Ensemble de sommets indépendant maximal appliqué au pooling sur graphes

Stevan Stanovic, Benoit Gaüzère, Luc Brun

Ensemble de sommets indépendant maximal appliqué au pooling sur graphes

Stevan Stanovic[1] Benoit Gaüzère[2] Luc Brun[1]

[1] Normandie Univ, ENSICAEN, CNRS, UNICAEN, GREYC UMR 6072, 14000 Caen, France
[2] Normandie Univ, INSA de Rouen, Univ. rouen, Univ. Le Havre, LITIS EA 4108, 76800 Saint-Étienne-du-Rouvray, France

{stevan.stanovic,luc.brun}@ensicaen.fr, benoit.gauzere@insa-rouen.fr

Résumé

Les réseaux de neurones convolutifs (CNN) ont permis des avancées majeures dans la classification d'images grâce à la convolution et au pooling. En particulier, le pooling sur image transforme une grille discrète connexe en une grille réduite de même connexité et permet aux fonctions de réduction de prendre en compte tous les pixels de l'image. Cependant, un pooling satisfaisant de telles propriétés n'existe pas pour les graphes. En effet, certaines méthodes sont restreintes à la sélection de sommets selon leur importance. Ceci induit la création de graphes réduits non connexes et une perte d'information importante. D'autres méthodes apprennent un partitionnement flou des sommets causant une hyper-connectivité du graphe réduit. Dans cette publication, nous proposons de pallier ces problématiques à l'aide de notre méthode de pooling, nommée MIVSPool. Elle est basée sur une sélection de sommets appelés sommets survivants à l'aide d'un ensemble de sommets indépendant maximal (MIVS) et d'une affectation des autres sommets aux survivants. Par conséquent, notre méthode donne la garantie de préserver la totalité de l'information du graphe lors de sa réduction. Les résultats expérimentaux montrent une augmentation de l'exactitude de la classification sur plusieurs jeux de données standards.

Mots Clef

Réseaux de neurones sur graphes, Pooling sur graphes, Classification de graphes, Ensemble de sommets indépendant maximal.

Abstract

Convolutional neural networks (CNN) have enabled major advances in image classification through convolution and pooling. In particular, image pooling transforms a connected discrete grid into a reduced grid with the same connectivity and allows the reduction functions to take into account all the pixels of the image. However, a pooling satisfying such properties does not exist for graphs. Indeed, some methods are restricted to the selection of vertices according to their importance. This induces the creation of disconnected reduced graph and an important loss of information. Other methods learn a fuzzy clustering of vertices causing a hyper-connectivity of the reduced graph. In this publication, we propose to overcome these problems using a new pooling method, named MIVSPool. This method is based on a selection of vertices called surviving vertices using a maximal independent vertex set (MIVS) and an assignment of the remaining vertices to the survivors. Consequently, our method guarantees to preserve all the information of the graph during its reduction. Experimental results show an increase in accuracy for graph classification on various standards datasets.

Keywords

Graph Neural Networks, Graph Pooling, Graph Classification, Maximal Independant Vertex Set.

1 Introduction

Les réseaux de neurones convolutifs (CNNs) [16] ont permis de grandes avancées dans la classification des images. Une image peut être définie comme une grille discrète connexe et cette propriété permet de définir simplement les opérations de convolution et de pooling.

Néanmoins, les réseaux sociaux, les molécules et les infrastructures de circulation ne sont pas décrites par une grille mais par des graphes. La convolution et le pooling doivent donc être adaptés pour ces réseaux et ce domaine de l'apprentissage profond s'appelle les réseaux de neurones sur graphes (GNNs) [11, 21].

L'adaptation de la convolution aux graphes est réalisée par l'apprentissage d'une nouvelle représentation des sommets en prenant en compte l'information des voisins de celui-ci. Aujourd'hui, il existe plusieurs convolutions [7, 12, 14, 22] qui peuvent être généralisées comme décrit dans [1, 10].

Pour le pooling sur graphes, l'adaptation est effectuée soit en sélectionnant des sommets [2, 6, 9, 15, 17], soit en apprenant un partitionnement flou de données [25].

Nous allons détailler la convolution et le pooling sur graphes ainsi que la reconstruction traditionnelle de la matrice d'adjacence du graphe réduit en Section 2. Dans la Section 3, nous présenterons notre méthode MIVSPool et nous expliquerons en quoi elle diffère des autres pooling.

Ensuite, nous présenterons en Section 4 les expériences réalisées. Nous y étudierons les propriétés de notre réseau, et enfin, nous comparerons notre méthode aux autres méthodes de l'état de l'art.

2 État de l'art

Les GNNs sont des réseaux de neurones appliqués aux graphes. Ils sont inspirés par les CNNs et sont comme ces derniers basés sur des opérations de convolution et de pooling.

