles que les deux variantes de K-SOM. Cela peut être dû au fait aux manipulations de la matrice des similarités que le K-SOM doit effectuer et qui peuvent être plus coûteuses. L'approche accélérée est particulièrement efficace pour les données non numériques, tandis que pour les données numériques, le SOM classique reste plus rapide.

Ainsi, cette extension est un avancement pour les SOM dans le cadre de données non vectorielles ou complexes, pour lesquelles l'approches classique n'est pas applicables.

2.3.2 FlowSOM

FlowSOM n'est pas une extension, mais plutôt une application intéressante utilisée notamment dans le domaine de la biologie et plus précisément pour l'analyse de données de cytométrie [16]. Cette approche en réalité ne propose pas une nouvelle variante de SOM, mais plutôt une nouvelle manière d'utiliser les SOM pour l'analyse de données de cytométrie avec une structure spécifique. Elle a comme objectif de faciliter la visualisation et l'interprétation des données de cytométrie, au lieu d'utiliser des scatter plots traditionnels, qui peuvent être difficiles à interpréter lorsque nous avons un grand nombre de dimensions et de points de données [16].

Basiquement, flowSOM suit l'algorithme d'apprentissage d'un SOM classique, mais le modèle est intégré dans un pipeline d'analyse de données pour optimiser le clustering, qui comprend des étapes de prétraitement, de clustering et de visualisation.

L'algorithme de flowSOM peut être résumé comme suit [16] :

- **Prétraitement des données** : nettoyage, normalisation et suppression des valeurs aberrantes.
- **Apprentissage du SOM** : un SOM est entraîné afin de projeter les données dans une grille structurée.
- **Construction d'un arbre couvrant minimal (MST)** : les noeuds du SOM sont reliés afin de capturer leur organisation topologique globale.
- **Meta-clustering** : les noeuds du SOM sont regroupés en clusters plus larges à l'aide d'une méthode de clustering.

L'étape de meta-clustering repose sur une sur-segmentation initiale des données via le SOM, où un nombre élevé de prototypes est utilisé pour ainsi capturer des plus petites structures dans les données, puis regrouper en un nombre réduit de clusters plus interprétables. Dans [16] pour avoir le nombre de clusters final, ils utilisent la méthode de l'elbow pour déterminer le nombre optimal de clusters à partir du MST. Elle consiste à détecter le point où la variation de la distance entre les clusters devient moins significative,

cela correspond à un point d'inflexion dans la courbe de la distance entre les clusters en fonction du nombre de clusters.

FlowSOM a montré des performances plus élevées que des méthodes traditionnelles d'analyse de données de cytométrie comme SPADE, notamment en termes de temps de calcul, de qualité de clustering et de visualisation des données. FlowSOM est donc une approche d'utilisation intéressante pour les cartes auto-organisées dans le contexte de l'analyse de données de cytométrie.

Ainsi, cette méthode fusionne les SOM avec d'autres techniques d'apprentissage automatique formant un pipeline plus large et permettant de répondre efficacement à un domaine d'application spécifique. Cela est aussi le cas pour les SOM hybrides.

2.3.3 SOM Hybrides

Les SOM hybrides sont une extension combinant les cartes auto-organisées avec d'autres techniques d'apprentissage automatique. Nous allons présenter deux type de SOM hybrides : SOM-CNN et SOM-MLP. Il en existe d'autres, mais nous allons nous limiter à ces deux types.

SOM-CNN

Cette extension combine les capacités de représentation des réseaux de neurones convolutionnels (CNN) avec les propriétés de structuration topologique des SOM. Selon les approches, les caractéristiques extraites par le CNN peuvent être utilisées comme entrée d'un SOM pour effectuer un clustering, ou inversement, les sorties d'un SOM peuvent servir d'entrée à un CNN pour des tâches supervisées. Ce type de réseau, employé notamment dans l'analyse d'images, utilise l'opération de convolution pour extraire des caractéristiques à partir des données d'entrée, ce qui est particulièrement efficace pour les données structurées en grille, comme les images.

Opération de convolution est définie de manière **discrète** comme suit :

$$(f * g)(t) = \sum_{a=-\infty}^{\infty} f(a) \cdot g(t - a)$$

Où f est la fonction d'entrée (par exemple, une image), g est le noyau de convolution (un petit filtre qui capture des caractéristiques spécifiques), a est l'indice de sommation et t est la position de sortie. Dans le cas des images, cette opération est généralisée en deux dimensions, où le noyau de convolution se déplace sur une grille bidimensionnelle.

Comme pour FlowSOM, cette approche ne propose pas une nouvelle variante de SOM, mais plutôt une nouvelle manière d'utiliser les SOM en les intégrant dans un pipeline d'apprentissage plus large. Le SOM est utilisé pour effectuer un clustering des données et produire une représentation structurée, qui est ensuite exploitée par un modèle supervisé tel qu'un CNN, afin d'améliorer les performances de prédiction.

Dans le document [1], ce type d'approche est testé pour savoir si la précision de detection de la resistance phénotypique des vecteurs de malaria est améliorée. Les résultats expérimentaux montrent une amélioration des performances en termes de précision de classification par rapport à un CNN seul, suggérant que l'intégration du SOM permet de mieux capturer la structure des données.

SOM-MLP

La combinaison d'un SOM avec un perceptron multicouche (MLP) est une autre approche hybride qui vise à tirer parti des avantages de chaque technique. L'approche du document [8] présente ce modèle hybride pour la prédiction manquante des logs de puits, en se basant sur des similitudes lithofaciques géologiques.

Dans cette architecture, le but du SOM est d'organiser les données dans un espace de faible dimension (souvent bidimensionnel) tout en préservant leur structure topologique, et de regrouper les observations similaires en clusters. Cette représentation peut ensuite être utilisée comme entrée d'un modèle supervisé. Le MLP, en tant que réseau de neurones profond capable de modéliser des relations non linéaires complexes, exploite ces informations pour effectuer des tâches de prédiction, telles que l'estimation de valeurs manquantes.

Les résultats expérimentaux montrent que l'approche SOM-MLP permet d'améliorer la précision de prédiction par rapport à des méthodes classiques, notamment dans des contextes où les données présentent une forte hétérogénéité.

Ainsi, certaines extensions de SOM, comme les cartes auto-organisées hybrides, ne se concentrent pas sur l'amélioration de l'algorithme d'apprentissage du SOM lui-même, mais plutôt sur l'intégration avec d'autres techniques d'apprentissage automatique pour créer des pipelines plus puissants et adaptés à des tâches spécifiques. Cet type d'approche permet de concevoir des modèles plus performants et mieux adaptés aux problématiques du monde réel, où les données sont souvent complexes et nécessitent une approche plus intégrée pour l'analyse et la prédiction.

2.3.4 Autres extensions

Nous avons présenté quelques extensions, mais il en existe d'autres, qui essayent de remédier différentes limites des SOM, que ce soit au niveau de la convergence, de la complexité, de la capacité à traiter des données non vectorielles, ou encore de l'intégration avec d'autres modèles d'apprentissage automatique.

Cela est représenté dans le document [6] qui présente une revue complète des principales évolutions des cartes auto-organisées.

Dans la suite de ce rapport, nous allons nous intéresser à l'extension deep embedded self-organizing maps (DESOM) qui mixe les SOM avec une technique moderne d'apprentissage profond : les auto-encodeurs.

2.4 Deep embedded self-organizing maps (DESOM)

L'apprentissage profond est une technique d'apprentissage automatique qui repose sur l'utilisation des réseaux de neurones profonds pour modéliser des relations complexes dans les données. Elle est souvent utilisée pour les tâches supervisées, mais peut également être appliquée pour des tâches non supervisées, notamment à travers les auto-encodeurs (AE). Ces derniers sont des réseaux de neurones conçus pour apprendre une représentation compacte des données d'entrée, appelée espace latent [5].

Cela a conduit à l'emergence des modèles hybrides qui combinent les cartes auto-organisées avec des techniques d'apprentissage profond. L'objectif est de tirer parti à la fois de la capacité de représentation des réseaux profonds et des propriétés topologiques des SOM. Une de ces approches est le Deep Embedded Self-Organizing Maps (DESOM),

qui intègre un auto-encodeur avec un SOM permettant un apprentissage conjoint de la représentation et du clustering [5].

Contrairement aux approches en deux étapes (réduction de dimension puis clustering), DESOM apprend simultanément un espace latent adapté au clustering et les prototypes du SOM. Ce processus permet de construire un espace latent dit SOM-friendly, dans lequel les données sont organisées de manière plus cohérente pour la quantification et la préservation de la topologie [5]. Le SOM agit ainsi comme une forme de régularisation qui améliore la structure topologique et la qualité du clustering dans cet espace latent.

Dans la suite de ce rapport, nous allons présenter les principes de fonctionnement de DESOM, son architecture et son algorithme d'apprentissage.

2.4.1 Autoencodeur (AE)

Les auto-encodeurs sont des réseaux de neurones conçus pour apprendre une représentation plus compacte des données d'entrée de manière auto-supervisée (self-supervised), avec pour objectif de reconstruire fidèlement ces données de sortie. Ils permettent ainsi d'extraire des caractéristiques pertinentes en minimisant la perte d'information entre l'entrée et sa reconstruction.

Un AE est composé de deux parties principales : un encodeur, qui transforme les données d'entrée x en une représentation de plus faible dimension appelée espace latent z, et un décodeur, qui reconstruit les données d'entrée à partir de cette représentation latente [5]. La figure 2.5 illustre cette architecture.

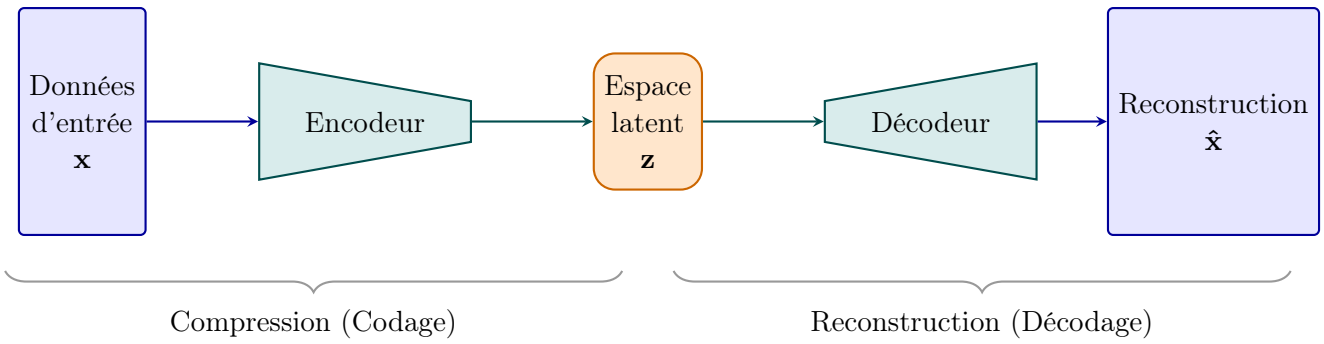

FIGURE 2.5 – Architecture générale d'un autoencodeur. L'encodeur compresse les données d'entrée dans un espace latent (goulot d'étranglement), tandis que le décodeur effectue l'opération inverse pour reconstruire $\mathbf{x}$.

Pour que le modèle puisse produire un espace latent de qualité, le modèle est entraîné en minimisant une fonction de perte qui mesure la différence entre les données d'entrée et leur reconstruction. Pour le DESOM, l'AE choisi apartient à la famille des auto-encodeur sous-complets (undercomplete autoencoder), qui impose une contrainte de dimensionnalité sur l'espace latent, c'est à dire que la dimension de l'espace latent est inférieure à celle des données d'entrée. Cette contrainte force au modèle à apprendre une réprésentation compréssée et évite que le modèle puisse utiliser la fonction d'identité pour la reconstruction [5].

Il existe d'autres types d'auto-encodeurs, comme les AE variationnels (VAE) qui introduisent une modélisation probabilistes de l'espace latent. Cependant, dans ce travail, nous nous concentrerons uniquement sur les AE deterministes produisant un espace latent continu.

2.4.2 Architecture du DESOM

L'architecture du DESOM présentée dans [5] repose sur la combinaison d'un auto-encodeur et d'une carte auto-organisée. L'objectif est d'intégrer le SOM dans l'architecture de l'auto-encodeur de manière à avoir un apprentissage conjoint d'une représentation latente des données et une organisation topologique de cette représentation.

