

Travail d'Étude et de Recherche (TER)

Belladjo Keroudine

15 mars 2022

Les différents domaines d'utilisation ou d'application de réseaux convolutifs de graphe (GCN)

Professeur encadrant : Marc Tommasi



Table des matières

1	Introduction	3
2	Définition et caractéristique des graphes	4
3	Fonctionnement d'un modèle de réseau convolutif de graphe	5
4	Comprendre les réseaux convolutifs de graphes pour la classification des noeuds	7
4.1	Chargement et préparation des données	8
4.2	Création des couches GCN et saisie des données à l'aide de StellarGraph	10
4.3	Entraînement et évaluation du modèle	11
5	Domaine d'applications des réseaux convolutionnels de graphes (GCN)	13
5.1	la vision par ordinateur	13
5.1.1	Image	16
5.1.2	Videos	18
5.1.3	Point clouds : Nuage de points	18
5.1.4	Mailles	19
5.2	Le traitement du langage naturel	19
5.3	Applications en sciences	20
5.3.1	Physique	20
5.3.2	Chimie, Biologie et Science des matériaux	21
5.4	Analyse des réseaux sociaux	22
5.5	Systèmes de recommandation à l'échelle du Web	22
6	Conclusion	23

1 Introduction

De nos jours d'importants ensembles de données du monde réel se présentent sous forme de graphes ou de réseaux : réseaux sociaux, réseaux d'interaction de protéines, le World Wide Web (www), graphes de connaissances etc. Cependant il n'y a pas très longtemps, très peu d'attention a été consacrée à la généralisation des modèles de réseau de neurones à de tels ensembles de données structurés [3]. L'ensemble d'outils traditionnels de deep learning (apprentissage profond), qui nous a permis à obtenir des résultats révolutionnaires en vision par ordinateur et en traitement automatique des langues (TAL), est peu utile lorsqu'il s'agit de graphes. Les réseaux de neurones classiques ne fonctionnent normalement qu'avec des données structurées comme des grilles de pixels de taille fixe (images) et des séquences (texte). Ces structures de données sont profondément plus simples que les graphes qui peuvent avoir une taille arbitraire, des caractéristiques multimodales, une topologie complexe et aucun ordre de noeud fixe [8] [2]. Pour résoudre ce problème, différentes méthodes de réseaux de neuronaux de graphes ont été proposées. Dans cette présentation, nous aborderons l'une des principales approches de deep learning (GCN) pour le traitement des données de graphes : les réseaux convolutifs de graphes. Ce sont des architectures puissantes permettant de résoudre des problèmes d'apprentissage en grandes dimensions. Ils se montrent particulièrement efficaces dans le cadre de l'apprentissage semi-supervisé. Ce dernier est une classe de techniques d'apprentissage automatique qui utilise un ensemble de données étiquetées et non-étiquetées. Il se situe ainsi entre l'apprentissage supervisé qui n'utilise que des données étiquetées et l'apprentissage non-supervisé qui n'utilise que des données non-étiquetées. Il a été démontré que l'utilisation de données non-étiquetées, en combinaison avec des données étiquetées, permet d'améliorer significativement la qualité de l'apprentissage [4]. D'où encore l'importance des GCN. Dans cette présentation, nous élaborons tout d'abord un aperçu des réseaux convolutifs de graphes. Ensuite, nous catégorisons différents réseaux convolutifs de graphes selon les domaines de leurs applications. Pour ce faire nous allons nous appuyer essentiellement sur des articles tels que l'article de Thomas Kipf [3], de Rostyslav Demush [2], et de Si et Tong [8].

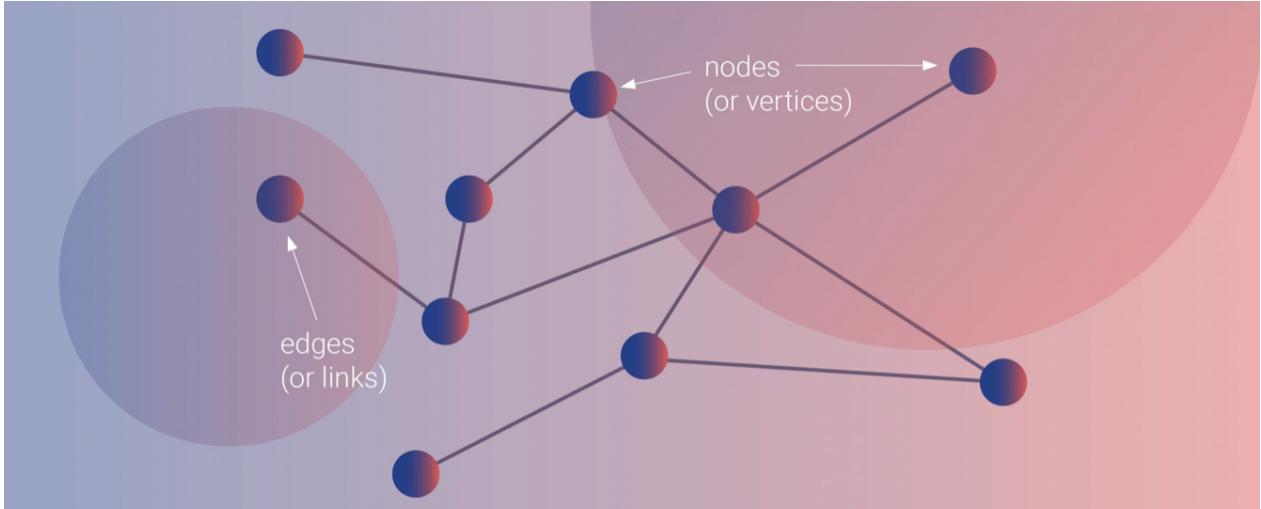


FIGURE 1 – Représentation d'un graphe [2] : Les arêtes peuvent être dirigées (graphe orienté) ou non, selon qu'il existe ou non des dépendances directionnelles entre les points de données.

2 Définition et caractéristique des graphes

D'après l'ouvrage de Olivier Cogis et Claudine Robert [1], La théorie des graphes est une théorie mathématique, qui définit des graphes comme des structures de données constituées de sommets (nœuds) et d'arêtes (connexion entre ces nœuds). Mathématiquement, ils sont définis comme $G = (E, V)$ où G est le graphe, V est l'ensemble des sommets et E est l'ensemble des arêtes de G , voici en figure 1 un exemple typique de graphe.

L'existence de la théorie des graphes est conditionnée par le fait que nous avons besoin d'un moyen informatique et mathématique pour représenter les relations entre les choses.

Les graphes sont donc importants pour représenter de nombreuses choses dans le monde réel entre autres les pixels dans une image, les adresses ou des emplacements (coordonnées) sur une carte, les neurones dans notre cerveau, les hiérarchies organisationnelles, des voisins physiques dans une communauté, les réseaux sociaux, les molécules (où les atomes peuvent être visualisés comme des nœuds et les liaisons chimiques comme des arêtes), les systèmes d'interaction protéine-protéine, les chaînes d'approvisionnement, graphes de connaissances, etc. Les graphes en tant que structure de données possèdent un pouvoir expressif énorme. Le but de la théorie des graphes pour l'apprentissage automatique en particulier est que la majeure partie de nos données peuvent être mieux comprises lorsque nous pouvons représenter ses relations. Par conséquent, nous

souhaitons un moyen d'intégrer ces relations afin de pouvoir ensuite travailler avec l'ensemble des données [2].

