République Algérienne Démocratique et Populaire Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Djilali BOUNAAMA de Khemis Miliana

Faculté des Sciences et de la Technologie

Département des Mathématiques et Informatique

Mémoire Présenté

Pour l'obtention de diplôme de

Master en Informatique

Option : Ingénierie du Logiciel et Systèmes Distribués

Titre:

Développement et évaluation d'une nouvelle approche D'exploration des sous graphes sous plateforme big data

Présenté par :

- Abbes Yousef
- Djemai Mohamed Zineddine

Devant le jury composé de :

- Mde.hachichi : Président
- Mr.khalfi : Examinateur
- Mr.BAhloul : Examinateur

Mr O. harbouche : Encadrant

Année Universitaire 2018/2019

Remerciement

Avant tout, nous remercions Allah le Tout Puissant pour nous avoir donné la force et le courage d'accomplir ce travail avec abnégation.

Nos vifs et sincères remerciements

S'adressent spécialement à,

Mr. Oussema harbouche ,

Dont nous avons eu la chance de l'avoir comme encadrant et qui a bien voulu nous confier ce travail riche d'expériences et nous guider dans chaque étape de sa consécration. Vous nous avez toujours réservé un chaleureux accueil, malgré vos obligations et les contraintes professionnelles. Vos talents ainsi que vos compétences et votre sens du devoir nous ont marqué à jamais. Vos encouragements inlassables, votre amabilité, votre gentillesse et votre patience méritent toute notre attention. Veuillez trouver ici l'expression de notre estime et notre considération.

Nous remercions également,

Les membres du jury de nous avoir fait l'honneur de juger cette thèse. Veuillez accepter l'expression de nos vives gratitudes

En fin,

A toutes les personnes qui ont contribué de près ou de loin, d'une manière directe ou indirecte à l'élaboration de ce travail de fin d'études

Dédicaces

Je dédie ce modeste travail à :

Ceux qui sont toujours présents dans mon Cœur

Mes très chers parents pour leurs soutiens,

Leurs patiences, prières et leurs sacrifices,

A mes très chers frères ilyes, qui

ont toujours été là pour moi, et qui m'ont donné un

magnifique modèle de labeur

et persévérance.

A toute ma famille surtout zaki et chrif .

A tous mes amis notamment : Abdel Haq,

Abdel Rahim, benhlima, Mohamed ,toufik

et à tous mes amies pour leur soutien moral.

Un merci spécial à walid , youcef ,amer , omar ,amine

A toute la promo du Ingénierie du Logiciel et Systèmes

Distribués 2018/2019

Djemai mohamed zineddine

Au nom d'Allah, le tout miséricordieux, le très miséricordieux

Tout d'abord, nous tenons à remercier le tout puissant de nous avoir donné le courage et la patience pour arriver à ce stade afin de réaliser ce travail que nous dédisions :

v A ma très chère mère.

v A mon très cher père.

Qui m'ont donné le courage et qui m'ont porté toujours

. Á tous les membres de ma famille et toute personne qui porte le nom abbes , aussi, je dédie ce travail à tous mes chers amis : Abde Alah , , Abde alhak , Hamza , zaki , younes ,Rahim , lakhder

Sans oublier tous mes collègues de l'ingénierie du logicel

pour tout ses efforts, leur aide, leur présence et leur soutien.

Á la fin je dédie très chaleureusement ce mémoire à mon binôme :

Sans oublier : Youcef , Omar , Abde alhak , krimo et Zino

Á tous ceux qui ont participé à ma réussite.

Abbes youcef

Sommaire

Développement et évaluation d'une nouvelle approche D'exploration des sous graphes sous plateforme big data

Remerciement 1

Dédicace.....2

Sommaire 4

Liste des tableau 8

Liste des figures.....8

Résumé.....10

Introduction générale 1

Chapitre I

I. apprentissage machine et apprentissage en profondeur 4

I.1. Introduction :.....4

I.2. Intelligence artificielle : 5

I.3. L'apprentissage automatique (Machine Learning):..... 5

I.3.1. Pourquoi L'apprentissage automatique:..... 6

I.3.2. Types de systèmes de L'apprentissage automatique :.....7

I.3.2 a- L'apprentissage supervisé : 7

I.3.2 b- L'apprentissage non supervisé : 9

I.4. L'apprentissage profond (Deep Learning):.....14

I.4.1. Fonctionnement du l'apprentissage profond :.....14

I.4.2. Applications de l'apprentissage profond :.....15

Conclusion.....16

Chapitre II

II. Théorie des graphes et extraction fréquente de sous-graphes :.....18

II.1. Introduction :.....18

II.2. Data mining:.....18

II .3. Extraction de graphes:……..……20

II.4. Applications de graphe:……..……..……21.

II.5. En quoi consiste l'exploration de graphes :……..…….……25

II.6. Extraction de sous-graphes fréquents (FSM) :…….……26

II.7. Théorie des graphes :……26

II.8. Définitions préliminaires :……26

II.8.1. Types de graphes :……..……..…26

II.8.2. Sous-graphes :…….…….……32

II.8.3. Isomorphisme de graphe :…… …….33

II.8.4. Automorphisme :…… ……...……..35

II.8.5. Structure …… ……..……..…35

II.8.6. Densité :…….36

II.8.6. Arbres :…….…….…….…….37

II.9. Aperçu de FSM :…….……..37

II.10. Formalisme :……... …….……..39

II.11. Détection d'isomorphisme de graphe :……..……40

II.12. Stratéie de recherche: …… ….……..…….…….42

Conclusion : ……...……..43

Chapitre III

III l'apprentissage en profondeur sur les graphes

III.1. Introduction……..……45

III.2. Notations et préliminaire…….……...……...…46

III.3. Graphe neural réseaux (GNNS)…….……..…….……..…48

III.4. Réseaux convolutionnels de graphe (GCNS) 51

III.4.1 Opérations de convolution…….……..….51

III. 4.1.1. Méthodes spectrales……..…….51

III.4.1.2. Aspect efficacité…….52

III.4..1.3. Aspect des graphes multiple……..……..…..54

III.4..1.4. frameworks……..…….……..55

III.4.2. Opérations de lecture……..…..……57

III.4.2.1 Statistiques ……..……..… 58

III.4.2.2. Regroupement hiérarchique …….….. 59

III.4.2.3 Autres …….……...……59

III.4.3. Améliorations et discussions…….……...…60

III.4.3.1. Mécanisme d'attention ……...…….60

III.4.3.2. Connexions résiduelles .. ……..……..60

III.4.3.3. Caractéristiques du Bord…….……..…..62

III.4.3.4 . Accélération par échantillonnage……..…..……..……..63

III.4.3.5 . Réglage inductif ……..……..……64

III.4.3.6 . Poids aléatoires…….64

III.5. Autoencodeurs de graphe (GAES) ……... 64

III 5.1. Autoencodeurs……..……65

III. 5.2 . Autoencodeurs variationnels…….…….…..68

III.5.3. Améliorations et discussions…….…...…69

III.5.3.1 Formationcontradictoire…….….69

III.5.3.2 Apprentissage inductif et codeur GCN.....70

III.5.3.3 Mesures de similarité…….……70

Conclusion …….……...…..…71

Chapitre IV :

IV. Graphes Des Réseaux De Neurones Convolutionnels Basés Sur La Saillance Du Vertex Quantique……...….72

IV .1 Introduction……73

IV .2. Concepts préliminaires ……..….73

IV .2.1.Randonnées quantiques à temps continu ……..……73

IV .2.2. Alignement transitif entre sommets de graphes……...……75

IV .3. Réseau neurones convolutionnel de graphe SPATIAL……76

IV .3.1. Alignement des structures de graphes sur la grille de sommets……...……77

IV .3.2. La couche de convolution de graphes spatiaux quantiques…….……79

IV .3.3. Les couches de réseau de neurones convolutionnels traditionnels……..81

IV .3.4. Discussion sur le modèle QSGCNN proposé ……81

IV .4.Expériences ……84

IV .4.1. Comparaisons avec les noyaux de graphes ……...…..84

IV .4.2. Comparaisons avec les méthodes d'apprentissage en profondeur……..…..87

Conclusion ……...…89

Conclusion général …….……90

Références bibliographiqu…….....91

Liste des tableaux

Tab II-1: Catégorisation d'algorithmes de test d'isomorphisme de (sous-) graphes correspondant exactement……...…..41

Tab. III 1. Some main distinctions of deep learning methods on graphes……49

Tab III 2.Tableau des notations couramment utilisées……...……..50

Tab IV .1 Informations sur les fichiers de données de graphe ……....85

Tab IV .2Précision du classement du classement pour comparaisons avec les noyaux graphes…..85

Tab IV .3.Exactitude du classement pour les comparaisons avec des réseaux de neuron graphes convolutionnel……...……85

Liste des figures

Figure I.1 : La relation entre IA, ML et DL.....4

Figure I.2 : L'approche traditionnelle.....6

Figure I.3 : Exemple de problème de classification.....8

Figure I.4 : Exemple de problème de régression 9

Figure I.5 : Exemple de données de $\mathbb{R}2$ et de densité estimée.....10

Figure I.6: Assemblage- Maximisation des attentes.....11

Figure I.7 : Malédiction de la dimoensionnalité……..……..12

Figure I.8 : Le cycle d'apprentissage par renforcement…….....12

Figure I.9: Principe général de (a) l'apprentissage automatique traditionnel et (b).....13

Figure I.10 : Schéma de fonctionnement de Deep Learning.....14

Figure II-1: Les étapes du processus de KDD (14)……..….19

Figure II-2: Concept associé à l'extraction de graphes…….…….20

Figure II-3: Représentation de graphes d'un composé chimique (13)……..…….22

Figure II-4: Sous-graphe fréquent extrait de divers composés chimiques……....22

Figure II-5: Réseau d'interaction protéine-protéine…….…..23

Figure II-6: Graphes de médias sociaux: il existe des groupes complexes, des liens……...24

Figure II-7: Règles d'évolution des graphes d'exploitation……...……25

Figure II-8: (a) graphe simple, (b) nom graphe simple ayant arête multiple (gauche) nom graphe simple ayant des boucles (droite)……...…27

Figure II-9: Graphes 3-régulier k = 3……27

Figure II-10: Graphes sans étiquette et étiquetés…….……...…..….28

Figure II-5: Réseau d'interaction protéine-protéine…….……...……...28

Figure II-11: Graphe non dirigé (à gauche), Graphe dirigé (à droite)……...…28

Figure II-12: Deux graphes complets et connectés avec 5 et 3 sommets respectivement, lorsqu'ils sont considérés ensemble, ils forment un graphe déconnecté……...……32

Figure II-13: (a) un graphes G, (b), (c), (d) représente respectivement un sous-graphe général, induit et connecté (15)……..…….33

Figure II-14: Isomorphisme de graphe……..……34

Figure II-15: Isomorphisme de sous-graphe…….....34

Figure II-16: Réseau (G)……..……..…36

Figure II-17: Répartition des algorithmes FSM ……..……...…..38

Figure II-18: Stratégies de recherche DFS et BFS……42

Figure III 1: La catégorisation des méthodes d'apprentissage en profondeur sur des graphes…...49

Figure III 2: Un exemple illustrant le fonctionnement de la convolution . Réimprimé avec permission……...……...…53

Figure III 3. Un exemple de classification hiérarchique d'un graphe. avec la permission……...…58

Figure III 4: Une illustration de l'attention multi-têtes où chaque couleur dénote une attention indépendante. Réimprimé avec permission……..……..…61

Figure III 5: Réseaux de connaissances ……...……... 62

Figure III 6: Méthode d'échantillonnage de noeud de FastGCN …….64

Figure III 7: Le framework de SDNE avec permission……..…….66

Figure III 8: Le cadre de DVNE avec permission……..69

Figure. IV .1. L'architecture du modèle QSGCNN proposé…….75

Figure. IV .2. Procédure de calcul de la matrice de correspondance……..……..79

الملخص

تعتبر الرسوم البيانية من اهم اساليب تمثيل البيانات و دراستها كما تعتبر الرسوم البيانية وسيلة هامة لفهم بعض العناصر المعقدة فانتشرت في الآونة الاخيرة عدة دراسات حول مجال تعدين البيانات و دراسة الرسوم البيانية و لعل اهم مجالات هذه الدراسات هو مجا دراسة الرسوم البيانية المتكررة لأهميتها في عدة مجالات الطبية و الشبكات الاجتماعية و بحثنا مركز على هذا المجال فقمنا بدراسة مجموعة من البيانات في شكل رسوم بيانية للعدة مركبات كيمائية تستعمل في العلاج الطبي محاولين استخراج و الرسوم البيانية الفرعية المتكررة للوصول لأنماط جديدة مستعملين احدث الاساليب التقنية او ما يعرف بأساليب الذكاء الاصطناعي مستعملين انواع الشبكات العصبية الاصطناعية في مجال ربما هو الاحدث و الأهم في مجال دراسة علوم الحاسوب

الكلمات الدالة : تعلم الي ، رسم بياني، الشبكات العصبية، التعلم المعمق.

Résumé

Les graphes sont l'une des méthodes les plus importantes de représentation et d'étude des données. Les graphes sont un moyen important de comprendre certains éléments complexes. Réseaux médicaux et sociaux et notre centre de recherche dans ce domaine, nous avons étudié un ensemble de données sous forme de graphes de plusieurs composés chimiques utilisés dans le traitement médical en essayant d'extraire et de sous-graphes récurrents pour atteindre de nouveaux modèles en utilisant les dernières techniques. Il est connu comme les méthodes des utilisateurs d'intelligence artificielle types de réseaux de neurones artificiels dans le domaine est probablement le dernier et le plus important dans l'étude de l'informatique

Mots-clés : apprentissage automatique, graphe, Réseaux de neurones, Apprentissage profond.

Abstract

Graphs are considered one of the most important methods of data representation and study. Graphs are an important means to understand some complex elements. Recently, several studies have been published on the field of data mining and studying graphs. Perhaps the most important areas of these studies is the study of repeated graphs of their importance in several fields. Medical and social networks and our research center on this area, we studied a set of data in the form of graphs of several chemical compounds used in medical treatment trying to extract and recurring sub-graphs to reach new patterns using the latest Yep technical or what is known as the methods of artificial intelligence users kinds of artificial neural networks in the field is probably the latest and most important in the study of computer science

Keywords : machine learning ,graph, Neural networks, Deep learning

Introduction générale

Introduction générale

Les graphes sont des structures de données courantes utilisées pour représenter et modéliser des systèmes réels, par exemple les réseaux sociaux, les molécules chimiques, la carte des routes dans un pays. L'extraction fréquente de sous-graphes qui fait l'objet du présent rapport est considérée comme une sous-section du domaine d'extraction graphe qui est largement utilisée pour identifier les sous-graphes dans de grands ensembles de données graphes dont le nombre d'occurrences est supérieur à un seuil de support minimum spécifié, également utilisés pour la classification des graphes, les indices de construction et les objectifs de clustering des graphes. L'exploration de sous-graphes fréquents est abordée sous différents angles et vue dans des directions différentes en fonction des attentes du domaine.

Ces dernières années,de nombreux auteurs ont déployé de nombreux algorithmes et outils pour convertir des données volumineuses en informations utiles et significatives .L'extraction fréquente de graphes est l'un des bras de ces techniques lorsque les données sont représentées sous forme de graphes .L'identification de graphes fréquents/sous-graphes dans une base de données graphe ou dans un seul grand graphe fait partie de l'exploration de graphe fréquente qui peut être utilisée pour des tâches de classification ,le clustering de graphes et les indices de construction. La découverte fréquente de sous-graphes est un processus d'identification des sous-graphes fréquemment présents à partir d'un ensemble de graphes (base de données de graphes) ou d'un seul graphe de grande taille avec fréquence d'occurrence n'est pas inférieur à un seuil spécifié.

L'apprentissage profond a été couronné de succès dans un certain nombre de domaines, allant de l'acoustique, des images au traitement du langage naturel. Cependant, l'application de l'apprentissage profond aux données graphes omniprésentes n'est pas négligeable en raison des caractéristiques uniques des graphes. Récemment, un nombre important d'efforts de recherche ont été consacrés à ce domaine, faisant grandement progresser les techniques d'analyse graphe. Dans cette enquête, nous passons en revue de façon exhaustive différents types de méthodes d'apprentissage profond appliquées aux graphes. Nous divisons les méthodes existantes en trois grandes catégories : les méthodes semi-supervisées, y compris les réseaux neuronaux graphes et les réseaux convolutionnels graphes, les méthodes non supervisées, y compris les autoencodeurs graphes, et les progrès récents, y compris le graphe recourant neuronal Réseaux et apprentissage du renforcement graphe. Nous fournissons

ensuite un aperçu complet de ces méthodes d'une manière systématique suivant leur histoire des développements.

- **Organisation du document**

Le reste de ce document est organisé comme suit :

Chapitre 1 : **l'apprentissage automatique et l'apprentissage en profondeur**

Dans ce chapitre on va parler d'apprentissage automatique et d'apprentissage en profondeur (notions de base et définitions dans ce domaine)

Chapitre 2 : **Théorie des graphes et extraction fréquente de sous-graphes**

Dans ce chapitre, nous présentons les théories, la terminologie et les algorithmes les plus importants liés aux graphes et aux sous-graphes.

Chapitre 3 : **Deep learning on graphe :**

Dans ce chapitre, nous aborderons les concepts les plus importants liés aux graphes dans le domaine de l'apprentissage en profondeur, ainsi que l'utilisation des types les plus importants de réseaux de neurones artificiels sur les graphes (GNN,GCN , AutoEncodeur) et les travaux les plus importants dans ce type de recherche.

Chapitre I

Apprentissage machine et apprentissage en profondeur

I.1. Introduction :

Depuis son point de départ dans les années 1950, notamment avec les travaux du mathématicien Alan TURING, l'intelligence artificielle (IA), discipline mathématique et technique destinée à reproduire l'intelligence humaine, s'est développée par cycles successifs parallèlement à la croissance de la puissance de calcul informatique disponible. Dans les années 1980, le concept d'apprentissage automatique («Machine Learning») se développe permettant à une machine de déduire une «règle à suivre » uniquement à partir de l'analyse de données. Cette période voit apparaître la majorité des algorithmes « apprenants » utilisés aujourd'hui (réseau de neurones, apprentissage par renforcement, machines à vecteurs de support, etc.). Ces avancées se concrétisent notamment par le succès de l'ordinateur Deep Blue face au grand maître d'échecs, Gary KASPAROV en 1992.À partir des années 2000, un nouveau cycle d'innovations en IA se met en place avec le formidable développement d'Internet et des très grandes infrastructures de calcul, offrant un accès à un volume de données encore jamais atteint dans l'histoire humaine. Avec cette capacité nouvelle, le développement des techniques d'apprentissage profond («Deep Learning») permet aux machines de commencer à surpasser les performances des meilleurs experts humains dans des domaines comme la reconnaissance visuelle, l'analyse documentaire ou la traduction.

Dans ce chapitre on va parler de quelques notions sur l'intelligence artificielle, apprentissage automatique (machine learning) , apprentissage profond(deep learning) avec les dernières actualités dans ces domaines et les liens entre eux.

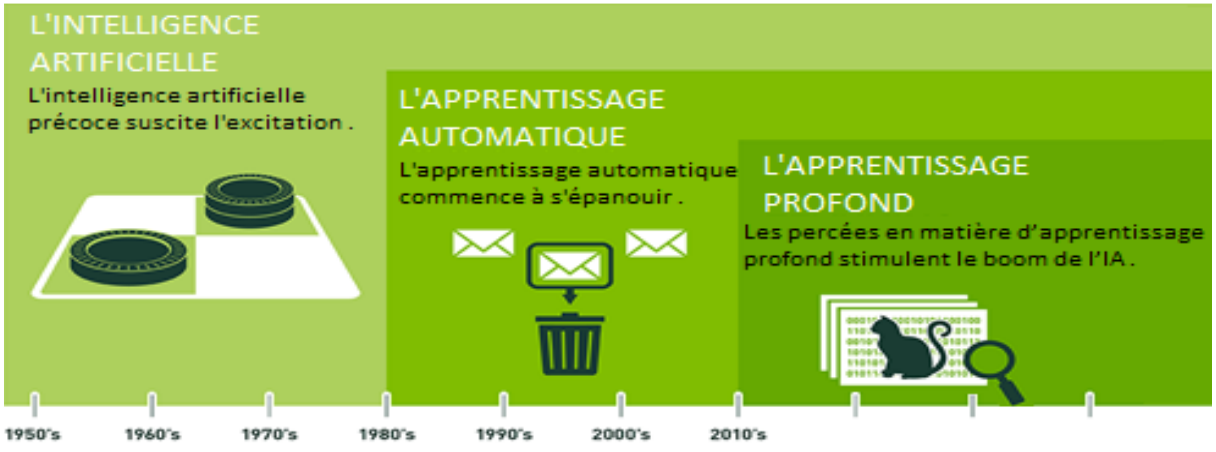


Figure I.1 : La relation entre IA, ML et DL

I.2. Intelligence artificielle :

L‘intelligence artificielle (IA) correspond à un ensemble de technologies qui permet de simuler l‘intelligence et d‘accomplir automatiquement des tâches de perception, compréhension et prise de décision. Ces techniques font particulièrement appel à l‘utilisation de l‘informatique, de l‘électronique, des mathématiques (notamment statistiques), des neurosciences et des sciences cognitives. [1]

Ces techniques montrent certains aspects de l'intelligence humaine. Mais comment est-ce arrivé? D'où vient cette intelligence? Cela nous amène au cercle suivant.

I.3. L'apprentissage automatique (Machine Learning):

L‘apprentissage automatique est devenu l‘un des sujets les plus importants au sein des organisations de développement qui cherchent des moyens novateurs de tirer parti des ressources de données pour aider l‘entreprise à acquérir un nouveau niveau de compréhension. Pourquoi ajouter l‘apprentissage automatique dans le mix ? Avec les modèles d‘apprentissage automatique appropriés, les organisations ont la capacité de prédire continuellement les changements dans l‘entreprise de sorte qu‘ils sont le mieux en mesure de prédire ce qui est à venir. Comme les données sont constamment ajoutées, les modèles d‘apprentissage automatique garantissent que la solution est constamment mise à jour. La valeur est simple : si vous utilisez les sources de données les plus appropriées et en constante évolution dans le contexte de l‘apprentissage automatique, vous avez la possibilité de prédire l‘avenir.

L‘apprentissage automatique est une forme d‘intelligence artificielle qui permet à un système d‘apprendre des données plutôt que par une programmation explicite. Cependant, l‘apprentissage automatique n‘est pas un processus simple.

L‘apprentissage automatique utilise divers algorithmes qui tirent des leçons itératives des données pour améliorer, décrire les données et prévoir les résultats. Comme les algorithmes ingèrent des données de formation, il est alors possible de produire des modèles plus précis basés sur ces données. Un modèle d‘apprentissage automatique est la sortie générée lorsque vous entraînez votre algorithme d‘apprentissage automatique avec des données. Après la formation, lorsque vous fournissez un modèle avec une entrée, vous recevrez une sortie. Par exemple, un algorithme prédictif créera un modèle prédictif. Ensuite, lorsque vous fournirez des données au modèle prédictif, vous recevrez une prévision fondée sur les données qui ont formé le modèle. L‘apprentissage automatique est maintenant essentiel pour créer des modèles analytiques. [2]

I.3.1. Pourquoi L'apprentissage automatique ?

Supposons que vous souhaitiez écrire le programme de filtrage sans utiliser de méthodes d‘apprentissage automatique. Dans ce cas, vous devrez suivre les étapes suivantes:

- Au début, vous examineriez à quoi ressemblent les spams. Vous pouvez les sélectionner pour les mots ou les expressions qu‘ils utilisent, comme «carte de débit», «libre», etc., ainsi que pour les modèles utilisés dans le nom de l‘expéditeur ou dans le corps du message.
- Deuxièmement, vous écririez un algorithme pour détecter les modèles que vous avez vus, puis le logiciel marquerait les e-mails en tant que spam si un certain nombre de ces modèles sont détectés.
- Enfin, vous testeriez le programme, puis renfaîtiez les deux premières étapes jusqu'à ce que les résultats soient suffisamment bons.

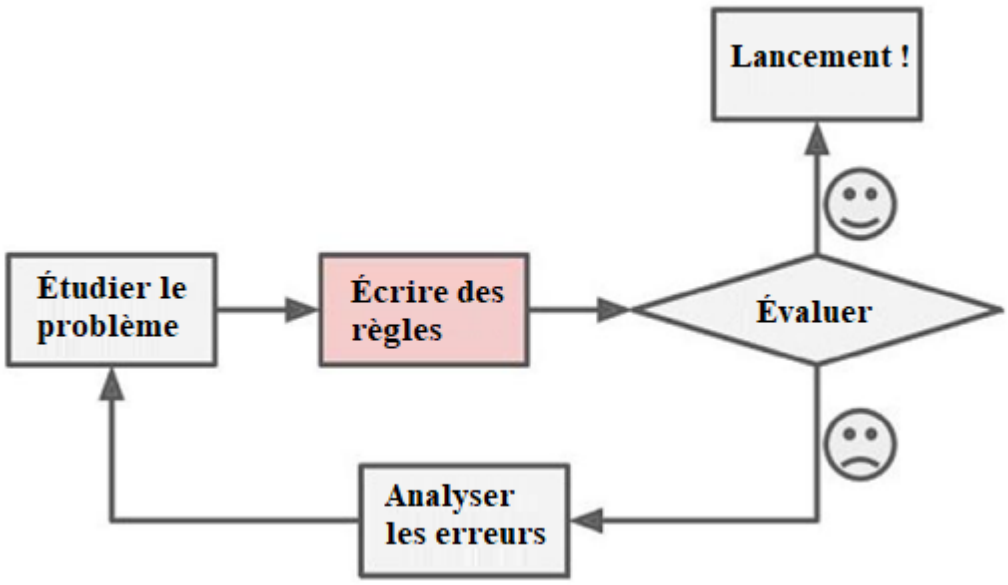


Figure I.2 : L'approche traditionnelle.

Comme le programme n'est pas un logiciel, il contient une très longue liste de règles difficiles à maintenir. Mais si vous avez développé le même logiciel en utilisant ML, vous pourrez le maintenir correctement. [3]

I.3.2. Types de systèmes de L'apprentissage automatique :

Il existe différents types de systèmes d'apprentissage automatique. Nous pouvons les diviser en catégories, selon que

- Ils ont été formés avec des humains ou non humains :

- Supervisé ;
- Non supervisé ;
- Semi-supervisé ;
- Apprentissage par renforcement.

- S'ils peuvent apprendre progressivement.
- S'ils travaillent simplement en comparant de nouveaux points de données pour trouver des points de données, ou peuvent détecter de nouveaux modèles dans les données, ils créeront ensuite un modèle. [3]

A. L'apprentissage supervisé :

Dans l'apprentissage supervisé (ou l'apprentissage avec un expert), les données sont des échantillons de motifs entrée-sortie.

Dans ce cas, une description concise des données est la fonction permettant de générer la sortie d'une entrée donnée. Ce problème est appelé apprentissage supervisé car les objets considérés sont associés déjà avec valeurs attendues (par ex classes et valeurs réelles).

Dans le problème d'apprentissage supervisé, un échantillon de couples entrée-sortie donné est appelé « échantillon d'entrainement »(ou ensemble d'entrainement), la tâche est de trouver une fonction déterministe qui mappe n'importe quelle entrée à une sortie qui est capable de prédire les futures observations entrée-sortie, et diminuer le possible les erreurs. Pour n'importe quelle valeur attendue d'un objet présenté dans l'échantillon d'entrainement elle retourne la valeur la plus apparue dans l'ensemble d'entrainement avec cet objet. [2]
On distingue l'apprentissage en classification ou régression selon le type de sorties.

● Classification :

Si l'espace de sorties n'a pas de structure sauf en cas d'égalité (ou pas) de deux éléments de sortie, ceci est appelé « le problème de classification d'apprentissage » (ou simplement classification). Chaque élément de l'espace de sortie est appelé « classe » (en anglais class).

L‘algorithme qui résout le problème de classification est appelé « classificateur ». Dans les problèmes de classification la tâche est d‘affecter de nouvelles entrées à un nombre de catégories ou classes discrètes. [2]

Exemple de problème de classification :

Objectif : Prédire qui gagne plus de 50k$ à partir de données de recensement. [4]

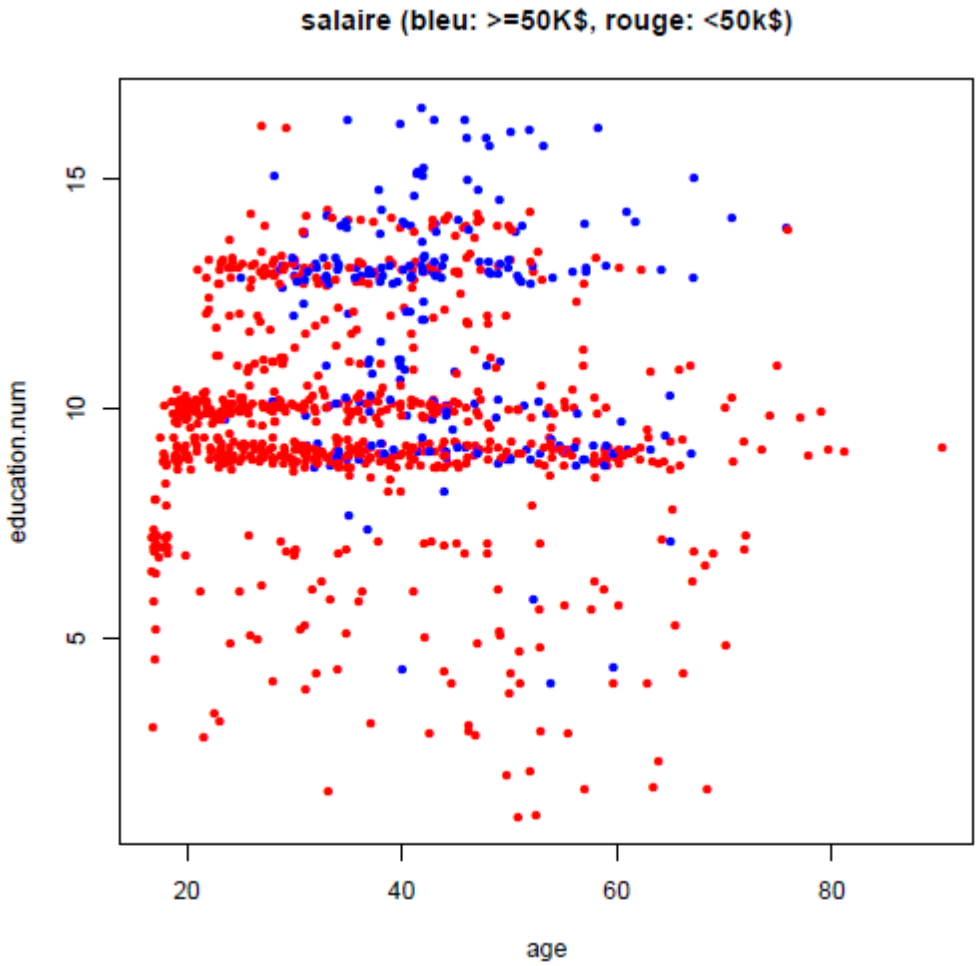


Figure I.3 : Exemple de problème de classification.