L'AE prend les données d'entrée x et les encode dans un espace latent z, qui est ensuite utilisé comme entrée pour le SOM. La couche SOM est entrainée à organiser les noeuds du SOM dans cet espace latent. L'architecture du DESOM est illustrée à la figure 2.6.

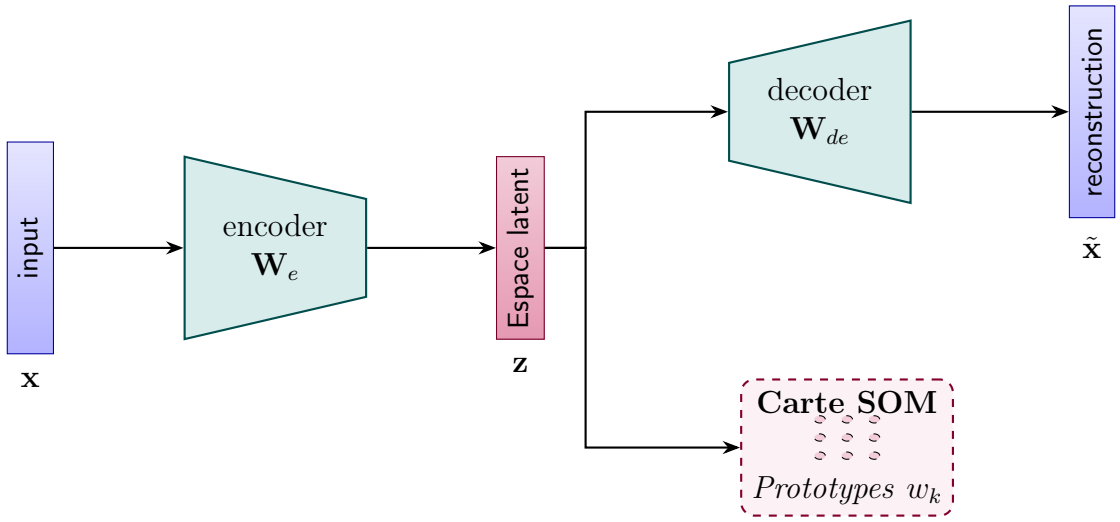

FIGURE 2.6 – Architecture du DESOM inspirée du document [5]. Les flèches noires indiquent la propagation avant (forward pass). L'encodeur W_e apprend à encoder les données d'entrée x dans un espace latent z, qui est ensuite utilisé par le décodeur W_{de}, pour la reconstruction de x et par le SOM pour l'apprentissage de la structure topologique et du clustering.

Cette architecture permet un réduction du temps moyen d'apprentissage par rapport à une approche en deux étapes, où nous entraînerons d'abord un auto-encodeur, puis un SOM sur les représentations apprises [5]. Le DESOM optimise simultanément les poids de l'autoencodeur et les prototypes de la carte. Cette optimisation conjointe permet d'obtenir un espace latent plus adapté à l'organisation topologique et au clustering.

Son algorithme d'apprentissage doit donc optimiser à la fois la fonction de perte de reconstruction de l'auto-encodeur et la fonction de perte du SOM en utilisant la méthode de descente de gradient.

2.4.3 Fonction de perte du DESOM

Le rôle de la fonction de perte dans le DESOM est central pour l'apprentissage conjoint de la représentation latente et de la carte auto-organisée. La fonction de perte du DESOM, $\mathcal{L}_{\text{DESOM}}$, est une combinaison de deux composantes principales : la perte de reconstruction de l'AE, $\mathcal{L}_{\text{R}}$, et la perte du SOM, $\mathcal{L}_{\text{SOM}}$ [5]. Il faut donc définir explicitement $\mathcal{L}_{\text{SOM}}$.

Pour un SOM classique, soit $\mathcal{L}^i_{\text{SOM}}$ la fonction de perte du SOM pour un échantillon d'entrée x_i, avec $w_{c(x_i)}$ comme noeud gagant (BMU), alors la fonction de perte du SOM

pour cet échantillon est donnée par :

$$\mathcal{L}^i_{\text{SOM}}(x_i, c(x_i), \{w_j\}) = \sum_{j=1}^{M} h_{c(x_i)j} \|x_i - w_j\|^2$$

Cette expression est donc la somme des distances au carré entre x_i et les prototypes, où $c(x_i)$ est l'indice du BMU pour l'échantillon x_i, $h_{c(x_i)j}$ est la fonction de voisinage entre w_j et le BMU.

Donc, la fonction de perte d'un SOM de manière générale est donnée par :

$$\mathcal{L}_{\text{SOM}} = \sum_{i=1}^{N} \mathcal{L}^i_{\text{SOM}}(x_i, c(x_i), \{w_j\}) \Leftrightarrow$$

$$\mathcal{L}_{\text{SOM}} = \sum_{i=1}^{N} \sum_{j=1}^{M} h_{c(x_i)j} \|x_i - w_j\|^2$$

La seconde composante pour $\mathcal{L}_{\text{DESOM}}$ est la fonction de perte de reconstruction $\mathcal{L}_{\text{R}}$ de l'auto-encodeur.

Soit W_e les poids de l'encodeur, W_{de} les poids du décodeur, $x_i \in \mathbb{R}^d$ les données d'entrée et $\tilde{x}_i \in \mathbb{R}^d$ la reconstruction de x_i par l'auto-encodeur, notons $f_{W_e} : \mathbb{R}^d \to \mathbb{R}^L$ la fonction d'encodage et $g_{W_{de}} : \mathbb{R}^L \to \mathbb{R}^d$ la fonction de décodage, avec $z_i = f_{W_e}(x_i)$ et $\tilde{x}_i = g_{W_{de}}(z_i)$. La fonction de perte de reconstruction $\mathcal{L}_{\text{R}}$ est généralement définie comme la moyenne de l'erreur quadratique (MSE) [5] :

$$\mathcal{L}_{\text{R}}(W_e, W_{de}) = \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}^i_{\text{R}} = \frac{1}{N} \sum_{i=1}^{N} \|x_i - \tilde{x}_i\|^2 \Leftrightarrow$$

En remplaçant $\tilde{x}_i$ par son expression, nous obtenons :

$$\mathcal{L}_{\text{R}}(W_e, W_{de}) = \frac{1}{N} \sum_{i=1}^{N} \|x_i - g_{W_{de}}(f_{W_e}(x_i))\|^2$$

Finalement, avec les deux fonctions de perte, nous pouvons définir la fonction de perte hybride du DESOM [5] :

$$\mathcal{L}_{\text{DESOM}}(W_e, W_{de}, \{w_j\}) = \mathcal{L}_{\text{R}}(W_e, W_{de}) + \gamma \mathcal{L}_{\text{SOM}}(W_e, \{w_j\})$$

Il est important de noter que la fonction de perte du SOM, $\mathcal{L}_{\text{SOM}}$, dépend à la fois des poids de l'encodeur W_e et les poids w_j de ses noeuds. Ces données d'entrée ne sont plus les données d'origine x_i, mais plutôt les représentations latentes $z_i \in \mathbb{R}^L$. Nous aurons donc pour un noeud S_j son vecteur de poids $w_j \in \mathbb{R}^L$.

Le paramètre $\gamma \in \mathbb{R}^+$, est un hyperparamètre qui équilibre l'importance relative de la perte de reconstruction et de la perte du SOM lors de l'apprentissage conjoint du DESOM. Une valeur élevée de γ donne plus d'importance à l'organisation topologique et au clustering dans l'espace latent, tandis qu'une valeur plus faible privilégie la qualité de reconstruction de l'autoencodeur.

Ainsi, la fonction de perte du SOM agit comme une régularisation guidée par la carte auto-organisatrice : elle encourage l'encodeur à produire un espace latent adapté à la quantification vectorielle et à la préservation de la topologie.

2.4.4 Processus d'apprentissage

Comme mentionné précédemment, l'apprentissage du DESOM est un processus conjoint qui optimise simultanément les poids de l'auto-encodeur et les noeuds du SOM. L'objectif est de minimiser la fonction de perte globale :

$$\mathcal{L}_{DESOM} = \mathcal{L}_R + \gamma \mathcal{L}_{SOM}$$

Cette optimisation permet à l'autoencodeur d'apprendre une représentation latente informative, tout en contraignant cet espace latent à être adapté à l'organisation topologique du SOM (SOM-friendly).

Pour effectuer cette optimisation, il faut calculer les gradients de $\mathcal{L}_{\text{DESOM}}$ par rapport aux poids de l'encodeur W_e, du décodeur W_{de} et des noeuds du SOM $\{w_j\}$. Cependant, le calcul du BMU (via l'opération d'argmin) dont dépend la fonction de voisinage n'est pas différentiable. Pour remédier ce problème, nous considérons $h_{c(x_i)j}$ comme des coefficients constants à chaque étape d'optimisation [5].

Cette contrainte, nous permet d'utiliser la rétropropagation et la descente de gradient pour mettre à jour les paramètres du modèle.

Les gradients de perte globale pour les poids W_e, W_{de} et w_j s'ecrivent alors [5] :

$$\begin{aligned}
\frac{\partial \mathcal{L}_{\text{DESOM}}}{\partial W_e} &= \frac{\partial \mathcal{L}_{\text{R}}}{\partial W_e} + \gamma \frac{\partial \mathcal{L}_{\text{SOM}}}{\partial W_e} \\
\frac{\partial \mathcal{L}_{\text{DESOM}}}{\partial W_{de}} &= \frac{\partial \mathcal{L}_{\text{R}}}{\partial W_{de}} \\
\frac{\partial \mathcal{L}_{\text{DESOM}}}{\partial w_j} &= \gamma \frac{\partial \mathcal{L}_{\text{SOM}}}{\partial w_j}
\end{aligned}$$

En calculant pour un point de données x_i, les gradients associés à la perte de reconstruction sont donnés par :

$$\begin{aligned}
\frac{\partial \mathcal{L}^i_{\text{R}}}{\partial W_e} &= 2(g_{W_{de}}(f_{W_e}(x_i)) - x_i) \frac{\partial g_{W_{de}}(f_{W_e}(x_i))}{\partial W_e} \\
\frac{\partial \mathcal{L}^i_{\text{R}}}{\partial W_{de}} &= 2(g_{W_{de}}(f_{W_e}(x_i)) - x_i) \frac{\partial g_{W_{de}}(f_{W_e}(x_i))}{\partial W_{de}}
\end{aligned}$$

Les gradients associés à la perte SOM sont :

$$\begin{aligned}
\frac{\partial \mathcal{L}^i_{\text{SOM}}}{\partial W_e} &= 2 \sum_{j=1}^{m} h_{c(x_i)j} (f_{W_e}(x_i) - w_j) \frac{\partial f_{W_e}(x_i)}{\partial W_e} \\
\frac{\partial \mathcal{L}^i_{\text{SOM}}}{\partial w_j} &= 2 h_{c(x_i)j} (w_j - f_{W_e}(x_i))
\end{aligned}$$

Avec ces gradients, nous pouvons mettre à jour les poids de l'encodeur, du décodeur et des prototypes du SOM à chaque itération de l'apprentissage à l'aide d'une descente de gradient stochastique par mini-batch [5]. Soit β un mini-batch de n_b échantillons et η le taux d'apprentissage. Les règles de mise à jour pour l'encodeur et le décodeur sont alors

données par :

$$W_e \leftarrow W_e - \frac{\eta}{n_b} \sum_{i \in \beta} \left(\frac{\partial \mathcal{L}_{\mathrm{R}}^i}{\partial W_e} + \gamma \frac{\partial \mathcal{L}_{\mathrm{SOM}}^i}{\partial W_e} \right)$$

$$W_{de} \leftarrow W_{de} - \frac{\eta}{n_b} \sum_{i \in \beta} \frac{\partial \mathcal{L}_{\mathrm{R}}^i}{\partial W_{de}}$$

La règle de mise à jour pour les prototypes du SOM est donnée par [5] :

$$w_j \leftarrow w_j - \frac{\eta}{n_b} \sum_{i \in \beta} \gamma \frac{\partial \mathcal{L}_{\mathrm{SOM}}^i}{\partial w_j}$$

En développant $\frac{\partial \mathcal{L}_{\mathrm{SOM}}^i}{\partial w_j}$ avec

$$z_i = f_{W_e}(x_i)$$

La règle de mise à jour devient :

$$w_j \leftarrow w_j + \frac{2\eta\gamma}{n_b} \sum_{i \in \beta} h_{c(x_i)j}(z_i - w_j)$$

Cette règle montre que chaque noeuds est déplacé vers les représentations latentes du mini-batch, avec une intensité pondérée par la fonction de voisinage du SOM.