3 Fonctionnement d'un modèle de réseau convolutif de graphe

Dans [3] Les réseau convolutif de graphe sont définis comme des modèles de neurones de graphe qui ont une architecture peu universelle car les paramètres de filtre sont généralement partagés sur tous les emplacements du graphe (ou un sous-ensemble de celui-ci). Pour ces modèles, le but est alors d'apprendre une fonction de signaux ou d'attributs sur un graphe $G = (E, V)$ qui prend en entrée :

- Un attribut x_i pour chaque nœud i, tout mis dans une matrice $X = N \times D$ appelé matrice des attributs (feature matrix) avec N le nombre de noeud et D nombre d'attributs en entrée
- Une description représentative de la structure du graphe sous forme matricielle ; généralement sous la forme d'une matrice d'adjacence A (ou une fonction de celle-ci)

Il produit une sortie au niveau du nœud Z (une matrice d'attribut $N \times F$ où F est le nombre d'attributs en sortie par noeud).

Toute couche de réseau de neurones peut alors être écrite comme une fonction non linéaire :

$$H^{(l+1)} = f(H^{(l)}, A) \quad (1)$$

où $H^{(0)} = X$ et $H^{(L)} = Z$ (ou Z les sorties de différents niveaux du graphe, L est le nombre de couches, A la matrcice d'adjcience ou de contiguïté.

$f(\cdot)$ est une fonction d'activation non linéaire comme ReLU(\cdot), Sigmoïde $\sigma(\cdot)$ et tangente hyperbolique $\tanh(\cdot)$.

GCN C'est aussi une couche de réseau neuronal, la façon dont il se propage d'une couche à l'autre est :

$$f(H^{(l)}, A) = \sigma(\widehat{D}^{\frac{-1}{2}} \widehat{A} \widehat{D}^{\frac{-1}{2}} H^{(l)} W^{(l)}) \quad (2)$$

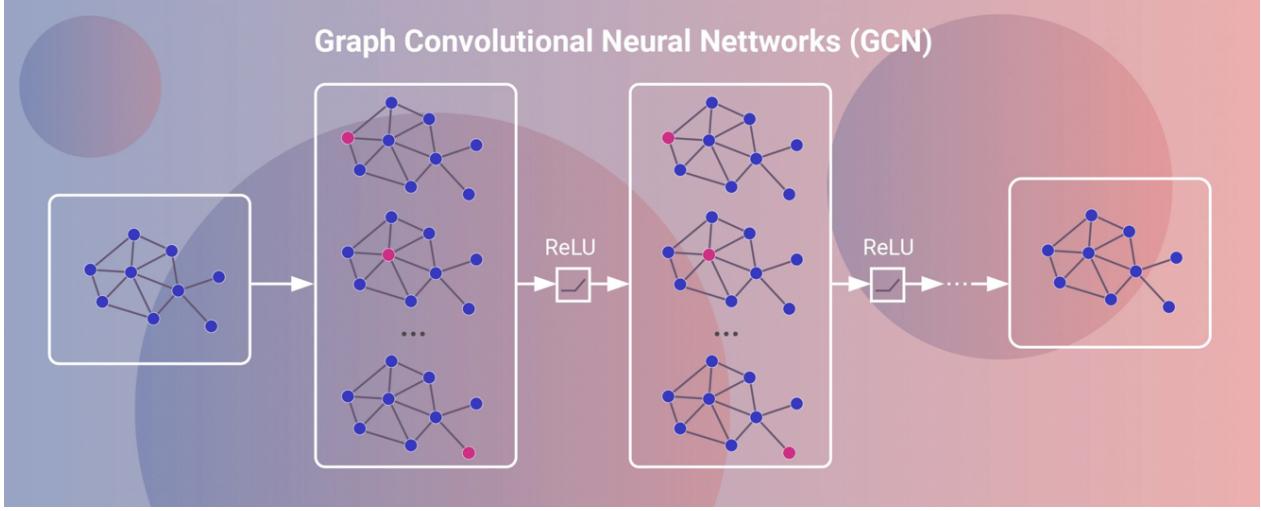


FIGURE 2 – Réseau convolutif de graphes multicouches (GCN) avec filtres de premier ordre.

Avec $\widehat{A} = A + I$, où I est la matrice d'identité et \widehat{D} est la matrice diagonale des degrés des nœuds de \widehat{A} , $W^{(l)}$ est une matrice de pondération pour la l -ième couche du réseau de neurones, σ est la fonction d'activation. $D^{-1/2}AD^{-1/2}$ est la matrice d'adjacence symétrique normalisée très similaire à la matrice Laplacienne normalisée symétrique.

La figure 2 représente un réseau convolutif de graphes multicouches. Ce dernier fonctionne comme suit : En premier lieu, chaque nœud reçoit des informations sur toutes les fonctionnalités de ses nœuds connectés et applique à ces valeurs une fonction d'agrégation telle que la somme ou la moyenne, qui garantit que toutes les représentations sortent de la même taille. Quelle que soit la fonction que nous choisissons, elle doit être invariante à la permutation et à l'ordre [2].

Ensuite, le vecteur résultant est passé à travers une couche de réseau neuronal dense, en d'autres termes le vecteur est multiplié par une matrice, puis une fonction d'activation non linéaire (ici ReLU) est utilisée en plus pour obtenir une nouvelle représentation vectorielle, voir la figure 2.

Ensuite, nous continuons à parcourir ces trois étapes :

- Pour chaque nœud, ça agrège les vecteurs de ses voisins (ceux mis à jour)
- ça passe ensuite la valeur résultante à travers une couche de réseau neuronal dense
- ça Applique en fin la non-linéarité.

On répète ces étapes autant de fois qu'on a de couches.

Si vous faites le constat, les nœuds des GCN auront une représentation différente à chaque couche :

- À la couche 0, ils seront les mêmes que les attributs de nœuds.
- Au niveau de la couche k , pour calculer une représentation d'un nœud, nous allons parcourir ses voisins, prendre leurs représentations de la couche précédente $k - 1$ et faire leur moyenne. Ensuite, à l'aide d'une matrice de paramètres nous les transformerons et ajouterons à cela les messages du nœud de $k - 1$. Le résultat sera transmise via une fonction non linéaire telle que ReLU.
- Enfin, une fois les transformations finies dans les couches cachées de la représentation du nœud, nous obtiendrons l'incorporation finale [2].

Il existe deux matrices de paramètres ajustables que nous appliquons à chaque couche , une pour transformer les valeurs de la représentation du nœud à partir de $k - 1$, et une autre pour transformer les messages agrégés des nœuds voisins (à nouveau à partir de $k - 1$).