2.1 Convolution sur graphes

Les convolutions sur graphes sont représentées par deux familles : les convolutions spectrales et spatiales.
Les approches spectrales, définies par Bruna et al. [5], sont basées sur une représentation des graphes dans le domaine de Fourier. Elles sont définies à l'aide de filtres utilisant le graphe Laplacien et sa décomposition en éléments propres. Par exemple, dans GCN [14], l'opération de convolution est définie à la couche l comme suit :

$$X^{(l+1)} = \sigma(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}X^{(l)}W^{(l)}) \quad (1)$$

où $\hat{A} = A + I$ est la matrice d'adjacence avec des boucles et ses degrés associés $\hat{D}$, $X^{(l)}$ est la matrice des caractéristiques des sommets et $W^{(l)}$ est une matrice de poids entraînable. Il existe également GIN [24] — une généralisation de GCN — et ChebNet [7] qui utilise des polynômes de Chebyshev.
Les approches spatiales ont été définies par Hamilton et al. [12] et se concentrent sur la structure du graphe. Elles sont composées de deux opérations : une agrégation sur le voisinage de chaque sommet et une combinaison du résultat précédent avec la valeur du sommet.
On notera aussi qu'il existe d'autres méthodes comme GAT [22] qui incorporent un mécanisme d'attention dans la convolution ou encore PAN [19] qui utilise une matrice de transition d'entropie maximale à la place d'une matrice Laplacienne.
Une récente étude comparative des approches spectrales et spatiales [1] a permis d'unifier les deux approches et de montrer expérimentalement leurs limites en termes de résolution spectrale.

2.2 Pooling sur graphes

Le pooling sur graphes se divise en deux catégories : pooling global et hiérarchique. Le pooling global réalise une agrégation de tous les sommets d'un graphe et résume celui-ci en un seul vecteur. Il existe plusieurs agrégations, certaines ordinaires comme la somme, la moyenne, le maximum ou plus complexe comme [18, 23, 26].
Le pooling hiérarchique permet de réduire la taille du graphe et de résumer ses informations en plusieurs sommets. Bien évidemment, le nombre de sommets du graphe réduit doit être inférieur à celui du graphe de base. Ce choix du nombre de sommets pour le graphe réduit peut être fixé par un nombre déterminé ou adapté en fonction du graphe d'origine à l'aide d'un ratio.

Notations. Pour toute couche de pooling l, on considère un graphe $\mathcal{G}^{(l)} = (\mathcal{V}^{(l)}, \mathcal{E}^{(l)})$ où $\mathcal{V}^{(l)}$ et $\mathcal{E}^{(l)}$ sont respectivement l'ensemble des sommets et des arêtes du graphe. Soit $n_l = |\mathcal{V}^{(l)}|$, on peut alors définir $\mathbf{A}^{(l)} \in \mathbb{R}^{n_l \times n_l}$ la matrice adjacente associée au graphe $\mathcal{G}^{(l)}$ où $\mathbf{A}^{(l)}_{ij} = 1$ si il existe une arête entre les sommets i et j sinon 0. On note également $\mathbf{X}^{(l)} \in \mathbb{R}^{n_l \times f_l}$ la matrice des caractéristiques des sommets du graphe $\mathcal{G}^{(l)}$ où f_l est la dimension des attributs.

Construction de l'ensemble de sommets survivants. Il existe plusieurs méthodes pour réduire la taille d'un graphe mais nous pouvons les regrouper en deux ensembles. Un premier ensemble consiste à sélectionner des sommets survivants sur un critère qui peut-être :

- un algorithme combinatoire comme dans NDP [2] où une approximation d'une coupe maximum sert à segmenter un graphe en deux parties
- le résultat d'un entraînement où une importance est calculée pour chaque sommet comme dans les méthodes Top-*k* [6, 9, 17].

Le second ensemble de méthodes apprend un partitionnement de $\mathcal{V}^{(l)}$ en n_{l+1} éléments qui formeront les sommets du graphe réduit comme pour DiffPool [25]. Le nombre de regroupement est fixé et l'apprentissage ne permet pas de prendre en compte la taille et la topologie variables des graphes.
Pour chaque méthode, une matrice de réduction $S^{(l)} \in \mathbb{R}^{n_l \times n_{l+1}}$ est créée où n_l et n_{l+1} sont respectivement les tailles du graphe et du graphe réduit. Cette matrice sert à construire les attributs et la matrice d'adjacence du graphe réduit. Chaque sommet i survivant appartenant à son propre regroupement, on suppose que l'on a $S^{(l)}_{i,i} = 1$.

Construction des attributs et de la matrice d'adjacence. La matrice de réduction $S^{(l)}$ est donc à la base de la construction du graphe réduit. Pour la majorité des méthodes, les deux équations suivantes permettent cette construction :

$$X^{(l+1)} = S^{(l)\top}X^{(l)} \quad (2)$$

$$A^{(l+1)} = S^{(l)\top}A^{(l)}S^{(l)} \quad (3)$$

L'équation 3 se réécrit comme suit pour tout couple (i, j) de sommets survivants :

$$(S^{(l)\top}A^{(l)}S^{(l)})_{i,j} = \sum_{k,l}^{n_l} A^{(l)}_{k,l}S^{(l)}_{k,i}S^{(l)}_{l,j} \quad (4)$$