Ces mis à jour sont utilisées pour la rétropropagation de l'erreur à travers le réseau. L'encodeur reçoit le signal provenant de la reconstruction et le signal provenant de la contrainte SOM, tandis que le décodeur et les noeuds du SOM sont mis à jour respectivement à partir la perte de reconstruction et de la perte SOM. La figure 2.7 illustre ce processus d'apprentissage conjoint.

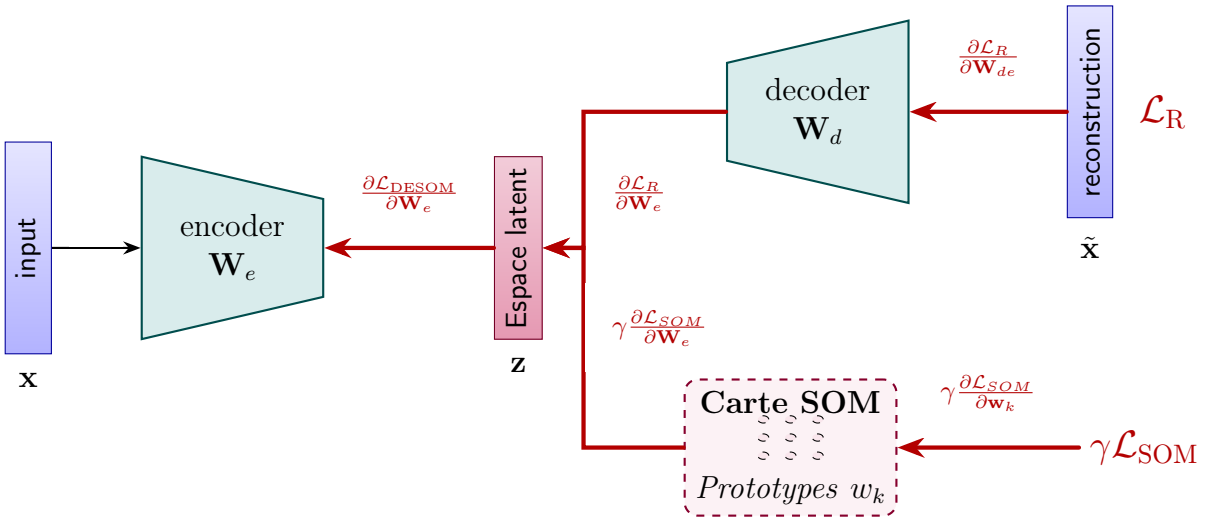

FIGURE 2.7 – Processus de rétropropagation (backward pass) du DESOM. Les flèches rouges indiquent le flux des gradients. Les poids de l'encodeur $\mathbf{W}_e$ sont mis à jour en fonction des erreurs combinées provenant de la reconstruction ($\mathcal{L}_{\mathrm{R}}$) et de l'organisation topologique du SOM ($\mathcal{L}_{\mathrm{SOM}}$). Les flèches noires indiquent la propagation avant (forward pass) pour la cohérence visuelle.

Donc la minimisation se produit via une descente de gradient stochastique comme c'est le cas dans le modèle de SOM stochastique [5].

L'algorithme 3 présente une description détaillée du processus d'apprentissage du DESOM. Les affectations aux BMU et les coefficients de voisinage sont considérés comme constants lors du calcul des gradients à chaque itération.

Algorithm 3 Algorithme d'apprentissage du DESOM

Require: Données d'entrée $\{x_i\}_{i=1}^{N}$, nombre de prototypes M, taux d'apprentissage η, hyperparamètre de régularisation γ, nombre d'itérations T.

1: Initialiser les poids de l'encodeur W_e, du décodeur W_{de} et des prototypes du SOM $\{w_j\}_{j=1}^{M}$ de manière aléatoire.
2: **for** $t = 1$ to T **do**
3: Sélectionner un mini-batch β de n_b échantillons de données d'entrée.
4: **for** chaque échantillon x_i dans le mini-batch β **do**
5: Calculer la représentation latente $z_i = f_{W_e}(x_i)$.
6: Trouver le BMU $c(z_i) = \arg\min_j \|z_i - w_j\|^2$.
7: Calculer les coefficients de voisinage $h_{c(z_i)j}$ pour chaque prototype w_j.
8: Calculer la reconstruction $\tilde{x}_i = g_{W_{de}}(z_i)$.
9: Calculer les pertes $\mathcal{L}_R^i$ et $\mathcal{L}_{SOM}^i$
10: **end for**
11: Calculer la fonction de coût totale du mini-batch :
12: $\mathcal{L}_{\text{DESOM}} = \frac{1}{n_b} \sum_{x_i \in \beta} (\mathcal{L}_{\text{R}}^i + \gamma \mathcal{L}_{\text{SOM}}^i)$
13: Calculer les gradients de $\mathcal{L}_{DESOM}^{\beta}$ par rétropropagation.
14: Mettre à jour W_e, W_{de} et $\{w_j\}_{j=1}^{M}$ par descente de gradient stochastique à mini-batch.
15: **end for**

Ainsi, cet algorithme fournit une description du processus d'apprentissage du DESOM. Cette méthode intégre le SOM dans l'architecture de l'auto-encodeur, en prenant les représentations latentes produite par l'encodeur comme entrée pour le SOM, rendant ainsi l'espace d'entrée SOM-friendly. Cette contrainte encourage l'apprentissage d'un espace latent qui améliore la qualité du clustering et la préservation de la topologie.

Chapitre 3

Étude empirique

Ce chapitre consistera à la comparaison expérimentale de trois modèles : un SOM implémenté depuis zéro, une approche en deux étapes combinant un autoencodeur et un SOM (AE+SOM), ainsi que le DESOM. Les expériences sont réalisées sur deux jeux de données d'images, MNIST et Fashion-MNIST.

L'objectif de cette étude permettra d'analyser les avantages et les limites de chaque approche selon certains critères : la qualité de quantification, la préservation de la topologie, la qualité du regroupement obtenu et le comportement visuel des cartes générées. Cette comparaison nous permettra aussi d'analyser les situations dans lesquelles le DESOM peut être benefique par rapport aux autres approches.

3.1 Données utilisées

Les expériences seront menées sur 2 jeux de données disponibles via la bibliothèque `torchvision` :

— **MNIST** : MNIST est un ensemble de données d'images de chiffres manuscrits. La tailles de chaque image est de 28×28 pixels. Nous avons normalisé les valeurs des pixels dans l'intervalle $[0, 1]$, puis nous avons aplati chaque image en un vecteur de 784 composantes.

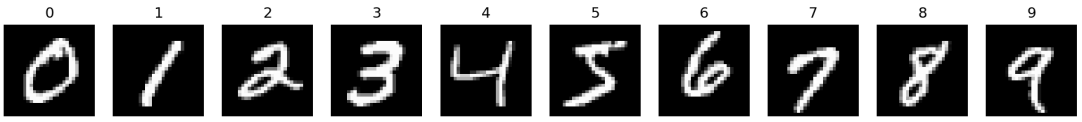

FIGURE 3.1 – Exemples d'images issues des jeux de données MNIST

— **Fashion-MNIST** : Fashion-MNIST est un ensemble de données d'images de vêtements et accessoires. Celui-ci est similaire à MNIST en termes de taille et de format, mais il est plus complexe en raison de la diversité des classes de vêtements et d'une plus grande similarité visuelle entre certaines classes. Nous avons appliqué les mêmes prétraitements que pour MNIST, en normalisant les valeurs des pixels dans $[0, 1]$ et en aplatissant les images en vecteurs de 784 composantes.

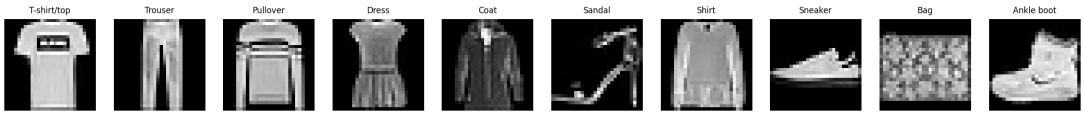

FIGURE 3.2 – Exemples d'images issues des jeux de données Fashion-MNIST

Pour des raisons de simplicité et de temps d'exécution, les expériences sont réalisées sur des sous-échantillons de ces jeux de données, composés de 10'000 échantillons d'entraînement et 2'000 échantillons de test. Les mêmes sous-échantillons sont utilisés pour tous les modèles afin de garantir une comparaison équitable.

3.2 Configuration expérimentale

Cette section présente la configuration expérimentale utilisée pour comparer les trois modèles étudiés. Pour une évaluation équitable, certains hyperparamètres sont partagé entre les modèles : la taille de la grille, la distance utilisée, la fonction de décroissance du rayon de voisinage et la topologie de la grille. L'implémentation est réalisée en **Python**, notamment avec les bibliothèques `NumPy` pour le batch SOM et `PyTorch` pour l'autoencodeur et le DESOM.

3.2.1 Configuration des modèles et hyperparamètres

Batch SOM

Nous avons décidé d'implémenter un SOM de type batch, car comme mentionné précédemment, il est généralement plus stable, plus rapide à converger que la version incrémentale et ne nécessite pas d'un taux d'apprentissage.

Les prototypes du SOM sont initialisés de manière aléatoire à partir des données d'entrée et la fonction de calcul du rayon $\sigma(t)$ est une autre version de la fonction de décroissance exponentielle présentée dans l'algorithme 2.

$$\sigma(t) = \sigma_{\text{initial}} \left(\frac{\sigma_{\text{final}}}{\sigma_{\text{initial}}} \right)^{\frac{t}{T}}$$

où t désigne l'itération courante et T le nombre total d'itérations.

Les hyperparamètres choisis pour le batch SOM sont les suivants :

Paramètre	Valeur
Taille de grille	8×8
Nombre d'itérations	250
Rayon initial	8
Rayon final	0.1
Initialisation des prototypes	aléatoire à partir des données
Distance utilisée	euclidienne
Topologie de la grille	rectangulaire
Historique de l'erreur de quantification	oui

L'historique de l'erreur de quantification est enregistré à chaque itération pour analyser la convergence du SOM.

AE+SOM

L'approche AE+SOM est une approche en deux étapes. Dans un premier temps, un auto-encodeur est entraîné pour avoir une représentation latente des données d'entrée, puis un SOM est entraîné sur ces représentations latentes.

Nous avons construit un auto-encodeur avec `PyTorch`. L'encodeur est composé d'une couche d'entrée de dimension 784, de trois couches cachées de taille respectivement $(128, 64, 128)$, avec fonction d'activation ReLU, et d'une couche latente de dimension 10. Le décodeur est symétrique à l'encodeur, avec des tailles de couches inversées et reconstruit les données d'entrée à partir de la représentation latente. La fonction de perte utilisée pour entraîner l'autoencodeur est l'erreur quadratique moyenne, et l'optimisation est effectuée avec Adam.

Après l'entraînement de l'auto-encodeur, le SOM est entraîné sur les représentations latentes produites par l'encodeur, en utilisant les mêmes hyperparamètres que pour le batch SOM.

Les hyperparamètres choisis pour l'AE+SOM sont les suivants :

Paramètre	Valeur
Dimension d'entrée	784
Dimension latente	10
Taille de grille	8×8
Nombre d'itérations pour le SOM et l'auto-encodeur	250
Taille du batch	32
Taux d'apprentissage	0.001
Rayon initial	8
Rayon final	0.1
Initialisation des prototypes	aléatoire
Distance utilisée	euclidienne
Topologie de la grille	rectangulaire
Historique de l'erreur de quantification	oui

Cette approche nous permet d'évaluer l'effet d'une réduction de dimension sans influence de la fonction de perte du SOM.

DESOM

Contrairement à l'approche AE+SOM, dans le DESOM, les deux composantes sont entraînées conjointement. Le SOM est donc intégré directement dans la fonction de perte du modèle.