● **Régression :**

Si l‘espace de sortie est formé par les valeurs des variables continues, pour un instant donné l‘échange de stock s‘indice sur quelques futures temps, la tache d‘apprentissage est connue comme un problème de régression ou « fonction d‘apprentissage ». Les exemples typiques de régression sont de prédire la valeur de partages dans le marché d‘échange de stock et d‘estimer la valeur d‘une mesure physique dans une section d‘un plat thermoélectrique. [2]

Exemple de problème de régression :

Prédire l'âge d'un ormeau (abalone) à partir de sa taille, son poids, etc. [4]

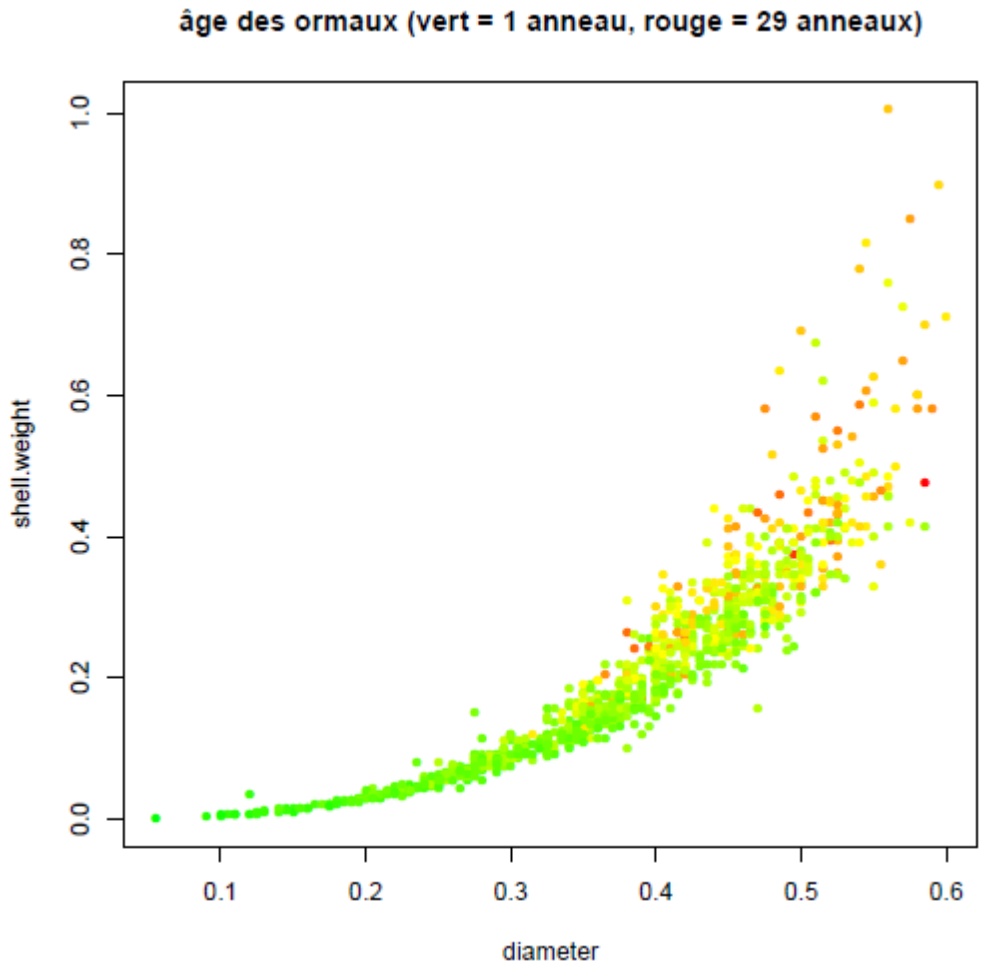


Figure I.4 : Exemple de problème de régression .

B. L'apprentissage non supervisé :

Dans l'apprentissage non supervisé il n'y a pas de notion de sortie désirée, on dispose seulement d'un nombre fini de données d'apprentissage, constituées ―d'entrées‖, sans qu'aucun label n'y soit rattaché. [5]

- **Estimation de densité :**

Dans un problème d'estimation de densité, on cherche à modéliser convenablement la distribution des données. L'estimateur obtenu **f̂(x)** doit pouvoir donner un bon estimé de la

densité de probabilité à un point de test x issu de la même distribution (inconnue) que les données d'apprentissage. [5]

Étant donné un ensemble de ***N*** observations vectorielles **D*N*={x1,…,x*N*}** décrites par ***d*** variables, donc **D*N*** ⊂ **R*d*** , l'objectif général de l'estimation de densité est de trouver la fonction de densité de probabilité ***f*** qui a généré l'échantillon aléatoire **D*N*** .

La figure suivante montre un exemple dans lequel les observations (représentées par les points bleus dans le plan horizontal) sont issues de $\mathbb{R}$**2** et la densité qui a généré ces points est la surface courbe. Le sommet (en rouge) sur cette surface correspond à la région du plan horizontal où les observations (les points bleus) sont les plus denses. [6]

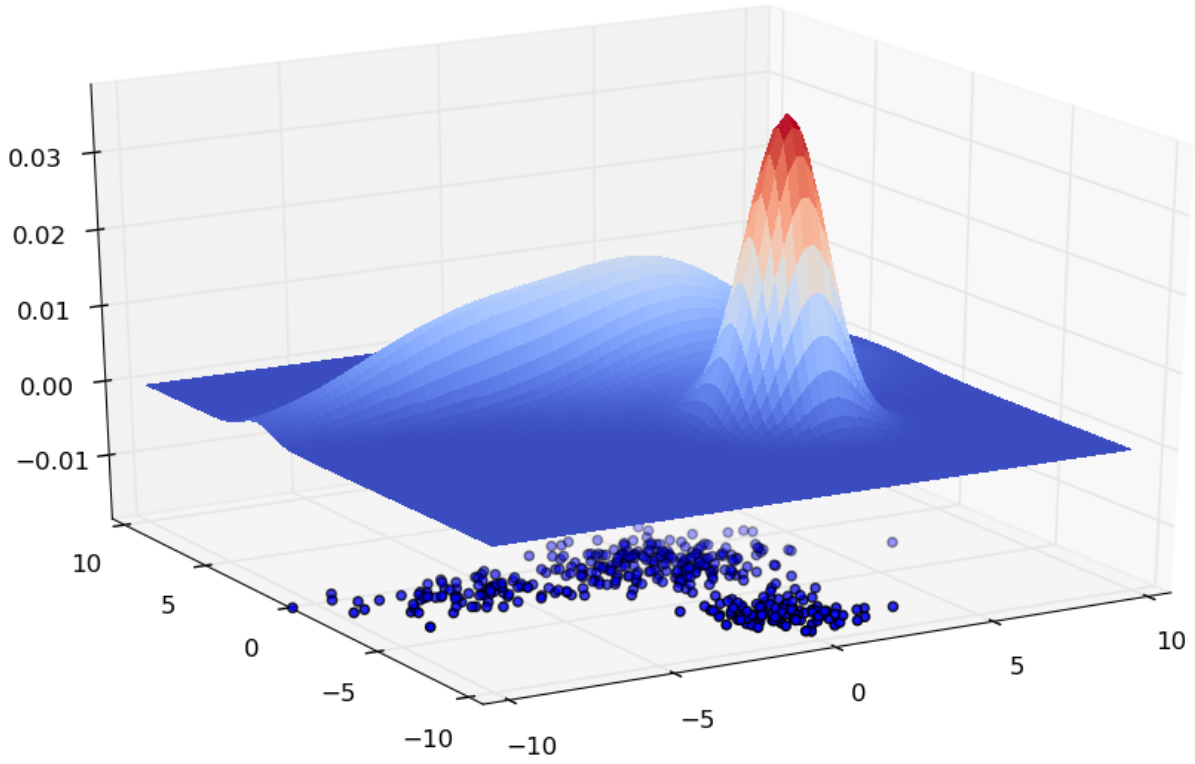


Figure I.5 : Exemple de données de $\mathbb{R}$2 et de densité estimée.

- ## Partitionnement (Clustering) :

Le problème du partitionnement est le pendant non-supervisé de la classification. Un algorithme de partitionnement tente de partitionner l'espace d'entrée en un certain nombre de ―classes _ en se basant sur un ensemble d'apprentissage fini, ne contenant aucune information de classe explicite.

Les critères utilisés pour décider si deux points devraient appartenir à la même classe ou à des classes différents sont spécifiques à chaque algorithme, mais sont très souvent liés à une mesure de distance entre points. [5]

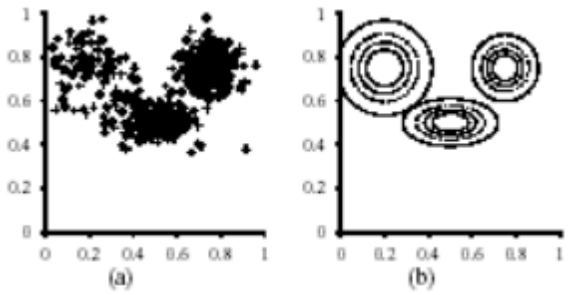


Figure I.6: Partitionnement - Maximisation des attentes.

- **Réduction de dimensionnalité :**

Le but d‘un algorithme de réduction de dimensionnalité est de parvenir à ―résumer‖ l‘information présente dans les coordonnées d‘un point en haute dimension ($x \epsilon Rn$, $n\ grand$) , par un nombre plus réduit de caractéristiques. Le but espéré est de préserver l‘information ―importante‖, de la mettre en évidence en la dissociant du bruit, et possiblement de révéler une structure sous-jacente qui ne serait pas immédiatement apparente dans les données d‘origine en haute dimension. L‘exemple le plus classique d‘algorithme de réduction de dimensionnalité est l‘Analyse en Composantes Principales (ACP). [7]

Raisons pour réduire la dimensionnalité :

- Malédiction de la dimensionnalité :

- Ajout d‘une dimension augmente exponentiellement l‘espace mathématique.
- 100 points équidistants de 0,01 en une dimension⇒1020 points nécessaires en 10 dimensions pour conserver la même densité.
- Grande dimensionnalité : complexité élevée en calculs et en mémoire.

- Epargner des coûts de mesures.

- Plus un modèle est simple, moins il y a de variance.

- Plus facile d‘expliquer avec moins de variables : extraction de connaissances.

- Visualiser les données : analyse de résultats. [7]

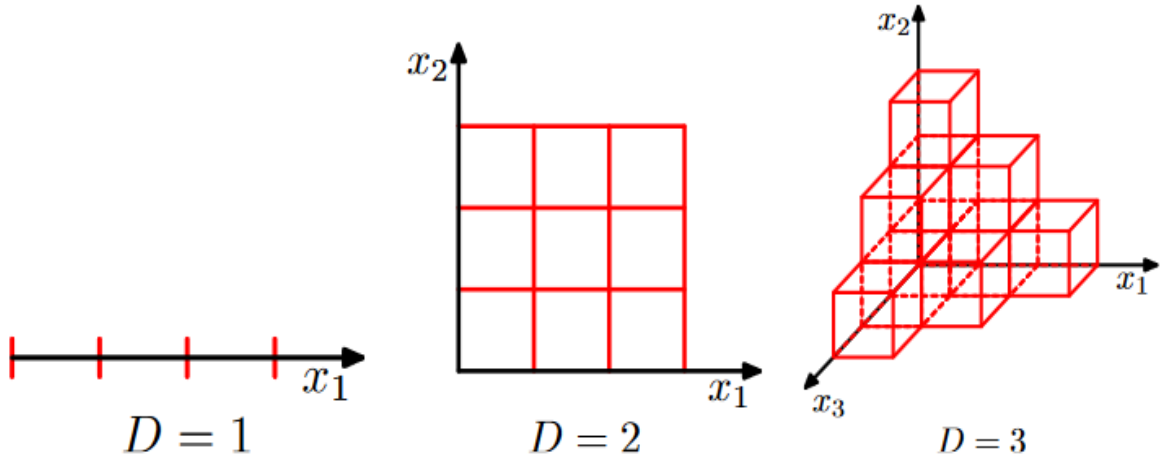


Figure I.7 : Malédiction de la dimensionnalité.

- **L'apprentissage par renforcement :**

Nous ne faisons ici que mentionner très succinctement le cadre général de l'apprentissage par renforcement, ce domaine étant hors du champ de notre sujet.

La particularité et la difficulté du cadre de l'apprentissage par renforcement est que les décisions prises par l'algorithme influent sur l'environnement et les observations futures.

L'exemple typique est celui d'un robot autonome qui évolue et effectue des actions dans un environnement totalement inconnu initialement. Il doit constamment apprendre de ses erreurs et succès passés, et décider de la meilleure politique à appliquer pour choisir sa prochaine action. [8]

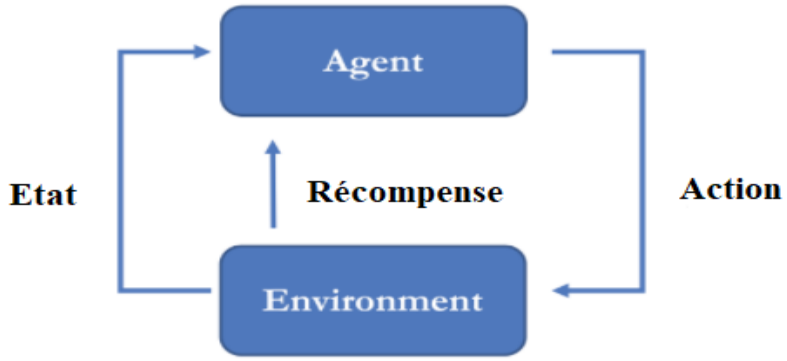


Figure I.8 : Le cycle d'apprentissage par renforcement

- **L’apprentissage par transfert :**

L‘apprentissage par transfert fournit un cadre théorique et méthodologique intéressant permettant d‘adapter un modèle appris à partir d‘un domaine et d‘une tâche source vers un nouveau domaine et/ou une nouvelle tâche cible. Le domaine fait référence à l‘espace de représentation des données ainsi qu‘à la répartition de ces données dans cet espace ; la tâche désigne à la fois l‘espace des labels et la fonction de prédiction qui associe un label à un exemple de test. Par exemple, pour une tâche de classification d‘expressions faciales, les exemples labélisés peuvent être des images de visage auxquelles on associe une catégorie d‘émotion (joie, colère, tristesse...). Si ces images ont été capturées dans des conditions spécifiques(en intérieur avec une caméra haute résolution par exemple), le système appris sur le domaine source ne pourra certainement pas être appliqué directement sur une image de visage prise en extérieur par une webcam. Pour autant il serait intéressant de transférer le domaine source vers le domaine cible car il est couteux et parfois impossible d‘annoter les données cibles. [9]

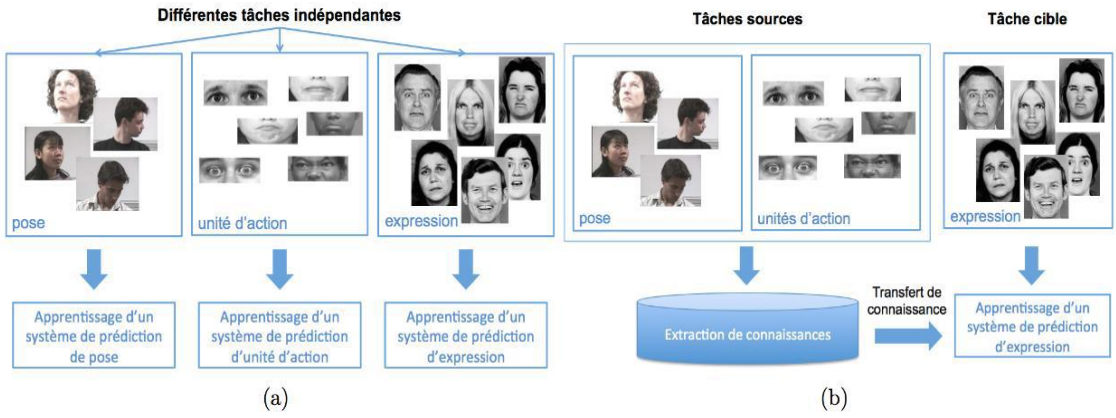


Figure I.9: Principe général de (a) l’apprentissage automatique traditionnel et (b) l’apprentissage par transfert.

Les domaines d‘application du Transfer Learning sont nombreux. Principalement, les méthodes de transfert de connaissance sont très souvent utilisées pour la reconnaissance d‘image ainsi que le traitement automatique du langage. Ces deux domaines d‘apprentissage sont très complexes et chronophages. C‘est pour cela que le Transfer Learning apporte un souffle nouveau pour tenter d‘optimiser ces traitements en exploitant au maximum des modèles déjà entraînés. Nous allons voir ici plusieurs méthodes de Transfer Learning. [10]

I.4. L’apprentissage profond (Deep Learning):

Le Deep Learning ou apprentissage profond est un type d'intelligence artificielle dérivé d‘apprentissage automatique (Machine Learning) où la machine est capable d'apprendre par elle-même, contrairement à la programmation où elle se contente d'exécuter à la lettre des règles prédéterminées. [11]

I.4.1. Fonctionnement du l’apprentissage profond :

Le Deep Learning s'appuie sur un réseau de neurones artificiels s'inspirant du cerveau humain. Ce réseau est composé de dizaines voire de centaines de « couches » de neurones, chacune recevant et interprétant les informations de la couche précédente. Le système apprendra par exemple à reconnaître les lettres avant de s'attaquer aux mots dans un texte, ou détermine s'il y a un visage sur une photo avant de découvrir de quelle personne il s'agit. [12]

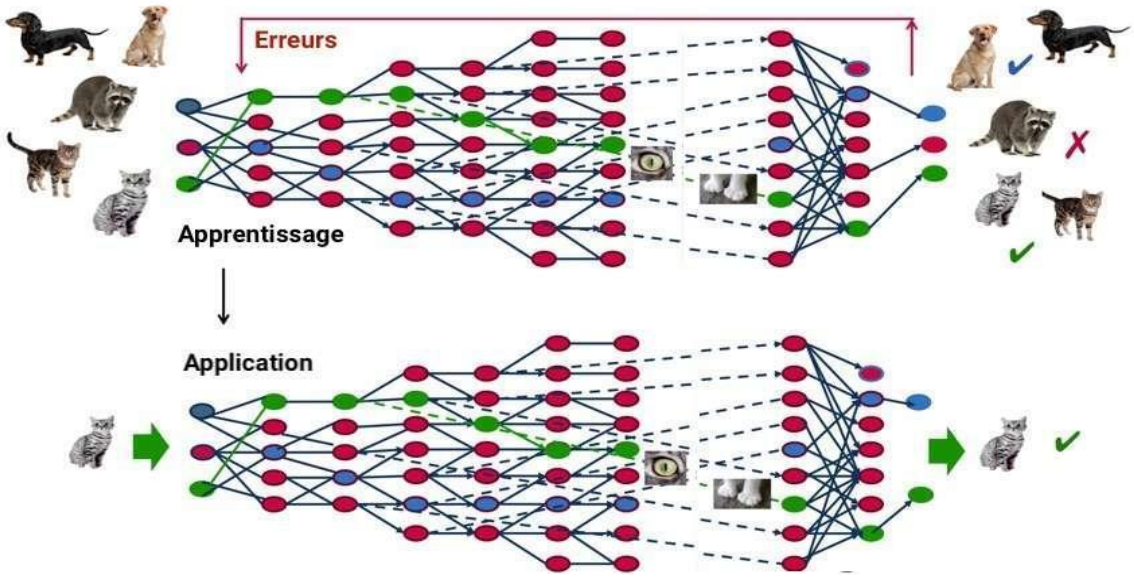


Figure I.10 : Schéma de fonctionnement de Deep Learning .

À travers un processus d‘auto-apprentissage, le Deep Learning est capable d‘identifier un chat sur une photo. À chaque couche du réseau neuronal correspond un aspect particulier de l‘image.

À chaque étape, les « mauvaises » réponses sont éliminées et renvoyées vers les niveaux en amont pour ajuster le modèle mathématique. Au fur et à mesure, le programme réorganise les informations en blocs plus complexes. Lorsque ce modèle est par la suite appliqué à d'autres cas, il est normalement capable de reconnaître un chat sans que personne

ne lui ait jamais indiqué qu'il n'ai jamais appris le concept de chat. Les données de départ sont essentielles : plus le système accumule d'expériences différentes, plus il sera performant. [12]

I.4.2. Applications de l’apprentissage profond :

Le Deep Learning est utilisé dans de nombreux domaines :

- Reconnaissance d'image, [13]
- Traduction automatique, [14]
- Voiture autonome, [15]
- Diagnostic médical, [16]
- Recommandations personnalisées, [17]
- Modération automatique des réseaux sociaux, [18]
- Prédiction financière et trading automatisé, [19]
- Identification de pièces défectueuses, [20]
- Détection de malwares ou de fraudes, [21]
- Chatbots (agents conversationnels), [22]
- Exploration spatiale, [23]
- Robots intelligents. [24]

L'apprentissage profond peut, par exemple, aider à :

- Mieux reconnaître des objets hautement déformables ;
- Analyser les émotions révélées par un visage photographié ou filmé ;
- Analyser les mouvements et position des doigts d'une main, ce qui peut être utile pour traduire les langues signées ;
- Améliorer le positionnement automatique d'une caméra, etc. ;
- Poser, dans certains cas, un diagnostic médical (ex. : reconnaissance automatique d'un cancer en imagerie médicale), ou de prospective ou de prédiction (ex. : prédiction des propriétés d'un sol filmé par un robot). [25]
- Reproduire une oeuvre artistique à partir d'une photo à l‘ordinateur . [25]

Conclusion :

Dans ce chapitre, nous avons parlé de l'intelligence artificielle et de son histoire. Nous avons également traité de l'apprentissage et des types de machine et donné des exemples de la différence entre ces types. Nous avons également expliqué l'apprentissage en profondeur et les domaines d'utilisation.

Chapitre 02

Théorie des graphes et fréquente Sous-graphes

II.1. Introduction

ce chapitre définit en détail le concept d'extraction de graphes et de concentrer principalement sur l'extraction minière fréquente (FSM), il est organisé en deux parties:

La partie 01 est principalement consacrée à la présentation simplifiée des notions de base relatives aux graphes et à leur théorie.

La partie 02 est consacrée à la présentation de «l'état de la technique» des algorithmes et techniques d'extraction fréquente de sous-graphes (FSM). Une enquête sur les recherches en cours dans le domaine et des solutions pour résoudre les principaux problèmes de recherche sont également présentées.

II.2. Data mining

L'exploration de données est une étape particulière dans le processus de découverte des connaissances dans les bases de données (KDD), qui permet d'extraire des connaissances statistiquement significatives et utiles d'un volume considérable d'ensembles de données brutes. L'extraction de connaissances aussi importantes peut être cruciale, parfois essentielle. phase suivante de l'analyse: la modélisation [26] Le processus de KDD est décrit à la figure

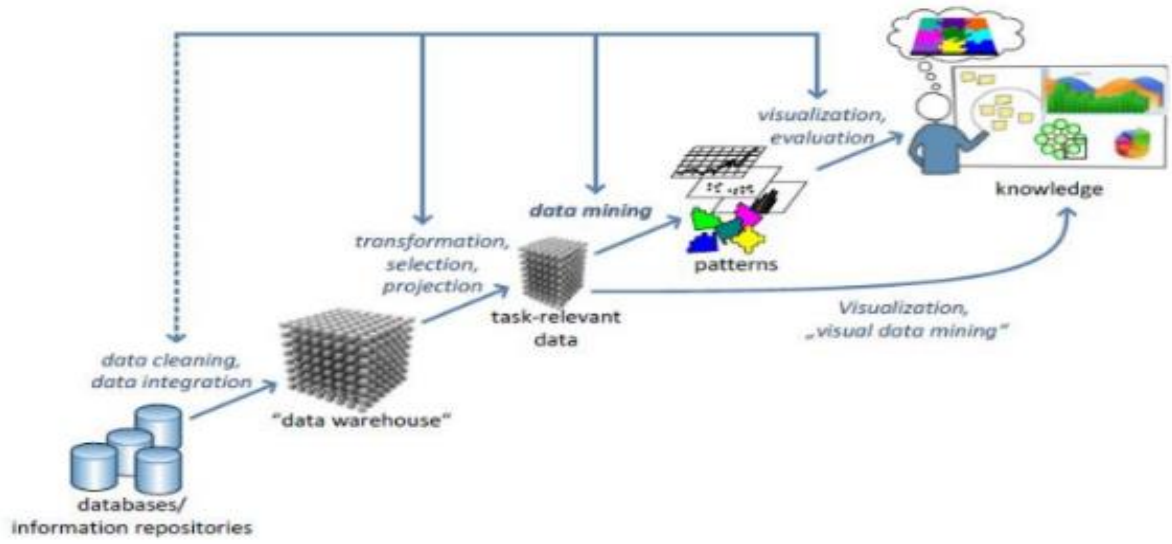


Figure II-1: Les étapes du processus de KDD

Les étapes supplémentaires du processus KDD incluent la sélection et la projection des données, ainsi que les étapes de visualisation et d'évaluation.

Au cours de la dernière décennie, le domaine de l'exploration de données est devenu un nouveau domaine de recherche, explorant des questions de recherche intéressantes et développant des applications difficiles de la vie réelle. Les formats de données objectives au début de l'exploration de données étaient limités aux tables et transactions relationnelles, chaque instance étant représentée par une ligne dans une table ou une transaction représentée par un ensemble.[27]

Cependant, au cours des dernières années, les études ont commencé à étendre les classes de données considérées aux données semi-structurées telles que les textes HTML et XML, les séquences symboliques et les relations représentées par des logiques avancées.

L'exploration de modèles fréquents, par exemple, est un thème central de la recherche sur l'exploration de données depuis plus de dix ans. Une littérature abondante a été consacrée à cette recherche et d'énormes progrès ont été réalisés, allant d'algorithmes efficaces et évolutifs pour l'extraction fréquente d'ensembles d'articles dans les bases de données de transactions, les règles d'association minière, telles que l'extraction de modèles séquentiels. Cependant, l'apparition et la variété des données semi-structurées complexes et la nécessité de découvrir des modèles structurels dans de grands ensembles de données, qui vont au-delà des ensembles, des séquences et des arbres, en vue d'une structure compliquée, rendent les ensembles d'éléments fréquents et les approches d'extraction de séquences fréquentes. inefficace et inadapté à ces exigences, l'émergence de graphes et l'exploitation minière structurelle fréquente comme solution à ces préoccupations.

Certes, les graphes en tant que structure de données peuvent répondre aux exigences de la modélisation de schémas de sous-structure et de relations complexes entre données, et ils représentent bien les objets complexes. De ce point de vue, on s'intéresse beaucoup à l'exploitation de données de graphes (souvent appelée exploration de données basée sur les graphes ou prochainement exploration de graphes). [27]

II.3. Extraction de graphes

En règle générale, l'exploration de graphes est le processus de découverte, d'extraction et d'analyse de modèles non triviales dans des données en forme de graphe. L'exploration de données basée sur les graphes ou l'exploration de graphes a une relation étroite avec l'exploration de données multi-relationnelle. Cependant, le principal objectif de l'extraction de graphes est de fournir de nouveaux principes et des algorithmes efficaces pour exploiter les sous-structures topologiques intégrées aux données de graphes, tandis que L'exploration de données multi-relationnelle a pour objectif principal de fournir des principes permettant d'exploiter et / ou d'apprendre les schémas relationnels, représentés par les langages logiques expressifs, le premier étant davantage axé sur la géométrie et le second davantage sur la logique et les relations [28] .

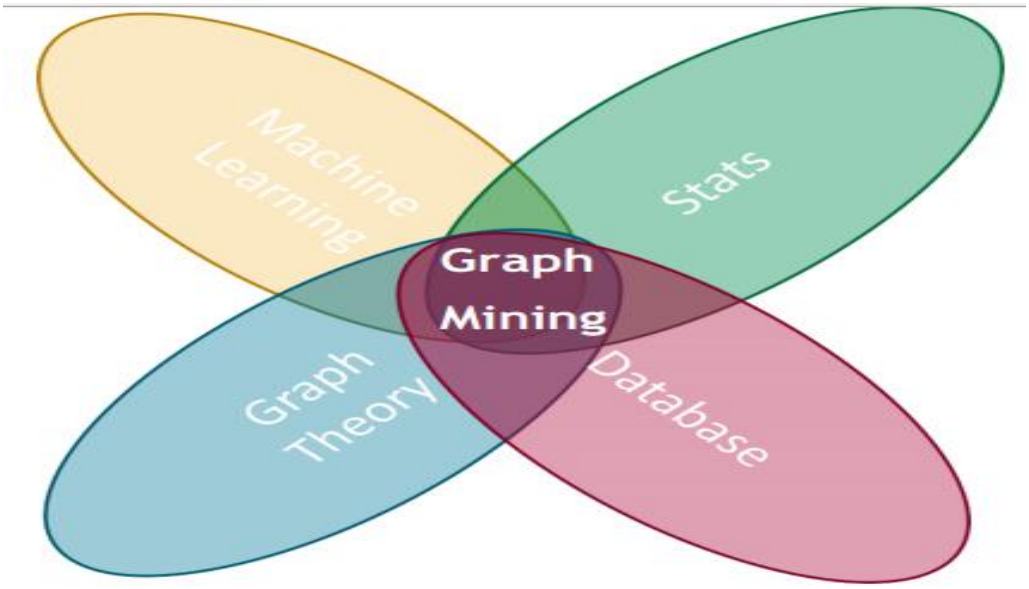


Figure II-2: Concept associé à l'extraction de graphes

Un certain nombre de sous-domaines de recherche populaires de l'extraction de graphes, Graphe des sous-domaines de la recherche minière...