Au cours de l'apprentissage, nous cherchons la meilleure manière d'enrichir les informations qu'un nœud apprend sur lui-même. Cela se résume à ceci : quelle transformation non linéaire nous voulons faire sur les vecteurs propres des attributs du nœud. Combien de modifications nous devons appliquer aux vecteurs des attributs des voisins. Selon les matrices que nous utilisons, nous pouvons faire en sorte que le nœud se concentre complètement sur les informations le concernant et ignore les messages de son environnement, ou l'inverse, ou un peu des deux, selon ce qui nous aide à obtenir les meilleures prédictions à la fin, voir le graphique 3 issu de [2].

4 Comprendre les réseaux convolutifs de graphes pour la classification des nœuds

Dans cette partie inspirée de [4], nous allons expliquer comment effectuer la classification des nœuds à l'aide du module *StellarGraph*. Ce module prend en charge de nombreux algorithmes d'apprentissage automatique (ML) de pointe sur les graphes dont l'algorithme de GCN. Il y a deux parties nécessaires pour pouvoir faire cette tâche :

1. Un graphe :On va utiliser l'ensemble de données *Cora*. Cet ensemble de données se compose de publications académiques comme nœuds et les

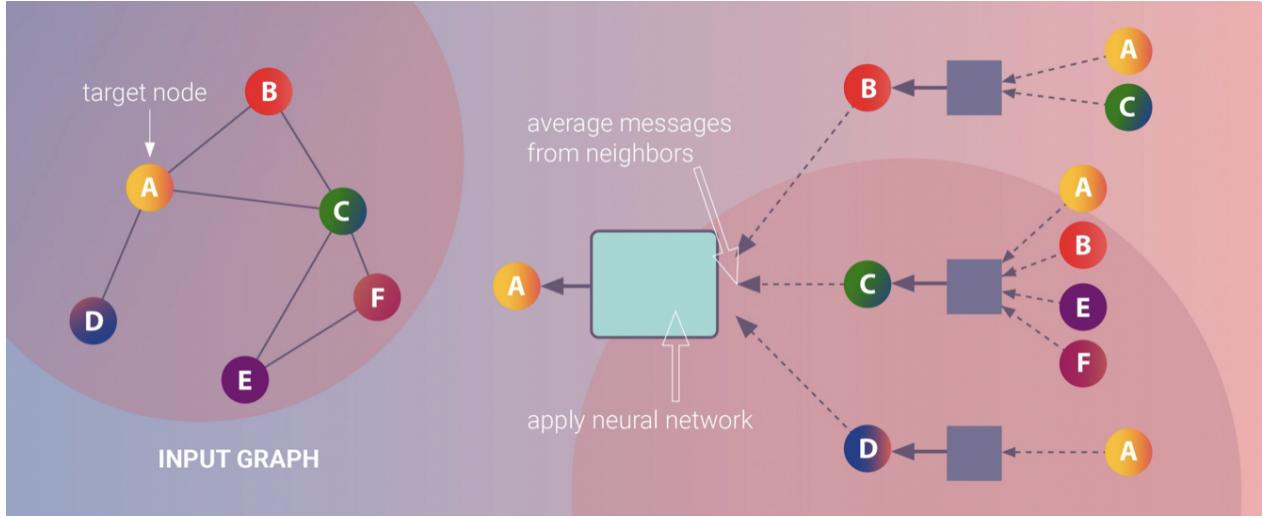


FIGURE 3 – Étant donné un graphe d’entrée, GCN prédit l’étiquette du nœud cible (le nœud en jaune (A)) en agrégeant les informations des nœuds voisins .

citations entre eux comme arêtes : si la publication A cite la publication B, alors le graphe a un arête de A à B. Les nœuds sont classés dans l’un des sept sujets, et notre modèle apprendre à prédire ce sujet ;

2. Un algorithme : Evidemment on va utiliser un Graph Convolution Network. Le cœur du modèle de réseau neuronal GCN est une couche de convolution de graphe. Cette couche est similaire à une couche dense conventionnelle, augmentée par la matrice de contiguïté du graphe pour utiliser les informations sur les connexions d’un nœud.

4.1 Chargement et préparation des données

Nous avons préparé les données à l'aide de Pandas et de *scikit – learn* : Nous avons chargé le graphe à partir de fichiers *CSV*, nous avons aussi effectué une introspection de base et le découpé en apprentissage de test et de validation pour l'apprentissage automatique, (confère le Notebook).

Premièrement on a importer les bibliothèques Python dont nous aurons besoin. Nous importons *stellargraph* sous le nom *sg* .

Nous récupérons un objet graphe *StellarGraph* contenant l'ensemble de données *Cora*. C'est un réseau de citations, qui est un graphe orienté de 2708 noeud et 5429 arêtes. Cette fonction est implémentée à l'aide de Pandas, nous avons Charger les données dans *StellarGraph*. Voir la figure 4.



```
print(G.info())
```

```
StellarGraph: Undirected multigraph
Nodes: 2708, Edges: 5429
```

Node types:

paper: [2708]

Features: float32 vector, length 1433

Edge types: paper-cites->paper

Edge types:

paper-cites->paper: [5429]

Weights: all 1 (default)

FIGURE 4 – Les données *Cora* Chargées dans *StellarGraph*

Notre objectif est de former un modèle d'apprentissage automatique de graphe qui prédira la classe "sujet" sur les nœuds. Ces sujets sont l'une des 7 classes, certaines classes étant plus courantes que d'autres (voir la table de la figure 5 qui énumère les différentes classes).

Nous allons à présent découper notre ensemble de données en un ensemble d'entraînement (train) et de test, (voir Noteook).

Pour la classe cible, nous utiliserons des vecteurs uniques qui seront comparés à la sortie *softmax* du modèle. En effet un *softmax* est une fonction d'activation qui convertit un vecteur de nombres en un vecteur de probabilités, où les probabilités de chaque valeur sont proportionnelles à l'échelle relative de chaque valeur dans le vecteur. Pour effectuer cette conversion, nous avons utilisé la transformation *LabelBinarizer* de *scikit – learn* pour interpréter les prédictions.