Nous avons utilisé la même architecture d'auto-encodeur que pour l'approche AE+SOM, avec les mêmes tailles de couches et fonctions d'activation. Cependant, le SOM utilisé a été implémenté avec `pytorch`, ce qui permet d'intégrer le SOM dans le processus d'apprentissage conjoint du DESOM. Cela permet d'optimiser simultanément les poids de l'auto-encodeur et les prototypes du SOM par rétropropagation. Pour l'apprentissage du DESOM, l'algorithme 3 est suivi, avec les mises à jour des poids effectuées à l'aide de la descente de gradient stochastique par mini-batch.

Paramètre	Valeur
Dimension d'entrée	784
Dimension latente	10
Taille de grille	8×8
Nombre d'époques	250
Taille du batch	32
Rayon initial	8
Rayon final	0.1
Taux d'apprentissage	0.001
γ	0.001

Le DESOM utilise un taux d'apprentissage pour l'optimiseur Adam utilisé pour la mise à jour des poids de l'auto-encodeur et des prototypes du SOM, la taille du batch est la taille du mini-batch utilisée pour la descente de gradient stochastique, le nombre d'époques correspond au nombre de passages complets à travers l'ensemble de données d'entraînement. Le choix des hyperparamètres ont été fait en s'inspirant des valeurs utilisées dans l'article [5].

Hyperparamètres communs

Les hyperparamètres communs aux trois modèles sont résumés dans le tableau suivant :

Paramètre	Valeur
Taille de grille principale	8×8
Distance utilisée	euclidienne
Topologie de la grille	rectangulaire
Rayon initial	8
Rayon final	0.1

3.2.2 Métriques d'évaluation

Pour évaluer les performances des modèles, nous utiliserons les métriques suivantes :

— **Erreur de quantification (QE)** : Cette métrique mesure la distance moyenne entre les données d'entrée et leurs prototypes correspondants dans le SOM. Un QE plus faible indique une meilleure quantification des données. Formellement, pour un échantillon d'entrée x_i et son BMU avec le vecteur $w_{c(x_i)}$, le QE est défini comme :

$$QE = \frac{1}{N} \sum_{i=1}^{N} \|x_i - w_{c(x_i)}\|$$

où N est le nombre total d'échantillons d'entrée.

— **Erreur de topologie (TE)** : Cette métrique mesure la préservation de la topologie dans le SOM. Un TE plus faible indique une meilleure préservation de la topologie. Le TE est défini comme la proportion d'échantillons d'entrée pour lesquels les deux prototypes les plus proches ne sont pas des voisins dans le codebook. Formellement, pour un échantillon d'entrée x_i, soit $w_{c_1(x_i)}$ et $w_{c_2(x_i)}$ les deux prototypes les plus proches, alors :

$$TE = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}(w_{c_1(x_i)}, w_{c_2(x_i)})$$

où $\mathbb{I}$ est la fonction indicatrice qui vaut 1 si les prototypes ne sont pas des voisins dans le codebook et 0 sinon.

— **Pureté** : Cela mesure la pureté des clusters formés par le SOM. Un score de pureté plus élevé indique que les clusters sont plus homogènes en termes de classes d'origine. La pureté est définie comme suit : pour chaque cluster, nous attribuons la classe majoritaire parmi les échantillons qui y sont assignés, puis nous calculons la proportion d'échantillons correctement classés. Formellement, soit C_k le cluster k et L_j la classe j, alors :

$$\text{Pureté} = \frac{1}{N} \sum_{k} \max_{j} |C_k \cap L_j|$$

où N est le nombre total d'échantillons d'entrée.

— **NMI (Normalized Mutual Information)** :C'est une mesure de la similarité entre les clusters formés par le SOM et les classes d'origine. Un score de NMI plus élevé indique une meilleure correspondance entre les clusters et les classes d'origine. Le NMI est défini comme suit : soit C l'ensemble des clusters et L l'ensemble des classes d'origine, alors :

$$NMI(C, L) = \frac{I(C; L)}{\sqrt{H(C)H(L)}}$$

où $I(C; L)$ est l'information mutuelle entre les clusters et les classes, et $H(C)$ et $H(L)$ sont les entropies des clusters et des classes respectivement.

3.2.3 Visualisation utilisées

Pour compléter l'analyse, nous allons également examiner les cartes générées par les modèles. Il y a deux types de visualisations que nous allons utiliser :

— **U-matrice** : Ce type de visualisation nous permet de voir les distances entres prototypes du SOM. Les Zones avec des distances plus élevées sont généralement les frontières entres les clusters.

— **Hit map** : Cela montre le nombre d'échantillons d'entrée qui sont assignés à chaque prototype du SOM. Cela permet d'observer la distribution des données d'entrée sur la carte et d'identifier les zones de forte densité.

3.3 Analyse des résultats

3.3.1 Résultats quantitatifs

Après avoir entraîné les trois modèles sur les deux jeux de données, nous avons calculé les métriques d'évaluation pour chaque modèle. Les valeurs en gras indiquent les meilleurs résultats pour chaque métrique.

MNIST

TABLE 3.1 – Résultats quantitatifs pour MNIST

Modèle	QE ↓	TE ↓	Pureté ↑	NMI ↑
SOM	5.416	**0.418**	0.782	0.543
AE+SOM	1.206	0.466	0.811	0.564
DESOM	**0.247**	0.555	**0.825**	**0.565**

Dans le tableau 3.1, l'erreur de quantification (QE) est significativement plus faible pour le DESOM par rapport au SOM et l'approche AE+SOM. Ce résultat indique que le DESOM a des prototypes assez proches des représentations encodées des données. Ainsi, la faible QE du DESOM ne signifie pas nécessairement que ses prototypes sont plus proches des images originales, mais plutôt que l'espace latent produit par l'auto-encodeur est mieux adapté pour la quantification par le SOM.

En revanche, nous observons une erreur de topologie (TE) plus élevée pour le DESOM que les deux autres modèles. Cela suggère que l'amélioration de la quantification peut se faire au détriment de la préservation locale de la topologie de la carte. En effet, cela est cohérent avec l'apprentissage conjoint du DESOM : l'encodeur est optimisé pour produire un espace latent favorable à la reconstruction et à la quantification par le SOM, mais cette optimisation ne garantit pas nécessairement une meilleure organisation topologique de la carte.

En ce qui concerne la pureté et le NMI, le DESOM obtient les meilleurs résultats. Cela suggère que l'espace latent produit par l'auto-encodeur du DESOM fournit une représentation plus adaptée pour le clustering que l'espace d'entrée d'origine. Cependant, la différence entre les trois modèles reste tout de même relativement faible pour les deux métriques.

Ainsi, pour le jeu de données MNIST, le DESOM se distingue par une erreur de quantification extrêmement basse et offre un meilleur compromis global pour ce jeu de données.

Fashion-MNIST

TABLE 3.2 – Résultats quantitatifs pour Fashion-MNIST

Modèle	QE ↓	TE ↓	Pureté ↑	NMI ↑
SOM	4.531	**0.398**	0.712	0.503
AE+SOM	0.887	0.406	0.722	0.510
DESOM	**0.177**	0.509	**0.738**	**0.516**

Comme pour MNIST, le tableau 3.2 montre que l'erreur de quantification du DESOM est nettement plus faible que celle des deux autres modèles. Cela renforce l'idée que l'apprentissage conjoint du DESOM permet d'obtenir un espace latent très adapté pour la quantification par le SOM.

De plus, comme obtenu sur MNIST, le DESOM obtient la plus grande erreur de topologie, ce qui appuie encore l'idée de compromis entre la quantification et la préservation

de la topologie. DESOM obtient les meilleures performances en termes de pureté et de NMI même si la différence reste relativement faible. Donc pour Fashion-MNIST, l'apprentissage conjoint du DESOM permet d'améliorer la quantification dans l'espace latent en échange d'une organisation topologique moins bonne.

Ainsi, même si le DESOM offre une meilleure quantification, les avantages en termes de clustering ne sont pas aussi marqués que pour les autres métriques, mais de manière globale, le DESOM offre un meilleur compromis pour ce jeu de données.

3.3.2 Analyse visuelle des cartes

Nous allons maintenant examiner les cartes générées par les trois modèles pour les deux jeux de données.

U-matrices

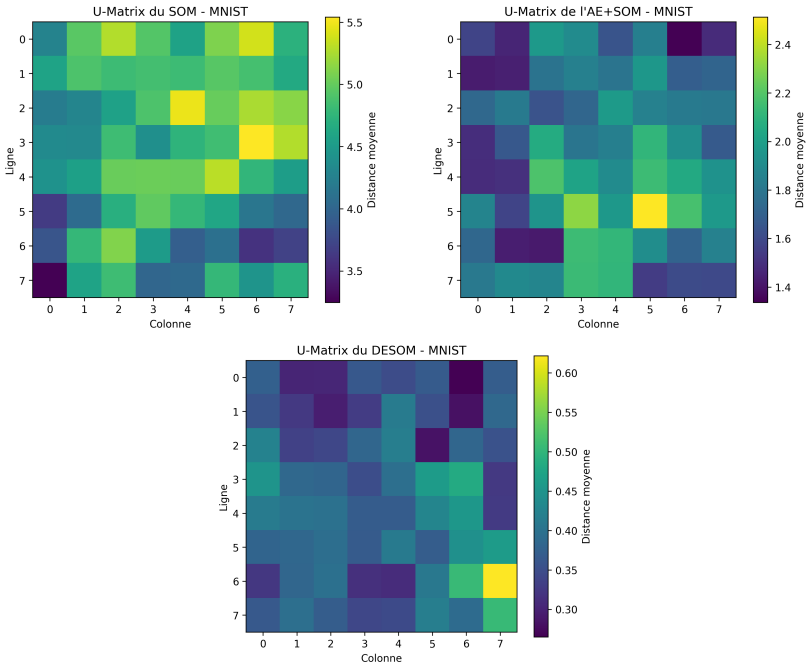

FIGURE 3.3 – U-matrix pour les trois modèles sur le jeu de données MNIST.

Dans la figure 3.3, les trois U-matrices présentent des organisations différentes pour MNIST.

Pour le SOM, nous pouvons observer une échelle de distance de 3.5 à 5.5. Cela indique des distances élevées entre les prototypes, notamment dans la partie supérieure à droite de la carte, cela montre des frontières plus marquées. L'approche AE+SOM semble présenter une échelle moins élevée que celui du SOM, avec des distances qui varient entre 2.4 et 1.4.

Le DESOM montre, quant à lui, a les distances plus faibles, avec une échelle entre 0.3 et 0.6. Ces observations reste cohérentes avec les résultats quantitatifs obtenus précédement : le DESOM présente une erreur de quantification nettement plus faible que les deux autres modèles. Cela s'explique par la création de l'espace latent de dimension plus faible, les vecteurs de cet espace sont donc plus proches les uns des autres que les vecteurs d'origine.

Cependant, nous remarquons que les contrastes entre les zones sont moins distincts pour le DESOM que pour les SOM. La priorité donnée à la quantification dans le DESOM

se fait au détriment d'une augmentation de l'erreur topologique. Donc, malgré que l'espace latent du DESOM soit favorable à la quantification, ce n'est pas nécessairement le cas pour l'organisation topologique de la carte.

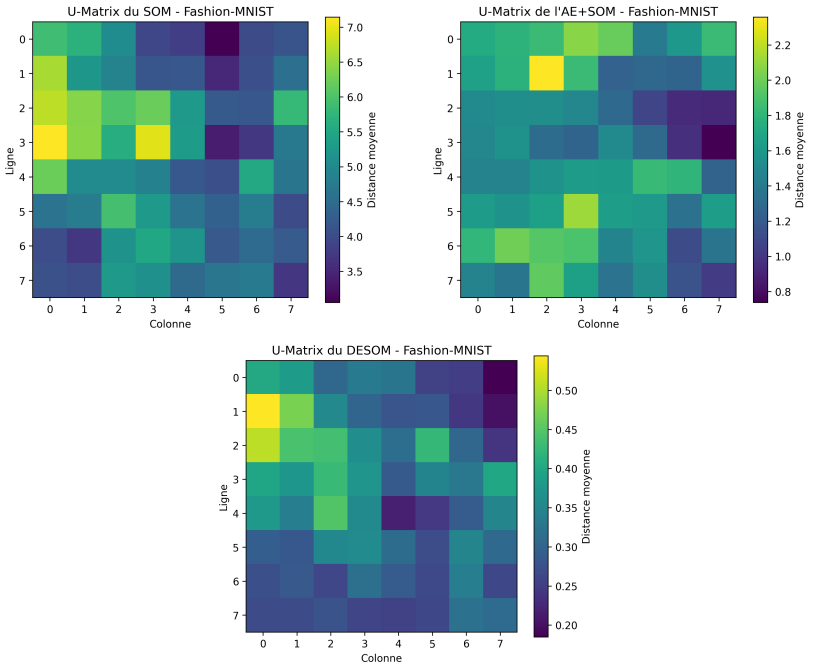

FIGURE 3.4 – U-matrix pour les trois modèles sur le jeu de données Fashion-MNIST.