Extraction fréquente de sous-graphes [29][31]

Extraction de modèles de graphes corrélés [32]

Extraction optimale de modèles de graphes [33]

Extraction de motifs de graphes approximatifs [34]

Récapitulatif de motif graphes [35]

Classification de graphes [36]

Regroupement de graphes [37] [38]

Indexation de graphes [39]

Recherche de graphes [40] [41]

Graphe des noyaux [42] [43]

Exploitation de liens [44][46]

Exploitation de structures Web [47]][48]

Extraction en flux de travail [49]

Exploitation de réseaux biologiques [50]

II.4. Applications de graphes:

En raison de sa capacité à modéliser des structures compliquées, la représentation de graphes des données est bien adoptée et prévalente dans divers domaines, notamment les interconnexions filaires / sans fil (réseaux), les objets 2D / 3D (reconnaissance de la vision et des modèles), les composés chimiques et les réseaux biologiques (chimie). / bio-informatique), circuits (conception assistée par ordinateur), données schématisées (XML), données RDF (Web sémantique), traces de programme (génie logiciel), médias sociaux, etc. [51]

En chimie, par exemple données chimiques, données chimiques (molécules, composés) est souvent représentée sous forme de graphes dans lesquels les nœuds correspondent à des atomes, et les liens correspondent à des liaisons entre les atomes.

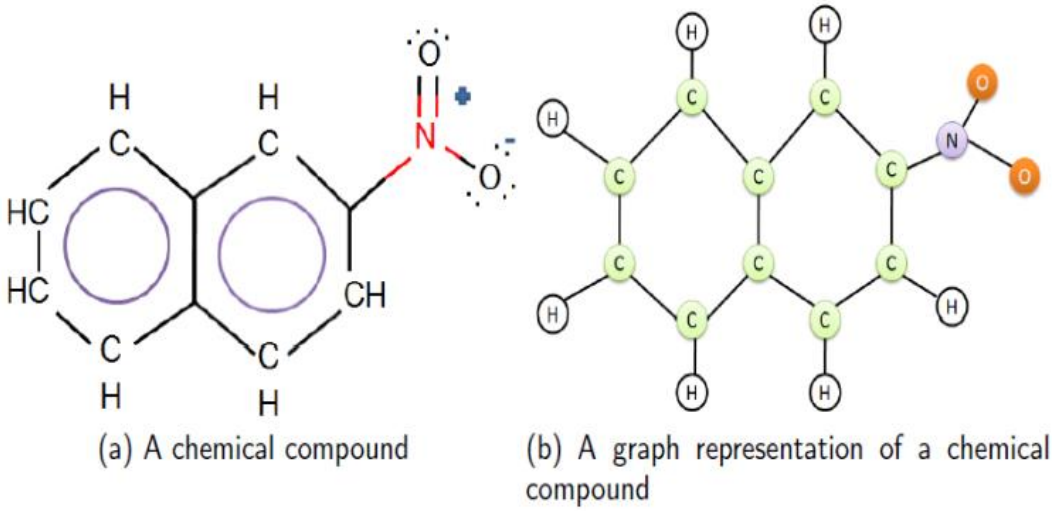


Figure II-3: Représentation de graphes d'un composé chimique

Bénéficier de systèmes de recherche et d'exploitation fréquents dans les composés chimiques, les chercheurs peuvent effectuer des dépistages, des conceptions et des découvertes de connaissances dans des ensembles de données composites et moléculaires à grande échelle [26]

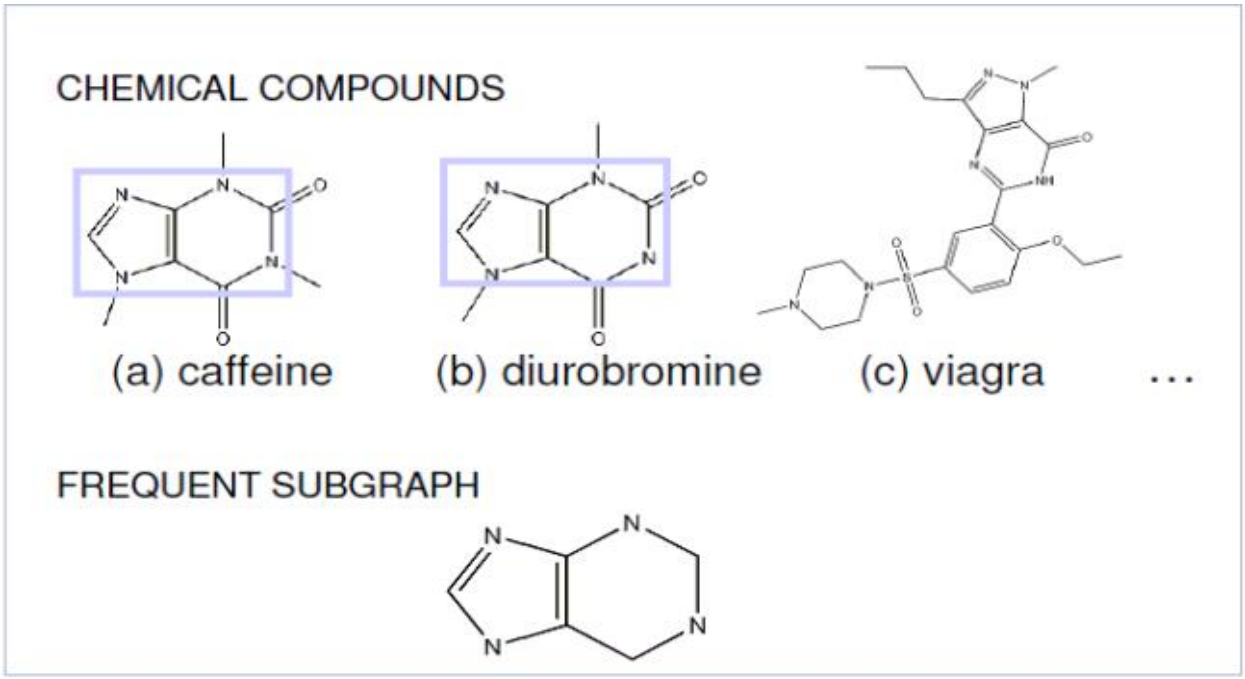


Figure II-4: Sous-graphe fréquent extrait de divers composés chimiques

En génie logiciel, un programme peut être modélisé par un graphe de flux de contrôle dans lequel les blocs de base sont représentés par des nœuds et les bords entre les nœuds indiquent le flux possible du programme. L'analyse des graphes de flux de contrôle peut éclairer

les comportements statiques et dynamiques des programmes et aider à détecter les logiciels malveillants et les logiciels malveillants. les virus.

De plus, les techniques avancées d'extraction de modèles fréquents peuvent nous aider à comprendre différentes relations et fonctions avancées. Par exemple, dans un réseau d'interaction protéine-protéine (PPI), un schéma fréquent pourrait révéler des fonctions inconnues d'une protéine. De même, dans un réseau social, un motif fréquent pourrait montrer une clique d'amis.

La figure II-5 montre un réseau d'interaction protéine-protéine, représenté par un graphe où les sommets désignent différentes protéines et dont les bords désignent les interactions physiques entre elles. La figure II-6 présente également un exemple de représentation graphe de réseau social.

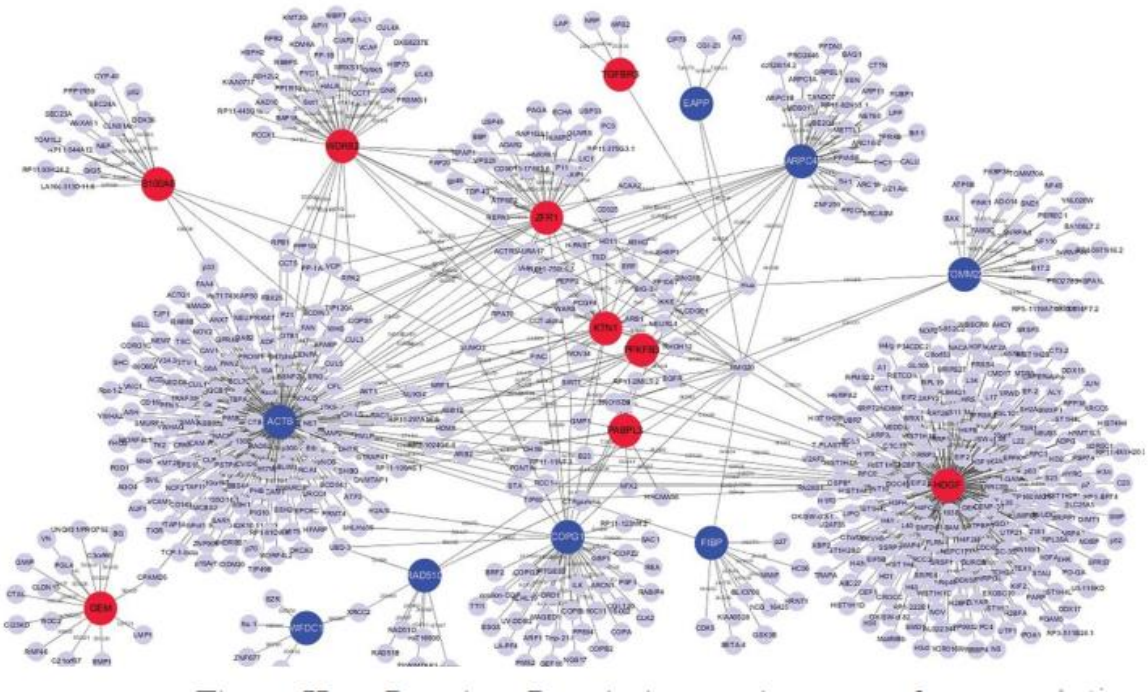


Figure II-5: Réseau d'interaction protéine-protéine

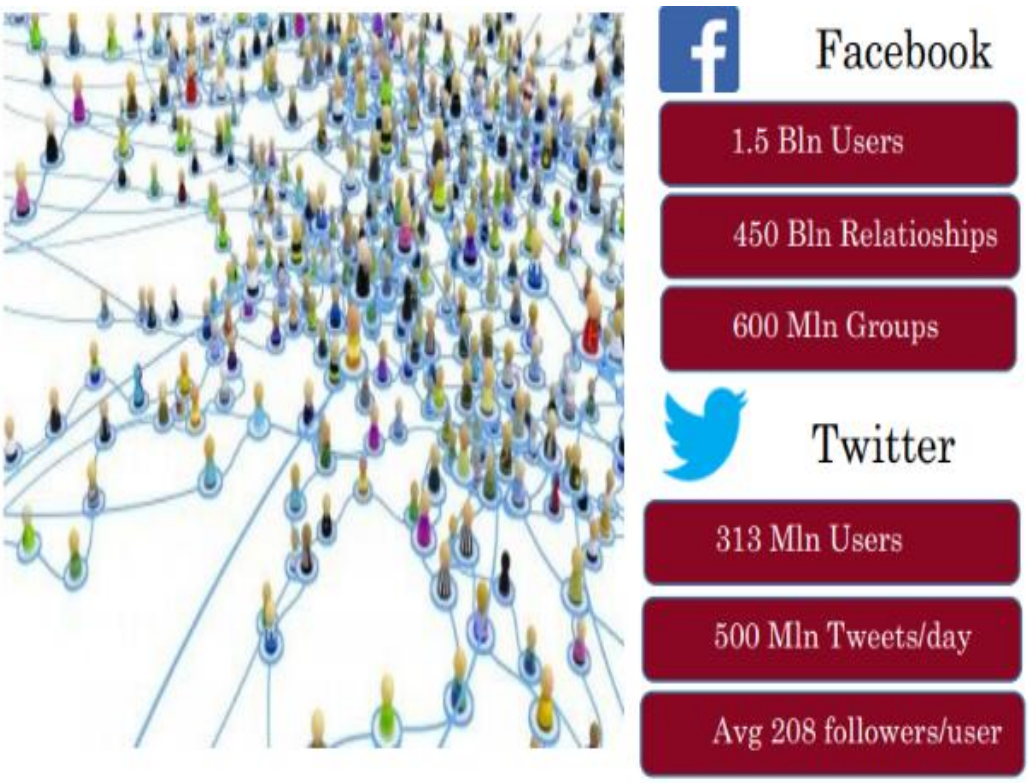


Figure II-6: Graphes de médias sociaux: il existe des groupes complexes, des liens, des préférences et des attributs.

Les réseaux de médias sociaux (graphes), ainsi que d'autres exemples de graphes de recommandation et de connaissances, présentent la particularité d'être complexes. Par conséquent, l'application de techniques d'exploration minière avancées pour trouver des modèles fréquents spécifiant une contrainte de fréquence minimale donnée peut aider à révéler les règles dérive de leur évolution.

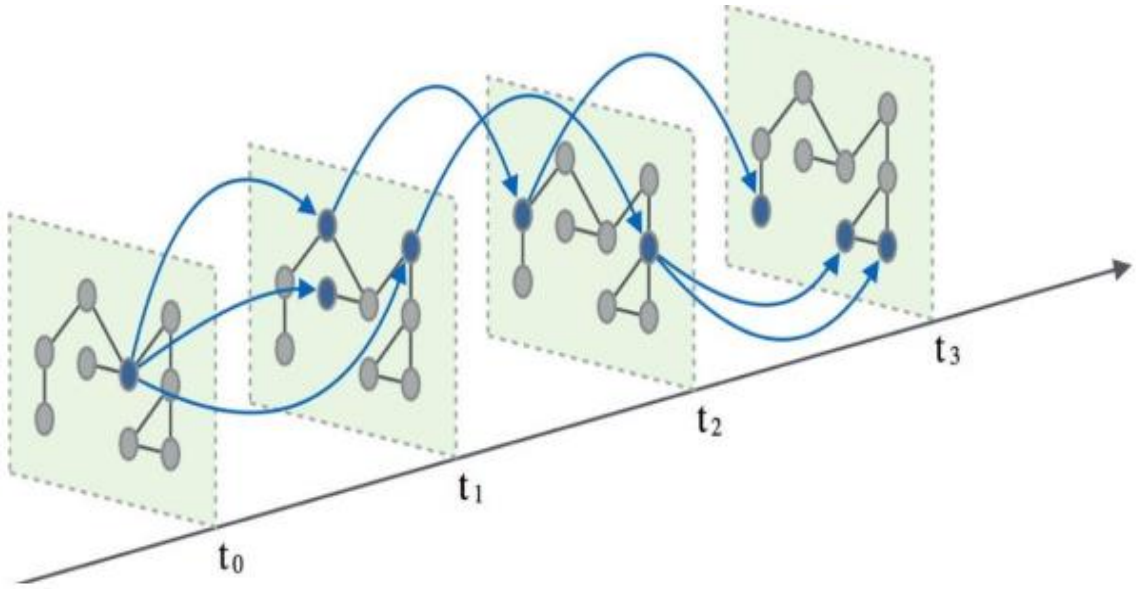


Figure II-7: Règles d'évolution des graphes d'exploitation

II.5. En quoi consiste l'exploration de graphes :

- Algorithmes de graphes de base (chemins les plus courts, BFS, DFS, isomorphismes, traversées, promenades aléatoires…)
- Stockage et indexation
- Représentations intelligentes pour la compacité
- Modélisation des problèmes sous forme de graphes
- Mesures de distance et mesures de similarité
- Algorithmes exacts, approximatifs et heuristiques
- Structures en évolution
- Interactivité et mises à jour en ligne (la plupart des problèmes ne peuvent pas être résolus de manière polynomiale)

II.6. Extraction de sous-graphes fréquents (FSM) :

Parmi les divers types de motifs de graphe, les sous-graphes fréquents sont très basiques et peuvent être découverts dans un ensemble de graphes (base de données de graphes) ou un seul grand graphe. Ils sont utiles pour caractériser les ensembles de graphes, distinguer différents groupes de graphes, classer et regrouper graphes et construction d'index de graphes, l'extraction fréquente de sous-graphes englobe toutes les techniques et méthodologies utilisées pour découvrir de tels modèles. [52]

Avant de parler dans les détails de la variante de l'extraction fréquente de sous-graphes algorithmes, passons d'abord en revue certains concepts et terminologies de base de la théorie des graphes.

II.7. Théorie des graphes

La théorie des graphes est une branche des mathématiques discrètes qui traite de la manière dont les objets sont connectés. Ainsi, la connectivité dans un système est une qualité fondamentale de la théorie des graphes. Le concept principal en théorie des graphes est celui d'un graphe. Pour un mathématicien, un graphe est l'application d'un ensemble sur lui-même, c'est-à-dire une collection d'éléments de cet ensemble et les relations binaires entre ces éléments. Les graphes sont des objets dimensionnels, mais ils peuvent être intégrés ou réalisés dans des espaces de dimensions supérieures .[53]

II.8. Définitions préliminaires :

Les paragraphes suivants présentent un certain nombre de définitions largement utilisées, utilisées plus loin dans ce chapitre

II.8.1. Types de graphes :

D'une manière générale**,** un graphe $G\ (V, E)$ est défini comme étant un ensemble V de sommets (nœuds) qui sont interconnectés par un ensemble E d'arêtes (liens). Le nombre d'éléments N dans V s'appelle l'ordre de G et on note $|\ V\ | = N$ et le nombre d'éléments M dans E est la taille de G et on note $|\ E\ | = M$

Simple et multigraphes: dans un graphe G, lorsque deux sommets quelconques de V sont reliés par plusieurs arêtes, le graphe est appelé multigraphe. Un graphe sans boucles et avec au

plus une arête entre deux sommets est appelé un graphe simple. Les graphes utilisés dans FSM sont supposés être des graphes étiquetés simples.

Figure II-8: (a) graphe simple, (b) nom graphe simple ayant arête multiple (gauche) nom graphe simple ayant des boucles (droite).

Dans un graphe *G (V, E)* pour une arête *e = {u, v},* on dit:

- *e* connecte *u* et *v*

- *u* et *v* sont les points finaux de e.

- *u* et *e* sont des incidents (*v* et *e* sont des incidents).

- *u* et *v* sont adjacents ou voisins

Le degré deg *(v)* d'un sommet *v* est le nombre d'arêtes qui lui sont incidentes. Un sommet de degré 0 est appelé isolé [54].

Graphes réguliers: un graphe *G* dans lequel tous les sommets sont de degré égal est appelé un graphe régulier, un graphe régulier de degré k est également appelé k – régulier

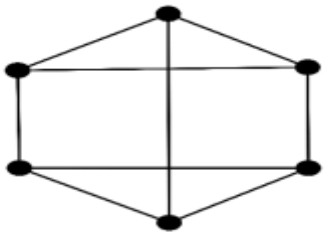
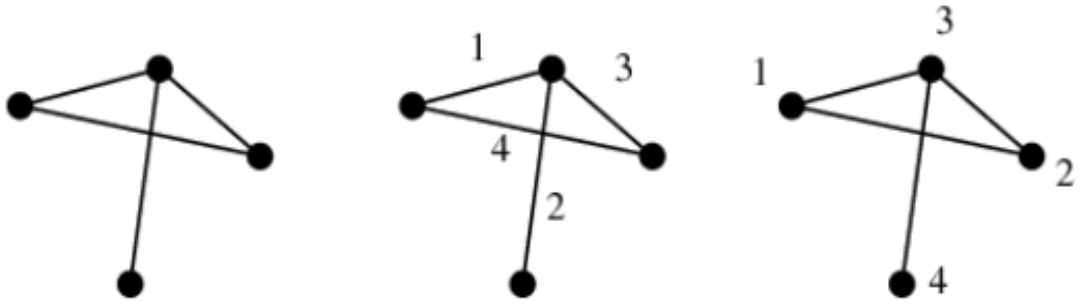


Figure II-9: Graphe 3-régulier k = 3.

Graphes étiquetés: un graphe étiqueté peut être représenté par $G\ (V, E, L_V, L_E, \varphi)$, où V est un ensemble de sommets, $E \subseteq V \times V$ est un ensemble d'arêtes; L_V et L_E sont des ensembles d'étiquettes de sommet et d'arête. respectivement; et φ est une fonction d'étiquette qui définit les correspondances $V \rightarrow L_V$ et $E \rightarrow L_E$. Lorsque les étiquettes de bord sont membres d'un ensemble ordonné (par exemple, les nombres réels), un graphe étiqueté peut être appelé graphe pondéré. [55]



graphe non étiqueté ***graphe marqué par des arêtes*** ***graphe étiqueté par sommet***

Figure II-10: Graphes sans étiquette et étiquetés

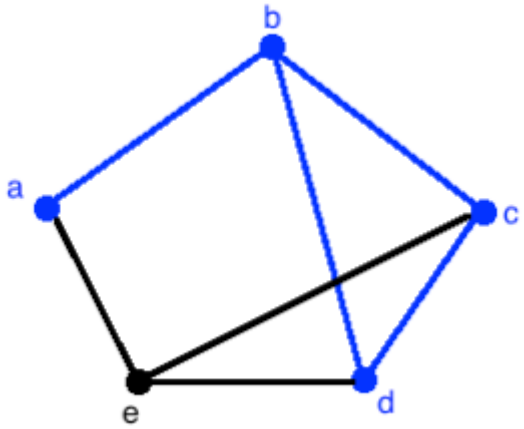
Graphes orientés et non orientés: un graphe G n'est pas orienté si $\forall e \in E$, e est une paire de sommets non ordonnée et est dirigé sinon

Figure II-11: Graphe non dirigé (à gauche), Graphe dirigé (à droite)

sentiers, chemins et cycles: la marche d'un graphe G est une alternance séquence de sommets et d'arêtes $v_0, v_1, v_1, \ldots\ldots . , v_{i-1}, e_i, v_i$ commençant et finissant par des sommets, dans lesquels chaque arête est incidente avec deux sommets qui le précèdent et le suivent immédiatement. Cette promenade rejoint les sommets v_0 et vi et peut également être désignée par $v_1, v_2, \ldots\ldots . , v_i$

(les arêtes étant évidentes par le contexte). La longueur l d'une promenade est le nombre d'occurrences d'arêtes dans celle-ci; v_0 est appelé le sommet initial de la promenade, tandis que v_i est appelé le sommet terminal de la promenade.

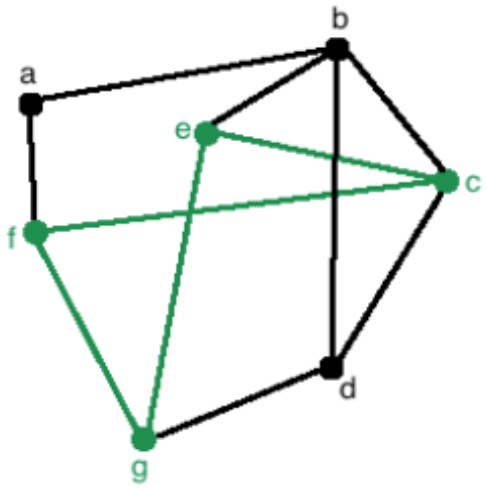
Par exemple, le graphe ci-dessous décrit une éventuelle promenade (en bleu). La marche est notée *abcdb*. Notez que les promenades peuvent avoir des bords répétés. Par exemple, si nous avions la marche *abcdcbce*, alors ce serait parfaitement bien même si certains bords sont répétés



Dans le graphe ci-dessus, la longueur de la promenade est abcdb 4 car elle traverse 4 bords.

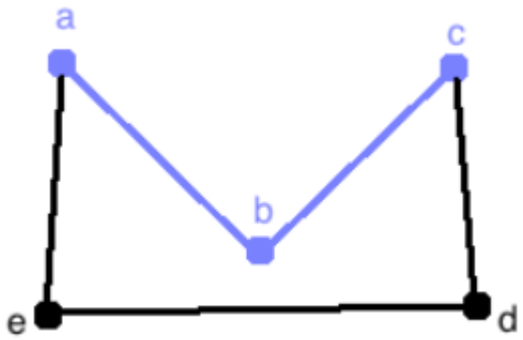
Une promenade fermée est une promenade v_i-v_i, c'est-à-dire une promenade qui commence et se termine au même sommet v_i. Sinon, une promenade est dite ouverte.

Par exemple, le graphe suivant montre une promenade fermée en vert:



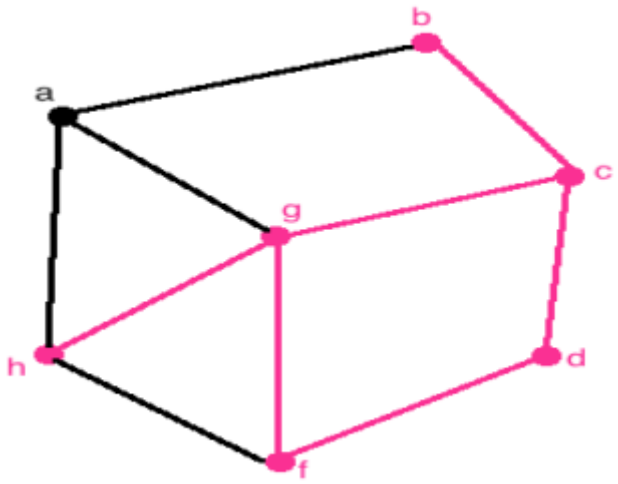
Notez que la promenade peut être définie par *cegfc* et que les sommets de début et de fin de la promenade sont c. C'est pourquoi cette promenade est fermée.

Une promenade est un sentier si tous les bords sont distincts. Jusqu'à présent, les deux précédents les exemples peuvent être considérés comme des traînées car il n'y a pas de bords répétés. Voici un autre exemple de sentier:



Notez que la promenade peut être définie comme abc. Il n'y a pas de bords répétés, donc cette promenade est aussi un sentier.

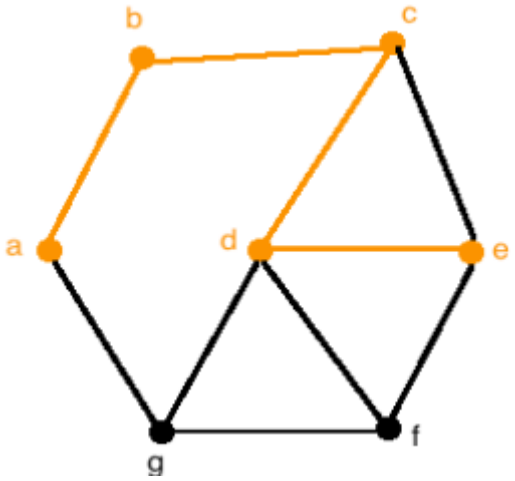
Regardons maintenant le graphe suivant:



Notez que cette promenade doit répéter au moins un bord.

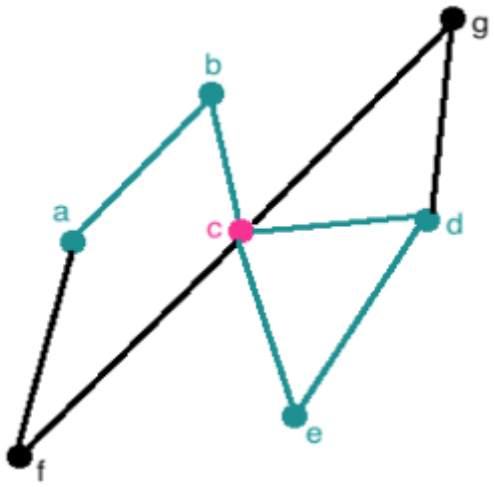
Une promenade est un chemin si tous les sommets (et donc nécessairement toutes les arêtes) sont distincts, Sinon dit, un chemin est défini comme un chemin ouvert sans sommets répétés. Notez que tous les chemins doivent donc être des chemins ouverts, puisqu'un chemin ne peut

à la fois commencer et se terminer au même sommet. Par exemple, la promenade de couleur orange suivante est un chemin:



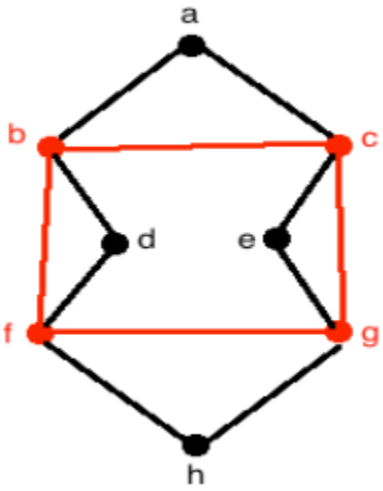
parce que la promenade ***abcde*** ne répète aucun bord.

Regardons maintenant le graphe suivant avec la marche des sarcelles. Cette promenade n'est PAS un chemin puisqu'elle répète un sommet, à savoir le sommet rose c:



De plus, un cycle dans *G* est défini comme un tracé fermé dans lequel aucun autre sommet n'est répété en dehors du sommet de début / fin. Et de plus, un graphe acyclique *G* est défini comme un graphe sans cycle.

Vous trouverez ci-dessous un exemple de cycle. Remarquez comment aucune arête ne se répète dans la promenade *bcgfb*, ce qui en fait un sentier, et que les sommets de début et de fin *b* sont identiques, ce qui les rend fermés.