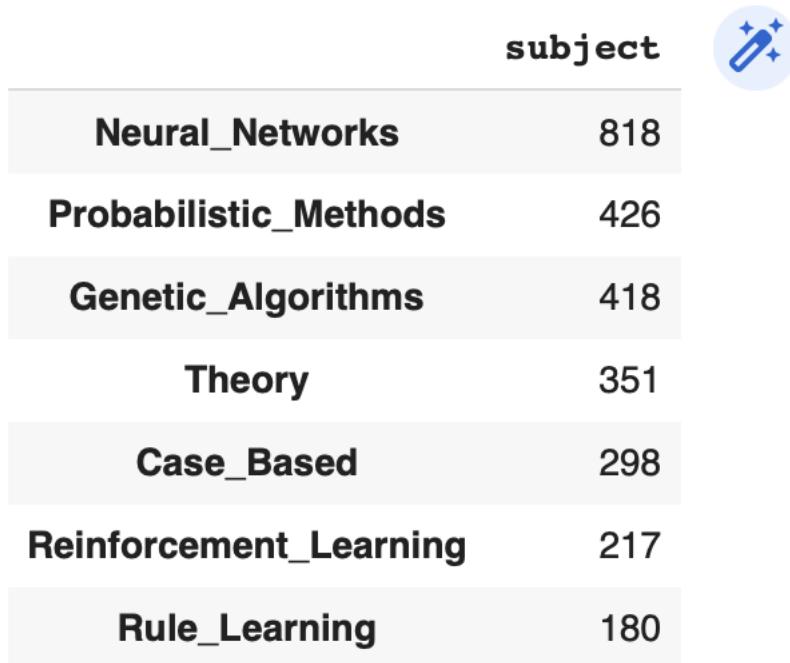


FIGURE 5 – Les 7 classes des données Cora

4.2 Création des couches GCN et saisie des données à l'aide de StellarGraph

Un modèle d'apprentissage automatique dans *StellarGraph* consiste en une paire d'éléments :

- les couches elles-mêmes, telles que la convolution de graphe, le décrochage et même les couches denses conventionnelles
- un générateur de données pour convertir la structure graphique de base et les caractéristiques des nœuds dans un format qui peut être introduit dans le modèle *Keras* pour la formation ou la prédiction. *Keras* est une API qui permet de créer des modèles de réseaux de neurones plus facilement

GCN est un modèle de lot complet et nous procédons ici à la classification des nœuds, ce qui signifie que la classe *FullBatchNodeGenerator* est le générateur approprié pour notre tâche. *StellarGraph* dispose de nombreux générateurs afin de prendre en charge tous ses nombreux modèles et tâches

La valeur x_out est un tenseur TensorFlow qui contient un vecteur à 16 dimensions pour les nœuds demandés lors de l'entraînement ou de la prédiction.

Les prédictions réelles de la classe (sujet) de chaque nœud doivent être calculées à partir de ce vecteur. *StellarGraph* est construit à l'aide de la fonctionnalité *Keras*, cela peut donc être fait avec une fonctionnalité *Keras* standard : une couche dense supplémentaire (avec une unité par classe) à l'aide d'une activation *softmax*. Cette fonction d'activation garantit que les sorties finales pour chaque noeud d'entrée seront un vecteur de "probabilités", où chaque valeur est comprise entre 0 et 1, et le vecteur entier est égal à 1. La classe prédite est l'élément avec la valeur la plus élevée.

4.3 Entraînement et évaluation du modèle

Créons maintenant le modèle *Keras* réel avec les tenseurs d'entrée x_inp et les tenseurs de sortie étant les prédictions des prédictions de la couche dense finale. Notre tâche est une tâche de prédiction de classe, donc une fonction de perte de validation croisée des classes est appropriée. Même si nous faisons de l'apprentissage automatique de graphe avec *StellarGraph*, nous travaillons toujours avec des valeurs de prédiction *Keras* conventionnelles, nous pouvons donc utiliser directement la fonction de perte de *Keras* (voir Notebook).

Pendant que nous formons le modèle, nous voudrons également suivre ses performances de généralisation sur l'ensemble de validation, ce qui signifie créer un autre générateur de données, en utilisant notre *FullBatchNodeGenerator* que nous avons créé.

Nous pouvons directement utiliser la fonctionnalité *EarlyStopping* offerte par *Keras* pour arrêter l'entraînement si la précision de la validation ne s'améliore plus.

Une fois que nous avons formé le modèle, nous pouvons afficher la fonction de perte de comportement et toute autre métrique à l'aide de la fonction *plot_history*. Dans ce cas, nous pouvons voir la perte et la précision sur les ensembles d'apprentissage et de validation (Voir la figure 6).

Comme la dernière partie de notre évaluation, vérifions le modèle par rapport à l'ensemble de test. Nous créons à nouveau les données requises pour cela en utilisant la méthode de flux sur notre *FullBatchNodeGenerator* d'en haut, et pouvons utiliser la méthode d'évaluation du modèle pour calculer les valeurs métriques pour le modèle formé.

Nous Obtenons maintenant les prédictions pour tous les nœuds. Nous allons utiliser notre *FullBatchNodeGenerator* pour créer l'entrée requise, puis utili-



```
sg.utils.plot_history(history)
```