Deux sommets survivants seront donc adjacents dans le graphe réduit si ils étaient adjacents dans le graphe initial ($A^{(l)}_{i,j} = S^{(l)}_{i,i} = S^{(l)}_{j,j} = 1$). De plus, les sommets survivants i et j seront également adjacents dans le graphe réduit si il existe un couple de sommets non-survivants adjacents (k, l) affectés respectivement à i et j ($A^{(l)}_{k,l} = S^{(l)}_{k,i} = S^{(l)}_{l,j} = 1$).
Pour les méthodes Top-*k*, on définit la matrice de réduction $S^{(l)}$ comme étant la sélection des colonnes des indices des

sommets survivants idx de la matrice d'identité I_{n_l} :

$$S^{(l)} = [I_{n_l}]_{:,idx} \tag{5}$$

On a donc $S^{(l)}_{i,i} = 1$ pour tout sommet survivant i et $S^{(l)}_{k,i} = 0$ pour tout non survivant. La matrice $S^{(l)}$ est alors appelée *sélective* car elle sélectionne les attributs des sommets survivants et supprime ceux des sommets non-survivants. De plus, dans ce cas, les lignes de $S^{(l)}$ correspondant aux sommets non-survivants sont à 0 et deux sommets survivants seront adjacents uniquement si ils étaient adjacents avant la réduction. Ceci induit la création de graphes réduits non connexes et une perte d'information importante.
À l'inverse, du fait de l'apprentissage, DiffPool tend à produire des matrices $S^{(l)}$ denses avec peu de valeurs non nulles. L'équation 4 nous montre que la matrice $A^{(l+1)}$ est alors également dense et le graphe correspondant a une densité proche de 1 (i.e. sera soit un graphe complet soit un graphe « presque » complet). La structure du graphe n'est par conséquent pas respectée. On dit que $S^{(l)}$ est *complète*.
On notera que l'on peut aussi s'abstraire de l'équation (3) comme dans NDP [2] où l'utilisation du complément de Schur joue le rôle d'une réduction de Kron. Cette réduction garantie la connexité du graphe réduit $\mathcal{G}^{(l+1)}$ et l'existence d'une arête entre les sommets i et j du graphe $\mathcal{G}^{(l+1)}$ s'il existait un chemin entre eux dans le graphe $\mathcal{G}^{(l)}$. Cependant, cette opération connecte tout couple de sommets survivants adjacents à un sommet supprimé. Elle augmente donc fortement la densité du graphe et il est nécessaire d'appliquer une étape d'élagage à la fin de NDP. Cela peut créer un graphe réduit non connexe. De surcroît, la complexité en temps de la réduction de Kron est approximativement de $\mathcal{O}((\frac{n_l}{2})^3)$, due en partie à l'inversion d'une partie de la matrice Laplacienne.
Contrairement aux autres méthodes, nous proposons de préserver la structure du graphe d'origine ainsi que les informations de ses attributs. Nous satisfaisons ces propriétés grâce à une sélection de sommets survivants bien proportionnée dans le graphe et à l'aide d'une affectation des autres sommets aux survivants. Nous avons nommé notre méthode de pooling MIVSPool.

3 Méthode proposée

Dans un premier temps, nous présentons l'algorithme de l'ensemble de sommets indépendant maximal de Meer [20] en reprenant le formalisme présenté par Brun et al [4]. Nous montrons ensuite comment nous avons adapté cet algorithme au pooling sur graphes.

3.1 Ensemble de sommets indépendant maximal (MIVS)

Avant de décrire l'algorithme, nous allons expliquer ce qu'est un ensemble indépendant maximal et comment il peut être appliqué aux sommets d'un graphe.

Ensemble indépendant maximal. Soit $\mathcal{X}$ un ensemble d'éléments et $\mathcal{N}$ une fonction de voisinage. Un sous-ensemble $\mathcal{J}$ de $\mathcal{X}$ est dit indépendant si :

$$\forall (x, y) \in \mathcal{J}^2 : x \notin \mathcal{N}(y) \tag{6}$$

$\mathcal{J}$ est donc un sous-ensemble de $\mathcal{X}$ tel qu'aucun élément de $\mathcal{J}$ soit relié par la fonction de voisinage $\mathcal{N}$. Les éléments de $\mathcal{J}$ sont appelés les éléments survivants.
Afin de définir la maximalité de l'ensemble indépendant $\mathcal{J}$ par rapport à $\mathcal{X}$, i.e. ne plus pouvoir sélectionner d'éléments dans $\mathcal{X}$, nous avons besoin de l'équation suivante :

$$\forall x \in \mathcal{X} - \mathcal{J}, \exists y \in \mathcal{J} : x \in \mathcal{N}(y) \tag{7}$$

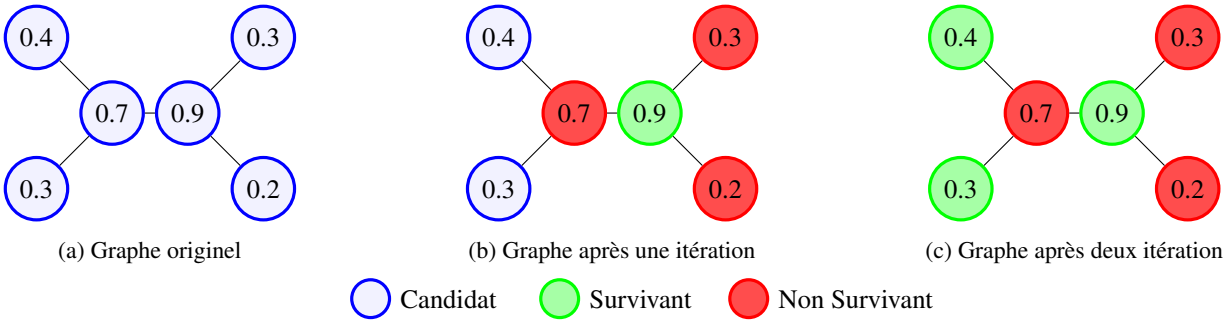
L'équation 7 stipule que chaque élément non-survivant doit être dans le voisinage d'au moins un survivant.
En utilisant les équations 6 et 7, on obtient un ensemble indépendant maximal. Il est à noter que nous parlons bien d'un maximal et non d'un maximum. Effectivement, notre ensemble indépendant maximal $\mathcal{J}$ n'est pas forcément celui dont la cardinalité est maximum par rapport à tous les ensembles indépendants maximaux de $\mathcal{X}$.
Si nous interprétons la construction d'un ensemble indépendant maximal comme une opération de sous-échantillonnage, l'équation 6 peut être interprétée comme une condition prévenant le sur-échantillonnage (deux sommets adjacents ne peuvent être simultanément sélectionnés). Inversement, l'équation 7 prévient le sous-échantillonnage : Tout sommet non sélectionné est à une distance 1 d'un sommet survivant.