Graphes connectés et complets

G est connecté, s'il contient un chemin pour chaque paire de sommets et est déconnecté sinon. G est complet (clique) si chaque paire de sommets est reliée par une arête. (44)

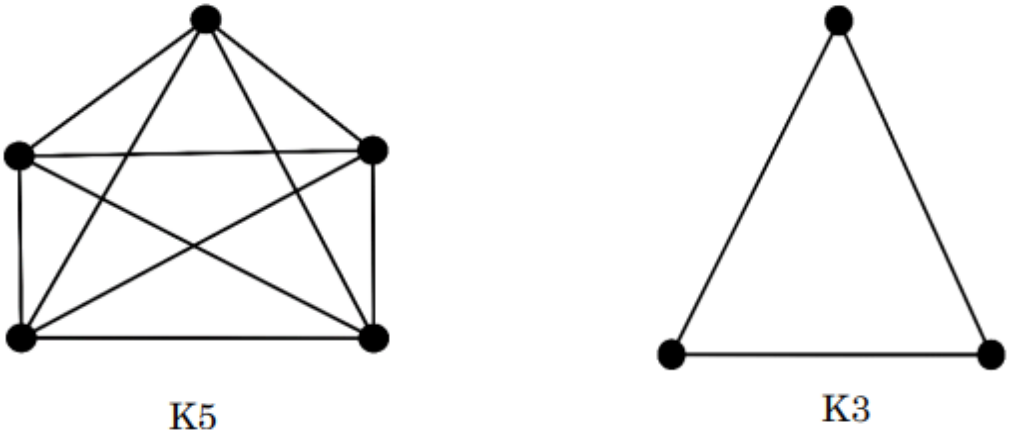


Figure II-12: Deux graphes complets et connectés avec 5 et 3 sommets respectivement, lorsqu'ils sont considérés ensemble, ils forment un graphe déconnecté.

II.8.2. Sous-graphes :

Étant donné deux graphes $G_1 (V_1, L_1, L_{V1}, L_{E1}, \varphi 1)$ et $G_2 (V_2, E_2, L_{V2}, L_{E2}, \varphi 2)$, G_1 est un sous-graphe général appelé de G_2, si G_1 est satisfait: (I) $V_1 V_2$, *et* $\forall v V_1$, $\varphi_1 (v) = \varphi_2 (v)$, *(ii)* $E_1 E_2$ *et* $(u, v) E1$, $\varphi_1 (u, v) = \varphi_2 (u, v)$. La figure II-13 (a) est un exemple du sous-graphe général dans lequel un sommet v_5 et des arêtes e_4; e_6; e_7; e_8; G_1 est un sous-graphe induit de G_2, si G_1 satisfait davantage: $Gu, v \in V_1, (u, v) E_1 (u, v) E_2$, simplement, un sous-graphe induit G_1 d'un graphe G_2 a un sous-

ensemble des sommets de *G2* et les mêmes arêtes entre paires de sommets que dans *G2*. en plus des conditions ci-dessus. *G2* est aussi un supergraphe de *G1* La figure II-13 (c) est un exemple du sous-graphe induit dans lequel un sommet v_5 est omis. Dans ce cas, seuls les bords e_8 *et* e_9 sont également omis, et e_4; e_6; e_7 sont conservés car ils existent entre v_1; v_3 et v_4 dans l'original G. [55]

La troisième classe importante et générique de la sous-structure est un sous-graphe connecté où $V_1 \subseteq V_2$, $E_1 \subseteq E$ et tous les sommets de V_1 sont mutuellement accessibles par certaines arêtes de E1. La figure II-13 (d) est un exemple où v_6 est omis de (c).

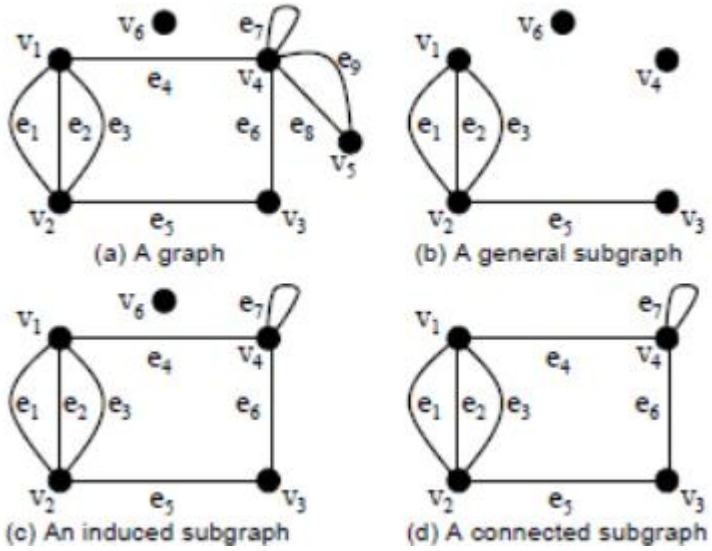


Figure II-13: (a) un graphes G, (b), (c), (d) représente respectivement un sous-graphe général, induit et connecté

II.8.3. Isomorphisme de graphe :

Un graphe G_1 $(V_1, L_1, L_{V1}, L_{E1}, \varphi_1)$ est isomorphe à un autre graphe G_2 $(V_2, L_2, L_{V2}, L_{E2}, \varphi_1)$ et est noté: G_1 - G_2 si et seulement si une bijection f: $V_1 \rightarrow V_2$ existe tel que: (i) $\forall u \in V_1$, $\varphi_1 (u) = \varphi_2 (f (u))$, *(ii)* $(u, v) E_1 (f (u), f (v)) \in E_2$, *(iii)* $(u, v) E_1$, $\varphi_1 (u, v) = \varphi_2 (f (u), f (v))$. La bijection f est un isomorphisme entre G_1 et G_2. Un graphe G_1 est un sous-graphe isomorphe à un graphe G_2, si et seulement s'il existe un sous-graphe G_2 tel que g1 soit isomorphe à g. Dans ce cas, on appelle g une incorporation de G_1 dans G_2. [55]

La figure II-15 et la figure II-16 présentent respectivement un isomorphisme de graphe et de sous-graphe:

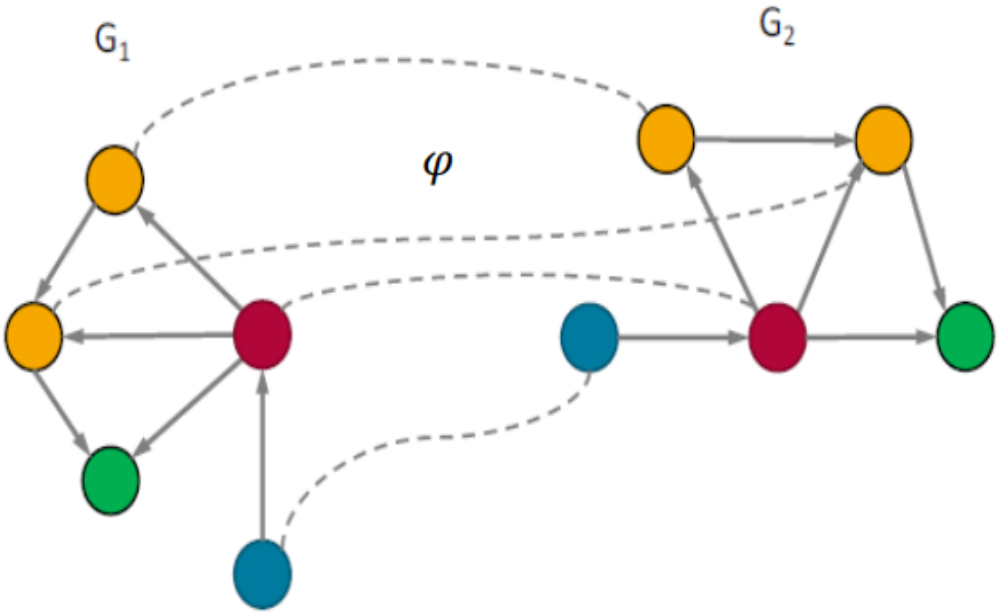


Figure II-14: Isomorphisme de graphe

Les deux graphes ci-dessus sont isomorphes, malgré leurs dessins différents

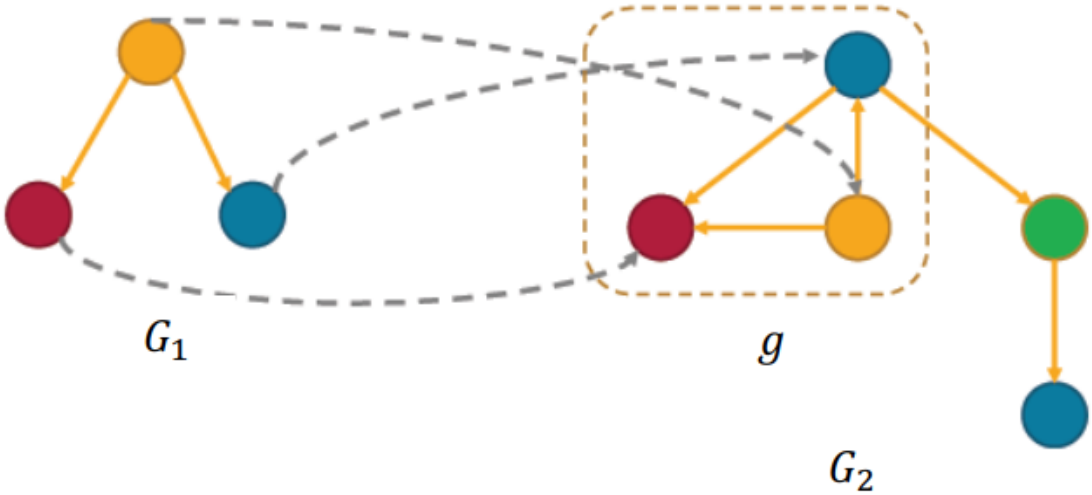


Figure II-15: Isomorphisme de sous-graphe.

Pensez à un isomorphisme de graphe comme à une structure préservant la bijection, et il découle de la définition que les graphes isomorphes sont identiques, mais dessinés différemment.

Généralement, le problème de la reconnaissance des graphes isomorphes est l'un des grands problèmes non résolus de la théorie des graphes. Construction de tous N! Les mappages possibles d'un graphe à un autre (où N est le nombre de sommets), bien qu'évidemment peu pratiques, restent le seul moyen sûr de vérifier l'isomorphisme des graphes.

Un invariant d'un graphe *G* est une quantité associée à *G* qui a la même valeur pour tout graphe isomorphe à *G*. Par conséquent, les invariants de graphe sont des quantités indépendantes de l'étiquetage des sommets d'un graphe. Par conséquent, le nombre de sommets et le nombre d'arêtes sont des invariants. Un ensemble complet d'invariants détermine un graphe allant jusqu'à l'isomorphisme.

II.8.4. Automorphisme :

Un mappage isomorphe des sommets d'un graphe *G* sur eux-mêmes (ce qui préserve également la relation d'adjacence) est appelé un automorphisme d'un graphe *G*. Evidemment, chaque graphe possède un automorphisme trivial appelé automorphisme d'identité. Pour certains graphes, c'est le seul automorphisme; ce sont les graphes d'identité. L'ensemble de tous les automorphismes d'un graphe G forme un groupe appelé groupe d'automorphisme de G. [56]

II.8.5. Structure:

Dans une base de données *G*, un réseau est une forme structurelle utilisée pour modéliser l'espace de recherche afin de rechercher des sous-graphes fréquents, chaque sommet représentant un sous-graphe connecté du graphe en *G*. Le sommet le plus bas représente le sous-graphe vide et les sommets du niveau le plus élevé. les graphes de *G*. Un sommet *p* est un parent du sommet q du réseau, si q est un sous-graphe de p, et que q est différent de *p* de exactement un bord. Le sommet q est un enfant de p. Tous les sous-graphes de chaque graphe $Gi \in G$ apparaissant dans la base de données sont présents dans le réseau et chaque sous-graphe n'y apparaît qu'une fois [55]

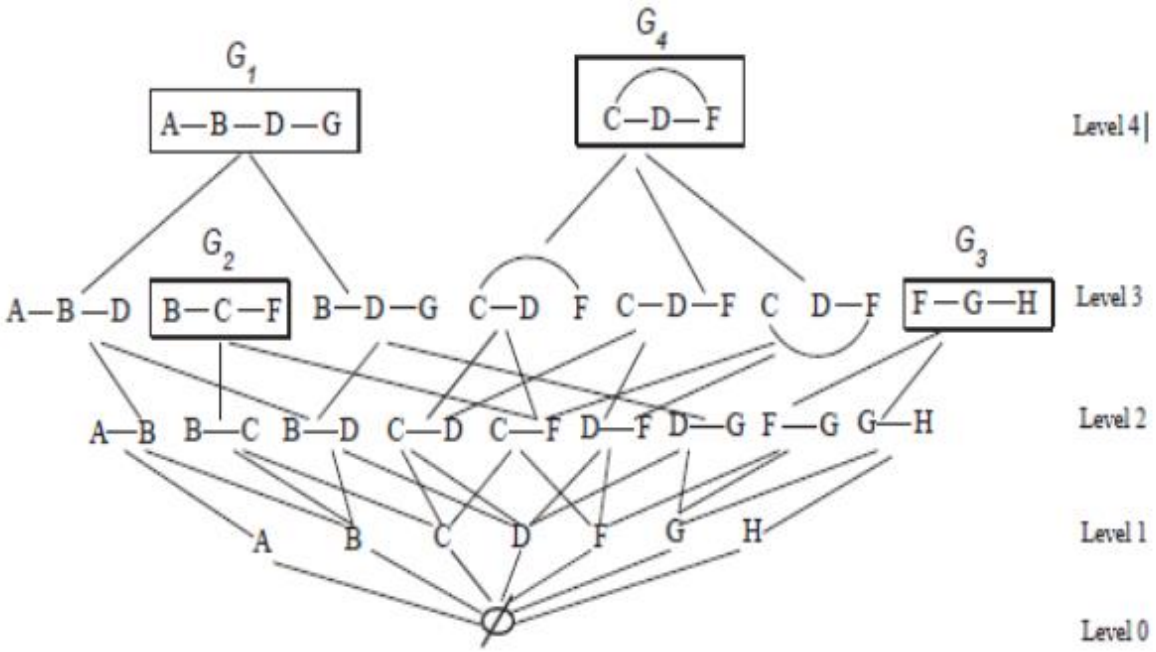


Figure II-16: Réseau (G)

Exemple: étant donné un ensemble de données graphes $G = \{G_1, G_2, G_3, G_4\}$, le réseau correspondant (G), est donné à la figure II-16. Sur la figure, le sommet le plus bas représente le sous-graphe vide et les sommets du niveau le plus élevé correspondent à G_1, G_2, G_3 et G_4. Les parents du sous-graphe B - D sont les sous-graphes A - B - D (joignant le bord A - B) et B - D - G (joignant le bord D - G). De même, les sous-graphes B - C et C - F sont les enfants du sous-graphe B - C - F.

II.8.6. Densité :

La densité d'un graphe $G = (V; E)$ est calculée par

$$\textbf{densité (G)} = \quad 2 \cdot \frac{|E|}{(|V| \,.\, (|V| - 1))}.$$

La densité du graphe mesure le rapport entre le nombre d'arêtes et le nombre maximal d'arêtes dans un graphe complet. Un graphe est dit dense si le rapport est proche de 1 et est considéré comme rare si le rapport est proche de 0. [26]

II.8.7. Arbres :

Arbre libre: un arbre libre est défini comme un graphe non dirigé, connecté et acyclique. Arbre non ordonné étiqueté: Un arbre non ordonné étiqueté (un arbre non ordonné, en abrégé) est un graphe acyclique dirigé désigné par $T(V, \varphi, E, v_r)$, où V est un ensemble de sommets de T; φ est une fonction d'étiquetage telle que $v_i \in V, \varphi(v_i) \rightarrow v_i$; $E \subseteq V \times V$ est un ensemble d'arêtes de T; et vr est un sommet distingué appelé racine de T. Pour vi ∈ V, il existe un chemin unique $(v_r, v_1, v_2, \ldots\ldots, v_i)$ de la racine v_r à v_i. Si un sommet v_i se trouve sur le chemin allant de la racine au sommet v_j, alors vi est un ancêtre de v_j et v_j est un descendant de v_i. Pour chaque bord (v_i, v_j) E, v_i est le parent de v_j et v_j est un enfant de v_i. Les sommets qui partagent le même parent sont des frères et soeurs. La taille de T est définie comme étant le nombre de sommets dans T. Un sommet sans enfant est un sommet de feuille; sinon c'est un sommet intermédiaire. Le chemin le plus à droite de T est le chemin allant du sommet de la racine à la feuille la plus à droite. La profondeur (niveau) d'un sommet est la longueur du chemin allant de la racine à ce sommet.
Arbre ordonné étiqueté: Un arbre ordonné étiqueté (un arbre ordonné, en abrégé) est un arbre étiqueté non ordonné mais avec un ordre de gauche à droite imposé aux enfants de chaque sommet.

Les autres formes de sous-arbres incluent les suivantes: sous-arbre de bas en haut, sous-arbre induit et sous-arbre intégré.

II.9. Aperçu de FSM :

L'extraction fréquente de sous-graphes (FSM) est l'essence même de l'extraction de graphes. Le domaine d'étude se concentre sur l'identification de sous-graphes fréquents dans des ensembles de données de graphes. Les objectifs de la recherche sont orientés vers: (i) des mécanismes efficaces pour générer des sous-graphes candidats (sans générer de doublons) et (ii) comment mieux traiter les sous-graphes candidats générés de manière à identifier les sous-graphes fréquents souhaités de manière efficace du point de vue du traitement et des procédures efficace.

les sous-graphes fréquents sont considérés comme tels, si leur nombre dépasse un seuil de support minimal. La figure II-8 (a) présente un aperçu du domaine de la FSM en termes de nombre d'algorithmes significatifs de FSM proposés de 1994 à aujourd'hui. La figure montre des périodes d'activité au début des années 90 (coïncidant avec l'introduction du concept de data

mining), suivies d'une autre période d'activité de 2002 à 2007. Aucun «nouvel» algorithme n'a été introduit ces dernières années. indiquant que le domaine est à maturité, même si de nombreux travaux ont été consacrés aux variantes des algorithmes existants .

Outre les activités de recherche associées aux États fédérés de Micronésie, l'importance de ces systèmes se reflète également dans ses nombreux domaines d'application. La figure II-8 (b) présente un aperçu du domaine d'application de FSM en termes de nombre d'algorithmes FSM rapportés dans la littérature et du domaine d'application spécifique vers lequel ils ont été dirigés. La figure montre que trois domaines d'application (chimie, Web et biologie) ont dominé l'utilisation des algorithmes FSM.

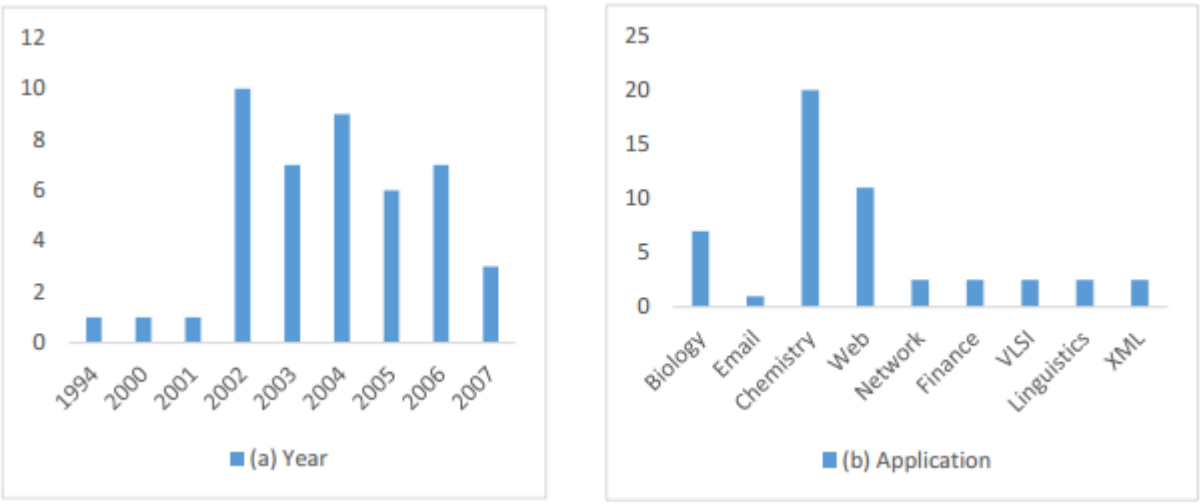


Figure II-17: Répartition des algorithmes FSM les plus significatifs par rapport à l'année d'introduction et au domaine d'application.

L'idée directe derrière FSM est de "développer" les sous-graphes candidats, soit en largeur (BFS), soit en profondeur (DFS) (génération candidate), puis de déterminer si les sous-graphes candidats identifiés apparaissent assez fréquemment dans le jeu de données de graphe pour qu'ils soient considérés comme intéressants (comptage des supports). Le Les deux principaux problèmes de recherche dans les États fédérés de Micronésie sont donc comment et avec efficacité:

(i) Générer les sous-graphes fréquents candidats.

(ii) Déterminer la fréquence d'apparition des sous-graphes générés .[55]

Pour générer efficacement un sous-graphe de candidats, il faut éviter de générer des candidats en double ou superflus. Le comptage des occurrences nécessite la comparaison

répétée de sous-graphes candidats avec des sous-graphes dans les données d'entrée, processus appelé vérification de l'isomorphisme. Les FSM, à bien des égards, peuvent être considérés comme une extension de l'extraction d'éléments fréquents (FIM) popularisée dans le contexte de l'exploration de règles d'association (voir par exemple [57]). Par conséquent, bon nombre des solutions proposées pour traiter les principales questions de recherche affectant les FSM reposent sur des techniques similaires trouvées dans le domaine de la FIM. Par exemple, la propriété de fermeture vers le bas (DCP) associée aux jeux d'éléments a été largement adoptée en ce qui concerne la génération de sous-graphes candidats, nous la définirons plus tard dans ce chapitre.

II.10. Formalisme :

Il existe deux formulations de problèmes distincts pour FSM: (i) un FSM basé sur une transaction de graphe et (ii) un FSM basé sur un graphe. Dans les FSM basés sur des transactions de graphes, les données d'entrée comprennent un ensemble de graphes de taille moyenne appelés transactions. Notez que le terme «transaction» est emprunté au champ Association Rule Mining. [58]

Comme dans son nom, dans un FSM à graphe unique, les données d'entrée comprennent un très grand graphe

Un sous-graphe g est considéré comme fréquent si son nombre d'occurrences est supérieur à une valeur de seuil prédéfinie. Le nombre d'occurrences d'un sous-graphe est généralement appelé son support, et le seuil correspondant est donc appelé seuil de support. La prise en charge de g peut être calculée à l'aide d'un comptage basé sur une transaction ou d'un comptage basé sur une occurrence. Le comptage basé sur les transactions ne s'applique qu'aux FSM basés sur des graphes, tandis que le comptage basé sur des événements peut être appliqué à un FSM basé sur des transactions ou à un seul FSM basé sur des graphes. Cependant, le comptage basé sur les occurrences est généralement utilisé avec un FSM basé sur un graphe.

Dans le comptage basé sur les transactions, le support est défini par le nombre de transactions de graphes dans lesquelles g se produit, un décompte par transaction, que g ait ou non été exécuté une ou plusieurs fois dans une transaction de graphes particulière. Ainsi, étant donné une base de données $G = \{G1, G2, G3.... , GT\}$ consistant en un ensemble de transactions graphes et un seuil de support σ a [0,1], puis l'ensemble des transactions graphes où se produit un sous-graphe g est défini par $\delta_G\ (g) = \{Gi \mid g \subseteq Gi\}$. Ainsi, le support de g est défini comme suit:

$Sup_G(g) = | \delta_G(g) | / T$

où $|\delta_G(g)|$ dénote la cardinalité de δ_G (g) et T le nombre de graphes (transactions) dans G. Par conséquent, g est fréquent si et seulement si $SupG(g) \geq \sigma$. Dans le comptage basé sur les occurrences, nous comptons simplement le nombre d'occurrences. de g dans le jeu d'entrées, (notez que nous ne sommes intéressés que par le FSM basé sur la transaction dans ce rapport.)

Le comptage basé sur les transactions offre l'avantage que la propriété bien connue de fermeture descendante peut être utilisée pour réduire de manière significative le temps système de calcul associé à la génération de candidats dans FSM. Dans le cas d'un comptage basé sur des occurrences, une mesure de fréquence alternative, qui conserve la propriété DC, doit être établie. ou des méthodes heuristiques adoptées pour maintenir le calcul aussi peu coûteux que possible .[55]

II.11. Détection d'isomorphisme de graphe :

Le noyau de FSM est la détection d'isomorphisme de (sous) graphe. On ne sait pas que l'isomorphisme de graphe puisse être résolu en temps polynomial ni NP-complet, tandis que l'isomorphisme de sous-graphe, dans lequel nous souhaitons établir si un sous-graphe est entièrement contenu dans un super graphe, est connu pour être NP-complet [59]. Lors de la restriction de la graphes en arbres, la détection d'isomorphisme de (sous) graphes devient la détection d'isomorphisme de (sous) arbres. La détection d'isomorphisme d'arbre peut être résolue dans un temps linéaire.

La détection de l'isomorphisme de sous-graphe est fondamentale pour FSM. Un nombre important de techniques «efficaces» ont été proposées, toutes visant à réduire, dans la mesure du possible, les coûts informatiques associés. Les techniques de détection d'isomorphisme de sous-graphe peuvent être grossièrement classées comme suit: correspondance exacte [60][61]ou correspondance tolérante aux erreurs [62][63]. La plupart des algorithmes FSM adoptent une correspondance exacte. Le tableau 2 présente une catégorisation des principaux algorithmes de détection d'isomorphisme de sous-graphe de correspondance exacte.

Tableau II-2: Catégorisation d'algorithmes de test d'isomorphisme de (sous-) graphes correspondant exactement.

Algorithmes	Techniques principales	Types correspondants
Ullman	Retour en arrière + regarder devant	Isomorphisme graphe et sous-graphe
SD	Matrice de distance + recul	Isomorphisme de graphe
Nauty	Théorie des groupes + étiquetage canonique	Isomorphisme de graphe
VF	Stratégie DFS + règles de faisabilité	Isomorphisme graphe et sous-graphe
VF2	VF's rationne + structures de données avancées	Isomorphisme graphe et sous-graphe

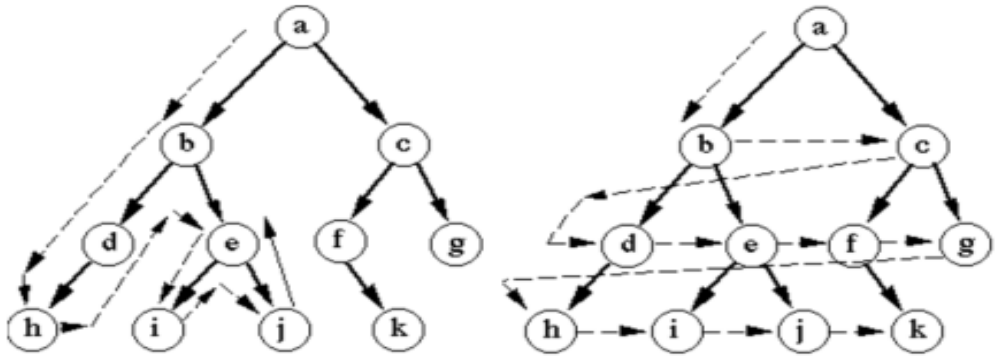
En référence au tableau 2, l'algorithme d'Ullmann utilise une procédure de retour en arrière avec une fonction d'anticipation pour réduire la taille de l'espace de recherche [60]. L'algorithme SD, à son tour, utilise une représentation matricielle de distance d'un graphe avec une procédure de retour en arrière pour réduire la recherche [64]. L'algorithme de Nauty [65] utilise la théorie des groupes pour transformer les graphes à associer en une forme canonique afin de permettre une vérification plus efficace de l'isomorphisme des graphes. Cependant, il a été noté [66]que la construction des formes canoniques peut conduire à une complexité exponentielle dans le pire des cas. Bien que Conte [66] ait considéré Nauty comme l'algorithme d'isomorphisme de graphes le plus rapide, Miyazaki dans [45] a démontré qu'il existait certaines catégories de graphes qui nécessitaient un temps exponentiel pour générer l'étiquetage canonique. Les méthodes VF [30] et VF2 [61] utilisent une stratégie de recherche en profondeur d'abord (DFS), assistée par un ensemble de règles de faisabilité pour élaguer l'arbre de recherche. VF2 est une version améliorée de VF, qui explore la recherche plus efficacement l'espace, de sorte que le temps d'adaptation et la consommation de mémoire sont considérablement réduits.

II.12. Stratégie de recherche:

Deux stratégies de recherche de base sont utilisées pour l'extraction de sous-graphes fréquents, la stratégie de recherche en profondeur d'abord (DFS) et la stratégie de recherche en profondeur d'abord (BFS).

La stratégie DFS (Depth First Search) est une méthode permettant de parcourir ou de rechercher des structures de données arborescentes ou graphes. Il commence à la racine (en sélectionnant un nœud comme racine dans le cas du graphe) et explore autant que possible le long de chaque branche avant de revenir en arrière.

La largeur de la première recherche (BFS) utilise la stratégie inverse qui est effectuée par DFS. Il commence à la racine de l'arborescence (ou à un nœud arbitraire d'un graphe, parfois appelé "clé de recherche"), puis explore tous les nœuds voisins à la profondeur actuelle avant de passer aux nœuds au niveau de profondeur suivant. Figure II-9 illustrer le processus de DFS et BFS



recherche en profondeur d'abord largeur-première recherche

Figure II-18: Stratégies de recherche DFS et BFS.

Conclusion :

Dans ce chapitre, nous avons présenté quelques concepts de base des graphes et de leurs théories et nous nous sommes concentrés sur l'utilisation répétée de sous graphes et Une explication de la façon dont l'algorithme (FSM) fonctionne pour trouver des sous-schémas tels que la création de filtres, la vérification de symétrie, la prise en charge du comptage et nous avons introduit des techniques associées pour les implémenter.