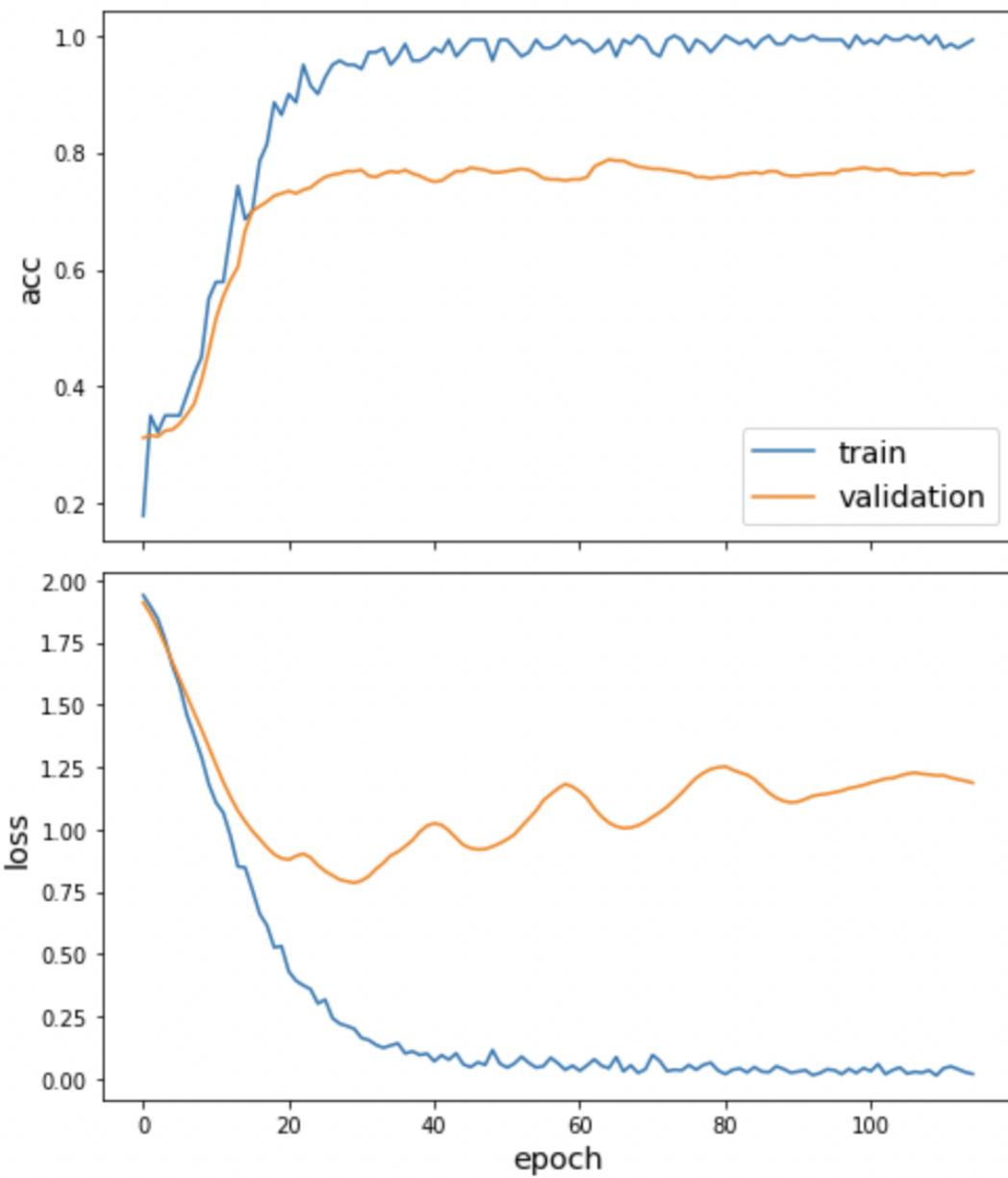


FIGURE 6 – Courbes de fonction de perte en apprentissage et en validation

sons l'une des méthodes du modèle : prédire. Cette fois, nous ne fournissons pas les étiquettes au flux, mais uniquement les nœuds, car nous essayons de prédire ces classes sans les connaître.

Ces prédictions seront la sortie de la couche *softmax*, donc pour obtenir les classes finales, nous utiliserons la méthode *inverse_transform* de notre spécification d'attribut cible pour ramener ces valeurs aux classes d'origine, voir le tableau de la figure 7.

En plus de simplement prédire la classe des exemples (nœuds cibles), il peut être utile d'obtenir une image plus détaillée des informations que le modèle a appris sur les nœuds et leurs voisinages (Figure 8).

5 Domaine d'applications des réseaux convolutionnels de graphes (GCN)

Les GCN peuvent également être classés en fonction de leurs domaines d'application. Dans cette section, nous introduisons principalement les applications des réseaux convolutifs de graphes dans la vision par ordinateur, le traitement du langage naturel, la science et d'autres domaines.

5.1 la vision par ordinateur

Selon Si Zhang et al. :

La vision par ordinateur a été l'un des domaines de recherche les plus en vogue au cours des dernières décennies. De nombreuses architectures de deep learning existants utilisés dans les problèmes de vision par ordinateur sont construites sur les réseaux de neurones à convolution classiques (CNN). Malgré les grands succès des CNN, il est difficile d'encoder les structures intrinsèques des graphes dans les tâches d'apprentissage spécifiques. En revanche, les réseaux convolutionnels de graphes (GCN) ont été appliqués pour résoudre certains problèmes de vision par ordinateur et ont montré des performances comparables, voire meilleures[8].

Dans ces sous-sections qui suivent, nous catégorisons ces applications en fonction du type de données.

	Prediction	Vrai
31336	Neural_Networks	Neural_Networks
1061127	Rule_Learning	Rule_Learning
1106406	Reinforcement_Learning	Reinforcement_Learning
13195	Reinforcement_Learning	Reinforcement_Learning
37879	Probabilistic_Methods	Probabilistic_Methods
1126012	Probabilistic_Methods	Probabilistic_Methods
1107140	Theory	Theory
1102850	Neural_Networks	Neural_Networks
31349	Neural_Networks	Neural_Networks
1106418	Theory	Theory
1123188	Probabilistic_Methods	Neural_Networks
1128990	Genetic_Algorithms	Genetic_Algorithms
109323	Probabilistic_Methods	Probabilistic_Methods

FIGURE 7 – Les classes prédictes

JTEC]

TSNE visualisation des intégrations GCN pour l'ensemble de données cora

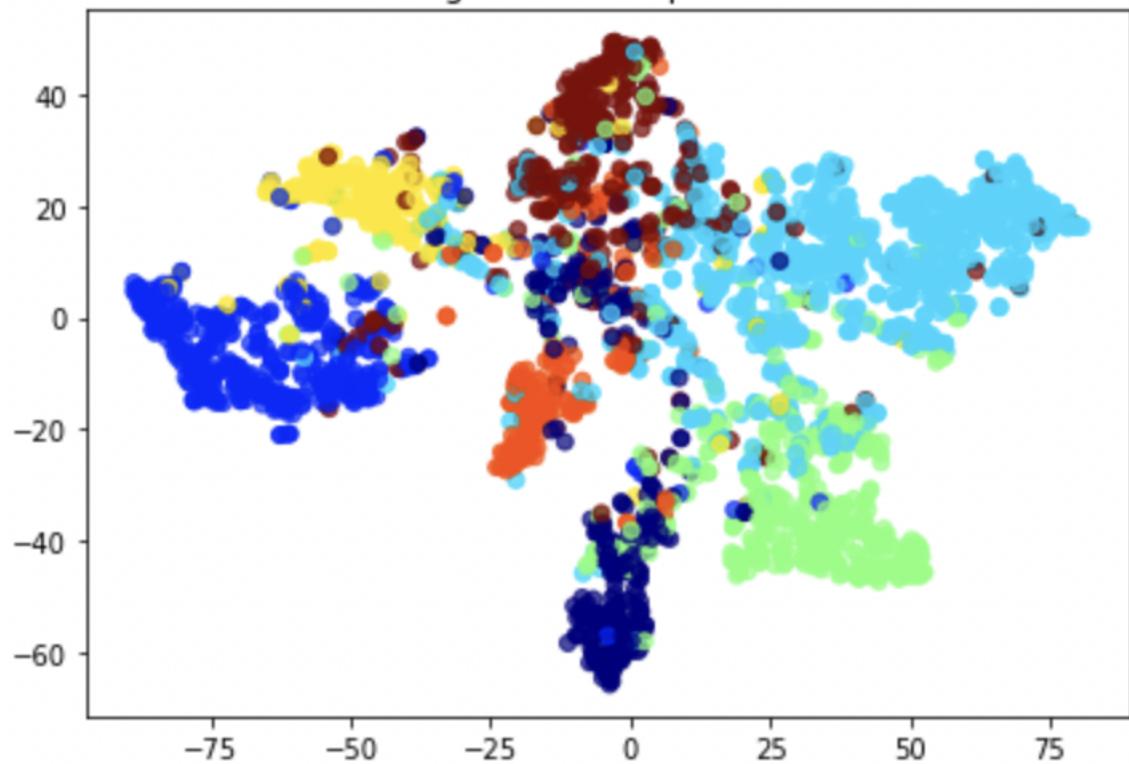


FIGURE 8 – Image détaillée montrant ce qu'a appris le modèle sur les noeuds et leurs voisinages