Dans la figure 3.4, les U-matrices pour Fashion-MNIST montre un comportement similaire à celle observée sur MNIST.

Le SOM présente des zones de distances élevées, notamment dans la partie supérieure à droite de la carte. Ces régions peuvent être interprétées comme des frontières entre régions de la carte.

L'approche AE+SOM montre des distances plus faibles et une distinction de zones moins marquée que pour le SOM, ce qui indique une organisation plus compacte.

Pour le DESOM, l'échelle est encore plus basse, avec des distances qui varient entre 0.2 et 0.5. Cela semble indiquer que l'espace latent est très compact, ce qui force les prototypes à être plus proches des représentations latentes des données. Cette observation est cohérente avec les résultats quantitatifs, le DESOM présente une meilleure erreur de quantification, mais les régions sont moins bien réparties donc une moins bonne organisation topologique.

Hit maps

Les hit maps présentées dans les figures 3.5 et 3.6 permettent d'analyser la répartition des données sur les prototypes de chaque carte. Une activation élevée (prototypes surchargés) indique une grande quantité de données d'entrée assignées à ce prototype, tandis qu'une activation faible indique peu ou pas de données assignées à ce prototype.

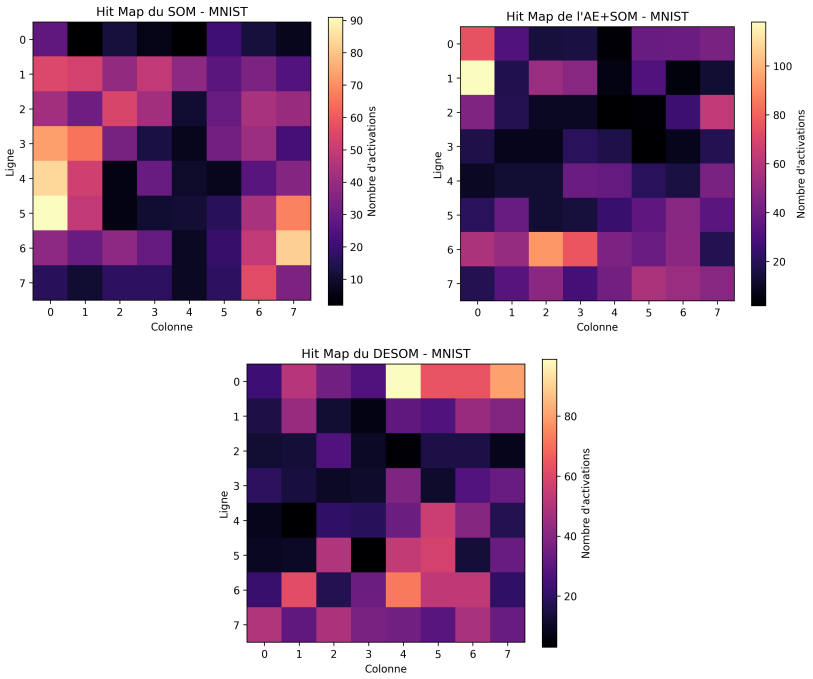

FIGURE 3.5 – Hit map pour les trois modèles sur le jeu de données MNIST.

Pour MNIST, les trois modèles présentent une repartition relativement hétérogène des activations. Le SOM contient plusieurs prototypes très chargés et d'autres très peu utilisés, ce qui veut dire qu'il y a une importante concentration des données dans certaines régions de la carte. L'approche AE+SOM semble avoir aussi plusieurs prototypes faiblement utilisés voire vides, mais les activations fortes sont plus localisées. Cela peut s'interpréter que l'espace latent concentre les représentations latentes dans quelques prototypes, tandis que d'autres restent peu utilisés. Le DESOM montre aussi une répartition hétérogène : certaines zones sont fortement utilisées, notamment en bordure de carte, tandis que plusieurs prototypes restent faiblement activés. Cependant, la carte semble être mieux repartie que pour AE+SOM. Cela suggère que, sur MNIST, aucune des trois cartes n'utilise parfaitement l'ensemble des prototypes de manière uniforme.

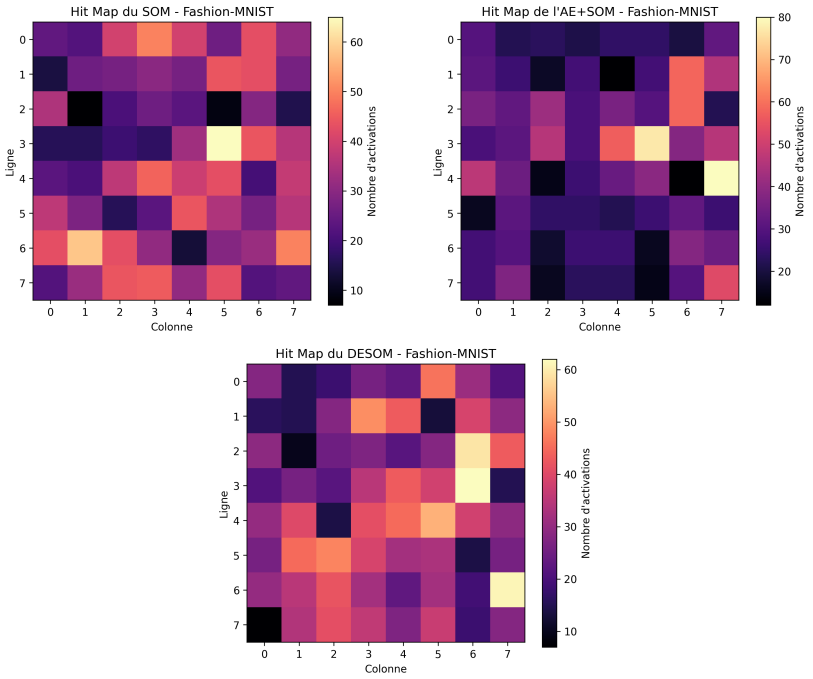

FIGURE 3.6 – Hit map pour les trois modèles sur le jeu de données Fashion-MNIST.

Pour Fashion-MNIST, les carte semble globalement plus équilibrée que pour MNIST. Le SOM et AE+SOM ont encore quelque prototypes surchargés, mais les données sont mieux réparties sur la carte. Le DESOM semble présenter une meilleure répartition des données d'entrée sur la carte, avec moins de prototypes vides et une répartition relativement homogène des activations.

Ainsi, pour les deux jeux de données, l'observation des hit maps complète l'analyse de U-matrices et des métriques quantitatives en montrant comment les observations sont distribuées sur les cartes. Elles indiquent que la qualité d'un modèle ne dépend pas seulement de la proximité entre prototypes et données, mais aussi de l'utilisation effective des prototypes de la carte. Sur MNIST, la répartition reste assez hétérogène pour les trois modèles, tandis que sur Fashion-MNIST, le DESOM semble produire une utilisation légèrement plus équilibrée de la carte.

Cartes colorées

Nous avons également produit des cartes colorées pour les trois modèles sur les deux jeux de données. Les vecteurs des prototypes sont de dimension 10 pour le AE+SOM et le DESOM et de dimension 784 pour le SOM. Afin de visualiser qualitativement les relations de similarité entre prototypes, nous avons projeté ces vecteurs en trois dimensions à l'aide d'une PCA, puis converti les trois valeurs obtenues en couleurs RGB. Les figures 3.7 et 3.8 présentent les cartes colorées pour les trois modèles sur les deux jeux de données.

Il convient toutefois de noter que cette représentation est avant tout qualitative. Les couleurs ne correspondent pas à des labels de classes, mais à une représentation visuelle de la position relative des prototypes après projection. Elle permet de visualiser la continuité relative entre prototypes voisins à l'intérieur d'une même carte.

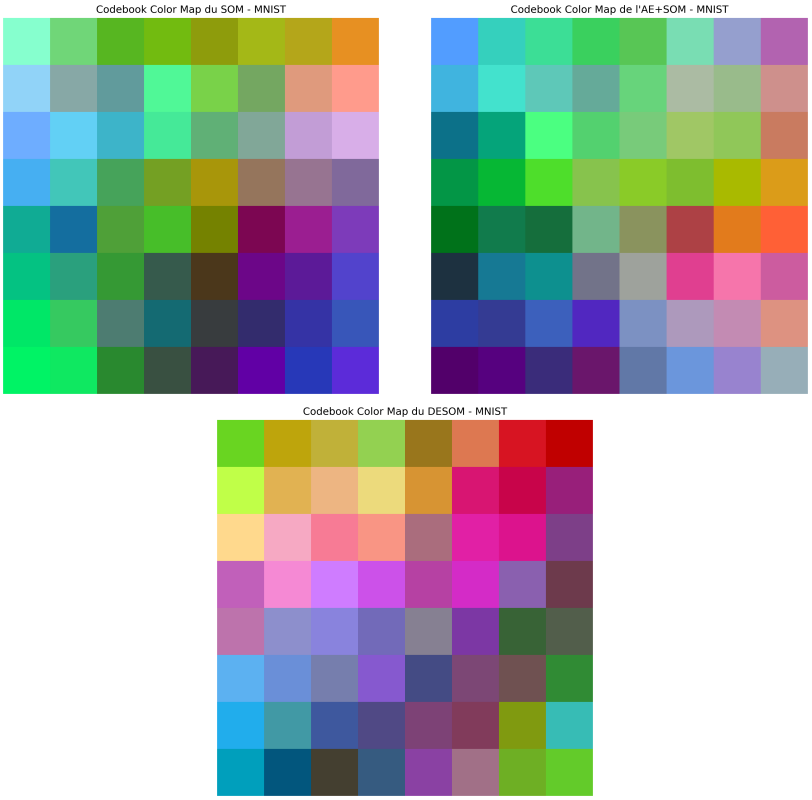

FIGURE 3.7 – Visualisation des cartes colorées pour les trois modèles sur le jeu de données MNIST.

Les cartes dans la figure 3.7 montrent des structures différentes pour chaque modèle. La carte du SOM présente une transition plus progressive, cela est cohérent avec le résultat obtenu pour l'erreur de topologie qui indique une meilleure préservation de la topologie. Ses frontières entre groupe de prototypes sont moins visibles. Pour l'approche AE+SOM, les transition sont plus marquées, mais les fontières se distinguent mieux que pour le SOM. La transition de la carte du DESOM est la plus hétérogène, cela est cohérent avec les résultats de l'erreur de topologie.

Cependant, les frontières de la carte du DESOM sont les plus marquées. Cela s'explique comme dit précédement par le fait que le DESOM privilégie la quantification dans l'espace latent, ce qui peut se faire au détriment de la préservation de la topologie.

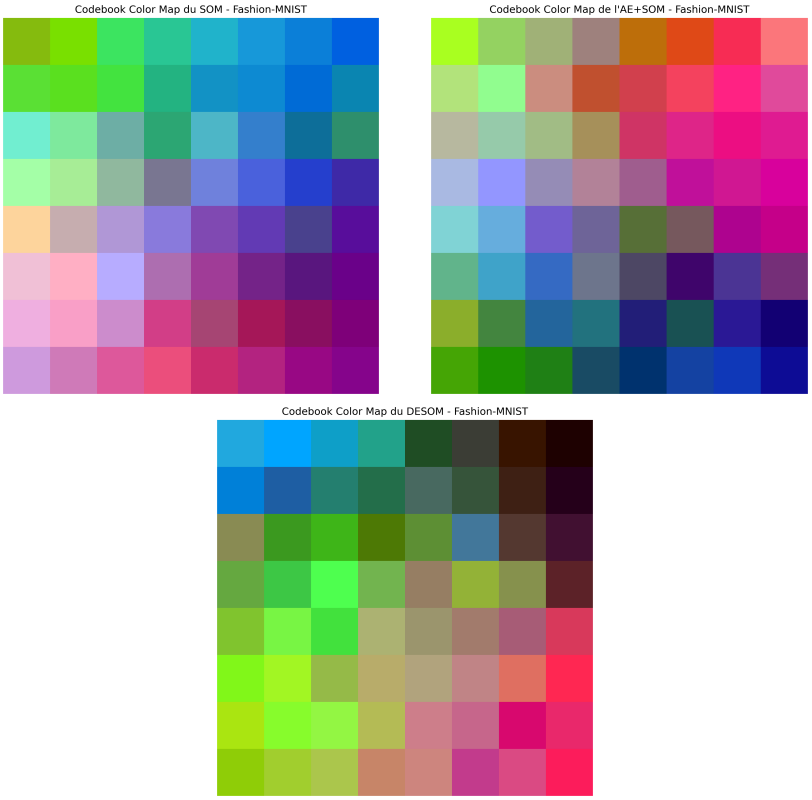

FIGURE 3.8 – Visualisation des cartes colorées pour les trois modèles sur le jeu de données Fashion-MNIST.