Chapitre III

Deep Learning on Graphs

Introduction

Au cours de la dernière décennie, l'apprentissage en profondeur a été un «joyau» de l'intelligence artificielle et de l'apprentissage automatique , montrant des performances supérieures en acoustique , en images et en traitement du langage naturel . Le pouvoir d'expression de l'apprentissage en profondeur pour extraire des modèles complexes sous-jacents aux données a été bien reconnu. En revanche, les graphes1 sont omniprésents dans le monde réel et représentent des objets et leurs relations, tels que les réseaux sociaux, les réseaux de commerce électronique, les réseaux de biologie et les réseaux de trafic. On sait également que les graphes ont des structures compliquées qui contiennent une valeur sous-jacente riche . En conséquence, la manière d'utiliser des méthodes d'apprentissage en profondeur pour l'analyse de données graphes a suscité une attention considérable de la part de la recherche ces dernières années. Ce problème est non trivial, car l'application des architectures d'apprentissage en profondeur traditionnelles aux graphes pose plusieurs problèmes:

Domaine irrégulier

Contrairement aux images, à l'audio et au texte qui ont une structure de grille claire, les graphes sont situés dans un domaine irrégulier, ce qui rend difficile la généralisation de certaines opérations mathématiques de base aux graphes. Par exemple, il n'est pas simple de définir une opération de convolution et de mise en commun pour les données de graphe, qui sont les opérations fondamentales des réseaux de neurones de convolution (CNN). Ceci est souvent appelé le problème de l'apprentissage en profondeur géométrique.

Structures et tâches variables

Le graphe lui-même peut être compliqué avec diverses structures. Par exemple, les graphes peuvent être hétérogènes ou homogènes, pondérés ou non pondérés, et signés ou non signés. En outre, les tâches relatives aux graphes varient également considérablement, allant de problèmes centrés sur les noeuds tels que la classification des noeuds et la prédiction de liens à des problèmes axés sur les graphes tels que la classification et la génération de graphes. Les différentes structures et tâches nécessitent un modèle différent architectures pour traiter des problèmes spécifiques.

Evolutivité et parallélisation

À l'ère des données volumineuses, les graphes réels peuvent facilement comporter des millions de nœuds et de bords, tels que les réseaux sociaux ou les réseaux de commerce

électronique. En conséquence, la conception de modèles évolutifs, de préférence avec une complexité temporelle linéaire, devient un problème clé. De plus, étant donné que les nœuds et les arêtes du graphe sont interconnectés et doivent souvent être modélisés dans leur ensemble, la conduite du calcul parallèle constitue un autre problème critique.

Interdiscipline

Les graphes sont souvent liés à d'autres disciplines, telles que la biologie, la chimie ou les sciences sociales. L'interdiscipline offre à la fois des opportunités et des défis: la connaissance du domaine peut être exploitée pour résoudre des problèmes spécifiques, mais l'intégration de la connaissance du domaine pourrait rendre la conception du modèle plus difficile. Par exemple, lors de la génération de graphes moléculaires, la fonction objective et les contraintes chimiques sont souvent non différentiables, de sorte que les méthodes de formation basées sur les gradients ne peuvent pas être facilement appliquées.

2 Notations et préliminaires :

Noun graphe est représenté par G = (V; E) où **V = {v1,…., vn}** est un ensemble de

N = │V │noeuds et E ⊆ V × V est un ensemble de M = │E│ arêtes entre nœuds

Nous utilisons $\mathbf{A} \in \mathbb{R}^{N\times N}$ pour désigner la matrice d'adjacence, où sa i^{th} ligne,

sa j^{th} colonne et un élément désigné par **A(i; :); A(: ; j); A(i; j),**respectivement.

Le graphe peut être dirigé / non dirigé et pondéré / non pondéré. Nous considérons principalement les graphes non signés, donc **A (i; j) ≥ 0**

Nous utilisons $\mathbf{F}^V$ **et** $\mathbf{F}^E$ pour désigner les entités correspondant aux nœuds et aux arêtes. Pour les autres variables, nous utilisons des caractères gras majuscules pour désigner les matrices et des caractères gras minuscules pour désigner les

vecteurs, par ex. **X** et x. La transposition de la matrice est notée et le $\mathbf{X}^T$ produit élément par élément est notée Les fonctions sont marquées par curlicue, par ex. F(·).

Préliminaires. Pour un graphe non orienté, sa $\mathbf{X}_1 \odot \mathbf{X}_2$. matrice laplacienne $\mathbf{L} = \mathbf{D} - \mathbf{A}$. est $\mathbf{D} \in \mathbb{R}^{N\times N}$ définie par ,où est une matrice de degrés diagonaux

$\mathbf{D}(i,i) = \sum_{i \neq i} \mathbf{A}(i,j)$. Sa propre décomposition est noté $\mathbf{L} = \mathbf{Q}\boldsymbol{\Lambda}\mathbf{Q}^{\mathbf{T}}$, où

$\boldsymbol{\Lambda} \in \mathbb{R}^{N \times N}$ est une matrice diagonale de valeurs propres triées par ordre croissant et

$\mathbf{Q} \in \mathbb{R}^{N \times N}$ sont les vecteurs propres correspondants. La matrice de transition est définie comme suit: , $\mathbf{P} = \mathbf{D}^{-1}\mathbf{A}$, où **P** (i; j) représente la probabilité d'une marche aléatoire à partir du nœud **vi** atterrissant au nœud **v**j. Les voisins k-step du nœud **vi** sont définis comme suit : $\mathcal{N}_k(i) = \{j | d(i,j) \leq k\}$,

où **d (i; j)** est la distance la plus courte entre le nœud **vi** et **vj,** c'est-à-dire que $\mathcal{N}_k(i)$ est un ensemble de nœuds accessibles à partir du nœud **vi** à l'intérieur de k-étapes. Pour simplifier les notations, nous supprimons l'indice du voisinage immédiat, c'est-à-dire $\mathcal{N}(i) = \mathcal{N}_1(i)$.

Pour un modèle d'apprentissage en profondeur, nous utilisons des indices supérieurs pour désigner des couches, par exemple. $\mathbf{H}^l$. Nous utilisons f_l pour désigner le nombre de dimensions de la couche **l** . La fonction d'activation sigmoïde est définie par $\sigma(x) = 1/(1+e^{-x})$ et l'unité linéaire du redresseur (ReLU) est définie par ReLU (x) = max (0; x). Une fonction d'activation non linéaire générale par élément est notée p(·).

Sauf indication contraire, nous supposons que toutes les fonctions sont différenciables de sorte que nous puissions apprendre les paramètres du modèle par rétro-propagation [14] en utilisant des optimiseurs généralement adoptés, tels que Adam [15], et des techniques de formation, telles que le décrochage [16]. Nous résumons les notations dans le tableau 2.

Les tâches d'apprentissage de modèles approfondis sur des graphes peuvent être classées dans deux domaines :

• **Tâches axées sur les nœuds :** les tâches sont associées à des nœuds individuels dans le graphe. Les exemples incluent la classification de nœud, la prédiction de lien et la recommandation de nœud.

• **Tâches centrées sur les graphes :** les tâches sont associées à l'ensemble du graphe. Les exemples incluent la classification de graphe, l'estimation de certaines propriétés du graphe ou la génération de graphes. Notez que cette distinction est plus conceptuelle que mathématiquement rigoureuse. D'une part, il existe des tâches associées aux structures mésoscopiques telles que la détection de communautés [17]. De plus, les problèmes centrés sur les nœuds peuvent parfois être étudiés en tant que problèmes centrés sur les graphes en

transformant les premiers en réseaux égocentriques [18]. Néanmoins, nous détaillerons la distinction entre ces deux catégories lorsque cela sera nécessaire.

3- Graphe neural réseaux (GNNS) :

nous passons en revue les méthodes profondes les plus primitives semi-supervisées. méthodes d'apprentissage des données graphes, réseaux de neurones graphes (GNN).

L'idée de GNN est simple: pour coder les informations structurelles du graphe, chaque nœud vi peut être représenté par un vecteur d'état de faible dimension si; 1 ≤ i ≤ N. Motivé par les réseaux de neurones récursifs [21], une définition récursive des états est adoptée [20]:

$$\mathbf{s}_i = \sum_{j \in \mathcal{N}(i)} \mathcal{F}\left(\mathbf{s}_i, \mathbf{s}_j, \mathbf{F}_i^V, \mathbf{F}_j^V, \mathbf{F}_{i,j}^E\right), \qquad (1)$$

où F (·) est une fonction paramétrique à apprendre. Après avoir obtenu si, une autre fonction paramétrique $O(\cdot)$ est appliquée pour les sorties finales :

$$\hat{y}_i = \mathcal{O}\left(\mathbf{s}_i, \mathbf{F}_i^V\right). \qquad (2)$$

Pour les tâches relatives aux graphes, les auteurs suggèrent d'ajouter un nœud spécial avec des attributs uniques correspondant au graphe entier. Pour apprendre les paramètres du modèle, la méthode semi-supervisée suivante est adoptée: après résolution itérative de l'équation. (1) jusqu'à un point stable en utilisant la méthode de Jacobi [22], une étape de descente de gradient est effectuée à l'aide de l'algorithme Almeida-Pineda [23], [24] afin de minimiser une fonction objectif spécifique à une tâche, par exemple la perte de carré entre les valeurs prédites et la vérité sur le terrain pour les tâches de régression; ensuite, ce processus est répété jusqu'à la convergence.

Avec deux équations simples dans les équations. (1) (2), GNN joue deux rôles importants. Rétrospectivement, GNN unifie certaines des premières méthodes de traitement de données graphes, telles que les réseaux de neurones récursifs et les chaînes de Markov [20]. En ce qui concerne l'avenir, le concept de GNN a de profondes inspirations: comme nous le montrerons plus tard, de nombreux GCN à la pointe de la technologie ont en fait une formulation similaire à celle de l'Eq. (1), dans le cadre de l'échange d'informations avec les quartiers immédiats. En fait, les GNN et les GCN peuvent être unifiés dans un cadre commun et GNN est équivalent à un GCN en utilisant.

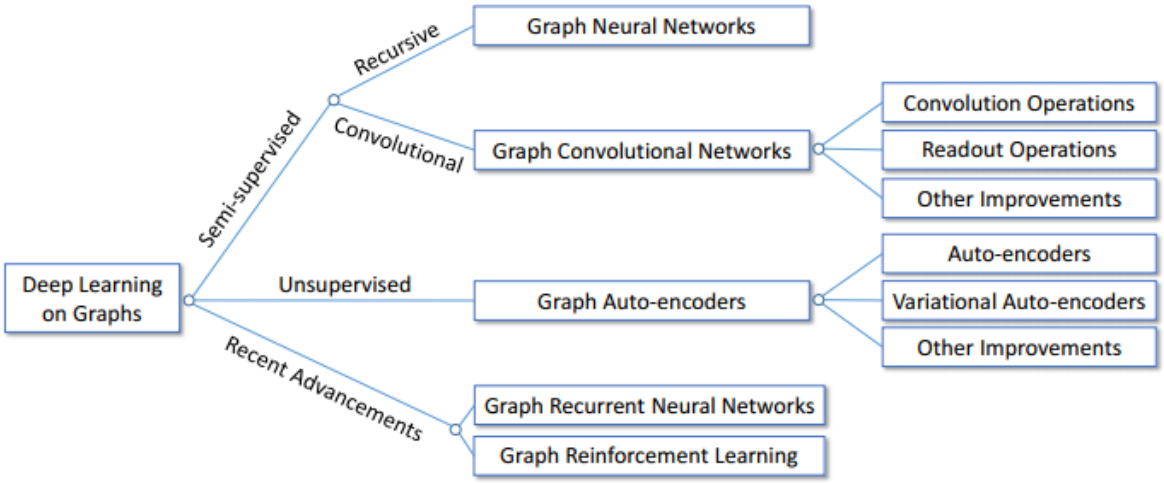


Fig. III 1. La catégorisation des méthodes d'apprentissage en profondeur sur des graphes.

Tableau. III 1. Some main distinctions of deep learning methods on graphs

Catégorie	Type	Attributs / étiquettes de nœuds	Homologues dans les domaines traditionnels
Graphe des réseaux de neurones	Semi-supervisé	Oui	Réseaux de neurones récursifs
Graph réseaux convolutifs	Semi-supervisé	Oui	Réseaux de neurones convolutifs
Autoencodeurs de graphes	Non supervisé	Partiel	Autoencodeurs / auto-encodeurs variationnels
Graph Recurrent Neural Networks	Various	Partiel	Recurrent Neural Networks
Apprentissage par renforcement graphe	Semi-supervisé	Oui	Apprentissage par renforcement

Tableau III 2.Tableau des notations couramment utilisées

$G = (V, E)$	Un graphe
N, M	Le nombre de nœuds et d'arêtes
$V = \{v_1, ..., v_N\}$	L'ensemble des nœuds
$\mathbf{F}^V, \mathbf{F}^E$	Attributs / caractéristiques pour les nœuds et les arêtes
$\mathbf{A}$	La matrice d'adjacence
$\mathbf{D}(i,i) = \sum_{j \neq i} \mathbf{A}(i,j)$	La matrice des degrés diagonaux
$\mathbf{L} = \mathbf{D} - \mathbf{A}$	La matrice laplacienne
$\mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T = \mathbf{L}$	La décomposition propre de L
$\mathbf{P} = \mathbf{D}^{-1}\mathbf{A}$	La matrice de transition
$\mathcal{N}_k(i), \mathcal{N}(i)$	voisins k-step et 1-step de **vi**
$\mathbf{H}^l$	La représentation cachée de l^{th} couche
f_l	Le nombre de dimensions de $\mathbf{H}^l$
$\rho(\cdot)$	Une activation non linéaire
$\mathbf{X}_1 \odot \mathbf{X}_2$	Produit par élément
$\mathbf{\Theta}$	Paramètres apprenables

Bien que conceptuellement important, GNN présente plusieurs inconvénients. Tout d'abord, pour s'assurer que l'équation (1) a une solution unique, F (·) doit être une «carte de contraction» [25], ce qui limite considérablement la capacité de modélisation. Deuxièmement, étant donné que de nombreuses itérations sont nécessaires entre les étapes de descente de gradient, GNN est coûteux en calcul. En raison de ces inconvénients et peut-être du manque de puissance de calcul (par exemple, l'unité de traitement graphe, GPU, n'est pas largement utilisée pour l'apprentissage en profondeur de nos jours) et du manque d'intérêts de recherche, GNN n'a pas largement connu de la communauté Une amélioration notable de GNN est la séquence de graphes gated Réseaux de neurones (GGS-NNs) [26] avec plusieurs modifications. Plus important encore, les auteurs remplacent la définition récursive de Eq. (1) avec les unités récurrentes gated (GRU) [27], supprime ainsi l'exigence de «carte de contraction» et prend en charge l'utilisation de techniques d'optimisation modernes. Plus précisément, Eq. (1) est remplacé par:

$$\mathbf{s}_i^{(t)} = (1 - \mathbf{z}_i^{(t)}) \odot \mathbf{s}_i^{(t-1)} + \mathbf{z}_i^{(t)} \odot \tilde{\mathbf{s}}_i^{(t)}, \qquad (3)$$

où **z** sont calculés par les portes de mise à jour, es est candidat à la mise à jour et t est le pseudo temps. Deuxièmement, les auteurs proposent d'utiliser plusieurs réseaux fonctionnant en séquence pour produire une sortie de séquence pouvant être appliquée à des applications telles que la vérification de programme [28].

GNN et ses extensions ont de nombreuses applications. Par exemple, CommNet [29] utilise GNN pour apprendre la communication multi-agents dans les systèmes d'intelligence artificielle en considérant chaque agent comme un nœud et en mettant à jour les états des agents

en communiquant avec d'autres pendant plusieurs instants avant de prendre une mesure. Interaction Network (IN) [30] utilise GNN pour le raisonnement physique en représentant les objets sous forme de nœuds, de relations en tant qu'arêtes et en utilisant le pseudo-temps comme système de simulation. VAIN [31] améliore CommNet et IN en introduisant des attentions pour peser différentes interactions. Relation Networks (RN) [32] propose d'utiliser GNN comme module de raisonnement relationnel pour renforcer d'autres réseaux de neurones et montrer des résultats prometteurs en ce qui concerne les problèmes de réponse visuelle à des questions.

4 RESEAUX CONVOLUTIONNELS DE GRAPHE (GCNS) :

Outre les GNN, les réseaux de convolution de graphes (GCN) constituent une autre classe de méthodes semi-supervisées pour les graphes. Étant donné que les GCN peuvent généralement être entraînés avec une perte spécifique à une tâche via une propagation en arrière, comme les CNN standard, nous nous concentrons sur les architectures adoptées. Nous discuterons d'abord des opérations de convolution, puis des opérations de lecture et des améliorations.

4.1 Opérations de convolution

4.1.1 Méthodes spectrales

Pour les CNN, la convolution est l'opération la plus fondamentale. Cependant, la convolution standard pour une image ou un texte ne peut pas être directement appliquée aux graphes en raison de l'absence de structure en grille [6]. Bruna et al. [33] introduisent d'abord la convolution pour les données de graphe du domaine spectral à l'aide du graphe laplacien matriciel L [54], qui joue un rôle similaire à la base de Fourier pour le traitement du signal [6]. Plus précisément, l'opération de convolution sur le graphe G est définie comme suit:

$$\mathbf{u}_1 *_G \mathbf{u}_2 = \mathbf{Q}\left(\left(\mathbf{Q}^T\mathbf{u}_1\right) \odot \left(\mathbf{Q}^T\mathbf{u}_2\right)\right), \qquad (4)$$

où $\mathbf{u}_1, \mathbf{u}_2 \in \mathbb{R}^N$ sont deux signaux définis sur les nœuds et **Q** sont vecteurs propres de **L**. Puis, en utilisant le théorème de convolution, le filtrage un signal **u** peut être obtenu sous la forme :

$$\mathbf{u}' = \mathbf{Q}\boldsymbol{\Theta}\mathbf{Q}^T\mathbf{u}, \qquad (5)$$

où $\mathbf{u}'$ est le signal de sortie, $\boldsymbol{\Theta} = \boldsymbol{\Theta}(\boldsymbol{\Lambda}) \in \mathbb{R}^{N\times N}$ est une matrice diagonale de filtres pouvant être appris et $\boldsymbol{\Lambda}$ sont des valeurs propres de **L**. Ensuite, une couche convolutionnelle est définie en appliquant différents filtres à différents signaux d'entrée et de sortie, comme suit :

$$\mathbf{u}_j^{l+1} = \rho\left(\sum_{i=1}^{f_l} \mathbf{Q}\boldsymbol{\Theta}_{i,j}^l\mathbf{Q}^T\mathbf{u}_i^l\right) \quad j = 1, ..., f_{l+1}, \qquad (6)$$

où $\mathbf{l}$ est la couche, $\mathbf{u}_j^l \in \mathbb{R}^N$ est j^{th} représentation cachée de les noeuds de la couche l^{th} ; sont des $\boldsymbol{\Theta}_{i,j}^l$ filtres pouvant être appris. L'idée de l'équation (6)

est similaire aux convolutions conventionnelles: passer les signaux d'entrée à travers un ensemble de filtres pouvant être appris pour agréger les informations, suivi d'une transformation non linéaire. En utilisant les fonctions de nœud $\mathbf{F}^V$ comme couche d'entrée et en empilant

plusieurs couches de convolution, l'architecture globale est similaire à celle des CNN. L'analyse théorique montre qu'une telle définition de l'opération de convolution sur les graphes peut imiter certaines propriétés géométriques des CNN, auxquelles nous renvoyons le lecteur [7] pour une étude complète.

Cependant, en utilisant directement l'équation. (6) nécessite l'apprentissage de $O(N)$ paramètres , ,ce qui peut ne pas être réalisable dans la pratique. De plus, les filtres dans le domaine spectral peuvent ne pas être localisés dans le domaine spatial. Pour atténuer ces problèmes, Bruna et al. [33] suggèrent d'utiliser les filtres lissés suivants:

$$diag\left(\mathbf{\Theta}_{i,j}^{l}\right) = \mathcal{K}\,\alpha_{l,i,j}, \qquad (7)$$

où **K** est un noyau d'interpolation fixe et sont $\alpha_{l,i,j}$ des coefficients d'interpolation apprises. Les auteurs ont également généralisé cette idée au paramètre où le graphe n'est pas donné, mais construit à partir de certaines caractéristiques brutes en utilisant une méthode supervisée ou non supervisée [34]. Cependant, deux limitations fondamentales restent non résolues. Premièrement, puisque tous les vecteurs propres de la matrice laplacienne sont nécessaires lors de chaque calcul, la complexité temporelle est au moins égale à $O(N^2)$par passe avant et arrière, ce qui n'est pas extensible pour des graphes à grande échelle. Deuxièmement, comme les filtres dépendent de la base propre **Q** du graphe, les paramètres ne peuvent pas être partagés entre plusieurs graphes de tailles et de structures différentes.

Ensuite, nous passons en revue deux lignes d'œuvres différentes qui tentent de résoudre ces deux limitations, puis nous les unifions à l'aide de cadres communs.

4.1.2 Aspect efficacité

Pour résoudre le problème d'efficacité, ChebNet [35] propose d'utiliser un filtre polynomial comme suit :

$$\mathbf{\Theta}(\mathbf{\Lambda}) = \sum_{k=0}^{K-1} \theta_k \mathbf{\Lambda}^k, \qquad (8)$$

Où $\theta_0, \ldots, \theta_{K-1}$ sont des paramètres que l'on peut apprendre et **K** est l'ordre polynomial. Ensuite, au lieu d'effectuer la décomposition propre, les auteurs réécrivent Eq. (8) utilisant l'extension Chebyshev [55]:

$$\mathbf{\Theta}(\mathbf{\Lambda}) = \sum_{k=0}^{K-1} \theta_k \mathcal{T}_k(\tilde{\mathbf{\Lambda}}), \qquad (9)$$

où $\tilde{\mathbf{\Lambda}} = 2\mathbf{\Lambda}/\lambda_{max} - \mathbf{I}$ sont les valeurs propres remises à l'échelle, λ_{max} est la valeur propre maximale, $\mathbf{I} \in \mathbb{R}^{N\times N}$ est la matrice d'identité et $\mathcal{T}_k(x)$ est le polynôme de Chebyshev d'ordre **k**. Le redimensionnement est nécessaire en raison de la base orthonormale des polynômes de Chebyshev. En utilisant le fait que polynôme des actes laplaciens

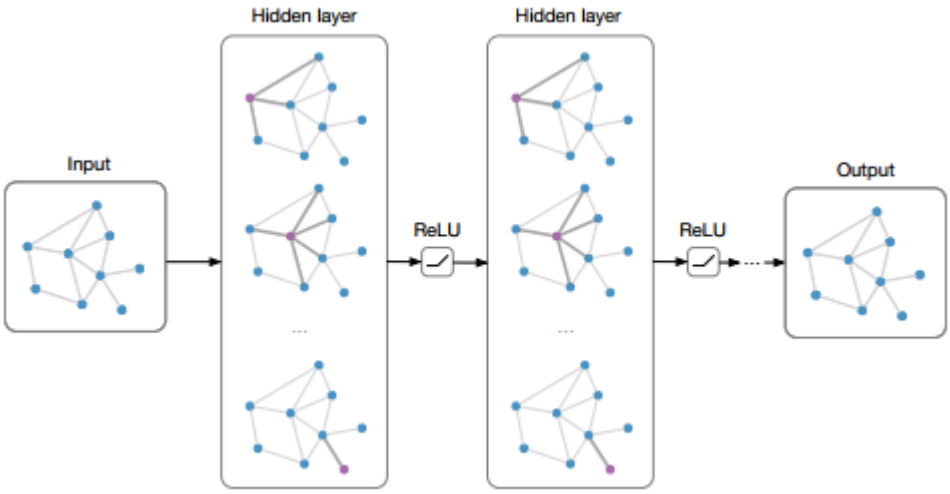


Fig. III 2. Un exemple illustrant le fonctionnement de la convolution . Réimprimé avec permission

en tant que polynôme de ses vecteurs propres, l'opération de filtrage dans Eq. (5) peut être réécrit comme:

$$\begin{aligned}\mathbf{u}' = \mathbf{Q}\boldsymbol{\Theta}(\boldsymbol{\Lambda})\mathbf{Q}^T\mathbf{u} &= \sum_{k=0}^{K-1}\theta_k\mathbf{Q}\mathcal{T}_k(\tilde{\boldsymbol{\Lambda}})\mathbf{Q}^T\mathbf{u} \\ &= \sum_{k=0}^{K-1}\theta_k\mathcal{T}_k(\tilde{\mathbf{L}})\mathbf{u} = \sum_{k=0}^{K-1}\theta_k\bar{\mathbf{u}}_k,\end{aligned} \tag{10}$$

où $\bar{\mathbf{u}}_k = \mathcal{T}_k(\tilde{\mathbf{L}})\mathbf{u}$ et $\tilde{\mathbf{L}} = 2\mathbf{L}/\lambda_{max} - \mathbf{I}$.

. Utilisation de la relation de récurrence du polynôme de Chebyshev

$\mathcal{T}_k(x) = 2x\mathcal{T}_{k-1}(x) - \mathcal{T}_{k-2}(x)$ and $\mathcal{T}_0(x) = 1, \mathcal{T}_1(x) = x$, $\bar{\mathbf{u}}_k$

peut également être calculé de manière récursive:

$$\bar{\mathbf{u}}_k = 2\tilde{\mathbf{L}}\bar{\mathbf{u}}_{k-1} - \bar{\mathbf{u}}_{k-2} \tag{11}$$

avec $\bar{\mathbf{u}}_0 = \mathbf{u}, \bar{\mathbf{u}}_1 = \tilde{\mathbf{L}}\mathbf{u}$. Maintenant, puisque seul le produit matriciel de $\tilde{\mathbf{L}}$ et certains vecteurs doivent être calculés, la complexité temporelle est **O (KM),** où M est le nombre d'arêtes et **K** est l'ordre polynomial, c'est-à-dire linéaire par rapport à la taille du graphe. Il est également facile de voir que ce filtre polynomial est strictement localisé en position **K**: après une convolution, la représentation du noeud vi ne sera affectée que par son voisinage en étape $\mathcal{N}_K(i)$Fait intéressant, cette idée est utilisée indépendamment dans l'incorporation au réseau pour préserver la proximité hiérarchique élevée [56], dont nous omettons les détails par souci de brièveté.

Une amélioration de ChebNet introduite par Kipf et Welling [36] simplifie encore le filtrage en utilisant uniquement les voisins de premier ordre comme suit:

$$\mathbf{h}_i^{l+1} = \rho\left(\sum_{j\in\tilde{\mathcal{N}}(i)} \frac{1}{\sqrt{\tilde{\mathbf{D}}(i,i)\tilde{\mathbf{D}}(j,j)}}\mathbf{h}_j^l\boldsymbol{\Theta}^l\right), \qquad (12)$$

Où $\mathbf{h}_j^l \in \mathbb{R}^{f_l}$ est la représentation cachée du noeud **vi** dans le l^{th} couche , $\tilde{\mathbf{D}} = \mathbf{D} + \mathbf{I}$ et $\tilde{\mathcal{N}}(i) = \mathcal{N}(i) \cup \{i\}$. Cela peut être écrit de manière équivalente sous la forme matricielle:

$$\mathbf{H}^{l+1} = \rho\left(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{H}^l\boldsymbol{\Theta}^l\right), \qquad (13)$$

où $\tilde{\mathbf{A}} = \mathbf{A}+\mathbf{I}$, c'est-à-dire l'ajout d'une connexion automatique. Les auteurs montrent que Eq. (13) est un cas particulier d'Eq. (8) en définissant K = 1 avec quelques modifications mineures. Ensuite, les auteurs soutiennent que l'empilement de telles couches a une capacité de modélisation similaire à celle de ChebNet et donne de meilleurs résultats. L'architecture est illustrée à la figure 2.

Un point important de ChebNet et de son extension est qu'ils connectent la convolution du graphe spectral à l'architecture spatiale, comme dans les GNN. En fait, la convolution dans l'équation (12) est très similaire à la définition des états dans GNN dans Eq. (1), sauf que la définition de convolution remplace la définition récursive. Dans cet aspect, GNN peut être considéré comme un réseau GCN utilisant un grand nombre de couches identiques pour atteindre des états stables [7].

4.1.3 Aspect des graphes multiples

En attendant, un parallèle de travaux se concentre sur la généralisation de l'opération de convolution à plusieurs graphes de tailles arbitraires. Les neurones FP [37] proposent une méthode spatiale utilisant également les voisins de premier ordre:

$$\mathbf{h}_i^{l+1} = \sigma\left(\sum\nolimits_{j\in\tilde{\mathcal{N}}(i)} \mathbf{h}_j^l\boldsymbol{\Theta}^l\right). \qquad (14)$$

Étant donné que les paramètres **Θ** peuvent être partagés par différents graphes et sont indépendants de la taille des graphes, les FP Neuronaux peuvent gérer plusieurs graphes de tailles arbitraires. Notez que Eq. (14) est très similaire à Eq. (12). Cependant, au lieu de considérer l'influence des degrés de nœud en ajoutant un terme de normalisation, les PF neuronaux proposent d'apprendre différents paramètres **Θ** pour des nœuds de degrés différents. Cette stratégie fonctionne bien pour les petits graphes tels que les graphes moléculaires, c'est-à-dire les atomes en tant que nœuds et les liaisons en tant qu'arêtes, mais peut ne pas être évolutive en graphes à grande échelle.

PATCHY-SAN [38] adopte une idée différente consistant à attribuer un ordre unique de nœuds à l'aide de la procédure d'étiquetage de graphe, telle que le noyau de Weisfeiler-Lehman [57], et organise les nœuds dans une ligne en utilisant cet ordre prédéfini. Pour imiter les CNN conventionnels, PATCHYSAN définit un «champ récepteur» pour chaque nœud **vi** en sélectionnant un nombre fixe de nœuds dans leurs voisinages à pas k puis $\mathcal{N}_k(i)$ adopte la norme CNN 1-D avec une normalisation appropriée. Étant donné que les nœuds de différents

graphes ont tous un «champ réceptif» de taille et d'ordre fixes, PATCHY-SAN peut tirer des enseignements de plusieurs graphes comme les CNN normaux. Cependant, l'inconvénient réside dans le fait que la convolution dépend fortement de la procédure d'étiquetage des graphes, étape de prétraitement non apprise. L'application forcée d'un ordre 1-D des nœuds n'est peut-être pas un choix naturel.