5.1.1 Image

La classification des images est très importante dans de nombreuses applications du monde réel. Dans la génération de graphe de scène, la relation sémantique entre les objets aide à comprendre la signification sémantique derrière la scène visuelle. les images non structurées peuvent être converties en données de graphe structurées et peuvent ainsi être appliquées à des réseaux convolutifs de graphes. Une autre application sur les images est la réponse visuelle aux questions. Dans [5], Narasimhan et Alexander proposent un modèle de deep learning basé sur un réseau convolutionnel de graphes pour utiliser les informations de plusieurs faits des images à partir de bases de connaissances pour faciliter la réponse aux questions, qui repose moins sur la récupération du seul fait correct des images. De plus, comme les images contiennent souvent plusieurs objets, comprendre les relations (c'est-à-dire les relations visuelles) entre les objets aide à caractériser les interactions entre eux, ce qui fait du raisonnement visuel un sujet brûlant en vision par ordinateur. Pour la détection des relations visuelles, Alexander propose un réseau convolutif de graphes pour exploiter à la fois les graphes sémantiques de mots et le graphe de scène spatial. Par ailleurs, Yao et al. [6] proposent une architecture de réseaux convolutifs de graphes et LSTM (Long short terme memory) pour explorer les relations visuelles pour le sous-titrage d'images. Un LSTM est une architecture de réseau de neurones récurrents utilisé dans le domaine de deep learning qui peut non seulement traiter des data points (points de données) uniques tels que des images, mais également des séquences complètes de données (telles que la parole ou la vidéo).

Aperçu de l'approche proposée en figure 9 : étant donné une image et une question, on utilise une technique de notation de similarité pour obtenir des faits pertinents à partir de l'espace des faits. Un LSTM prédit la relation à partir de la question pour réduire davantage l'ensemble des faits pertinents et ses entités. Un encastrement d'entité est obtenu en concaténant l'encastrement des concepts visuels de l'image , l'encastrement LSTM de la question , et l'encastrement LSTM de l'entité . Chaque entité forme un nœud unique dans le graphe et les relations constituent les arêtes. Un GCN suivi d'un MLP effectue une évaluation conjointe pour prédire la réponse. Notre approche est formée de bout en bout.

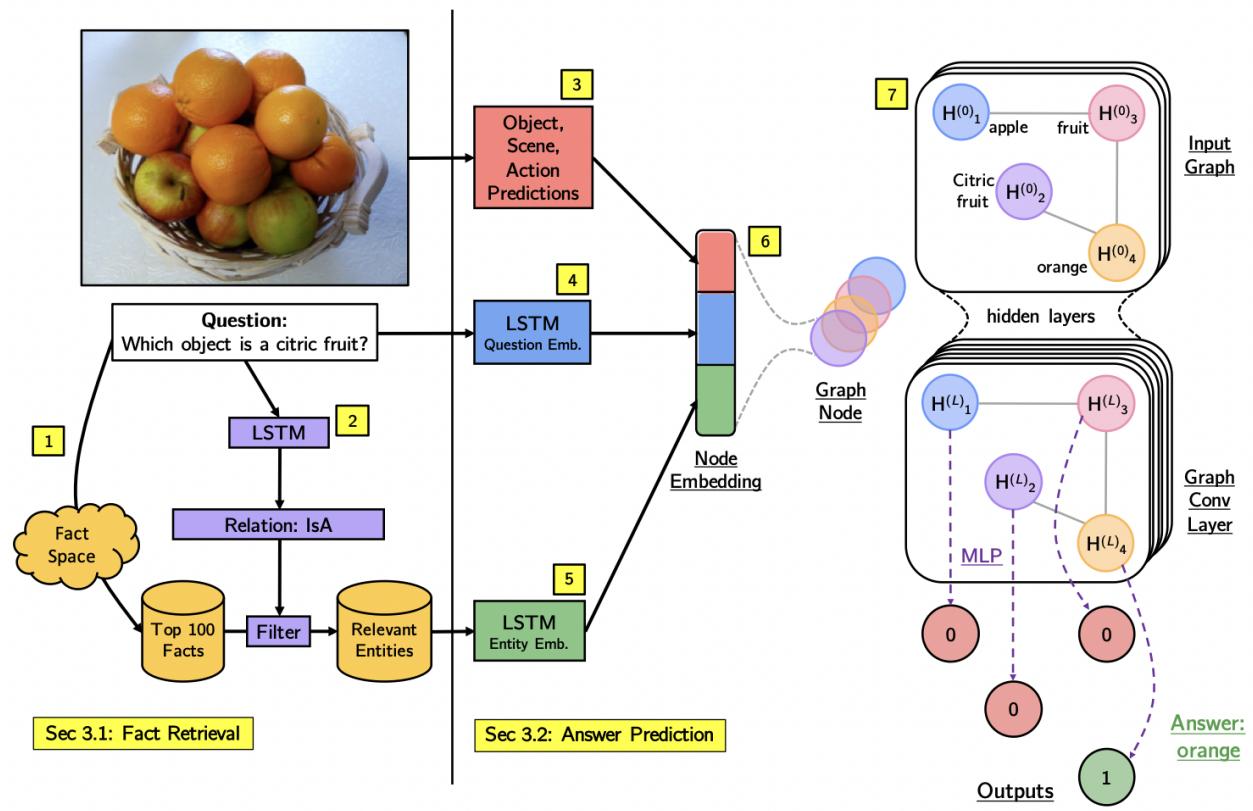


FIGURE 9 – La classification d’images par LSTM et GCN

5.1.2 Videos

Dans cette partie on vous fait part des explications données par Si Zhang et al. concernant l'utilisation des GCN dans le domaine des vidéos. L'une des applications à fort impact des vidéos est la reconnaissance d'action qui peut aider à la compréhension de la vidéo. *Un modèle convolutif de graphe spatio-temporel est conçu pour éliminer le besoin d'affectation de pièces artisanales et peut atteindre une plus grande puissance expressive. Une autre méthode basée sur le squelette où un processus de construction de graphe généralisé est proposé pour capturer la variation dans les séquences de squelette et le graphe généralisé est ensuite envoyé à un réseau convolutif de graphe pour l'apprentissage de la variation.* Wang et Gupta représentent la vidéo d'entrée comme un graphe de région spatio-temporelle qui construit deux types de connexions (c'est-à-dire la similarité d'apparence et la proximité spatio-temporelle), puis reconnaît les actions en appliquant des réseaux convolutionnels de graphe. Zhang et al. proposent un réseau tenseur convolutif pour la reconnaissance d'action [8] .

5.1.3 Point clouds : Nuage de points

Les nuages de points fournissent une représentation géométrique flexible pour de nombreuses applications en infographie et en vision par ordinateur. Suivis par le pionnier PointNet, les réseaux de neurones profonds de pointe prennent en compte les caractéristiques locales des nuages de points. Cependant, on ignore les relations géométriques entre les points. EdgeConv, d'autre part, est proposé pour capturer la structure géométrique locale tout en conservant la propriété d'invariance de permutation et surpassé les autres approches existantes dans la tâche de segmentation des nuages de points. Un modèle de réseau convolutif de graphes régularisé a été proposé pour la segmentation sur les nuages de points dans lequel le Laplacien de graphes est mis à jour dynamiquement pour capturer la connectivité des caractéristiques apprises. FeaStNet construit sur des réseaux convolutifs de graphes détermine dynamiquement l'association entre les poids de filtre et le voisinage de graphe, montrant une performance comparable dans l'étiquetage des parties. Si Zhang et al. [8]. proposent un réseau convolutif de graphes spectraux locaux pour la classification et la segmentation des nuages de points. Pour la classification des nuages de points, d'autres méthodes basées sur la convolution de graphes. Dans [7], Ying et al. proposent un modèle génératif localisé en utilisant la convolution de graphes pour générer des nuages de points

3D.