Pour Fashion-MNIST, les trois cartes paraissent globalement plus régulières que sur MNIST. La carte du SOM présente des transitions relativement lisses, cela suggère une organisation progressive des prototypes sur la grille. Cette observation est cohérente avec son erreur topologique plus faible. Nous remarquons aussi une amélioration de la distiction des frontières. La carte de AE+SOM montre aussi une transitionassez régulière, avec des zones de couleurs proches regroupées localement. Les transitions semblent être un peu plus marquées que pour le SOM, ce qui peut indiquer une organisation moins progressive que pour le SOM. Pour ce qui est de la carte du DESOM, nous avons aussi une structure globalement plus régulière, mais avec des transitions plus abruptes entre prototypes voisins. Cette observation est cohérente avec les résultats quantitatifs obtenus pour ce jeu de données, où le DESOM présente une erreur de topologie plus élevée que les deux autres modèles.

Ainsi, les cartes colorées pour les deux jeux de données confirment qualitativement le compromis observé dans les métriques : le DESOM privilégie l'apprentissage d'un espace latent favorable à la quantification, en échange d'une organisation topologique moins régulière.

Visualisation des prototypes

Les figures 3.9 et 3.10 présentent les prototypes appris par chaque modèle pour les deux jeux de données. Cette visualisation permet d'observer directement ce que représentent les prototypes des cartes. Pour le SOM, les prototypes sont directement visualisés comme des images de 28×28 pixels, tandis que pour les approches AE+SOM et DESOM, les prototypes sont appris dans l'espace latent de dimension 10. Nous les reconstruisons à

l'aide du décodeur correspondant pour être visualisés sous forme d'images de 28×28 pixels.

Nous observons que, pour MNIST, les trois modèles ont des prototypes globalement reconnaissables. Le SOM a certains prototypes qui paraissent ambigus et semblent mélanger des caractéristiques de plusieurs chiffres. Cela peut être lié à la difficulté de représenter toute la variabilité du jeu de données dans une carte de taille limitée.

L'approche AE+SOM semble produire aussi des prototypes reconnaissables. Nous observons des transitions relativement progressives entre certaines régions de la carte. Plusieurs zones montrent des familles qui semble proche visuellement, par exemples les prototypes qui resembles aux chiffres 7 et 1, ou encore aux chiffres 0 et 6. Cela pourrait être lié au fait que l'espace latent conserve une partie de la structure visuelle des données.

Le DESOM a aussi des prototypes reconnaissable, mais certains apparaissent moins distincts ou plus flous. Une cause possible de cette observation est l'apprentissage de ses prototypes : ils sont appris dans un espace latent, puis reconstruits par le décodeur. Il y a donc une certaine dépendance de la qualité de reconstruction de l'auto-encodeur. Un prototype peut devenir donc une moyenne de plusieurs exemples proches. Ces observations sont cohérentes avec les résultats précédents : le DESOM obtient une très faible erreur de quantification dans l'espace latent, mais sans garantir une organisation visuelle parfaitement progressive sur MNIST.

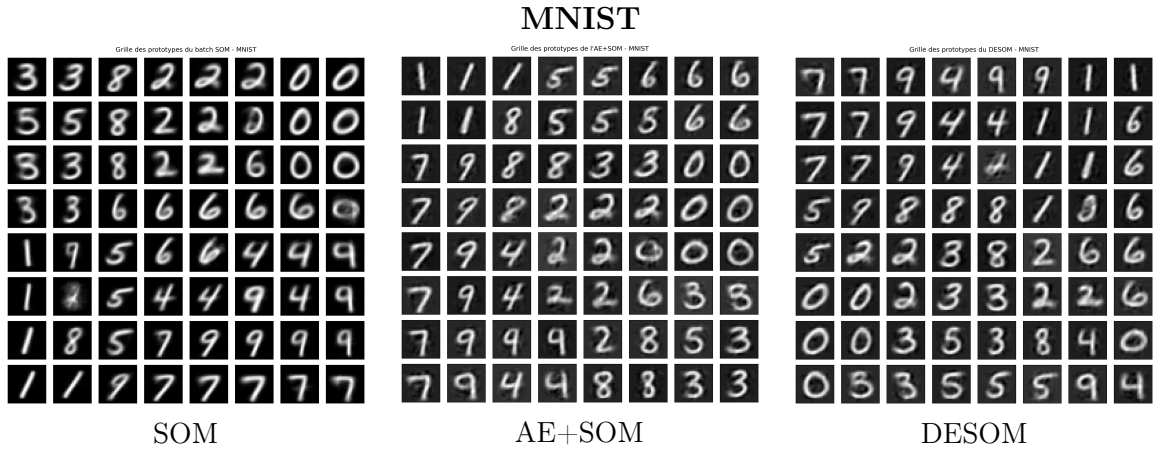

FIGURE 3.9 – Visualisation des prototypes pour les trois modèles sur le jeu de données MNIST.

Les trois cartes des prototypes sont aussi interprétable pour Fashion-MNIST. Le SOM apprend des prototypes correspondant à différentes catégories d'objets, comme des pulls, des pantalons, des sacs ou des chaussures. La carte présente une organisation visuelle assez claire, avec certaines régions associées à des familles d'objets similaires. Par exemple, les chaussures semblent être regroupées dans la partie de la carte, tandis que les vêtements du haut sont regroupés dans une autre région.

L'approche AE+SOM produit aussi des prototypes relativement nets, notamment pour les chaussures, les pantalons et les vêtements du haut. Cependant, il semble avoir des transitions plus abruptes entre catégories, par exemple entre les vetement et les chaussures. Cela rejoint les résultats précédents, où cette approche n'obtient pas les meilleurs résultats en termes de préservation de la topologie.

Le DESOM apprend aussi des prototype reconnaissables, mais certaines transitions semblent plus marquées entre prototypes voisins, par exemple lorsq'une région de chaussure est proche d'une région de vêtements. Cela est cohérent avec les résultats quantitatifs

obtenus pour ce jeu de données, où le DESOM présente une erreur de topologie plus élevée que les deux autres modèles. Cependant, les prototypes du DESOM semblent globalement plus nets, ce qui peut être lié au fait que certaines catégories de fashion-mnist possèdent des structures visuelles plus régulières, ce qui facilite la reconstruction de prototypes moyens par le décodeur.

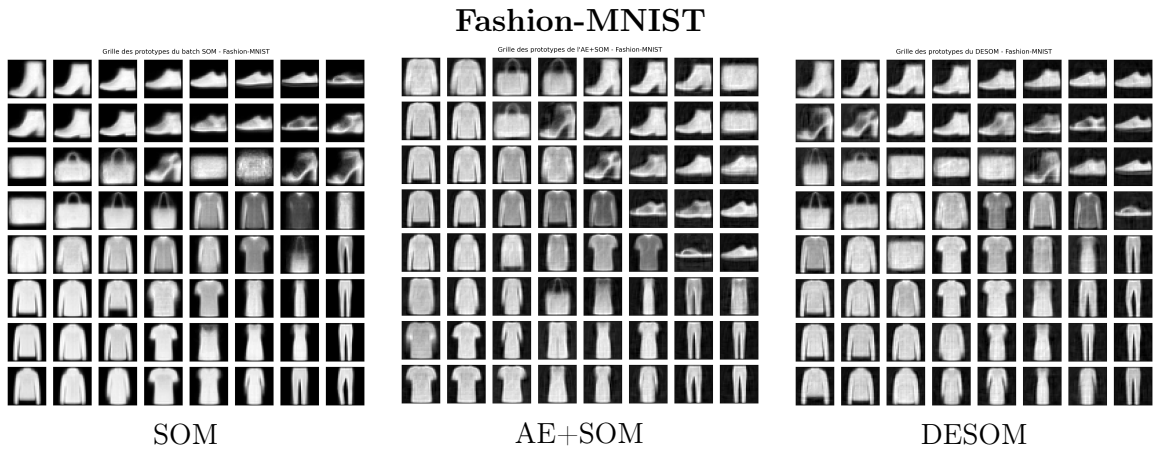

FIGURE 3.10 – Visualisation des prototypes pour les trois modèles sur le jeu de données Fashion-MNIST.

Ainsi, grâce à les visualisation de plusieurs types de cartes, nous avons pu renforcer qualitativement les observations faites à partir des métriques quantitatives. Cela nous a permis aussi de mieux comprendre les compromis entre les différents modèles, notamment en ce qui concerne la quantification et la préservation de la topologie.

3.3.3 Courbes de l'erreur de quantification et fonction de perte

Cette section présente les courbes de l'erreur de quantification pour le batch SOM et AE+SOM, ainsi que les courbes de la fonction de pertes pour le DESOM. Ces courbes permettent l'analyse de la convergences des différents modèles, ainsi que de l'évolution de l'erreur de quantification au cours de l'entraînement.

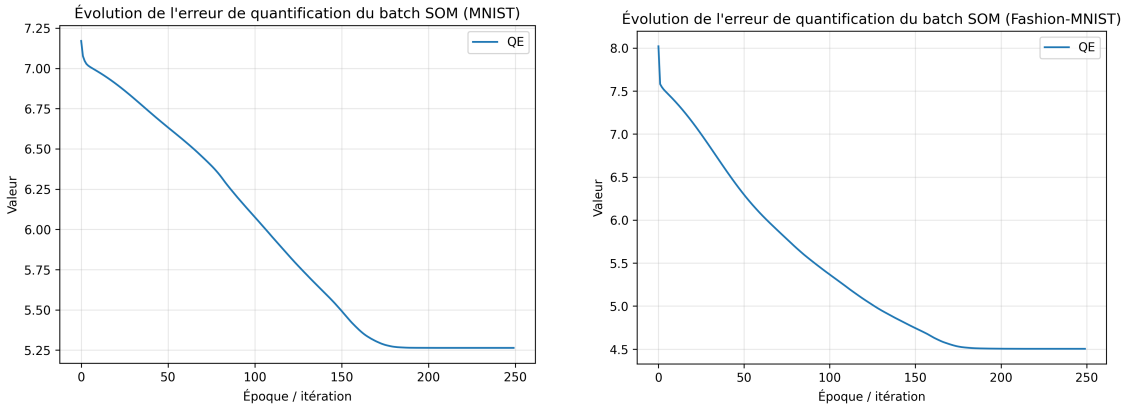

FIGURE 3.11 – Courbes d'erreur de quantification pour le batch SOM sur le jeu de données MNIST et Fashion-MNIST.

Pour le batch SOM, les courbes d'erreur de quantification montrent une diminution progressive de l'erreur au cours des itérations, indiquant que les prototypes s'ajustent

progressivement pour mieux représenter les données d'entrée. Dans les deux cas, l'erreur de quantification diminue rapidement au début de l'entraînement, puis ralentit après un certain nombre d'itérations, aux alentours de 175 à 190 itérations et elle atteint un plateau vers la fin de l'entraînement. Ce plateau suggère que le modèle a convergé et que les prototypes ne sont plus modifiés de manière significative.

Nous observons également que la convergence est relativement régulière pour les deux jeux de données. Sur MNIST, l'erreur de quantification se stabilise autour de 5.25, tandis que sur Fashion-MNIST elle se stabilise autour de 4.5.