DCNN [39] adopte une autre approche pour remplacer la base propre de la convolution par une base de diffusion, c'est-à-dire que le «champ récepteur» des nœuds est déterminé par la probabilité de transition de diffusion entre les nœuds. Plus précisément, la convolution est définie comme suit:

$$\mathbf{H}^{l+1} = \rho\left(\mathbf{P}^K \mathbf{H}^l \mathbf{\Theta}^l\right), \qquad (15)$$

où $\mathbf{P}^K = (\mathbf{P})^K$ est la probabilité de transition d'un processus de diffusion de longueur K (c'est-à-dire une marche aléatoire), K est une longueur de diffusion prédéfinie et est une matrice $\mathbf{\Theta}^l \hat{\in} \mathbb{R}^{f_l \times f_l}$ diagonale de paramètres apprenables $\mathbf{\Theta}^l$. Comme seule $\mathbf{P}^K$ la dépend de la structure du graphe, les paramètres l peuvent être partagés entre des graphes de tailles arbitraires. Cependant, le calcul de la a$\mathbf{P}^K$ la complexité temporelle $O\left(N^2K\right)$, rendant ainsi la méthode non évolutive pour les graphes à grande échelle.

La DGCN [40] propose en outre d'adopter conjointement les bases de diffusion et de communication en utilisant un réseau convolutionnel à double graphe. Plus précisément, DGCN utilise deux convolutions: une en tant que Eq. (13) et l'autre remplace la matrice d'adjacence par une matrice d'informations mutuelles mutuelles par points (PMI) [58] de la probabilité de transition, c.-à-d.

$$\mathbf{Z}^{l+1} = \rho\left(\mathbf{D}_P^{-\frac{1}{2}} \mathbf{X}_P \mathbf{D}_P^{-\frac{1}{2}} \mathbf{Z}^l \mathbf{\Theta}^l\right), \qquad (16)$$

Où **X*p*** est la matrice PPMI et est $\mathbf{D}_P(i,i) = \sum_{j \neq i} \mathbf{X}_P(i,j)$ la matrice de degrés diagonale de XP. Ensuite, deux convolutions sont réunies en minimisant les différences quadratiques moyennes entre H et Z. Une procédure d'échantillonnage aléatoire de passage est également proposée pour accélérer le calcul de la probabilité de transition. Les expériences démontrent que de telles convolutions doubles sont efficaces même pour des problèmes à un seul graphe.

4.1.4 framworks :

Sur la base des deux lignes de travail ci-dessus, les MPNN [41] proposent un cadre unifié pour l'opération de convolution de graphes dans le domaine spatial, utilisant une fonction de transmission de message :

$$\begin{aligned} \mathbf{m}_i^{l+1} &= \sum_{j \in \mathcal{N}(i)} \mathcal{F}^l\left(\mathbf{h}_i^l, \mathbf{h}_j^l, \mathbf{F}_{i,j}^E\right) \\ \mathbf{h}_i^{l+1} &= \mathcal{G}^l\left(\mathbf{h}_i^l, \mathbf{m}_i^{l+1}\right), \end{aligned} \qquad (17)$$

où $\mathcal{F}^l$ et $\mathcal{G}^l$ sont des fonctions de message et des fonctions de mise à jour de vertex qui doivent être apprises, et ml sont les «messages» transmis entre les nœuds. Conceptuellement, les MPNN proposent un cadre dans lequel chaque nœud envoie des messages en fonction de ses états et met à jour ses états en fonction des messages reçus des voisins immédiats. Les auteurs montrent que le cadre ci-dessus inclut de nombreuses méthodes antérieures telles que [26], [33], [34], [36], [37], [50], à titre de cas particuliers. En outre, les auteurs proposent d'ajouter un nœud «maître» connecté à tous les nœuds afin d'accélérer le passage des messages sur de longues distances et de scinder la représentation masquée en différents «tours» afin d'améliorer la capacité de généralisation. Les auteurs montrent qu'une variante spécifique des NPN peut atteindre les performances les plus récentes en matière de prévision des propriétés moléculaires.

Parallèlement, Graph SAGE [42] prend une idée similaire à celle de l'équation. (17) avec plusieurs fonctions d'agrégation comme suit:

$$\begin{aligned} \mathbf{m}_i^{l+1} &= \text{AGGREGATE}^l(\{\mathbf{h}_j^l, \forall j \in \mathcal{N}(i)\}) \\ \mathbf{h}_i^{l+1} &= \rho\left(\mathbf{\Theta}^l\left[\mathbf{h}_i^l, \mathbf{m}_i^{l+1}\right]\right), \end{aligned} \tag{18}$$

où [·; ·] Est la concaténation et AGGREGATE (·) est la fonction d'agrégation. Les auteurs suggèrent trois fonctions d'agrégation:

mémoire élémentaire, mémoire à court terme (LSTM) [59] et regroupement comme suit :

$$\text{AGGREGATE}^l = \max\{\rho(\mathbf{\Theta}_{\text{pool}}\mathbf{h}_j^l + \mathbf{b}_{\text{pool}}), \forall j \in \mathcal{N}(i)\}, \tag{19}$$

où $\mathbf{\Theta}_{\text{pool}}$ et $\mathbf{b}_{\text{pool}}$ sont les paramètres à apprendre et max {} est le maximum élément par élément. Pour la fonction d'agrégation LSTM, puisqu'un ordre des voisins est nécessaire, les auteurs adoptent l'ordre aléatoire simple.

Le réseau de modèles de mélange (MoNet) [43] essaie également d'unifier les travaux antérieurs de GCN ainsi que CNN pour les variétés dans un cadre commun en utilisant une "mise en correspondance de modèles":

$$h_{ik}^{l+1} = \sum_{j \in \mathcal{N}(i)} \mathcal{F}_k^l(\mathbf{u}(i,j))\mathbf{h}_j^l, k = 1, ..., f_{l+1}, \tag{20}$$

où **u** (i; j) sont les pseudo-coordonnées de la paire de nœuds **vi** et **vj**, est une $\mathcal{F}_k^l(\mathbf{u})$ fonction paramétrique à apprendre, dimension h_{ik}^l est le k^{tn} de $\mathbf{h}^l$. En d'autres termes, sert de $\mathcal{F}_k^l(\mathbf{u})$ noyau de pondération pour combiner des quartiers. MoNet suggère ensuite d'utiliser le noyau gaussien:

$$\mathcal{F}_k^l(\mathbf{u}) = \exp\left(-\frac{1}{2}(\mathbf{u} - \boldsymbol{\mu}_k^l)^T(\mathbf{\Sigma}_k^l)^{-1}(\mathbf{u} - \boldsymbol{\mu}_k^l)\right), \tag{21}$$

où $\boldsymbol{\mu}_k^l$ sont les vecteurs moyens et $\mathbf{\Sigma}_k^l$ sont les matrices de covariance diagonales à apprendre. Les pseudo-coordonnées sont définies en degrés comme dans [36], c.-à-d.

$$\mathbf{u}(i,j) = \left(\frac{1}{\sqrt{\mathbf{D}(i,i)}}, \frac{1}{\sqrt{\mathbf{D}(j,j)}}\right). \qquad (22)$$

Les réseaux de graphes (GN) [9] proposent récemment un cadre plus général permettant aux réseaux GCN et GNN d'apprendre trois ensembles de représentations: $\mathbf{h}_i^l, \mathbf{e}_{ij}^l, \mathbf{z}^l$ en tant que représentation des nœuds, des arêtes et du graphe entier, respectivement. Les représentations sont apprises à l'aide de trois fonctions d'agrégation et de trois fonctions de mise à jour, comme suit :

$$\begin{aligned}
\mathbf{m}_i^l &= \mathcal{G}^{E\to V}(\{\mathbf{h}_j^l, \forall j \in \mathcal{N}(i)\}) \\
\mathbf{m}_V^l &= \mathcal{G}^{V\to G}(\{\mathbf{h}_i^l, \forall v_i \in V\}) \\
\mathbf{m}_E^l &= \mathcal{G}^{E\to G}(\{\mathbf{h}_{ij}^l, \forall (v_i, v_j) \in E\}) \\
\mathbf{h}_i^{l+1} &= \mathcal{F}^V(\mathbf{m}_i^l, \mathbf{h}_i^l, \mathbf{z}^l) \\
\mathbf{e}_{ij}^{l+1} &= \mathcal{F}^E(\mathbf{e}_{ij}^l, \mathbf{h}_i^l, \mathbf{h}_j^l, \mathbf{z}^l) \\
\mathbf{z}^{l+1} &= \mathcal{F}^G(\mathbf{m}_E^l, \mathbf{m}_V^l, \mathbf{z}^l),
\end{aligned} \qquad (23)$$

où $\mathcal{F}^V(\cdot), \mathcal{F}^E(\cdot), \mathcal{F}^G(\cdot)$ sont des fonctions de mise à jour correspondantes pour les nœuds, les arêtes et le graphe entier respectivement et sont $\mathcal{G}(\cdot)$ des fonctions de transmission de messages avec des exposants indiquant les directions de transmission des messages. Par rapport aux NPP, les GN introduisent des représentations de bord et la représentation entière du graphe, ce qui rend le cadre plus général.

les opérations de convolution ont évolué du domaine spectral au domaine spatial et des voisins multi-étapes aux voisins immédiats. Actuellement, la collecte d'informations auprès de voisins immédiats comme Eq. (13) et dans le cadre des équations d'équations. (17) (18) (23) sont les choix les plus courants pour l'opération de convolution de graphe.

4.2 Opérations de lecture

En utilisant les opérations de convolution, des fonctionnalités utiles pour les nœuds peuvent être apprises pour résoudre de nombreuses tâches centrées sur les nœuds. Cependant, pour traiter des tâches centrées sur les graphes, les informations des nœuds doivent être agrégées pour former une représentation au niveau des graphes. Dans la littérature, on appelle généralement cette opération l'opération de lecture ou de graissage grossière. Ce problème n'est pas trivial car les convolutions de foulée ou la mise en commun dans les CNN conventionnels ne peuvent pas être utilisées directement en raison de l'absence de structure de grille.

Invariance d'ordre. Une condition essentielle pour l'opération de lecture de graphe est que l'opération soit invariante à l'ordre des noeuds, c'est-à-dire que si nous modifions les indices des noeuds et des arêtes à l'aide d'une fonction bijective entre deux ensembles de sommets, la représentation de l'ensemble du graphe ne devrait pas changer. Par exemple, le fait qu'un médicament puisse traiter certaines maladies doit être indépendant de la façon dont le médicament est représenté sous forme de graphe. Notez que, puisque ce problème est lié au problème de l'isomorphisme de graphe connu comme NP [60], nous ne pouvons trouver qu'une fonction invariante à l'ordre, mais pas l'inverse en temps polynomial, c'est-à-dire que même deux graphes ne sont pas isomorphes. avoir la même représentation.

4.2.1 Statistiques

Les opérations les plus élémentaires qui sont invariantes à l'ordre sont des statistiques simples comme la mise en pool de sommes, moyennes ou maximales [37], [39], c.-à-d.

$$\mathbf{h}_G = \sum_{i=1}^{N} \mathbf{h}_i^L \text{ or } \mathbf{h}_G = \frac{1}{N}\sum_{i=1}^{N} \mathbf{h}_i^L \text{ or } \mathbf{h}_G = \max\left\{\mathbf{h}_i^L, \forall i\right\}, \tag{24}$$

où $\mathbf{h}_G$ est la représentation du graphe **G** et $\mathbf{h}_i^L$ est la représentation du nœud **vi** dans la couche finale **L**. Toutefois, ces statistiques du premier moment peuvent ne pas être suffisamment représentatives pour représenter le graphe en entier.

les auteurs suggèrent d'envisager la distribution des représentations de nœud en utilisant des histogrammes flous [61]. L'idée de base des histogrammes flous est de construire plusieurs "histogrammes".

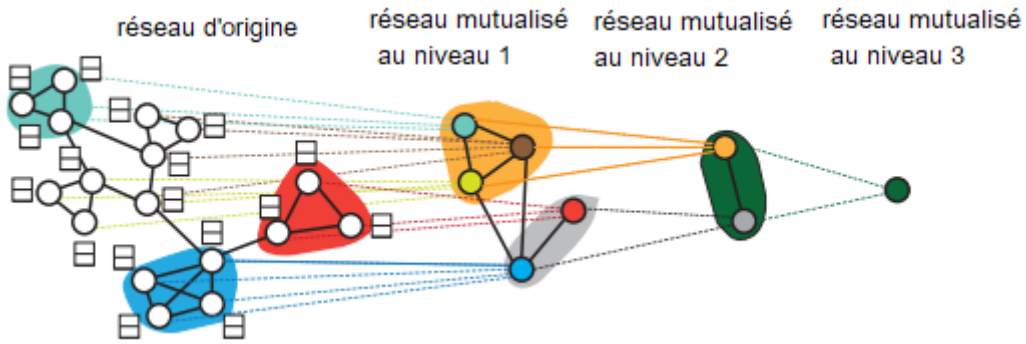


Fig. III 3. Un exemple de classification hiérarchique d'un graphe. avec la permission

«bacs», puis calcule les appartenances $\mathbf{h}_i^L$ de i à ces bacs, c'est-à-dire en ce qui concerne les représentations des nœuds en tant qu'échantillons et les fait correspondre à des modèles prédéfinis, puis renvoie la concaténation des histogrammes finaux. De cette manière, les nœuds avec la même somme / moyenne / maximum mais avec des distributions différentes peuvent être distingués

Une autre approche couramment utilisée pour collecter des informations consiste à ajouter une couche entièrement connectée (FC) en tant que couche finale [33], c'est-à-dire

$$\mathbf{h}_G = \mathbf{\Theta}_{FC}\mathbf{H}^L, \tag{25}$$

où $\mathbf{\Theta}_{FC}$ sont les paramètres de la couche FC. Éq. (25) peut être considéré comme une somme pondérée des caractéristiques combinées au niveau du nœud. Un des avantages est que le modèle peut apprendre différentes pondérations pour différents nœuds, au prix de l'impossibilité de garantir l'invariance des ordres.

4.2.2 Regroupement hiérarchique

Plutôt qu'une dichotomie entre la structure au niveau des nœuds ou des graphes, les graphes présentent des structures hiérarchiques riches [62], qui peuvent être explorées à l'aide de méthodes de classification hiérarchique illustrées à la figure 3. Par exemple, une classification agglomérée basée sur la densité [63] est utilisée. dans Bruna et al. [33] et le groupement spectral multi-résolution [64] est utilisé dans Henaff et al. [34] ChebNet [35] et MoNet [43] ont adopté Graclus [65], un autre algorithme de classification hiérarchique glouton permettant de fusionner deux nœuds à la fois, ainsi qu'une méthode de regroupement rapide en les réorganisant en un arbre binaire équilibré. ECC [48] adopte une autre méthode de classification hiérarchique par décomposition propre [66]. Cependant, ces méthodes de classification hiérarchique sont toutes indépendantes de l'opération de convolution, c'est-à-dire qu'elles peuvent être effectuées en tant qu'étape de pré-traitement et ne pas être formées de bout en bout.

Pour résoudre ce problème, DiffPool [44] propose un algorithme de classification hiérarchique différenciable, formé conjointement avec les convolutions de graphes. Plus précisément, les auteurs proposent d'apprendre une matrice d'affectation de grappes souples dans chaque couche à l'aide des représentations cachées:

$$\mathbf{S}^l = \mathcal{F}\left(\mathbf{A}^l, \mathbf{H}^l\right), \qquad (26)$$

où $\mathbf{S}^l \in \mathbb{R}^{N_l \times N_{l+1}}$ est la matrice d'affectation des N_l grappes, est le nombre de grappes de la couche **l** et F (·) est une fonction à apprendre. Ensuite, les représentations de nœud et la nouvelle matrice d'adjacence pour ce graphe «grossier» peuvent être obtenues en prenant la moyenne selon $\mathbf{S}^l$ comme suit:

$$\mathbf{H}^{l+1} = (\mathbf{S}^l)^T \hat{\mathbf{H}}^{l+1}, \mathbf{A}^{l+1} = (\mathbf{S}^l)^T \mathbf{A}^l \mathbf{S}^l, \qquad (27)$$

où $\hat{\mathbf{H}}^{l+1}$ est obtenu en appliquant une couche de convolution sur $\mathbf{H}^l$, c'est-à-dire réduire le graphe N_l de noeuds N_{l+1} à noeuds dans chaque couche après l'opération de convolution. Toutefois, comme l'affectation de grappes est souple, les connexions entre les grappes ne sont pas rares et la complexité temporelle de la méthode est en principe de .

$O(N^2)$

4.2.3 Autres

Outre les méthodes susmentionnées, il existe d'autres opérations de lecture qui méritent d'être discutées.

Dans GNNs [20], les auteurs suggèrent d'ajouter un nœud spécial connecté à tous les nœuds pour représenter l'ensemble du graphe. De même, les GN [9] prennent l'idée d'apprentissage direct de la représentation du graphe entier en recevant des messages de tous les nœuds et de toutes les arêtes.

Les MPNN adoptent set2set [67], une modification du modèle seq2seq invariante à l'ordre des entrées. Plus précisément, set2set utilise un modèle Lecture-Process-et-Écriture qui reçoit toutes les entrées en même temps, calcule les mémoires internes à l'aide du mécanisme d'attention et de LSTM, puis écrit les sorties.

Comme mentionné précédemment, PATCHY-SAN [38] prend l'idée d'imposer un ordre de nœuds à l'aide d'une procédure d'étiquetage de graphe, puis de recourir au pooling 1-D standard, comme dans CNN. La capacité de cette méthode à préserver l'invariance des ordres dépend de la procédure d'étiquetage des graphes, qui est un autre domaine de recherche qui dépasse le cadre de cet article [68]. Toutefois, imposer un ordre pour les nœuds peut ne pas être un choix naturel pour la collecte d'informations de nœud et pourrait nuire aux performances des tâches en aval. En bref, les statistiques telles que la moyenne ou la somme sont les opérations de lecture les plus simples, tandis que les algorithmes de classification hiérarchique formés conjointement avec les convolutions de graphes constituent une solution plus avancée mais sophistiquée. Pour des problèmes spécifiques, d'autres méthodes existent également

4.3 Améliorations et discussions

4.3.1 Mécanisme d'attention

Dans les GCN précédents, les voisinages de nœuds sont combinés avec des poids égaux ou prédéfinis. Cependant, l'influence des voisins peut varier considérablement, ce qui devrait être mieux appris au cours de la formation que prévu. S'inspirant du mécanisme d'attention [69], les réseaux d'attention au graphe (GAT) [45] introduisent les attentions dans les GCN en modifiant la convolution dans Eq (12) comme suit :

$$\mathbf{h}_i^{l+1} = \rho\left(\sum_{j\in\hat{\mathcal{N}}(i)} \alpha_{ij}^l \mathbf{h}_j^l \mathbf{\Theta}^l\right), \qquad (28)$$

où α_{ij}^l est l'attention définie comme suit:

$$\alpha_{ij}^l = \frac{\exp\left(\text{LeakyReLU}\left(\mathcal{F}\left(\mathbf{h}_i^l\mathbf{\Theta}^l, \mathbf{h}_j^l\mathbf{\Theta}^l\right)\right)\right)}{\sum_{k\in\hat{\mathcal{N}}(i)}\exp\left(\text{LeakyReLU}\left(\mathcal{F}\left(\mathbf{h}_i^l\mathbf{\Theta}^l, \mathbf{h}_k^l\mathbf{\Theta}^l\right)\right)\right)}, \qquad (29)$$

où F (· ; ·) est une autre fonction à apprendre, telle qu'un petit réseau entièrement connecté. Les auteurs suggèrent également d'utiliser des attentions multiples et indépendantes et de concaténer les résultats, c'est-à-dire l'attention multi-têtes de [69].

4.3.2 Connexions résiduelles et sautantes

Semblable à ResNet [70], des connexions résiduelles peuvent être ajoutées aux GCN existants pour ignorer certaines couches. Par exemple, Kipf & Welling [36] ajoutent des connexions résiduelles dans Eq. (13) comme suit :

$$\mathbf{H}^{l+1} = \rho\left(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{H}^l\mathbf{\Theta}^l\right) + \mathbf{H}^l. \qquad (30)$$

Ils montrent expérimentalement que l'ajout de telles connexions résiduelles peut augmenter la profondeur du réseau, c'est-à-dire le nombre de couches de convolution dans les GCN, ce qui est similaire aux résultats de ResNet.

Le réseau de colonnes (CLN) [46] reprend une idée similaire en utilisant les connexions résiduelles suivantes avec des poids pouvant être appris :

$$\mathbf{h}_i^{l+1} = \boldsymbol{\alpha}_i^l \odot \widetilde{\mathbf{h}}_i^{l+1} + (1 - \boldsymbol{\alpha}_i^l) \odot \mathbf{h}_i^l, \qquad (31)$$

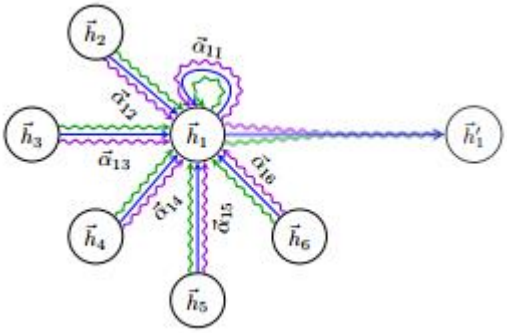


Fig. III 4. Une illustration de l'attention multi-têtes proposée dans les GAT [45] où chaque couleur dénote une attention indépendante. Réimprimé avec permission

où $\widetilde{\mathbf{h}}_i^{l+1}$ est similaire à l'équation (13) et $\boldsymbol{\alpha}_i^l$ sont des poids calculés comme suit:

$$\boldsymbol{\alpha}_i^l = \rho\left(\mathbf{b}_\alpha^l + \boldsymbol{\Theta}_\alpha^l \mathbf{h}_i^l + \boldsymbol{\Theta}_\alpha^{\prime l} \sum_{j \in \mathcal{N}(i)} \mathbf{h}_j^l\right), \qquad (32)$$

où $\mathbf{b}_\alpha^l, \boldsymbol{\Theta}_\alpha^l, \boldsymbol{\Theta}_\alpha^{\prime l}$ y a quelques paramètres à apprendre. Notez que Eq. (31) est très similaire à la GRU comme dans GGS-NNs [26], mais l'architecture globale est basée sur des couches convolutives au lieu de pseudo-temps.

Les réseaux de connaissances sautants (JK-Nets) [47] proposent une autre architecture permettant de connecter la dernière couche du réseau avec toutes les couches cachées précédentes, c'est-à-dire «sautant» toutes les représentations vers la sortie finale, comme l'illustre la figure 5. Ainsi, le modèle peut apprendre à exploiter de manière sélective des informations provenant de différentes localités. Formellement, JK-Nets peut être formulé comme suit:

$$\mathbf{h}_i^{final} = \text{AGGREGATE}(\mathbf{h}_i^0, \mathbf{h}_i^1, ..., \mathbf{h}_i^K), \qquad (33)$$

où $\mathbf{h}_i^{final}$ est la représentation finale du nœud vi, AGGREGATE (·) est la fonction d'agrégation et K le nombre de couches masquées. JK-Nets utilise trois fonctions d'agrégation similaires à GraphSAGE [42]: la concaténation, le pooling maximal et l'attention de LSTM.

Les résultats expérimentaux montrent que l'ajout de connexions sauteuses peut améliorer les performances de plusieurs architectures GCN.

4.3.3 Caractéristiques du bord

Les GCN précédents se concentrent principalement sur l'utilisation des fonctionnalités de nœud. Une autre source d'information importante est constituée par les fonctions périphériques, que nous allons brièvement aborder dans cette section.

Pour les entités de bord simples avec des valeurs discrètes, telles que les types de bord, une méthode simple consiste à entraîner différents paramètres pour différents types de bord et à agréger les résultats. Par exemple, les neurones FP [37] entraînent différents paramètres pour des nœuds de degrés différents, ce qui correspond à la caractéristique d'arête cachée des types de liaison dans un graphe moléculaire, et additionnent les résultats. CLN [46] entraîne différents paramètres pour différents types d'arêtes dans un graphe hétérogène et effectue la moyenne des résultats. Convolution conditionnée par les bords (ECC) [48] entraîne également différents paramètres en fonction des types de bords et les applique à la classification des graphes. Les GCN relationnels (R-GCN) [49] prennent une idée similaire dans les graphes de connaissance en formant différents poids pour différents types de relation. Cependant, ces méthodes ne peuvent gérer que des caractéristiques de bord discrètes limitées.

DCNN [39] propose une autre méthode pour convertir chaque arête en un nœud connecté aux nœuds de tête et de queue de l'arête. Ensuite, les entités périphériques peuvent être traitées comme des entités nodales.

Kearnes et al. [50] proposent une autre architecture utilisant le «module de tissage». Plus précisément, ils apprennent des représentations à la fois

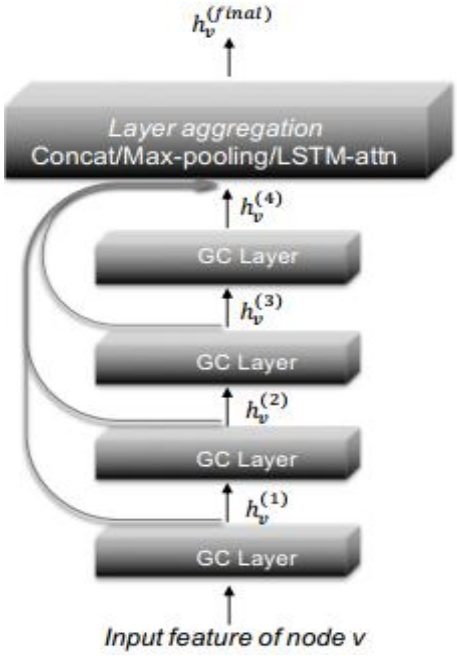


Fig. III 5. Réseaux de connaissances sautés proposés dans [47] où la dernière couche est connectée à toutes les couches cachées pour exploiter de manière sélective les informations provenant de différentes localités. Réimprimé avec permission

les nœuds et les arêtes et échangent des informations entre eux dans chaque module d'armure avec quatre fonctions différentes: nœud à nœud (NN), nœud à bord (NE), bord à bord (EE) et bord à nœud (EN):

$$\begin{aligned}
\mathbf{h}_i^{l'} &= \mathcal{F}_{NN}(\mathbf{h}_i^0, \mathbf{h}_i^1, ..., \mathbf{h}_i^l) \\
\mathbf{h}_i^{l''} &= \mathcal{F}_{EN}(\{\mathbf{e}_{ij}^l | j \in \mathcal{N}(i)\}) \\
\mathbf{h}_i^{l+1} &= \mathcal{F}_{NN}(\mathbf{h}_i^{l'}, \mathbf{h}_i^{l''}) \\
\mathbf{e}_{ij}^{l'} &= \mathcal{F}_{EE}(\mathbf{e}_{ij}^0, \mathbf{e}_{ij}^1, ..., \mathbf{e}_{ij}^l) \\
\mathbf{e}_{ij}^{l''} &= \mathcal{F}_{NE}(\mathbf{h}_i^l, \mathbf{h}_j^l) \\
\mathbf{e}_{ij}^{l+1} &= \mathcal{F}_{EE}(\mathbf{e}_{ij}^{l'}, \mathbf{e}_{ij}^{l''}),
\end{aligned} \tag{34}$$

où 1 sont des représentations du bord **(vi; vj)** dans la 2 couche et F (·) sont des fonctions pouvant être apprises avec des indices représentant les directions de passage des messages. En empilant plusieurs de ces modules, les informations peuvent se propager en passant alternativement entre les nœuds et les représentations d'arêtes. Notez que dans les fonctions Node-to-Node et Edge-to-Edge, des connexions sautantes similaires à JKNets [47] sont ajoutées implicitement. Les réseaux de graphes [9] proposent également d'apprentissage de la représentation des arêtes et mettent à jour les représentations des nœuds et des arêtes à l'aide de fonctions de transmission de messages, comme indiqué à la section 4.1, qui contient le «module tissage» comme cas particulier.

4.3.4 Accélération par échantillonnage

Le problème d'efficacité est l'un des principaux goulots d'étranglement dans la formation des GCN pour la création de graphes à grande échelle. Dans cette section, nous passons en revue certaines méthodes d'accélération pour GCN.

Comme indiqué précédemment, de nombreux réseaux GCN suivent le cadre de regroupement des informations provenant des quartiers. Cependant, étant donné que de nombreux graphes réels suivent la distribution de la loi de puissance [71], c'est-à-dire que peu de nœuds ont des degrés très élevés, l'expansion des voisins peut croître extrêmement rapidement. Pour résoudre ce problème, GraphSAGE [42] échantillonne uniformément un nombre fixe de voisins pour chaque noeud au cours de la formation. PinSage [51] propose en outre d'échantillonner les voisins en utilisant des marches aléatoires sur des graphes ainsi que plusieurs améliorations de la mise en œuvre, par exemple. coordination entre CPU et GPU, un pipeline de réduction de déduction de carte, etc. Il a été démontré que PinSage fonctionnait bien dans un graphe réel à l'échelle d'un milliard.

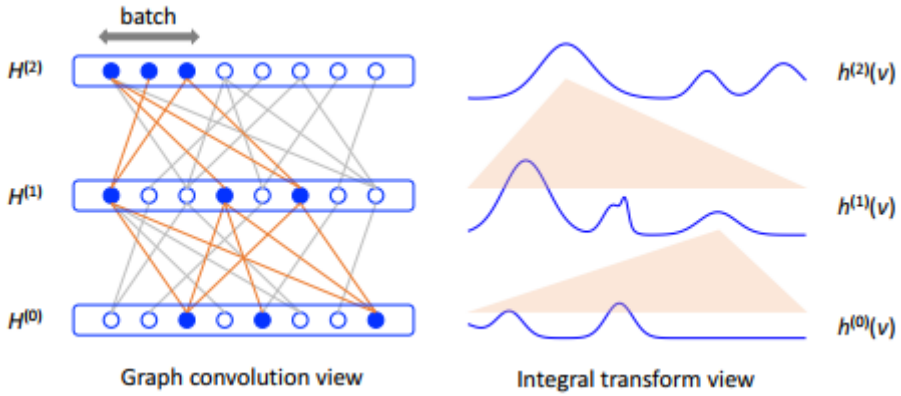


Fig. III 6. Méthode d'échantillonnage de noeud de FastGCN . FastGCN interprète les convolutions de graphes

FastGCN [52] adopte une stratégie d'échantillonnage différente. Au lieu d'échantillonner les voisins de chaque nœud, les auteurs suggèrent d'échantillonner les nœuds de chaque couche de convolution en interprétant les nœuds comme étant Les échantillons et les convolutions graphes sous forme de transformations intégrales dans des mesures de probabilité, comme indiqué dans la figure 6. FastGCN montre également que les nœuds d'échantillonnage via leurs degrés normalisés peuvent réduire la variance et améliorer les performances.