5.1.4 Mailles

Une application sur les maillages que nous considérons dans cette présentation est la correspondance de forme, c'est-à-dire pour trouver des correspondances entre des collections de formes 3D. Au-delà des méthodes classiques basées sur CNN , plusieurs approches basées sur des réseaux convolutifs de graphes ont été proposées. De plus, dans [8], Zhang et al. proposent de combiner des réseaux convolutifs de graphes avec un auto-encodeur variationnel pour la tâche de complétion de formes.

5.2 Le traitement du langage naturel

La classification de texte est l'un des problèmes les plus classiques du traitement du langage naturel. Avec les documents comme noeuds et les relations de citation entre eux comme arêtes, le réseau de citation peut être construit, dans lequel les attributs de nœud sont souvent modélisés par le sac de mots. Dans ce scénario, le moyen le plus simple pour classer les documents dans différentes classes consiste à classer les nœuds. De nombreux modèles de réseaux convolutifs de graphes ont été proposés. Une autre façon consiste à visualiser les documents au niveau du graphe (c'est-à-dire que chaque document est modélisé sous forme de graphe) et à classer les textes par classification de graphe. En outre, TextGCN modélise un corpus entier en un graphe hétérogène et apprend simultanément l'incorporation de mots et l'incorporation de documents, suivi d'un classificateur softmax pour la classification de texte [6].

Liang Yao et Chengsheng Mao [6] ont fait construire dans leur article un grand graphe de texte hétérogène qui contient des nœuds de mots et des nœuds de documents afin que la concurrence globale des mots puisse être explicitement modélisée et que la convolution du graphe puisse être facilement adaptée, comme le montre la figure 10. Le nombre de nœuds dans le graphe de texte V est le nombre de documents (taille du corpus) plus le nombre de mots uniques (vocabulaire).

L'extraction d'informations est souvent la pierre angulaire de nombreuses applications liées à la programmation neurolinguistique (PNL) et les réseaux convolutifs de graphes y ont été largement appliqués et ses problèmes de variantes. Par exemple, GraphIE utilise d'abord un réseau de neurones récurrent

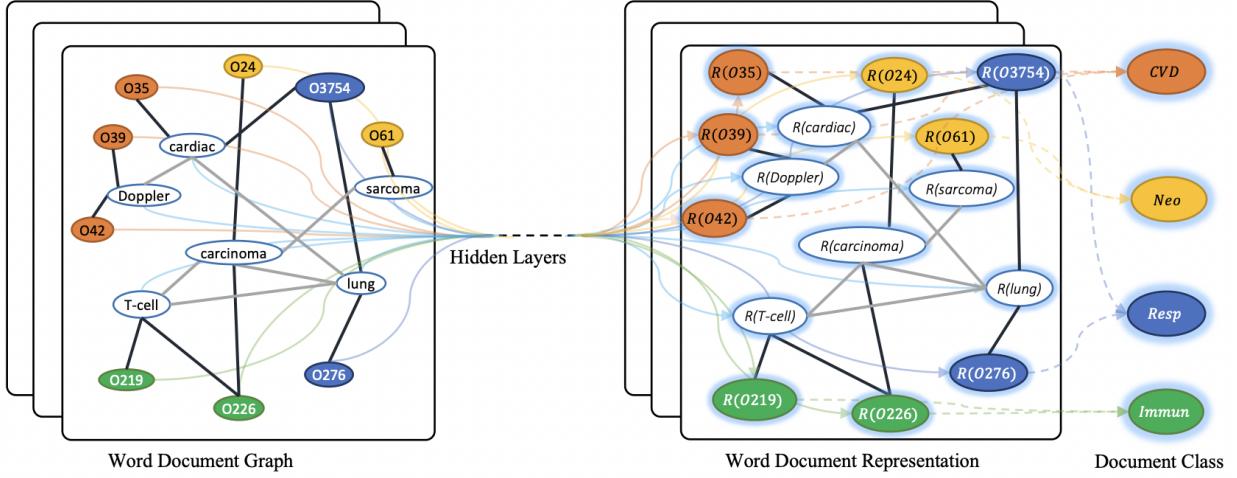


FIGURE 10 – Schéma du TextGCN, Exemple tiré [6] : Les nœuds commençant par « O » sont des nœuds de document, les autres sont des nœuds de mot. Les arêtes gras noirs sont des arêtes de mot de document et les arêtes fins gris sont des arêtes de mot-mot. $R(x)$ signifie la représentation (incorporation) de x . Différentes couleurs signifient différentes classes de documents (seulement quatre exemples de classes sont affichés pour éviter l'encombrement). MCV : Maladies cardiovasculaires, Néo : Tumeurs, Resp : Maladies des voies respiratoires, Immun : Maladies immunologiques.

pour générer des représentations cachées de mots ou de phrases sensibles au contexte local, puis apprend les dépendances non locales entre les unités textuelles, suivi d'un décodeur pour l'étiquetage au niveau du mot. GraphIE peut être appliqué à l'extraction d'informations telles que l'extraction d'entités nommées. Des réseaux convolutifs de graphes ont été conçus pour l'extraction de relations entre les mots et l'extraction d'événements.

Les réseaux convolutifs de graphes peuvent être utilisés pour injecter un biais sémantique dans les encodeurs de phrases et améliorer les performances . De plus, le modèle de réseau convolutif de graphe itéré dilaté est conçu pour l'analyse de dépendance

5.3 Applications en sciences

5.3.1 Physique

Ici Zhang et al. nous montrent comment les GCN sont appliqués en physique.

En physique des particules, les jets font référence aux pulvérisations collimatées de hadrons énergétiques et de nombreuses tâches sont liées aux jets, y compris les problèmes de classification et de régression as-

sociés aux particules progénitrices à l'origine des jets. Récemment, des variantes du réseau de neurones à passage de messages ont été conçues pour classer les jets en deux classes : les jets basés sur la chromodynamique quantique et les jets basés sur le boson W. ParticleNet, construit sur des convolutions d'arête, est une architecture de réseau de neurones personnalisée qui fonctionne directement sur les nuages de particules pour le marquage par jet. En outre, le modèle de réseau convolutif graphique a également été appliqué pour la classification des signaux IceCube. Une autre application intéressante est de prédire la dynamique physique, par exemple, comment un cube se déforme lorsqu'il entre en collision avec le sol. Utilisation des GCN dans la représentation d'objet basée sur un graphe hiérarchique qui décompose un objet en particules et relie les particules au sein d'un même groupe, ou aux ancêtres et descendants [8].