Ensemble de sommets indépendant maximal (MIVS). Un ensemble de sommets indépendant maximal (MIVS) d'un graphe $\mathcal{G}^{(l)} = (\mathcal{V}^{(l)}, \mathcal{E}^{(l)})$ est donc un ensemble indépendant maximal où la fonction de voisinage est déduite de l'ensemble des arêtes $\mathcal{E}^{(l)}$. En adaptant les équations 6 et 7, on peut sélectionner les sommets survivants $\mathcal{V}^{(l+1)}$ comme décrit ci-dessous :

$$\forall (v, v') \in (\mathcal{V}^{(l+1)})^2 : (v, v') \notin \mathcal{E}^{(l)} \tag{8}$$

$$\forall v \in \mathcal{V}^{(l)} - \mathcal{V}^{(l+1)}, \exists v' \in \mathcal{V}^{(l+1)} : (v, v') \in \mathcal{E}^{(l)} \tag{9}$$

Les équations 8 et 9 conditionnent la procédure du MIVS en stipulant que deux sommets survivants ne peuvent être voisins et qu'un sommet non-survivant doit être voisin d'au moins un sommet survivant. Néanmoins, ces deux équations n'expliquent pas comment sélectionner ces sommets. Une initialisation simple a été proposée par Meer [20] où celui-ci applique une procédure itérative en affectant une variable aléatoire de loi uniforme $\mathcal{U}([0, 1])$ à chaque sommet. Dans cette procédure, chaque sommet survivant correspond à un maximum local par rapport à ses voisins en fonction de cette variable aléatoire. Les sommets adjacents à ces sommets survivants sont marqués non-survivants du fait de l'équation 8. Les sommets restants (non encore marqués) sont étiquetés candidats et l'algorithme itère la sélection de sommets survivants et non-survivants sur cet ensemble de sommets (voir Figure 1). La convergence de l'algorithme est garantie car, à chaque itération, au moins un sommet candidat est marqué comme survivant.



(a) Graphe originel (b) Graphe après une itération (c) Graphe après deux itération

FIGURE 1 – Évolution d'un MIVS. À chaque sommet est affecté une valeur aléatoire.

Algorithme de Meer. L'algorithme de Meer [20] associe un triplet (x_i, p_i, q_i) à chaque sommet $v_i \in \mathcal{V}_l$ où x_i est une valeur aléatoire uniforme et où p_i et q_i sont des booléens décrivant respectivement si le sommet v_i survit et si celui-ci est toujours candidat. La condition d'arrêt de l'algorithme est obtenue quand tous les q_i sont $Faux$, i.e. lorsque l'équation 9 est satisfaite.
À l'initialisation de l'algorithme, tous les p_i sont $Faux$ et tous les q_i sont $Vrais$ et, pour une itération k, les variables $p_i^{(k)}$ et $q_i^{(k)}$ sont mises à jour par :

$$\begin{aligned} p_i^{(k+1)} &= p_i^{(k)} \vee (q_i^{(k)} \wedge \\ &\quad (x_i = \max\{x_j | (v_i, v_j) \in \mathcal{E}^{(l)} \wedge q_j^{(k)}\})) \\ q_i^{(k+1)} &= \wedge_{j|(v_i,v_j)\in\mathcal{E}^{(l)}} \overline{p}_j^{(k+1)} \end{aligned} \quad (10)$$

En d'autres termes, un survivant à l'itération $k+1$ est survivant à l'itération k ou est un candidat dont la variable aléatoire (x_i) est plus grande que celle de ses voisins candidats. Un sommet est candidat à l'itération $k+1$ si il n'est pas adjacent à un survivant. Notons que le voisinage inclus le sommet central. Cette procédure n'implique que des calculs locaux et est donc parallélisable.

3.2 Adaptation du MIVS à l'apprentissage profond

Dans cette sous-section, nous détaillons l'adaptation de l'algorithme de Meer [20] à notre pooling sur graphes.

Système du calcul de score. Dans l'algorithme de Meer [20], la variable aléatoire de loi $\mathcal{U}([0,1])$ joue un rôle important dans la sélection de l'ensemble des sommets survivants. Nous proposons de modifier cette variable de façon à ce que les x_i utilisés dans l'algorithme soient entraînés et représentent l'importance des sommets. Pour ce faire, nous allons utiliser un mécanisme d'attention comme dans SagPool [17] où un vecteur de valeurs est renvoyé par un GCN [14]. Nous appelons ce vecteur de valeurs le score $s \in \mathbb{R}^{n_l \times 1}$. En utilisant ce score dans notre MIVS, nous sélectionnons un ensemble $\mathcal{V}^{(l+1)}$ de sommets survivants correspondant à des maxima locaux de la fonction d'intérêt codée par le vecteur s. Une répartition uniforme de ces sommets sur le graphe est assurée par le calcul du MIVS. Notons que, contrairement à DiffPool, l'apprentissage n'est pas effectué sur la matrice S mais sur la fonction de score s. Ceci permet de nous adapter aux différentes topologies de graphes.