Chen et al. [53] proposent en outre une autre méthode d'échantillonnage pour réduire la variance. Spécifiquement, les activations historiques des nœuds sont utilisées comme variables de contrôle, ce qui permet des tailles d'échantillons arbitrairement petites. Les auteurs prouvent également que cette méthode a une garantie théorique et surpasse les méthodes d'accélération existantes dans les expériences.

4.3.5 Réglage inductif

Un autre aspect important des GCN s'applique au paramètre inductif, c'est-à-dire à la formation sur un ensemble de noeuds / graphes et au test sur un autre ensemble de noeuds / graphes invisibles au cours de la formation. En principe, ceci est réalisé en apprenant une fonction de mappage sur les caractéristiques données, qui ne dépendent pas de la base du graphe et peuvent être transférées sur des nœuds / graphes. Le réglage inductif est vérifié dans GraphSAGE [42], GATs [45] et FastGCN [52]. Cependant, les GCN inductifs existants ne conviennent que pour les graphes avec des caractéristiques. Comment mener un apprentissage inductif pour des graphes sans caractéristiques, généralement appelé le problème de l'échantillon non échantillonné [72], reste largement ouvert dans la littérature.

4.3.6 Poids aléatoires

Les GCN avec des poids aléatoires constituent également une direction de recherche intéressante, similaire au cas des réseaux de neurones généraux [73]. Par exemple, les PF neuronaux [37] avec des poids aléatoires élevés présentent des performances similaires à celles des empreintes digitales circulaires, certaines caractéristiques fabriquées à la main pour les molécules indexées par hachage. Kipf et Welling [36] montrent également que les GCN non

entraînés peuvent extraire certaines caractéristiques de nœud significatives. Cependant, une compréhension générale des GCN avec des poids aléatoires reste inexplorée.

5- AUTOENCODEURS DE GRAPHE (GAES)

L'auto-codeur (AE) et ses variantes sont largement utilisés pour l'apprentissage non supervisé [74], qui conviennent à l'apprentissage des représentations de nœud pour les graphes sans information supervisée. nous allons d'abord présenter les autoencodeurs de graphes, puis passer à des autoencodeurs de graphes graphes et à d'autres améliorations. Les principales caractéristiques des EAG étudiés sont résumées dans le tableau 4.

5.1 Autoencodeurs

L'utilisation d'EA pour les graphes provient de Sparse Autoencoder (SAE) [75] 3. L'idée de base est que, en considérant la matrice d'adjacence ou ses variations comme les caractéristiques brutes des nœuds, les AE peuvent être utilisés comme technique de réduction de dimension pour apprendre des représentations de nœud de faible dimension. En particulier, SAE adopte la perte de reconstruction L2 suivante :

$$\min_{\mathbf{\Theta}} \mathcal{L}_2 = \sum_{i=1}^{N} \left\| \mathbf{P}\left(i,:\right) - \hat{\mathbf{P}}\left(i,:\right) \right\|_2 \qquad (35)$$
$$\hat{\mathbf{P}}\left(i,:\right) = \mathcal{G}\left(\mathbf{h}_i\right), \mathbf{h}_i = \mathcal{F}\left(\mathbf{P}\left(i,:\right)\right),$$

où **P** est la matrice de transition, $\hat{\mathbf{P}}$ est la matrice reconstruite $\mathbf{h}_i \in \mathbb{R}^d$ est la représentation basse dimension du nœud vi, F (·) est le codeur, $\mathcal{G}(\cdot)$ est le décodeur, d<< N est la dimensionnalité et **Θ** sont des paramètres. Le codeur et le décodeur sont des perceptrons multicouches avec de nombreuses couches cachées. En d'autres termes, SAE essaie de compresser les informations de **P** (i ; **:**) en vecteurs de faible dimension **hi** et reconstruire le vecteur d'origine. SAE ajoute également un autre terme de régularisation de parcimonie. Après avoir obtenu la représentation en basse dimension , **hi** k-signifie [85] est appliqué à la tâche de regroupement de nœuds, ce qui s'avère empiriquement plus performant que les référentiels d'apprentissage non approfondis. Cependant, l'analyse théorique étant incorrecte, le mécanisme sous-jacent à cette efficacité reste inexpliqué.

La structure en réseau en profondeur (SDNE) [76] complète le casse-tête en montrant que la perte de reconstruction de niveau 2 dans l'équation. (35) correspond en fait à la proximité de second ordre, c'est-à-dire que deux nœuds partagent des représentations verrouillées similaires s'ils ont des voisinages similaires, ce qui est bien étudié dans la science des réseaux comme le filtrage collaboratif ou la fermeture du triangle [5]. Motivé par les méthodes d'intégration de réseau, qui montrent que la proximité de premier ordre est également importante [86], SDNE modifie la fonction objectif en ajoutant un autre terme similaire à celui des cartes propres à Laplacian [54]:

$$\min_{\mathbf{\Theta}} \mathcal{L}_2 + \alpha \sum_{i,j=1}^{N} \mathbf{A}(i,j) \left\| \mathbf{h}_i - \mathbf{h}_j \right\|_2, \qquad (36)$$

c'est-à-dire que deux nœuds doivent également partager des représentations de verrouillage similaires s'ils sont directement connectés. Les auteurs ont également modifié la perte de reconstruction L2 en utilisant la matrice d'adjacence et en attribuant des poids différents à des éléments nuls et non nuls:

$$\mathcal{L}_2 = \sum_{i=1}^{N} \|(\mathbf{A}(i,:) - \mathcal{G}(\mathbf{h}_i)) \odot \mathbf{b}_i\|_2 , \qquad (37)$$

où **bij** = 1 si A (i; j) = 0 et **bij** = β> 1 sinon et β est un autre hyper-paramètre.

Motivé par une autre ligne de travail, un travail contemporain DNGR [77] remplace la matrice de transition P dans Eq. (35) avec la matrice d'information de point mutuel positif (PPMI) [58] d'une probabilité de surf aléatoire. De cette manière, les entités brutes peuvent être associées à une probabilité de marche aléatoire du graphe [87]. Cependant, la construction de la matrice en entrée peut nécessiter une complexité temporelle de $O(N^2)$, qui n'est pas évolutive pour les graphes à grande échelle.

GC-MC [78] adopte en outre une approche différente pour les auto-codeurs en utilisant GCN dans [36] en tant que codeur:

$$\mathbf{H} = GCN\left(\mathbf{F}^V, \mathbf{A}\right), \qquad (38)$$

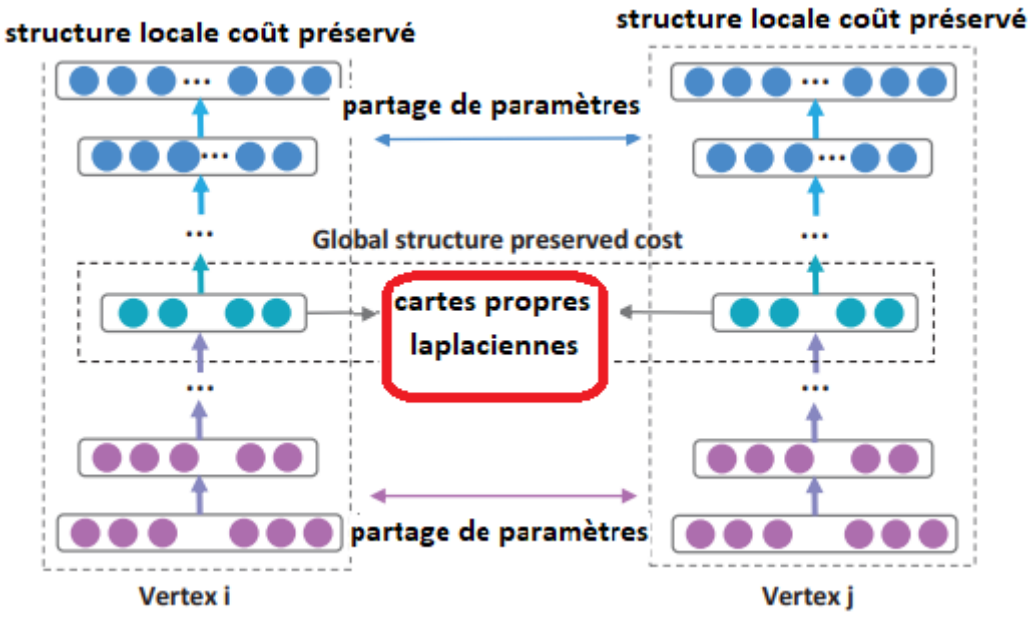


Fig. III 7.Le framework de SDNE réimprimé de [76] avec permission

le décodeur est une fonction linéaire simple :

$$\hat{\mathbf{A}}(i,j) = \mathbf{H}(i,:)\boldsymbol{\Theta}_{de}\mathbf{H}(j,:)^T, \qquad (39)$$

où **Θ** de sont les paramètres du codeur. De cette façon, les caractéristiques des nœuds peuvent être naturellement incorporées. Pour les graphes sans caractéristiques de nœud, un codage à

chaud des nœuds peut être utilisé. Les auteurs démontrent l'efficacité de GC-MC sur le problème de recommandation des graphes bipartites.

Au lieu de reconstruire la matrice d'adjacence ou ses variations, DRNE [79] propose une autre modification pour reconstruire directement les vecteurs de dimension réduite des nœuds en agrégeant les informations de voisinage à l'aide de LSTM. Plus précisément, DRNE minimise la fonction objectif suivante :

$$\mathcal{L} = \sum_{i=1}^{N} \|\mathbf{h}_i - \text{LSTM}\left(\{\mathbf{h}_j | j \in \mathcal{N}(i)\}\right)\|. \qquad (40)$$

Comme LSTM nécessite une séquence d'entrées, les auteurs suggèrent de classer les voisinages des nœuds en fonction de leurs degrés. Un échantillon de voisins est également adopté pour les nœuds avec des degrés élevés afin d'éviter que la mémoire ne soit trop longue. Les auteurs prouvent qu'une telle méthode peut préserver une équivalence régulière et de nombreuses mesures de centralité de nœuds

Contrairement aux travaux précédents qui cartographe les nœuds en vecteurs de faible dimension, Graphe 2 Gauss (G2G) [80] propose de coder chaque nœud sous forme de distribution $\mathbf{h}_i = \mathcal{N}\left(\mathbf{M}(i,:), diag\left(\mathbf{\Sigma}(i,:)\right)\right)$

$$\mathbf{M}(i,:) = \mathcal{F}_{\mathbf{M}}(\mathbf{F}^V(i,:)), \mathbf{\Sigma}(i,:) = \mathcal{F}_{\mathbf{\Sigma}}(\mathbf{F}^V(i,:)), \qquad (41)$$

gaussienne. capturer les incertitudes des nœuds. Plus précisément, les auteurs utilisent un mappage détaillé des attributs de nœud sur les moyennes et les variances de la distribution gaussienne en tant que codeur:

où F_M (·) et F_Σ (·) sont des fonctions paramétriques doivent être apprises. Ensuite, au lieu d'utiliser une fonction de décodeur explicite, ils utilisent des contraintes par paires pour apprendre le modèle :

$$\begin{aligned} &\text{KL}\left(\mathbf{h}_j || \mathbf{h}_i\right) < \text{KL}\left(\mathbf{h}_{j'} || \mathbf{h}_i\right) \\ &\forall i, \forall j, \forall j' \; s.t. \; d(i,j) < d(i,j'), \end{aligned} \qquad (42)$$

où **d (i; j)** est la distance la plus courte entre le noeud vi et v_j et $\text{KL}[q(\cdot)||p(\cdot)]$

est la divergence de Kullback-Leibler (KL) entre q (·) et p (·) [89]. En d'autres termes, les contraintes garantissent que la divergence KL entre les paires de noeuds a le même ordre relatif que la distance du graphe. Cependant, depuis l'équation (42) est difficile à optimiser, la perte d'énergie [90] est utilisée comme une relaxation :

$$\mathcal{L} = \sum_{(i,j,j') \in \mathcal{D}} \left(E_{ij}^2 + \exp^{-E_{ij'}}\right), \qquad (43)$$

où $\mathcal{D} = \{(i,j,j') | d(i,j) < d(i,j')\}$ et $E_{ij} = \text{KL}(\mathbf{h}_j || \mathbf{h}_i)$.

Une stratégie d'échantillonnage non biaisée est en outre proposée pour accélérer le processus de formation.

5.2 Autoencodeurs variationnels

Contrairement aux anciens auto-encodeurs, Variational Autoencoder (VAE) est un autre type de méthode d'apprentissage en profondeur qui combine la réduction de dimension avec des modèles génératifs [91]. VAE a été introduit pour la première fois dans la modélisation des données de graphe dans [81], où le décodeur est un simple produit linéaire :

$$p\left(\mathbf{A}|\mathbf{H}\right)=\prod_{i,j=1}^{N}\sigma\left(\mathbf{h}_i\mathbf{h}_j^T\right), \qquad (44)$$

où **hi** est supposé suivre une distribution postérieure gaussienne Pour le codeur des matrices $q\left(\mathbf{h}_i|\mathbf{M},\mathbf{\Sigma}\right)=\mathcal{N}\left(\mathbf{h}_i|\mathbf{M}(i,:),diag\left(\mathbf{\Sigma}(i,:)\right)\right).$ de moyenne et de variance, les auteurs adoptent GCN

$$\mathbf{M}=GCN_{\mathbf{M}}\left(\mathbf{F}^V,\mathbf{A}\right),\log\mathbf{\Sigma}=GCN_{\mathbf{\Sigma}}\left(\mathbf{F}^V,\mathbf{A}\right). \quad (45)$$

Ensuite, les paramètres du modèle peuvent être appris en minimisant la limite inférieure variationnelle .

$$\mathcal{L}=\mathbb{E}_{q(\mathbf{H}|\mathbf{F}^V,\mathbf{A})}\left[\log p\left(\mathbf{A}|\mathbf{H}\right)\right]-\mathrm{KL}\left[q\left(\mathbf{H}|\mathbf{F}^V,\mathbf{A}\right)||p(\mathbf{H})\right]. \qquad (46)$$

Cependant, étant donné que le graphe complet doit être reconstruit, la complexité temporelle est complexe $O(N^2)$.

Motivée par SDNE et G2G, DVNE [82] propose un autre VAE pour les données de graphe en représentant également chaque nœud sous forme de distribution gaussienne. Contrairement aux travaux antérieurs qui adoptent la divergence **KL** -comme mesure, DVNE utilise la distance de Wasserstein [92] pour préserver la transitivité des similitudes de nœuds. Similaire à SDNE.

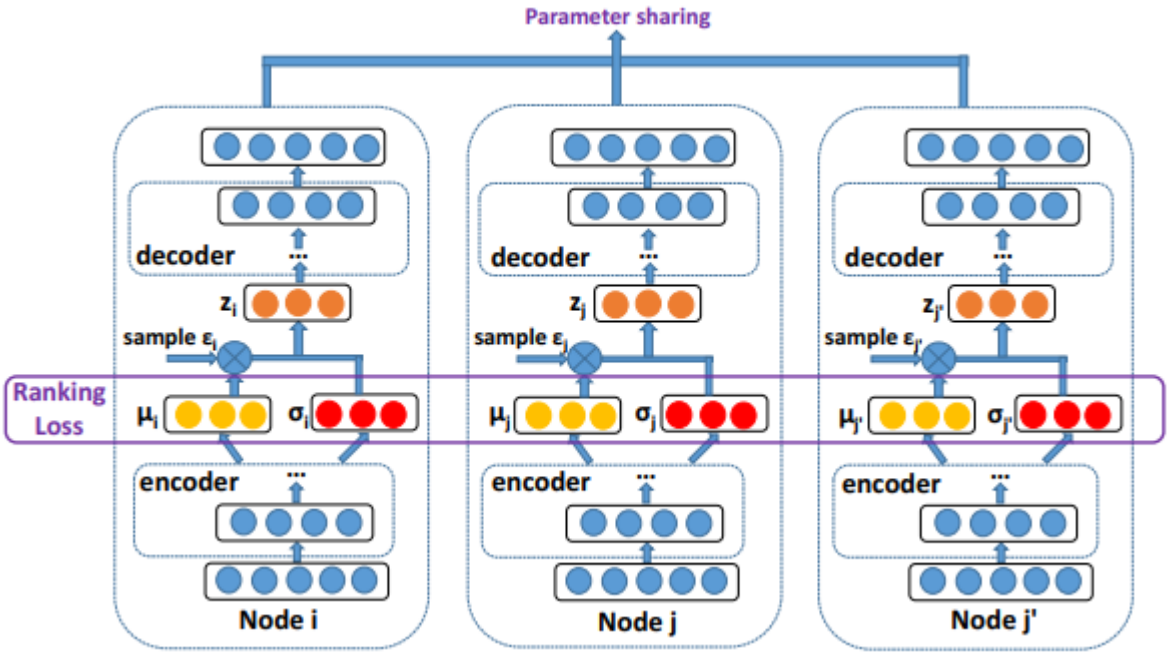


Fig. III 8. Le cadre de DVNE avec permission.

et G2G, DVNE préservent également la proximité des premier et deuxième ordres dans la fonction objectif :

$$\min_{\mathbf{\Theta}} \sum_{(i,j,j')\in\mathcal{D}} \left(E_{ij}^2 + \exp^{-E_{ij'}}\right) + \alpha\mathcal{L}_2, \qquad (47)$$

où $E_{ij} = W_2(\mathbf{h}_j||\mathbf{h}_i)$ est le 2^{nd} Distance de Wasserstein entrdeux distributions gaussiennes **hg** et **hi** et est $\mathcal{D} = \{(i,j,j')|j \in \mathcal{N}(i), j' \notin \mathcal{N}(i)\}$ un ensemble de tous les triplets correspondant à la perte de classement de proximité de premier ordre. La perte de reconstruction est définie comme :

où P est la matrice de transition et Z sont des échantillons tirés de H. Le cadre est illustré à la figure 8. Ensuite, la fonction objectif peuvent être minimisés comme des VAE classiques en utilisant l'astuce de reparamétrage [91].

$$\mathcal{L}_2 = \inf_{q(\mathbf{Z}|\mathbf{P})} \mathbb{E}_{p(\mathbf{P})}\mathbb{E}_{q(\mathbf{Z}|\mathbf{P})} \|\mathbf{P} \odot (\mathbf{P} - \mathcal{G}(\mathbf{Z}))\|_2^2, \qquad (48)$$

5.3 Améliorations et discussions

Outre ces deux catégories principales, plusieurs améliorations méritent d'être discutées.

5.3.1 Formation contradictoire

Le programme de formation à la confrontation, en particulier le réseau de confrontation générative (GAN), a récemment été un sujet brûlant dans l'apprentissage automatique [93]. L'idée de base du GAN est de construire deux modèles liés, un discriminateur et un générateur. Le but du générateur est de "tromper" le discriminateur en générant de fausses données, tandis que le discriminateur vise à distinguer si un échantillon est issu de données réelles ou généré par le générateur. Ensuite, les deux modèles peuvent tirer profit d'une formation commune utilisant un jeu minimax.

Le programme de formation en confrontation est incorporé aux GAE en tant que terme de régularisation supplémentaire dans [83]. L'architecture globale est illustrée à la figure 9. Plus précisément, le codeur est utilisé en tant que générateur et le discriminateur vise à distinguer si une représentation latente provient du générateur ou d'une distribution antérieure. De cette manière, l'auto-codeur est obligé de faire correspondre la distribution précédente en tant que régularisation. La fonction objectif est :

$$\min_{\mathbf{\Theta}} \mathcal{L}_2 + \alpha\mathcal{L}_{GAN}, \qquad (49)$$

où $\mathcal{L}_2$ est similaire à la perte de reconstruction définie dans les jeux de valeurs, et est $\mathcal{L}_{GAN}$

$$\min_{\mathcal{G}} \max_{\mathcal{D}} \mathbb{E}_{\mathbf{h} \sim p_{\mathbf{h}}} [\log \mathcal{D}(\mathbf{h})] + \mathbb{E} \left[\log \left(1 - \mathcal{D} \left(\mathcal{G}(\mathbf{F}^V, \mathbf{A}) \right) \right) \right], \tag{50}$$

où $\mathcal{G}(\mathbf{F}^V, \mathbf{A})$ est le codeur de convolution dans Eq. (45), D (·) est un discriminateur de la perte par entropie croisée et ph est la distribution antérieure. Dans cet article, un simple préalable gaussien est adopté et les résultats expérimentaux démontrent l'efficacité du système de formation contradictoire.

5.3.2 Apprentissage inductif et codeur GCN

Semblables aux GCN, les GAE peuvent être appliqués au réglage inductif si des attributs de nœud sont incorporés dans le codeur. Cela peut être réalisé en utilisant des GCNs comme encodeur, comme dans [78], [81], [83], ou en apprenant directement une fonction de mappage à partir de caractéristiques comme dans [80]. Les informations de bord n'étant utilisées que pour l'apprentissage des paramètres, le modèle peut être appliqué à des noeuds non visibles pendant l'entraînement. Ces travaux montrent également que, bien que les réseaux GCN et GAE reposent sur différentes architectures, il est possible de les utiliser conjointement, ce qui constitue, à notre avis, une orientation future prometteuse.

5.3.3 Mesures de similarité

Dans les GAE, de nombreuses mesures de similarité sont adoptées, telles que la perte de reconstruction L2, les cartes propres laplaciennes et la perte de classement pour les événements indésirables, ainsi que la divergence de KL et la distance de Wasserstein pour les VAE. Bien que ces mesures de similarité reposent sur des motivations différentes, la manière de choisir une mesure de similarité appropriée pour une tâche et une architecture données reste floue. Plus de recherche est nécessaire pour comprendre les différences sous-jacentes entre ces mesures

Conclusion :

Dans ce chapitre, nous avons examiné les différentes architectures de méthodes d'apprentissage basées sur des graphes profonds(GNNs,GCNs,GAEs), ainsi que leur distinction. Nous avons également mentionné les équations de base appliquées aux graphes dans chaque type de réseau de neurones (GNNs,GCNs,GAEs).

Chapitre IV

Réseaux de neurones convolutifs basés sur des graphes

Sommet quantique

1.Introduction :

Ce chapitre l'auteur propose un nouveau modèle de réseau de neurones à convolution de graphes spatiaux quantiques (QSGCNN) pouvant directement apprendre une fonction de classification pour les graphes de tailles arbitraires. Contrairement aux modèles à la pointe de la technologie GCNN (Graph Convolution Neural Network), le modèle QSGCNN proposé par l'auteur intègre le processus d'identification des sommets alignés de manière transitive entre les graphes et transforme les graphes de taille arbitraire en structures de grille de sommet alignées de taille fixe.

IV .2 Concepts préliminaires :

IV .2.1.Randonnées quantiques à temps continu :

L'un des principaux objectifs de l'auteur est de développer une nouvelle couche torsadée de graphes spatiaux pour extraire les entités de sommets à plusieurs échelles en propageant progressivement les informations relatives à chaque sommet aux sommets voisins, ainsi qu'au sommet lui-même. Cela nécessite généralement des informations de connexion entre chaque sommet et ses sommets voisins. La plupart des méthodes existantes utilisent la matrice de vertex de chaque graphe dans la formulation du cadre de propagation d'informations [100], [101], [102], [103]. Afin de saisir les caractéristiques de sommet les plus riches de la couche de convolution de graphes proposée, l'auteur propose dans ce travail d'utiliser le processus de propagation d'informations de sommet de la marche quantique en temps continu. C'est Analogique quantique de la marche aléatoire classique à temps continu [104].

La principale raison de s'appuyer sur les marches quantiques est que, contrairement aux marches aléatoires classiques, dont l'état est décrit par un vecteur à valeurs réelles et où l'évolution est régie par une matrice doublement stochastique, le vecteur d'état des marches quantiques est à valeurs complexes et son évolution est régie par une matrice unitaire temporelle. Ainsi, l'évolution de la marche quantique est réversible, ce qui implique qu'elle est ne possède pas de distribution limite. En conséquence, le comportement des marches quantiques est significativement différent de celui de leurs homologues classiques et possède un certain nombre de propriétés importantes, par exemple, il permet une interférence. Cette interférence, à son tour, aide à réduire le problème instable des marches aléatoires, comme un marcheur quantique faisant marche arrière sur un bord le fait avec une phase opposée. De plus, étant donné que l'évolution de la marche quantique n'est pas dominée par les composantes basse fréquence du spectre laplacien, elle a une meilleure capacité à distinguer différentes structures de graphes. Dans la section III, l'auteur montrerque la couche convolutionnelle de graphes associée au quantum en temps continu permet non seulement de réduire le problème de bascule apparaissant dans certains noyaux de graphes et modèles de réseaux convolutionnels de graphes à la pointe de la technologie, mais différentes structures de graphes.

Dans cette sous-section, L'auteur a brièvement étudié le concept d'étapes quantiques en temps continu. Plus précisément, il utilise la matrice de mélange moyenne pour capturer le comportement moyenné dans le temps de la marche quantique et pour mesurer les informations

$$\sum_{u\in V} \alpha_u(t)\alpha_u^*(t) = 1, \text{ et } \quad \alpha_u(t)\alpha_u^*(t) \in [0,1], \text{ for all } u \in V,\ t \in \mathbb{R}^+$$

quantiques transmises entre les sommets des graphes. La marche quantique en continu est un Similaire quantique de la marche aléatoire en continu [104], où cette dernière modélise un processus de diffusion markovien sur les sommets d'un graphe par le biais des transitions entre les sommets adjacents. Supposons qu'un échantillon de graphe soit noté G (V; E) avec l'ensemble de sommets V et l'ensemble de bords E. Comme pour la marche aléatoire classique, l'espace d'état de la marche quantique est l'ensemble de sommets V. Son état au temps *t* est une combinaison linéaire complexe des états de base sont l'amplitude et les deux complexes. En outre, indique la probabilité que le promeneur visite le sommet u au même moment t, Contrairement à la version classique, la marche quantique en temps continu évolue en fonction de l'équation de Schrodinger.

$$\partial/\partial t \, |\psi_t\rangle = -iH \, |\psi_t\rangle \,, \qquad (1)$$

où H représente le système hamiltonien. Dans ce travail, l'auteur utilise la matrice de communication comme hamiltonien. Le comportement d'une marche quantique sur le graphe G (V; E) à l'instant t peut être résumé à l'aide de la matrice de mélange [105]

$$Q_M(t) = U(t) \circ U(-t) = e^{iHt} \circ e^{-iHt}, \qquad (2)$$

où le symbole $Q_M(t)$ d'opération représente le produit e^{iHt} et e^{-iHt}. Hadamard de $Q_M(t)_{uv}$ Parce que U est unitaire, est une matrice doublement stochastique et chaque entrée indique la probabilité que la promenade se rende au sommet v au moment t où la marche commence initialement au sommet u. Cependant, M (t) ne peut pas converger, car U (t) préserve également les normes. Pour surmonter ce problème, l'auteur pouve imposer la convergence en prenant une moyenne temporelle. Plus précisément, $Q = \lim_{T\to\infty} \int_0^T Q_M(t)dt$, prenons la moyenne de Cesaro et définissons la matrice de mélange moyenne comme où chaque entrée *Qvivj* de la matrice de mélange moyenne Q représente la probabilité moyenne pour une marche quantique de visiter le sommet vj à partir du sommet vi, et Q est toujours une matrice doublement stochastique. De plus, Godsil [105] a indiqué que les entrées de Q sont des nombres rationnels. l'auteur pouve facilement calculer Q à partir du spectre de l'hamiltonien. Plus précisément, supposons que la matrice de communication A de G soit l'Hamiltonien H. Soit

$\lambda_1, \ldots, \lambda_{|V|}$ représenter le $|V|$ valeurs propres distinctes de H et Pj est la représentation matricielle de la projection orthogonale sur l'espace propre associé à la λ_j, i.e., $H = \sum_{i=1}^{|V|} \lambda_j \mathbb{P}_j$.