. Ils proposent ensuite un réseau convolutif de graphes hiérarchiques pour apprendre les prédictions physiques.

5.3.2 Chimie, Biologie et Science des matériaux

Dans cette partie Si Zhang et al. nous expliquent comment les GCN sont utilisées dans les domaines de la Chimie, la biologie et en science des matériaux.

L'apprentissage des molécules a attiré beaucoup d'attention en chimie, en découverte de médicaments et en science des matériaux. Par exemple, les réseaux convolutifs de graphes ont été utilisés pour la prédiction des empreintes moléculaires. Dans la découverte de médicaments, DeepChemStable , un mode de réseau convolutif basé sur l'attention, est utilisé pour la prédiction de la stabilité chimique d'un composé. En outre, en modélisant les interactions protéine-protéine, les interactions cible médicament-protéine dans un graphe multimodal, les convolutions de graphe peuvent être appliquées pour prédire les effets secondaires de la polypharmacie . Une autre application importante en chimie est la prédiction des propriétés moléculaires. Les réseaux de neurones à passage de messages (MPNN), un cadre général de réseau de neurones à graphes, peuvent être utilisés pour prédire les propriétés quantiques d'une molécule. PotentialNet implique d'abord des convolutions de graphes sur des liaisons chimiques pour apprendre

les caractéristiques des atomes, puis implique à la fois une propagation basée sur les liaisons et basée sur la distance spatiale et conduit enfin la collecte de graphes sur les atomes de ligand, suivi d'une couche entièrement connectée pour les prédictions de propriétés moléculaires. La prédiction de l'interface des protéines est un problème difficile avec des applications importantes dans la découverte de médicaments. De plus, un réseau neuronal convolutif permet d'apprendre directement les propriétés matérielles à partir de la connexion des atomes dans le cristal [8]

5.4 Analyse des réseaux sociaux

En plus des applications dans les problèmes classiques des sciences sociales, telles que la détection de communauté et la prédiction de liens , les réseaux convolutifs de graphes ont été appliqués dans de nombreux autres problèmes. DeepInf vise à prédire les influences sociales en apprenant les caractéristiques latentes des utilisateurs. Vijayan et al. proposent d'utiliser des réseaux convolutifs de graphes pour la prévision du nombre de retweets [8]. De plus, les fausses nouvelles peuvent également être détectées par des convolutions de graphes. Les réseaux convolutifs de graphes ont été largement utilisés pour la recommandation sociale qui vise à tirer parti des interactions utilisateur-élément ou utilisateur-utilisateur pour améliorer les performances de la recommandation. Dans [7] Ying et al. proposent un modèle de réseau convolutif de graphes très efficace PinSage basé sur GraphSAGE qui exploite les interactions entre les broches et les cartes dans Pinterest. Al. proposent un cadre de filtrage collaboratif de graphe neuronal qui intègre les interactions utilisateur-item dans le réseau convolutif du graphe et exploite explicitement les signaux collaboratifs.

5.5 Systèmes de recommandation à l'échelle du Web

PinSage est un algorithme GCN hautement évolutif capable d'apprendre les incorporations de noeuds dans des graphes à l'échelle du Web contenant des milliards d'objets. Dans [7], Rex Ying et Ruining He expliquent qu'ils ont déployé PinSage chez Pinterest et évalué de manière exhaustive la qualité des intégrations apprises sur un certain nombre de tâches de recommandation, avec des métriques hors ligne, des études d'utilisateurs et des tests A/B démontrant

tous une amélioration substantielle des performances de recommandation. Notre travail démontre l'impact que les méthodes de convolution de graphes peuvent avoir dans un système de recommandation de production, et nous pensons que PinSage peut être étendu à l'avenir pour résoudre d'autres problèmes d'apprentissage de représentation de graphes à grande échelle, y compris le raisonnement des graphes de connaissances et le regroupement de graphes.

6 Conclusion

Les réseaux convolutifs de graphes sont devenus un sujet très passionnant à la fois dans l'apprentissage automatique et dans d'autres domaines connexes, différents modèles de GCN ont été proposés pour résoudre différents problèmes. Dans cette présentation, nous avons fait une revue de littérature pour comprendre le fonctionnement et le domaine d'utilisation ou d'application des réseaux convolutifs de graphes. Pour se faire, nous avons mis en évidence avec quelques exemples détaillés d'un point de vue unique. En somme Les réseaux convolutifs de graphes occupent de nos jours une place très importantes dans notre société , on les retrouve partout en raison de leur puissance significative dans le traitement de données structurées en graphes. Ils sont en particulier très utilisés dans le domaines scientifiques tels que l'informatique, sciences sociales, biologie et physique.

Nous sommes ravis de cette première expérience dans le milieu de la recherche car nous avons pu découvrir des démarches et une nouvelle manière de travailler et pour cela, nous tenons à remercier sincèrement Monsieur Marc Tommasi pour l'encadrement et le suivi qu'il a assuré tout au long du projet.

Références

- [1] Olivier COGIS et Claudine ROBERT. *Théories des graphes, Problème, Théorème, Algorithme*. Eyrolles, 2003.
- [2] Rostyslav DEMUSH. *Gentle Introduction to Graph Neural Networks and Graph Convolutional Networks*. URL : <https://perfectial.com/blog/graph-neural-networks-and-graph-convolutional-networks/>. (accessed: 01.27.2021).
- [3] Thomas N KIPF et Max WELLING. « Semi-Supervised Classification with Graph Convolutional Networks ». In : *arXiv preprint arXiv:1609.02907* (2016).

- [4] Thomas N. KIPF et Max WELLING. « Semi-Supervised Classification with Graph Convolutional Networks ». In : *CoRR* abs/1609.02907 (2016). arXiv : 1609.02907. URL : <http://arxiv.org/abs/1609.02907>.
- [5] Medhini NARASIMHAN, Svetlana LAZEBNIK et Alexander G. SCHWING. « Out of the Box: Reasoning with Graph Convolution Nets for Factual Visual Question Answering ». In : *CoRR* abs/1811.00538 (2018). arXiv : 1811.00538. URL : <http://arxiv.org/abs/1811.00538>.
- [6] Liang YAO, Chengsheng MAO et Yuan LUO. « Graph Convolutional Networks for Text Classification ». In : *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence*. AAAI'19/IAAI'19/EAAI'19. Honolulu, Hawaii, USA : AAAI Press, 2019. ISBN : 978-1-57735-809-1. DOI : 10.1609/aaai.v33i01.33017370. URL : <https://doi.org/10.1609/aaai.v33i01.33017370>.
- [7] Rex YING et al. « Graph Convolutional Neural Networks for Web-Scale Recommender Systems ». In : *CoRR* abs/1806.01973 (2018). arXiv : 1806.01973. URL : <http://arxiv.org/abs/1806.01973>.
- [8] Si ZHANG et al. « Graph convolutional networks: a comprehensive review ». In : *Computational Social Networks* 6.1 (2019), p. 11. DOI : 10.1186/s40649-019-0069-y. URL : <https://doi.org/10.1186/s40649-019-0069-y>.