Affectation des sommets non-survivants. Afin de construire notre matrice de réduction $S^{(l)}$ et prendre en considération les informations de tous les sommets, nous avons besoin d'affecter les sommets non-survivants. Pour rappel, l'équation 9 stipule que chaque non-survivant possède au moins un voisin survivant. Nous allons donc affecter chaque sommet non-survivant à un de ses voisins survivants. Le voisin choisit sera celui dont le score est le plus grand. On obtient alors une matrice de réduction $S^{(l)}$ avec n_{l+1} regroupements correspondant aux sommets survivants :

$$\begin{aligned} &\forall v_j \in \mathcal{V}^{(l)} - \mathcal{V}^{(l+1)}, \exists! v_i \in \mathcal{V}^{(l+1)} | \mathcal{S}_{ji}^{(l)} = 1 \\ &\text{avec } i = argmax(s_k, (v_k, v_j) \in \mathcal{E}^{(l)} \wedge p_k^N) \end{aligned} \quad (11)$$

où N est le nombre d'itérations nécessaires pour obtenir un MIVS et s le vecteur de score.

Construction des attributs réduits. Sachant que la fonction de score s code l'importance des sommets de $\mathcal{G}^{(l)}$, nous proposons de tenir compte du score de chaque sommet dans le calcul des attributs des sommets survivants. Ceci est obtenue grâce à une moyenne pondérée par s de l'ensemble des sommets agrégés aux sommets survivants :

$$X_i^{(l+1)} = \frac{1}{\sum_{j|S_{ji}^{(l)}=1} s_j} \sum_{j|S_{ji}^{(l)}=1} s_j X_j^{(l)} \quad (12)$$

Notons que cette définition permet d'une part de renforcer l'importance des sommets de plus haut score et d'autre part fait intervenir le score de tous les sommets de $\mathcal{G}^{(l)}$ dans le calcul de $\mathcal{G}^{(l+1)}$, Ce dernier point facilite l'apprentissage de la fonction s qui est indépendant de la topologie des graphes. L'équation 12 peut être factorisée en une transformation similaire à l'équation 2 en substituant $S^{(l)}$ par la matrice $S^{(l)'}$:

$$S^{(l)'} = D_2 D_1 S^{(l)} \text{avec} \begin{cases} D_1 &= diag(s) \\ D_2 &= diag(\frac{1}{\mathbf{1}^\top D_1 S^{(l)}}) \end{cases} \quad (13)$$

Construction de la matrice d'adjacence réduite. La construction de la matrice d'adjacence réduite $A^{(l+1)}$ est obtenue grâce à l'équation 3 : Deux sommets survivants i et j seront adjacents si ils étaient adjacent dans le graphe initial $G^{(l)}$ ou si il existait dans $G^{(l)}$ deux sommets k et l non-survivants et affectés respectivement à i et j. Notons que contrairement à la transformée de Kron, notre définition de $S^{(l)}$ n'induit pas une adjacence entre tous les sommets survivants adjacents à un sommet non survivant.

4 Expériences

Dans cette section, nous allons évaluer notre MIVSPool sur deux jeux de données de référence. Nous allons également étudier les propriétés de MIVSPool et comparer notre méthode de pooling aux autres méthodes.

4.1 Jeux de données

Les deux jeux de données sont D&D [8] et PROTEINS [3, 8]. Ils sont respectivement composés de 1178 et 1113 graphes de protéines. Chaque sommet d'une protéine est un acide aminé et deux acides aminés sont reliés par une arête s'ils sont distants de moins de 6 Ångström. Dans les deux jeux de données, le but est de prédire si une protéine est une enzyme ou non. La principale différence entre eux est que D&D possède des graphes avec plus de sommets que ceux de PROTEINS (voir Table 1).

4.2 Architecture du réseau

L'architecture du réseau de neurones se compose de trois blocs constitués d'une opération de convolution suivi d'un pooling sur graphes. Pour capturer l'information obtenue après chaque bloc, une couche de Readout est appliquée. Le Readout utilisé est une concaténation de la moyenne et du maximum des caractéristiques des sommets $X^{(l)}$. À la fin de notre réseau, les trois Readout sont sommés et le résultat est envoyé dans une couche de Perceptron Multicouche. La classification est alors obtenue grâce à une couche de Softmax. Le schéma de l'architecture est disponible en Figure 2. Cette architecture est la même que celle décrite dans SagPool [17] et Cangea et al [6].