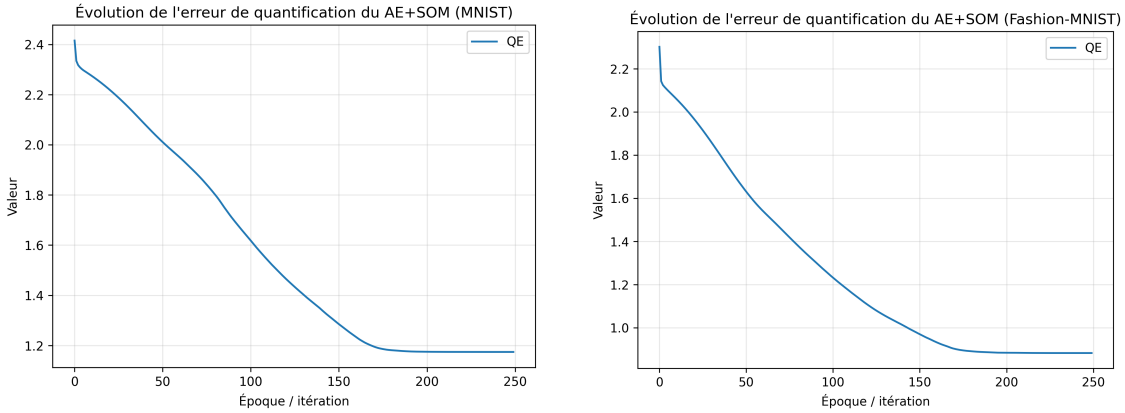

FIGURE 3.12 – Courbes d'erreur de quantification pour l'approche AE+SOM sur le jeu de données MNIST et Fashion-MNIST.

La figure 3.12 montre la courbe d'erreur de quantification pour le SOM entraîné sur les représentations latentes de l'auto-encodeur. Cette courbe montre une diminution progressive de l'erreur au cours des itérations, indiquant que les prototypes du SOM s'adaptent progressivement aux représentations produites par l'encodeur. Sur MNIST, la QE se stabilise autour de 1.2, tandis que sur Fashion-MNIST elle se stabilise autour de 0.9. Ces valeurs pour le SOM sur les représentations latentes sont plus faibles que pour le batch SOM, ce qui suggère que l'auto-encodeur a réussi à produire un espace latent plus adapté pour la quantification par le SOM.

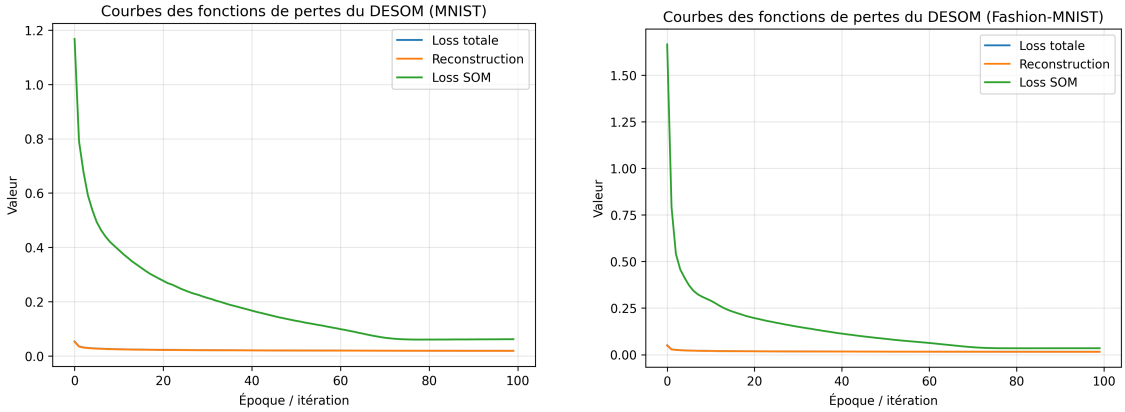

FIGURE 3.13 – Courbes d'erreur de quantification pour le DESOM sur le jeu de données MNIST et Fashion-MNIST.

La figure 3.13 montre la courbe de la fonction de perte de la composante SOM du DESOM. Pour les deux jeux de données, la composante SOM diminue fortement au

début de l'entraînement, puis décroît plus lentement avant d'atteindre un plateau. Cela indique que les prototypes s'adaptent rapidement aux représentations latentes produites par l'encodeur, puis se stabilisent progressivement. Cette évolution est cohérente avec les résultats quantitatifs, où le DESOM obtient une très faible erreur de quantification dans l'espace latent.

La perte de reconstruction reste beaucoup plus faible que la composante SOM sur l'échelle affichée. Ainsi, la perte totale est visuellement très proche de la courbe d'erreur de construction. L'hyperparamètre γ est en partie responsable de cette observation, car il est fixé à une valeur relativement faible ($\gamma = 0.001$), donnant ainsi plus d'importance à la composante de reconstruction. Cette observation peut expliquer pourquoi le DESOM obtient une très bonne quantification latente, tout en ne garantissant pas toujours la meilleure préservation topologique, comme observé sur MNIST.

Sur Fashion-MNIST, la composante SOM diminue de manière régulière et atteint un plateau plus bas que sur MNIST. Cela est cohérent avec les résultats précédents, où le DESOM obtient les meilleurs scores pour l'ensemble des métriques sur Fashion-MNIST.

3.3.4 Impact de la taille de la grille

Dans cette section, nous avons analysé l'impact de la taille de la grille sur les performances des modèles. Pour cela, nous avons entraîné les trois modèles avec différentes tailles de grilles : $(6, 6), (8, 8), (10, 10), (12, 12)$ et $(15, 15)$. Nous avons ensuite calculé les métriques d'évaluation pour chaque taille de grille afin d'observer comment la performance évolue en fonction de ce paramètre.

La figure 3.14 présente l'évolution de l'erreur de quantification selon la taille de la grille pour les trois modèles, sur MNIST et Fashion-MNIST. Cette figure montre que, pour les deux jeux de données, l'erreur de quantification diminue lorsque le nombre de prototypes augmente. Cela est attendu : une grille plus grande permet d'avoir plus de prototypes, ce qui peut permet de représenter plus finement la diversité des données d'entrée.

Cependant, cette diminution de QE reste relativement faible pour AE+SOM et DESOM, en particulier pour les tailles de grilles plus grandes. Cela suggère que l'espace latent appris permet déjà une quantification plus compacte, l'ajout supplémentaire de prototypes apporte relativement peu d'amélioration.

Nous observons également que, pour les plus grandes tailles de grille, les courbes tendent à se stabiliser. Cela indique qu'au-delà d'une certaine taille, augmenter le nombre de prototypes n'apporte pas nécessairement une amélioration significative de la quantification.

Pour le SOM nous observons une diminution, un peu plus marquée, de l'erreur de quantification lorsque la grille augmente. Cela peut être lié au fait que le SOM est directement entraîné sur les données d'entrée, et une plus grande grille permet d'avoir plus de prototypes pour mieux représenter la diversité des données d'entrée.

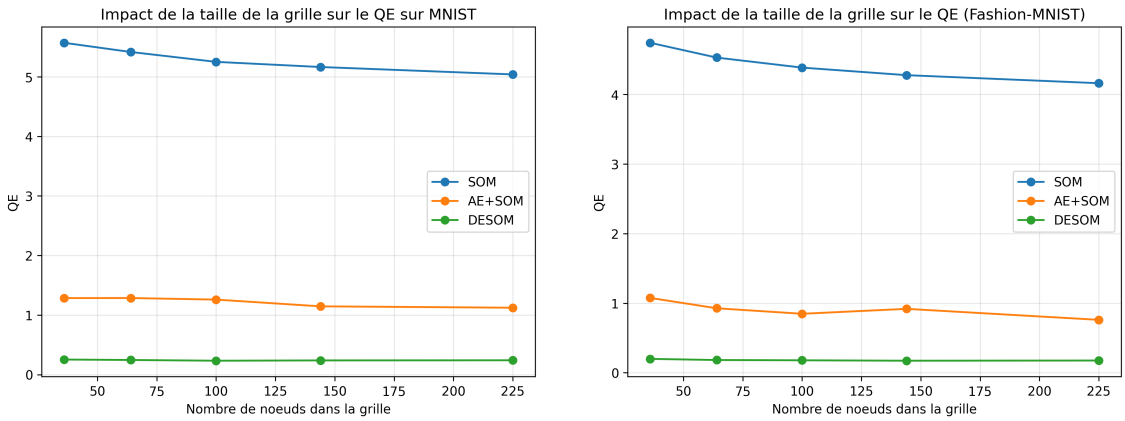

FIGURE 3.14 – Évolution de l'erreur de quantification selon la taille de la grille pour les trois modèles sur MNIST et Fashion-MNIST.

Pour l'erreur de topologie, la figure 3.15 montre que l'augmentation de la taille de la grille n'entraîne pas une amélioration systématique de la préservation de la topologie. Contrairement à l'erreur de quantification, qui tend généralement à diminuer lorsque le nombre de prototypes augmente, l'erreur de topologie présente des variations irrégulières. Une hypothèse explicative de ce phénomène est la sensibilité du SOM à l'initialisation aléatoire des poids : comme pour l'algorithme k-means, l'apprentissage peut converger vers des configurations sous-optimales selon l'initialisation, introduisant une variabilité dans l'erreur de topologie indépendamment de la taille de la grille.

Sur MNIST, l'approche AE+SOM obtient les valeurs les plus stables et atteint son meilleur score de TE pour les tailles de grilles intermédiaires. Le SOM présente une erreur plus faible pour les petites grilles, mais celle-ci augmente lorsque le nombre de nœuds devient plus important. Le DESOM présente les variations les plus marquées et obtient globalement les erreurs de topologie les plus élevées sur ce jeu de données.

Sur Fashion-MNIST, les courbes montrent une tendance similaire, avec une augmentation de l'erreur de topologie pour les tailles de grilles plus grandes, en particulier pour le DESOM. Le SOM et l'approche AE+SOM obtiennent des résultats relativement proches pour les tailles de grille intermédiaires et grandes. Le DESOM, en revanche, voit son erreur topologique augmenter avec la taille de la grille, ce qui suggère que l'organisation topologique de la carte devient plus difficile à stabiliser lorsque le nombre de prototypes augmente.

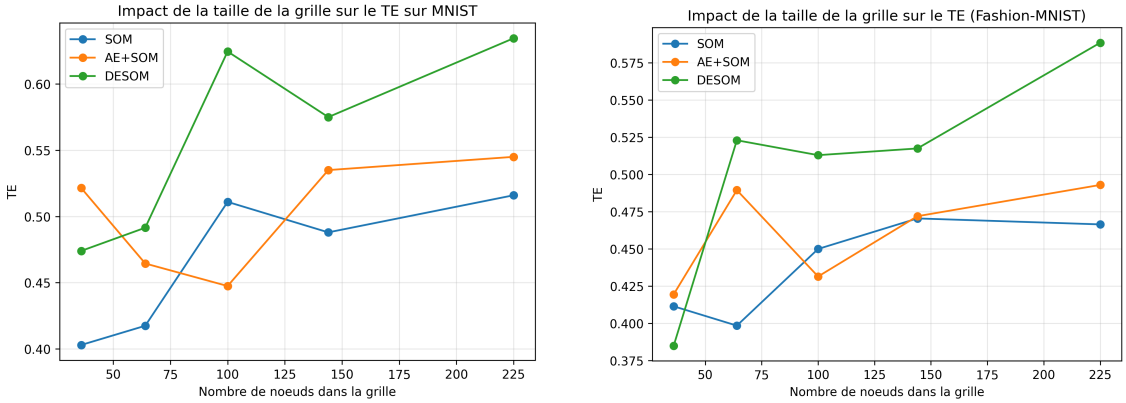

FIGURE 3.15 – Évolution de l'erreur topologique selon la taille de la grille pour les trois modèles sur MNIST et Fashion-MNIST.

Nous observons dans la figure 3.16 que la pureté a tendance à augmenter lorsque la taille de la grille augmente, pour les trois modèles. Cette tendance est particulièrement visible pour le SOM classique. Cette augmentation globale s'explique en partie par la définition de la pureté : lorsque le nombre de prototypes augmente, les données peuvent être réparties dans un plus grand nombre de clusters. Chaque cluster contient alors potentiellement des observations plus homogènes, ce qui peut conduire à une meilleure correspondance entre les prototypes et les classes d'origine.

Pour le jeu de données MNIST, le AE+SOM obtient des scores de pureté plus élevés notamment pour les grilles $(10, 10)$ et $(15, 15)$. Il faut remarquer que l'augmentation du nombre de prototypes peut réduire l'écart entre les modèles sur cette métrique. Pour la plus grande grille, les trois modèles obtiennent des scores de pureté relativement proches.