4.3 Procédure d'entraînement

Pour évaluer notre pooling, nous proposons de reprendre la procédure d'entraînement de [6, 17] mais d'évaluer notre réseau sur 27 configurations en faisant varier le taux d'apprentissage, la dimension des couches intermédiaires et la diminution des poids (régularisation L2, « weight decay » en anglais) (voir Table 2). Nous fixons également la dimension du batch à 128 et le nombre d'époques à 150. Dans nos expériences, nous avons donc évalué notre réseau de neurones avec 27 validations croisées (une pour chaque configuration) de 10 blocs (« 10-fold cross validation » en anglais) où 80% de notre jeu de données est attribué aux données d'entraînement, 10% aux données de test et le restant aux données de validation. Pour chaque bloc d'une validation croisée, l'erreur minimale est calculée à partir des données de validation et ces paramètres d'apprentissage sont

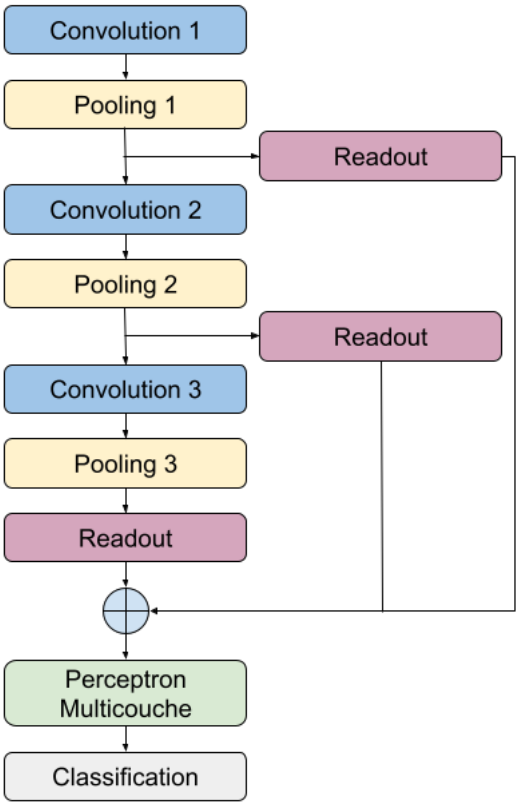


FIGURE 2 – Architecture de notre réseau de neurones

utilisés sur l'ensemble de test. Pour une validation croisée, on obtient alors 10 valeurs d'erreur minimale et d'exactitude que l'on moyenne. Parmi les 27 validations, nous choisissons de sélectionner la configuration dont l'erreur minimale moyenne est la plus faible. Leurs exactitudes et leurs écarts-types sont reportés dans nos expériences. Notons que les écarts-types ont été calculés sur une seule exécution, nos futurs travaux prévoient d'affiner l'estimation de ces écarts-types grâce à de multiples exécutions.

4.4 Étude des propriétés du réseau

Avant de comparer notre MIVSPool par rapport aux autres méthodes de pooling sur graphes, nous proposons d'étudier les propriétés de notre réseau.

Étude de la fonction score. Comme les méthodes Top-*k*, MIVSPool est composé de différentes fonctions : le calcul de l'importance des sommets, la sélection des sommets survivants en fonction de cette importance, et enfin, la reconstruction du graphe réduit. Ces fonctions sont toutes indépendantes les unes des autres. Par conséquent, le calcul du score s peut être vu comme une fonction. Par exemple, dans les méthodes Top *k*, il existe plusieurs façon de calculer l'importance des sommets. Dans notre étude de la fonction score, nous avons fait varier celle-ci de différentes manières en utilisant :

— une loi uniforme $\mathcal{U}([0, 1])$ comme dans Meer [20]
— une projection d'un vecteur normalisé entraînable

Jeu de données	Nombres de Graphes	Nombres de Classes	Nombre moyen de sommets	Nombre moyen d'arêtes
D&D	1178	2	284.32	715.66
PROTEINS	1113	2	39.06	72.82

TABLE 1 – Statistiques des jeux de données

Hyperparamètres	Intervalles
Taux d'apprentissage	1e-2, 1e-3, 1e-4
Dimension des couches intermédiaires	32, 64, 128
Diminution des poids (régularisation L2)	1e-3, 1e-4, 1e-5

TABLE 2 – Hyperparamètres utilisés dans la grille de recherche

Base du score	D&D	PROTEINS
MIVSPool$_{rand}$	77.33 ± 4.03	74.31 ± 4.02
MIVSPool$_{Top-k}$	77.84 ± 3.66	**74.40** ± 4.78
MIVSPool$_{SagPool}$	**78.44** ± 3.45	73.23 ± 3.25

TABLE 3 – Étude de MIVSPool selon la fonction de score

sur les attributs $X^{(l)}$ comme dans [6, 9]

— un mécanisme d'attention à l'aide d'une convolution sur graphes comme dans SagPool [17]

Nous nommons ces variations respectivement MIVSPool$_{rand}$, MIVSPool$_{Top-k}$ et MIVSPool$_{SagPool}$. Les résultats reportés Table 3 montrent que le choix de la fonction score a une influence marginale sur l'exactitude. Par contre, l'utilisation de la fonction score entraînable SagPool permet d'obtenir des écarts-types réduits et une exactitude comparable aux autres heuristiques. Dans la suite de notre article, nous choisissons de prendre MIVSPool$_{SagPool}$ comme référence que nous noterons simplement MIVSPool.

Étude du MIVS en nombre d'itérations. Comme nous l'avons vu, l'algorithme MIVSPool produit un ensemble indépendant maximal au bout d'un certain nombre d'itérations. Une variante proposée par Jolion [13] est d'exécuter un nombre d'itération fixé sans garantir que la condition « ne plus avoir de sommets candidats » soit atteinte. Cette variante permet de limiter le nombre de sommets non-survivants et les sommets candidats restants après la dernière itération deviennent survivants. Nous avons étudié

Nombre d'itérations	D&D	PROTEINS
Une itération	76.39 ± 5.02	69.54 ± 6.41
Deux itérations	74.95 ± 3.49	71.70 ± 5.04
Trois itérations	75.55 ± 3.18	**73.77** ± 4.35
MIVSPool	**78.44** ± 3.45	73.23 ± 3.25

TABLE 4 – Étude de MIVSPool en fonction du nombre d'itérations

Jeu de données	Pooling 1	Pooling 2	Pooling 3
PROTEINS	2.10 ± 0.69	2.17 ± 0.90	1.62 ± 0.71
D&D	3.10 ± 0.62	3.44 ± 0.80	2.67 ± 0.70

TABLE 5 – Nombre moyen d'itérations de MIVS pour chaque étape de pooling.

l'impact d'un arrêt forcé à la première, seconde et troisième itérations et comparé les exactitudes avec notre MIVSPool de référence. Les résultats sont reportés dans la Table 4. On observe une progression de l'exactitude en fonction de l'augmentation des itérations et cela confirme l'utilité de notre MIVS. Pour PROTEINS, on remarque que l'on obtient des résultats proches de ceux à convergence dès que le nombre d'itérations est égal à 3. Cela est sûrement dû au faible nombre de sommets par graphes de ce jeu de données.

Évolution du nombre d'itérations après chaque étape de pooling. Nous venons d'étudier l'exactitude fournie par notre algorithme en bornant le nombre d'itérations du MIVS. Nous proposons maintenant d'étudier le nombre moyen d'itérations à convergence dans le MIVS pour chaque étape de pooling. Pour rappel, l'architecture de notre réseaux se compose de trois couches de pooling (Figure 2). Nous avons reporté les résultats dans la Table 5. Malgré une différence importante entre la taille des graphes de PROTEINS et D&D (Table 1), on constate que le nombre d'itérations reste comparable entre les deux jeux de données et est inférieur à 5. Sachant qu'une itération du MIVS sur un graphe $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ nécessite de l'ordre de $|\mathcal{V}|d_{max}$ calculs, où d_{max} est le degré maximum de $\mathcal{G}$, le calcul du MIVS sur un des graphes de nos deux jeux de données est borné par $5|\mathcal{V}|d_{max}$. Cette complexité est bien inférieure à celle nécessitée par la transformation de Kron (Section 2.2)

Évolution des densités et des degrés moyens. Nous avons reporté sur la Table 6, les densités et degrés moyens des jeux de données avant et après chaque couche de pooling. Nous remarquons que le cardinal des sommets $|\mathcal{V}|$ est divisé par un ratio de 2,52 en moyenne. Si l'on enlève la première réduction ce ratio tombe à 2,13. Nous notons aussi une augmentation régulière de la densité qui s'explique en partie par la diminution de la taille des graphes et par la préservation de leur connexité. En effet, le graphe connexe de n sommets de densité minimale est un chemin de densité $\frac{2}{n}$. Cette borne inférieure n'est pas négligeable pour de petites valeurs de n. Par exemple, à la couche de Pooling 3 et pour le jeu de données PROTEINS, un graphe

Jeu de données	Étapes	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{E}\rvert$	Densité moyenne	Degré moyen
PROTEINS	Initial	39.06 ± 45.78	72.82 ± 84.64	0.21 ± 0.20	3.73 ± 0.42
	Pooling 1	13.29 ± 17.03	19.29 ± 31.63	0.38 ± 0.26	2.29 ± 0.82
	Pooling 2	6.54 ± 7.83	7.73 ± 14.70	0.56 ± 0.29	1.75 ± 0.73
	Pooling 3	4.12 ± 4.32	3.86 ± 7.53	0.73 ± 0.29	1.41 ± 0.60
D&D	Initial	284.32 ± 272.12	715.66 ± 694.20	0.03 ± 0.02	4.98 ± 0.59
	Pooling 1	77.24 ± 74.50	152.18 ± 157.17	0.08 ± 0.06	3.82 ± 0.65
	Pooling 2	29.36 ± 27.68	57.73 ± 62.20	0.20 ± 0.13	3.68 ± 0.79
	Pooling 3	12.96 ± 12.41	23.47 ± 29.03	0.38 ± 0.20	3.20 ± 0.95

TABLE 6 – Évolution de la densité et du degré après une étape de pooling.

Pooling	D&D	PROTEINS
DiffPool [25]	66.95 ± 2.41	68.20 ± 2.02
gPool [9]	75.01 ± 0.86	71.10 ± 0.90
SagPool [17]	76.45 ± 0.97	71.86 ± 0.97
MIVSPool	**78.44** ± 3.45	**73.23** ± 3.25

TABLE 7 – Comparaison de MIVSPool avec d'autres méthodes de pooling hiérarchique

de 4 sommets a une densité minimale de 0,5. Enfin, on observe une diminution lente du degré moyen.

4.5 Comparaison de MIVSPool par rapport aux autres méthodes

Dans le but de prouver l'efficacité de MIVSPool, nous avons effectué une comparaison avec les méthodes de références sur les jeux de données PROTEINS et D&D. Comme notre architecture et notre procédure d'entraînement sont similaires à celles de SagPool [17], nous avons reporté ces résultats en ajoutant les nôtres (voir Table 7). Les méthodes de pooling de référence sont donc DiffPool, gPool et SagPool.

Les résultats montrent que l'exactitude moyenne de MIVSPool surpasse les méthodes de référence quelque soit la fonction de score utilisée (voir également Table 3). Ce résultat est toutefois à relativiser par l'importance de l'écart-type sur les exactitudes. Ces forts écarts-types sont principalement dû au faible nombre d'exécutions que nous avons pu effectuer. Ce dernier point fera l'objet de nos futurs travaux.