Pour le jeu de données Fashion-MNIST, le DESOM obtient les meilleurs scores de pureté pour la plupart des tailles de grilles, cela peut s'expliquer par l'apprentissage conjoint du DESOM qui permet d'obtenir des représentations latentes favorables au regroupement des classes. Néanmoins, pour la plus grande grille, les performances des trois modèles deviennent relativement proches. Le AE+SOM obtient le score le plus grand avec la grille $(15, 15)$.

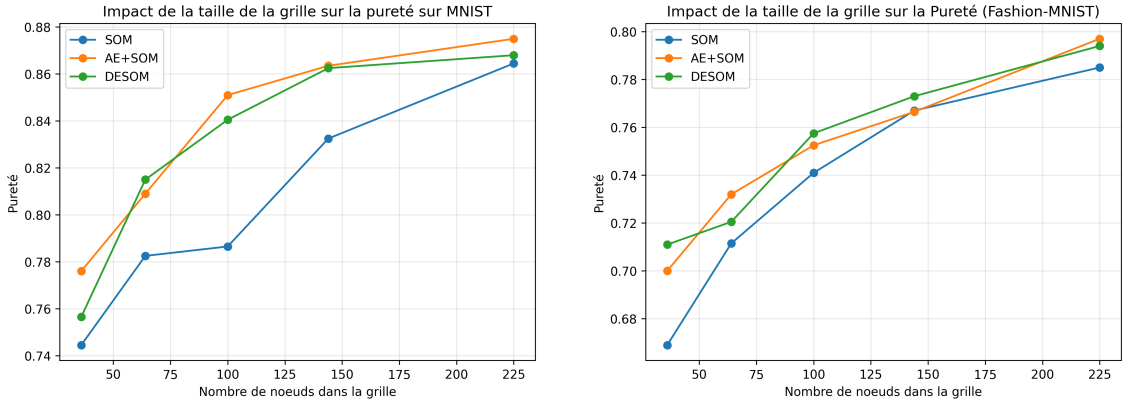

FIGURE 3.16 – Évolution de la pureté selon la taille de la grille pour les trois modèles sur MNIST et Fashion-MNIST.

L'évolution du NMI en fonction de la taille de la grille, présentée dans la figure 3.17, montre une tendance différente à celle observée pour la pureté. Alors que la pureté augmente

globalement avec le nombre de prototypes, le NMI a plutôt tendance à diminuer lorsque la taille de la grille devient plus grande. Cela peut être lié au fait que le NMI mesure la similarité entre les clusters formés, et une plus grande taille de grille peut fragmenter les classes sur un plus grand nombre de prototypes. Cette fragmentation peut réduire la similarité globale entre les clusters obtenus et les classes réelles, ce qui se traduit par une baisse du NMI.

Pour le jeu de données MNIST, le AE+SOM obtient les meilleurs résultats pour toutes les tailles de grilles. Cependant, lorsque la taille de la grille augmente d'avantage, le NMI diminue pour AE+SOM et DESOM, ainsi se rapprochant des performances du SOM. Le SOM présente une diminution plus raide, mais ses scores restent assez proches de ceux des autres modèles pour les grandes grilles. Ces résultats montrent qu'une augmentation du nombre de prototypes ne conduit pas nécessairement à une meilleure correspondance globale avec les classes réelles.

Pour fashion-MNIST, les trois modèles présentent également une diminution globale du NMI lorsque la taille de la grille augmente. Le DESOM obtient le meilleurs scores pour la plus petite grille et reste compétitif pour les tailles intermédiaires, mais son avantage n'est pas systématique. Pour les grandes grilles, les scores des trois modèles deviennent relativement proches. Cela indique que, pour Fashion-MNIST également, l'augmentation de la taille de la grille peut améliorer la pureté tout en diminuant la cohérence globale entre les clusters et les classes

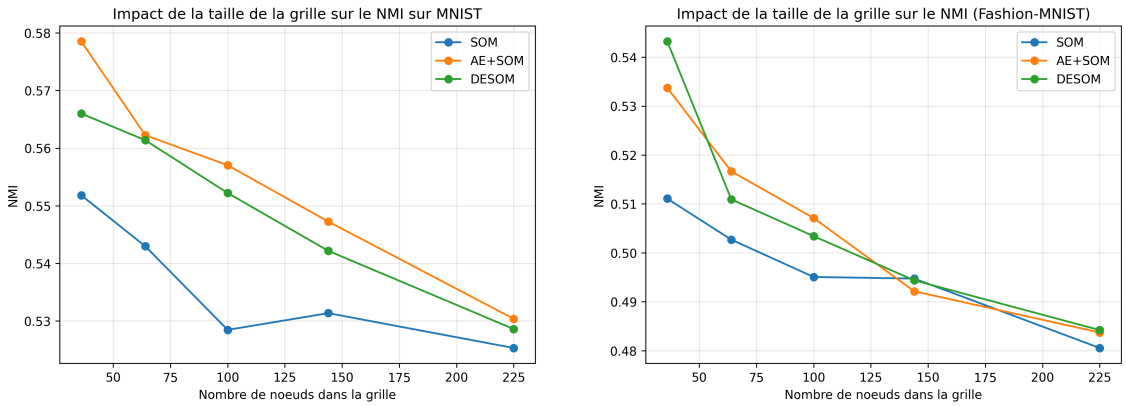

FIGURE 3.17 – Courbes de NMI selon la taille de la grille pour les trois modèles sur MNIST et Fashion-MNIST.

Ainsi, cette étude montre que la taille de la grille a un impact significatif pour l'erreur topologique, la pureté et le NMI. Une grille plus grande permet d'avoir une meilleure pureté, mais elle peut dégrader le NMI et l'erreur de topologie. Quant à l'erreur de quantification, elle tend à diminuer avec l'augmentation de la grille, mais cette diminution reste relativement faible voir inexistante pour les approches basées sur l'apprentissage d'un espace latent, ce qui suggère que l'espace latent appris est déjà très adapté pour la quantification par le SOM. Dans nos expériences, les tailles intermédiaires semblent offrir un compromis plus raisonnable entre quantification, pureté, NMI et préservation topologique.

Chapitre 4

Conclusion

Dans ce rapport, nous avons étudié les cartes auto-organisées qui constituent une familles de modèles d'apprentissage compétitif non supervisé. Nous avons commencé par présenter leur principes général, en distingant les prototypes utilisés des neurones formels utilisés dans les réseaux de neurones. Nous avons continué en décrivant leur structure de base, leur algorithme de mis à jour, ainsi qu'un variant de SOM appelé batch SOM qui est plus adapté aux applications pratiques.

Nous avons également discuté sur certaines limites du SOM, notamment leur coût de calcul lorsque la taille des données augmente et et les difficultés liées à la préservation simultanée de la quantification et de la topologie. Nous avons présenté quelques extensions qui proposent des solutions à ces problèmes.

Dans la seconde partie de ce document, nous nous sommes concentrés sur une approche hybride particulière qui intégre le SOM dans le cadre du deep learning : DESOM (Deep Embedded Self-Organizing Map). Cette méthode combine un auto-encodeur avec une carte auto-organisée afin d'apprendre de manière conjointe une représentation latente des données d'entrée et une organisation topologique des prototypes. Une étude empirique a été réalisée sur deux sous-ensembles des jeux de données : MNIST et Fashion-MNIST.

Les résultats obtenus montrent que le DESOM est une solution intéressante pour le traitement des données de grande dimension. Il permet d'obtenir une erreur de quantification très faible dans l'espace latent, indiquant que l'espace latent appris est bien adapté pour la quantification par le SOM (SOM-friendly). Cependant, l'amélioration de la quantification a un compromis en termes de préservation de la topologie de la carte. L'erreur de quantification faible est acompagnée d'une erreur topologique plus élevée dans nos expériences.

L'étude empirique a également montré que les approches comme AE+SOM et DESOM peuvent obtenir de bonne performances en clustering, notamment en terme de pureté et de NMI. Néanmoins, l'écart entre les modèles reste faible et les résultats dépendent du choix des hyperparamètres, de la taille de la grille et de la qualité de l'espace latent appris.

Il existe plusieurs études possibles pour prolonger ce travail. Une étude sur l'hyperparamètre γ du DESOM pourrait être intéressante, car ce hyperparamètre contrôle l'importance relative de l'organisation faite par le SOM. Dans ce document γ est fixé à une valeur relativement faible. Un meilleur choix de γ pourrait potentiallement diminuer l'erreur topologique tout en coservant une bonne quantification. Il serait également pertinent de produire les expérience sur les jeux de données complets, afin d'obtenir des résultats plus robustes.

En conclusion, les carte auto-organisées seul sont aujourd'hui moins compétitives

que certaines approches modernes dans un contexte où les données sont de plus en plus volumineuses de grande dimension et complexes. Cependant, leur principe reste pertinent, car elles offrent une manière de combiner quantification, clustering et visualisation topologique. Les méthodes récentes comme le DESOM ou les approches appliquées telles que FlowSOM demontrent que le principe des SOM reste d'actualité, en particulier lorsqu'elles sont combinées avec des techniques d'apprentissage automatique. Ainsi, les SOM restent un outil de référence en apprentissage non supervisé et comme base pour certains modèles hybrides interprétables et adaptés à l'analyse de données complexes.

Bibliographie

[1] Komi Mensah Agboka, Elfatih M. Abdel-Rahman, Daisy Salifu, Brian Kanji, Frank T. Ndjomatchoua, Ritter A.Y. Guimapi, Sunday Ekesi, and Landmann Tobias. Towards combining self-organizing maps (som) and convolutional neural network (cnn) for improving model accuracy : Application to malaria vectors phenotypic resistance. *MethodsX*, 14 :103198, 2025.

[2] Christopher M. Bishop. *Pattern Recognition and Machine Learning*, chapter 9, pages 423–459. Information Science and Statistics. Springer, 2006.

[3] Ingwer Borg and Patrick J. F. Groenen. *Modern Multidimensional Scaling : Theory and Applications*. Springer, 2 edition, 2005.

[4] Arthur Flexer. Limitations of self-organizing maps for vector quantization and multidimensional scaling. *Advances in neural information processing systems*, 9, 1996.

[5] Florent Forest, Mustapha Lebbah, Hanene Azzag, and Jérôme Lacaille. Deep embedded self-organizing maps for joint representation learning and topology-preserving clustering. *Neural Computing and Applications*, 33 :1–31, 2021.

[6] Axel Guérin, Pierre Chauvet, and Frédéric Saubion. A survey on recent advances in self-organizing maps, 2024.

[7] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2) :251–257, 1991.

[8] Wakeel Hussain, Miao Luo, Muhammad Ali, Izhar Sadiq, Erasto E. Kasala, Tariq Aziz, and Zuriyat Batool. Hybrid modeling of deep neural networks and unsupervised machine learning algorithms for missing well log prediction based on geological lithofacies similarities. *Journal of Applied Geophysics*, 241 :105846, 2025.

[9] Teuvo Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78 :1464–1480, 1990.

[10] Teuvo Kohonen. Essentials of the self-organizing map. *Neural Networks*, 37 :52–65, 2013.

[11] Jérôme J. Mariette, Fabrice Rossi, Madalina Olteanu, and Nathalie Villa-Vialaneix. Accelerating stochastic kernel SOM. In *XXVth European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, page 6 p., Bruges, Belgium, April 2017.

[12] Leland McInnes, John Healy, and James Melville. UMAP : Uniform manifold approximation and projection for dimension reduction, 2018.

[13] John W. Sammon. A nonlinear mapping for data structure analysis. *IEEE Transactions on Computers*, C-18(5) :401–409, 1969.

[14] Stephane Marchand-Maillet. Cours d'intelligence artificielle, 2026.

[15] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9 :2579–2605, 2008.

[16] Sofie Van Gassen, Britt Callebaut, Mary J. Van Helden, Bart N. Lambrecht, Piet Demeester, Tom Dhaene, and Yvan Saeys. Flowsom : Using self-organizing maps for visualization and interpretation of cytometry data. *Cytometry Part A*, 87 :636–645, 2015.

[17] Wikipedia contributors. Neurona, 2025.

[18] Wikipedia contributors. Artificial neuron, 2026.

[19] Wikipedia contributors. Dirac delta function, 2026.