5 Conclusion

Nous avons présenté notre méthode de pooling sur graphes MIVSPool. Elle est basée sur une sélection de sommets survivants à l'aide d'un ensemble de sommets indépendant maximal (MIVS) et d'une affectation des sommets non-survivants aux survivants. À l'inverse des méthodes de référence, notre méthode permet de préserver la totalité des informations du graphe lors de sa réduction. Les résultats expérimentaux tendent à montrer une amélioration de l'exactitude pour la classification de graphes.

Dans nos futures recherches, nous comptons étoffer notre MIVSPool avec d'autres fonctions score et approfondir l'étude de méthodes permettant d'affecter les sommets non survivants aux survivants.

Remerciements

Ce travail a bénéficié des moyens de calcul du mésocentre CRIANN (Centre Régional Informatique et d'Application Numériques de Normandie).

Références

[1] Muhammet Balcilar, Guillaume Renton, Pierre Héroux, Benoit Gaüzère, Sébastien Adam, and Paul Honeine. Analyzing the expressive power of graph neural networks in a spectral perspective. In *ICLR*. OpenReview.net, 2021.

[2] Filippo Maria Bianchi, Daniele Grattarola, Lorenzo Livi, and Cesare Alippi. Hierarchical representation learning in graph neural networks with node decimation pooling. *CoRR*, abs/1910.11436, 2019.

[3] Karsten M. Borgwardt, Cheng Soon Ong, Stefan Schönauer, S. V. N. Vishwanathan, Alexander J. Smola, and Hans-Peter Kriegel. Protein function prediction via graph kernels. In *Proceedings Thirteenth International Conference on Intelligent Systems for Molecular Biology 2005, Detroit, MI, USA, 25-29 June 2005*, pages 47–56, 2005.

[4] Luc Brun and Walter Kropatsch. *Image Processing and Analysing With Graphs : Theory and Practice*, chapter Hierarchical Graph Encoding. CRC Press, 2012.

[5] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. In Yoshua Bengio and Yann LeCun, editors, *ICLR*, 2014.

[6] Catalina Cangea, Petar Velickovic, Nikola Jovanovic, Thomas Kipf, and Pietro Liò. Towards sparse hierarchical graph classifiers. *CoRR*, abs/1811.01287, 2018.

[7] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, edi-

tors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[8] Paul D. Dobson and Andrew J. Doig. Distinguishing enzyme structures from non-enzymes without alignments. *Journal of Molecular Biology*, 330(4) :771–783, 2003.

[9] Hongyang Gao and Shuiwang Ji. Graph u-nets. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *ICML*, volume 97, pages 2083–2092. PMLR, 2019.

[10] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In Doina Precup and Yee Whye Teh, editors, *ICML*, volume 70 of *Proceedings of Machine Learning Research*, pages 1263–1272. PMLR, 2017.

[11] M. Gori, G. Monfardini, and F. Scarselli. A new model for learning in graph domains. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks*, volume 2, pages 729–734, 2005.

[12] William L. Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 1024–1034, 2017.

[13] Jean-Michel Jolion and Annick Montanvert. The adaptive pyramid : A framework for 2d image analysis. *CVGIP Image Underst.*, 55(3) :339–348, 1992.

[14] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *CoRR*, abs/1609.02907, 2016.

[15] Boris Knyazev, Graham W. Taylor, and Mohamed R. Amer. Understanding attention and generalization in graph neural networks. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 4204–4214, 2019.

[16] Yann LeCun, Léon Bottou, Genevieve B. Orr, and Klaus-Robert Müller. Efficient backprop. In Grégoire Montavon, Genevieve B. Orr, and Klaus-Robert Müller, editors, *Neural Networks : Tricks of the Trade - Second Edition*, volume 7700 of *Lecture Notes in Computer Science*, pages 9–48. Springer, 2012.

[17] Junhyun Lee, Inyeop Lee, and Jaewoo Kang. Self-attention graph pooling. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *ICML*, volume 97 of *Proceedings of Machine Learning Research*, pages 3734–3743. PMLR, 2019.

[18] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard S. Zemel. Gated graph sequence neural networks. In Yoshua Bengio and Yann LeCun, editors, *ICLR*, 2016.

[19] Zheng Ma, Junyu Xuan, Yu Guang Wang, Ming Li, and Pietro Liò. Path integral based convolution and pooling for graph neural networks. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems : 33*, 2020.

[20] Peter Meer. Stochastic image pyramids. *Comput. Vis. Graph. Image Process.*, 45(3) :269–294, 1989.

[21] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Trans. Neural Networks*, 20(1) :61–80, 2009.

[22] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. *CoRR*, abs/1710.10903, 2017.

[23] Oriol Vinyals, Samy Bengio, and Manjunath Kudlur. Order matters : Sequence to sequence for sets. In Yoshua Bengio and Yann LeCun, editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

[24] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks ? *CoRR*, abs/1810.00826, 2018.

[25] Rex Ying, Jiaxuan You, Christopher Morris, Xiang Ren, William L. Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. *CoRR*, abs/1806.08804, 2018.

[26] Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. An end-to-end deep learning architecture for graph classification. In Sheila A. McIlraith and Kilian Q. Weinberger, editors, *AAAI*, pages 4438–4445. AAAI Press, 2018.