Ensuite, l'auteur pouve réécrire la matrice de mélange moyenne Q comme

$$Q = \sum_{j=1}^{|V|} \mathbb{P}_j \circ \mathbb{P}_j. \qquad (3)$$

IV .2. 2. Alignement transitif entre sommets de graphes

L'auteur introduise une nouvelle méthode d'alignement de vertex transitif. Pour ce faire, il commence par identifier une famille de prototypes

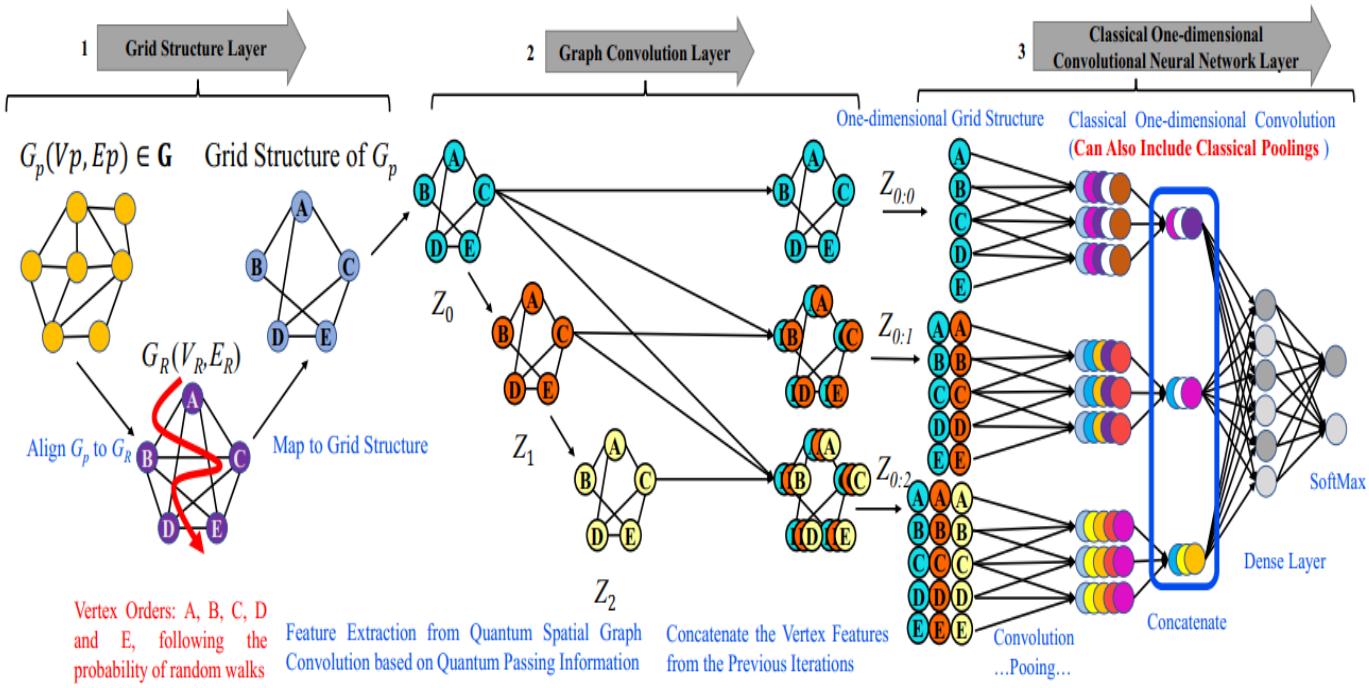


Fig. IV .1. L'architecture du modèle QSGCNN proposé.

représentations qui reflètent les caractéristiques principales des représentations vectorielles de sommets sur un ensemble de graphes G. Supposons qu'il y a n sommets sur tous les graphes de G et que les représentations vectorielles à K de dimension correspondante de ces sommets soient: $\mathbf{R}^K = (\mathrm{R}_1^K, \mathrm{R}_2^K, \ldots, \mathrm{R}_n^K)$. il utilise k-signifie [106] pour identifier M centrioles sur toutes les représentations en . Plus $\mathbf{R}^K$ précisément, étant donné M grappes $\boldsymbol{\Omega}$ = ($c1, c2, \ldots\ldots; c_M$), le but de k-signifie est de minimiser la fonction objectif

$$\arg\min_{\Omega} \sum_{i=1}^{M} \sum_{\mathrm{R}_j^K \in c_i^K} \|\mathrm{R}_j^K - \mu_i^K\|^2, \qquad (4)$$

où i μ_i^K est la moyenne des $\mathbf{R}^K$ représentations vectorielles de vertex appartenant au i-ème groupe ci. Puisque l'équation (4) minimise la somme des distances euclidiennes carrées entre les points R_j^K sommets et Le point médian la c_i^K, grappe Les points médians $\{\mu_1^K, \cdots, \mu_i^K, \cdots, \mu_M^K\}$ M peut être vu comme une famille de représentations prototypes de dimension K qui encapsulent des caractéristiques représentatives sur tous les graphes de G. Laisser $\mathbf{G} = \{G_1, \cdots, G_p, \cdots, G_q, \cdots, G_N\}$ graphes. Pour chaque graphe Gp (Vp; Ep) ∈ G et chaque sommet vi ∈Vp associé à sa représentation vectorielle K-dimensionnelle on $\mathrm{R}_{p;i}^K$, $\mathbf{PR}^K = \{\mu_1^K, \ldots, \mu_j^K, \ldots, \mu_M^K\}$ commence par identifier l'ensemble des représentations prototypes K-dimensionnelles . l'auteur utilise k-signifie [106] pour identifier M Le point médian sur toutes les représentations en . Plus précisément, étant donné M grappes $\boldsymbol{\Omega}$ = ($c1, c2, \ldots\ldots; c_M$), le but de k-signifie est de minimiser la fonction objectif

$$A_p^K(i,j) = \|\mathrm{DB}_{p;i}^K - \mu_j^K\|_2. \qquad (5)$$

et chaque élément représente la $R_p^K(i,j)$ distance entre la représentation vectorielle de v∈Vp et la $R_{p;i}^K$ représentation du prototype Si la valeur de $\mu_j^K \in \mathbf{PR}^K$. est le plus petit de $A_p^K(i,j)$ la rangée i, on dit que est aligné sur $R_{p;i}^K$, c'est-à-dire μ_j^K, que le sommet vi est aligné sur le j-ème prototype représentation. Notez que pour chaque graphe, deux sommets ou plus peuvent être alignés sur la même représentation prototype. l'auteur enregistre les informations de correspondance à l'aide de la matrice de correspondance de niveau K

$$C_p^K(i,j) = \begin{cases} 1 \text{ si } A_p^K(i,j) \text{ est le plus petit de la rangée i} \\ 0 \text{ autrement} \end{cases}$$

Pour une paire de graphes Gp et Gq, si leurs sommets vp et vq sont alignés sur la même représentation prototype PR_j^K, l'auteur dise que vp et vq sont également alignés. Ainsi, il pouves identifier les informations d'alignement transitif entre les sommets de tous les graphes de G, en faisant correspondre leurs sommets à un ensemble commun de points de référence, c'est-à-dire les représentations prototypes.

Discussion : Le processus d'alignement illustré par l'équation (5) et l'équation (6) peut s'expliquer par la fonction objectif des k-signifie définie par l'équation (4). En effet, identifier le plus petit $A_p^K(i,j)$ dans la rangée de A_p^K élément équivaut à attribuer la représentation $\mathrm{R}_{p;i}^K$ de $v_i \in V_p$ au cluster c_i^K dont le vecteur moyen est μ_i^K. vectorielle En conséquence, la procédure d'alignement proposée peut être vue comme un processus d'optimisation qui minimise progressivement la somme des carrés de groupes de sommets intérieurs sur les sommets de tous les graphes et peut établir des informations de correspondance de sommets fiables sur tous les graphes.

IV .3. Réseau de neurones convolutifs à graphes spatiaux quantiques

Dans cette section, l'auteur développe un nouveau modèle de réseau de neurones convolutifs à graphes spatiaux quantiques (QSGCNN). L'architecture du modèle proposé est illustrée à la Fig.1. Plus précisément, l'architecture est composée de trois étapes séquentielles, c'est-à-dire 1) la construction de la structure de grille et la couche d'entrée, 2) la couche de convolution du graphe spatial quantique et 3) le réseau de neurones à convolution traditionnel et les couches Softmax. Plus précisément, la construction de la structure de grille et la couche d'entrée a) d'abord mappe des graphes de tailles arbitraires en structures de grille de taille fixe avec des ordres de sommets cohérents, et b) insère les structures de grille dans le modèle QSGCNN proposé. Avec les structures de grille de graphe d'entrée à portée de main, la couche de convolution de graphe spatial quantique extrait en outre des entités de sommet multi-échelles en propageant des informations sur les entités de sommet entre les sommets alignés de la grille.

Puisque les entités de sommet extraites de la couche de convolution de graphe préservent les ordres de sommet originaux des structures de grille, le réseau de neurones à convolution traditionnel et la couche Softmax peuvent lire les entités de sommet extraites et prédire la classe de graphe.

IV .3.1. Alignement des structures de graphes sur la grille de sommets

Dans cette sous-section, l'auteur montre comment mapper des graphes de différentes tailles sur des structures de grille de vertex alignées de taille fixe et sur les matrices de adjacence de vertex de grille alignées de taille fixe correspondantes. Pour l'ensemble des graphes G définis précédemment, supposons que G_p (V_p; E_p; A_p) ∈ G est un exemple de graphe, avec V_p représentant l'ensemble des sommets, E_p représentant l'ensemble des arêtes et A_p représentant la matrice d'adjacence des sommets. Supposons que chaque sommet $v_p \in V_p$ soit représenté sous la forme d'un vecteur de caractéristique c-dimensionnelle. Ensuite, les caractéristiques de tous les n sommets peuvent être codées à l'aide de la matrice n × c, c'est-à-dire, $X_p \in \mathbb{R}^{n \times c}$.

Notez que la ligne de X_p suit le même ordre de sommet de A_p. Si les graphes de G sont des graphes à attribution de vertex, X_p peut être la matrice de codage one-hot des étiquettes de vertex. Pour les graphes non attribués, l'auteur propose d'utiliser le degré de sommet comme étiquette de sommet. Sur la base de la méthode d'alignement de vertex transitive introduite à la section II, pour chaque graphe $G_p \in G$, il commence par calculer la matrice de correspondance de sommet de niveau K, qui enregistre les C_p^K informations de correspondance entre la représentation de sommet vectoriel de dimension K de G_p et les représentations de prototype de dimension K dans de G. La ligne et la colonne $\mathbf{PR}^K = \{\mu_1^K, \ldots, \mu_i^K, \ldots, \mu_M^K\}$ de sont indexées par les C_p^K sommets dans V_p et les représentations de prototype dans $\mathbf{PR}^K$., respectivement. Avec , l'auteur calcule la C_p^K matrice de caractéristiques de sommet alignée de niveau K pour G_p comme

$$\widehat{X}_p^K = (C_p^K)^T X_p, \qquad (7)$$

où $\widehat{X}_p^K$ est une matrice M × c et chaque ligne de $\widehat{X}_p^K$ représente la caractéristique d'un sommet aligné correspondant. De plus, l'auteur calcule également la matrice de communication des sommets alignés associée de niveau K pour G_p comme

$$\widehat{A}_p^K = (C_p^K)^T (A_p)(C_p^K), \qquad (8)$$

où $\widehat{A}_p^K$ est une matrice M × M. Avec la matrice de correspondance C_p^K à part, $\widehat{X}_p^K$ et C_p^K sont calculés à partir de la matrice des caractéristiques de sommet et de la matrice de adjacence, $\mathbf{PR}^K$ respectivement, en $\widehat{X}_p^K$ mappant les informations de caractéristique et de adjacence d'origine de chaque sommet $v_p \in V_p$ à celle des nouveaux sommets alignés indexés par les prototypes correspondants . En d'autres termes, et $\widehat{A}_p^K$ encapsulent la caractéristique d'origine et les informations structurelles de G_p. Notez également que selon Eq. 6 chaque sommet $v_p \in V_p$ peut être aligné sur plus d'un prototype, et donc en général, est une matrice pondérée d'adjacence .

Afin de construire la structure de grille alignée de taille fixe pour chaque graphe $G_p \in$ G, l'auteur deve établir un ordre cohérent pour les sommets des graphes de G. Puisque les sommets de tous les graphes sont alignés sur les mêmes représentations de prototype, l'auteur détermine les ordres de vertex en réordonnant les représentations du prototype. À cette fin, il construise un prototype de graphe qui capture la similarité par paires entre les représentations du prototype. Compte tenu de ce graphe, une approche pourrait consister à trier les représentations du prototype en fonction de leur $\mu_j^K \in \mathbf{PR}^K \in \mathbf{PR}\mu_k^K \in \mathbf{PR}^K$ degré. Cela équivaudrait à

trier les prototypes par ordre de similitude moyenne avec les prototypes restants. Plus précisément, il calcule le graphe prototype G_R (V_R; E_R) qui caractérise les informations de relation entre les représentations prototypes à K dimensions, en , chaque sommet $v_j \in V_R$ représentant la représentation prototype et chaque bord $(v_j; v_k) \in E_R$ représentant la similarité entre et La similarité entre deux sommets de G_R est calculée comme suit:

$$s(\mu_j^K, \mu_k^K) = \exp(-\frac{\|\mu_j^K - \mu_k^K\|_2}{K}). \qquad (9)$$

Le degré de chaque représentation prototype μ_j^K is $D_R(\mu_j^K) = \sum_{k=1}^{M} s(\mu_j^K, \mu_k^K)$. . l'auteur trie les représentations prototypes K-dimensionnelles en $\mathbf{PR}^K$ en fonction de leur degré $D_R(\mu_j^K)$. .Ensuite, l'auteur réorganise $\tilde{X}_p^K$ et $\hat{A}_p^K$ en conséquence.

Enfin, pour construire des structures de grille fiables pour les graphes, il utilise dans ce travail les représentations basées sur la profondeur en tant que représentations vectorielles de vertex pour calculer la matrice de correspondance de vertex requise au niveau K,. Plus précisément, la représentation en C_p^K fonction de la profondeur de chaque sommet est calculée en mesurant les entropies sur une famille de sous-graphes d'expansion de k-couches enracinées au sommet [107], où le paramètre k varie de 1 à K. En outre, il a été montré que une telle représentation d'un sommet en K de dimension peut être vue comme une représentation de sommet imbriquée qui encapsule un flux de contenu d'information riche en entropie imbriqué de chaque sommet local à la structure graphe globale [107], en fonction de la profondeur. Le processus de calcul de la matrice de correspondance associée aux représentations en profondeur est illustré à la C_p^K Fig.2. Lorsque l'auteur faise varier la plus grande couche K des sous-graphes d'expansion de 1 à L (c'est-à-dire, $K \leq L$), il calcule la structure finale de la grille de sommets alignés pour chaque graphe $G_p \in G$ comme suit :

$$\hat{X}_p = \sum_{K=1}^{L} \frac{\hat{X}_p^K}{L}, \qquad (10)$$

et la matrice d'adjacence de vertex de grille alignée associée comme

$$\hat{A}_p = \sum_{K=1}^{L} \frac{\hat{A}_p^K}{L}, \qquad (11)$$

où $\hat{X}_p$ est une matrice M × c et $\hat{A}_p$ est une matrice M × M.

Discussion : Les équations (10) et (11) transforment les graphes d'origine $G_p \in G$ avec un nombre variable de nœuds $|V_p|$ en une nouvelle structure de graphe alignée avec le même nombre de sommets, où est l'entité correspondante $\hat{X}_p$ du sommet de la grille alignée. matrix et est la matrice d'adjacence du $\hat{A}_p$ sommet de la grille alignée correspondante. Etant donné que pour tout graphe $G_p \in G$, les lignes de sont systématiquement indexées par les $\hat{X}_p$ mêmes représentations prototypes, la structure de grille de sommets de taille fixe peut être directement utilisée comme entrée d'un réseau de neurones à $\hat{X}_p$ convolution traditionnel. En d'autres termes, on peut appliquer un filtre convolutionnel classique de taille fixe à glissez sur les lignes $\hat{A}_p$ $\hat{X}_p$ de et apprenez la fonctionnalité de $\hat{X}_p$ $G_p \in G$. Enfin, notez que et

encapsulent avec précision la fonctionnalité d'origine et les informations structurelles de Gp, respectivement.

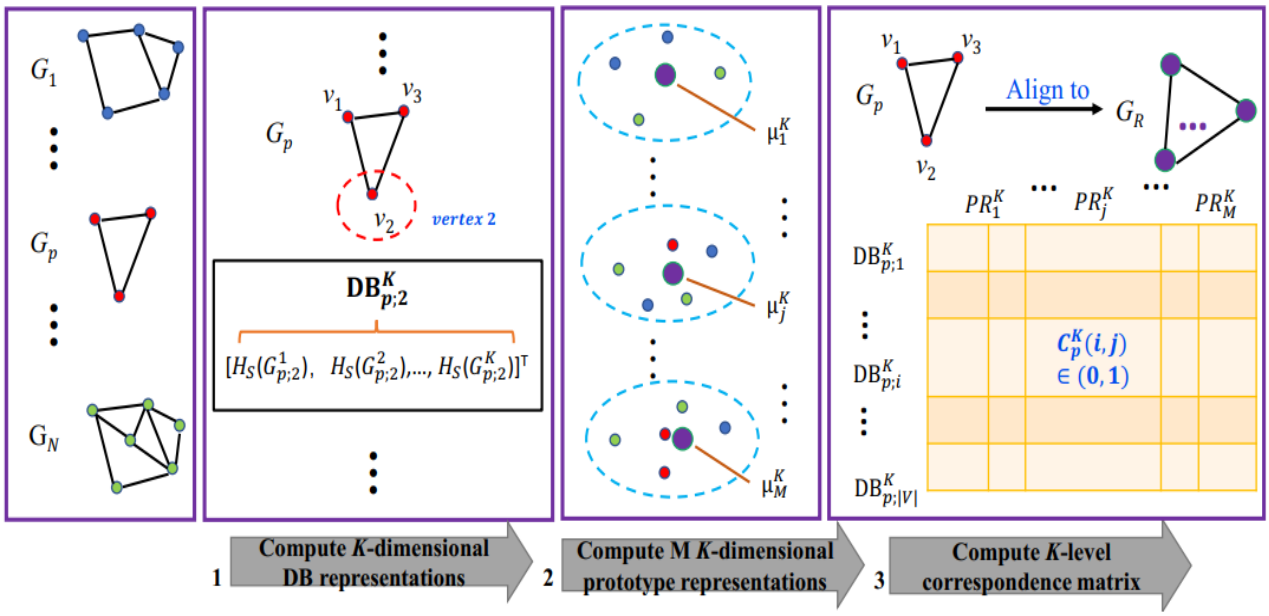


Fig. IV .2. Procédure de calcul de la matrice de correspondance

IV .3.2. La couche de convolution de graphes spatiaux quantiques :

Dans cette sous-section, l'auteur propose une nouvelle couche de convolution de graphes spatiaux quantiques pour extraire davantage les $\hat{X}_p \in \mathbb{R}^{M \times c}$ caractéristiques des $\hat{A}_p \in \mathbb{R}^{M \times M}$ sommets de chaque graphe. Ceci est défini par la propagation d'informations quantiques entre les sommets alignés de la grille. À cette fin, il utilise la matrice de mélange moyenne de la marche quantique en temps continu sur la matrice adjacente de vertex de grille associée associée. Pour l'exemple de graphe Gp (Vp; Ep), l'auteur passe la structure de grille de sommet alignée et la matrice adjacente de vertex de grille de grille associée de Gp en tant qu'entrée de la couche de convolution de graphe spatial quantique. La couche de convolution de graphes spatiaux proposée prend la forme suivante, à savoir:

$$Z = \mathrm{Relu}(Q\hat{X}_p W), \qquad (12)$$

où Relu est la fonction des unités linéaires rectifiées (c'est-à-dire une fonction d'activation non linéaire), Q est la matrice de mélange moyenne de la marche quantique en temps continu sur de Gp définie dans la section II-A, est $\hat{A}_p$ la matrice des paramètres $W \in \mathbb{R}^{c \times c}$ pouvant être entraînés de la couche convolutionnelle de graphe proposée, et est la matrice d'activation de sortie $Z \in \mathbb{R}^{M \times c}$.

La couche de convolution de graphe spatial quantique proposée définie par Eq (12) comprend trois étapes. Dans la première étape, l'opération est appliquée $\hat{X}_p W$ pour transformer la matrice d'informations de sommet de grille alignée en une nouvelle matrice d'informations de sommet de grille alignée. À son tour, cela mappe les caractéristiques c-dimensionnelles de

chaque sommet de grille aligné en nouvelles caractéristiques c, c.-à-d. mappe les canaux de caractéristique c en c0 canaux dans la couche suivante. Les poids W sont partagés entre tous les sommets de la grille alignés. La deuxième étape calcule QY, où . Cela propage les informations de caractéristiques de chaque sommet de grille aligné vers les sommets restants, ainsi que le sommet lui-même, en termes d'informations de visite de sommet de balades quantiques.

Notons plus précisément que Qij encapsule la probabilité moyenne pour une promenade quantique en temps continu de visiter le j-ème sommet de la grille aligné à partir du i-ème sommet de la grille aligné, et Ici, je peux être égal $(Q\hat{X}_p')_i = \sum_j Q_{ij} Y_j$. à j, c'est-à-dire que Q inclut les informations de boucle automatique pour chaque sommet. Ainsi, la ième ligne de la matrice résultante de QXbp0 est la somme des caractéristiques du i-ème sommet de la grille aligné et des autres sommets de la grille alignés associés à la probabilité de visite moyenne des marches quantiques allant du i-ème sommet aux sommets restants. comme le i-ème sommet lui-même. La dernière étape applique la fonction d'unité linéaire rectifiée à et génère le résultat de la convolution du graphe $Q\hat{X}_p'$.

La convolution de graphe spatial quantique proposée propage les informations de sommet de grille alignées en termes d'informations de visite de sommet associées à la marche quantique en temps continu entre les sommets. Pour extraire davantage les entités multi-échelles des sommets de la grille alignés, l'auteur empile plusieurs couches de convolution de graphes définies par l'équation (12) comme suit:

$$Z_{t+1} = \mathrm{Relu}(QZ_tW_t), \qquad (13)$$

où Z0 est la structure de grille de sommets alignée en entrée est la $\hat{X}_p$, $Z_t \in \mathbb{R}^{M \times c_t}$ sortie de la deuxième couche de convolution de graphe spatial et $W_t \in \mathbb{R}^{c_t \times c_{t+1}}$ est la matrice de paramètres pouvant être entraînée mappant les canaux ct en canaux c_{t+1}

Après chaque Couche convolutionnelle de graphe spatial quantique, l'auteur ajoute également une couche pour concaténer horizontalement la sortie Z^t associée aux sorties des couches convolutionnelle de graphes spatiaux précédentes de 1 à t - 1, ainsi que l'entrée d'origine en tant que Z0: t, c'est-à- Z^0 dire En conséquence, pour la $Z_{0:t} = [Z_0, Z_1, \ldots, Z^t]$ sortie $Z_{0:t} \in \mathbb{R}^{M \times \sum_{i=0}^{t} c_i}$. concaténée Z0:t, chacune de ses lignes peut être considérée comme la nouvelle entité multi-échelle pour le sommet de la grille correspondant.

Discussion: Notez que la convolution de graphe spatial quantique proposée extrait uniquement les nouvelles fonctionnalités du sommet de la grille et ne modifie pas les ordres des sommets. En conséquence, la sortie et la sortie concaténée Z^t Z0:t préservent la propriété de structure de grille de l'entrée d'origine et peuvent être directement $Z_0 = \hat{X}_p$ utilisé comme entrée du réseau neuronal convolutionnelle traditionnel. Ceci fournit un moyen élégant de combler le fossé entre la couche de convolution de graphe spatial quantique proposée et le réseau de neurones de convolution traditionnel, créant ainsi une architecture d'apprentissage en profondeur intégrant la représentation du graphe et l'apprentissage dans la couche de convolution de graphe spatial quantique et la couche de convolution traditionnelle pour les problèmes de classification des graphes.

IV .3.3. Les couches de réseau de neurones convolutionnels traditionnels :

Après les couches de convolution de graphe spatial quantique proposées, l'auteur obtene une structure de grille de sommets concaténée $Z_{0:t} \in \mathbb{R}^{M \times \sum_{z=0}^{t} c_z}$. où chaque ligne de $Z_{0:t}$ représente la caractéristique à plusieurs échelles pour un sommet de grille correspondant. Comme il a mentionné ci-dessus, chaque structure de grille $Z_{0:t}$ peut être directement utilisée comme entrée du réseau de neurones à convolution (CNN) traditionnel. Spécifiquement, la partie CNN unidimensionnelle classique de la Fig.1 présente l'architecture des couches CNN traditionnelles associées à chaque $Z_{0:t}$. Ici, chaque structure de grille de sommets concaténée $Z_{0:t}$ est vue sous la forme d'une structure de grille de vertex M × 1 (sur la figure 1 = 5) et chaque sommet est représenté par une caractéristique dimensionnelle de c'est-à-dire, le canal de chaque grille le sommet est Ensuite, l'auteur ajoute une couche convolutionnelle unidimensionnelle. L'opération de $\sum_{z=0}^{t} c_z$ convolution peut être effectuée en faisant glisser un filtre $\sum_{z=0}^{t} c_z$. de taille fixe de taille k × 1 (sur la figure 1, k = 3) sur les sommets voisins dans l'espace. Après cela, plusieurs couches MaxPooling et les couches convolutives unidimensionnelles restantes peuvent être ajoutées pour apprendre les modèles locaux de la séquence de sommets de grille alignée. Enfin, lorsque l'auteur faise varier t de 0 à T (sur la Fig.1, T = 2), il obtiendre des représentations de motif extraites T +1. il concaténe les motifs extraits de chaque $Z_{0:t}$ et ajoutons un calque entièrement connecté suivi d'un calque Softmax.

IV .3.4. Discussion sur le modèle QSGCNN proposé :

Le modèle QSGCNN proposé est associé à des modèles de réseau de convolution de graphes et à des noyaux de graphes à la pointe de la technologie. Cependant, il existe un certain nombre de différences théoriques significatives entre le modèle QSGCNN proposé et ces méthodes de pointe, expliquant l'efficacité du modèle proposé. Dans cette sous-section, l'auteur discute des relations entre ces méthodes et démontrons les avantages du modèle proposé.

Premièrement, de manière similaire à la convolution de graphes spatiaux quantiques du modèle QSGCNN proposé, à la convolution de graphes associée du réseau de neurones convolutionnels à graphes profonds (DGCNN) [103] et à la convolution de graphes spectraux du réseau de neurones convolutionnels à graphes approximatifs rapides (FAGCNN) [108] propage également les entités entre les sommets du graphe. Plus précisément, les convolutions de graphe des modèles DGCNN et FAGCNN utilisent la matrice de communication de graphes ou la matrice laplacienne normalisée pour déterminer comment transmettre les informations entre les sommets. En revanche, notre convolution de graphe spatial quantique utilise la matrice de mélange moyenne de la marche quantique en temps continu associée au graphe. Comme il a mentionné à la section II-A, la marche quantique n'est pas dominée par les valeurs de basse fréquence du spectre laplacien et offre donc une meilleure capacité à distinguer différentes structures de graphes. En conséquence, la méthode proposée peut extraire davantage de caractéristiques de sommet discriminantes.

Deuxièmement, afin de maintenir l'échelle des entités de sommet après chaque couche de convolution de graphe, la convolution de graphe du modèle DGCNN [103] et la convolution de graphe spectrale du modèle FAGCNN [108] doivent permettre une multiplication par l'inverse du matrice de degré de sommet. Par exemple, la couche de convolution de graphes du modèle DGCNN associée à un graphe à n sommets est

$$Z = f(\tilde{D}^{-1}\tilde{A}XW), \qquad (14)$$

où $\tilde{A} = A + I$ est la matrice d'adjacence du graphe avec les boucles propres ajoutées, est la $\tilde{D}$ matrice de degrés de $\tilde{A}$. $X^{n\times c}$ est la matrice de caractéristiques de sommet, chaque rangée représentant les caractéristiques de dimension c d'un sommet, $W^{c\times c'}$ est la matrice de paramètres pouvant être entraînés, f est une fonction d'activation non linéaire (par exemple, la fonction Relu) et $Z^{n\times c'}$ est la sortie. D'une manière similaire à la convolution de graphe spatial quantique proposée définie dans l'équation (12), XW mappe les caractéristiques de dimension c de chaque sommet en un ensemble de nouvelles caractéristiques de dimension c0. De plus, propage les informations de caractéristiques de chaque sommet sur les sommets voisins, ainsi $\tilde{A}Y$ $(Y := \hat{X}_p W)$ que sur le sommet lui-même. La i-ème ligne i de la matrice résultante e représente les caractéristiques extraites du i-ème sommet et correspond (AY) à la somme de Y_i lui- (AY) même et de Y_j à partir des sommets du i-ème sommet. et correspond à la somme de Y_i lui-même et de Y_j à partir des sommets voisins du i-ème sommet. Multiplier par l'inverse de peut être considéré comme le $\tilde{D}$ (i.e., $\tilde{D}^{-1}$) processus de normalisation et d'attribution de poids égaux entre le i-ème sommet et chacun de ses voisins. En d'autres termes, la convolution de graphe du modèle DGCNN considère les influences mutuelles entre les sommets spécifiés pour l'opération de convolution comme identiques. En revanche, la convolution de graphe spatial quantique du modèle QSGCNN proposé défini dans l'équation (12) attribue une distribution de probabilité de visite moyenne quantifiée à des sommets spécifiés, chaque sommet ayant une probabilité de visite différente du poids. Par conséquent, l'entité de sommet extraite est la somme pondérée des entités de sommet spécifiées. En conséquence, la convolution de graphe spatial quantique du modèle QSGCNN proposé maintient non seulement l'échelle de la caractéristique, mais discrimine également les influences mutuelles entre les sommets spécifiés en fonction des différentes probabilités de visite lors de la convolution.

Troisièmement, comme dans le modèle QSGCNN proposé, le modèle de réseau de neurones à convection graphe (PSGCNN) basé sur PATCHY-SAN [102] et le modèle DGCNN [103] doivent réorganiser l'ordre des sommets de chaque structure de graphe et transformer chaque graphe en diagramme fixe. taille de grille de vertex Spécifiquement, le modèle PSGCNN forme d'abord les structures de grille, puis exécute le CNN classique standard sur les structures de grille. Le modèle DGCNN trie les sommets à travers un SortPooling associé aux entités de sommet extraites de plusieurs couches de convolution de graphes spatiaux. Malheureusement, les modèles PSGCNN et DGCNN trient les sommets de chaque graphe en fonction du descripteur de structure local, en ignorant les informations de correspondance de sommet cohérentes entre différents graphes. En revanche, le modèle QSGCNN proposé s'associe à une procédure d'alignement de vertex transitive pour transformer chaque graphe en une grille de vertex de taille fixe alignée. structure. De ce fait, seul le modèle QSGCNN proposé peut intégrer les informations de correspondance structurelle précises sur tous les graphes en cours d'investigation.

Quatrièmement, lorsque le modèle PSGCNN [102] et le modèle DGCNN [103] forment des structures de grille de vertex de taille fixe, certains vertices de rang inférieur sont ignorés. De plus, le réseau d'empreintes digitales de graphes (NGFN) [100] et le réseau de neurones de convolution par diffusion (DCNN) [101] tendent à capturer les caractéristiques des graphes au

niveau mondial en faisant la somme des entités de sommets au niveau local extraites au moyen d'un SumPooling. couche, étant donné que le modèle NGFN et le modèle DCNN ne peuvent pas directement former de structures de grille de vertex. Cela entraîne une perte d'informations importante pour les entités de sommet au niveau local. En revanche, les structures de grille de vertex alignées requises et la Les matrices de communication des sommets associés au modèle QSGCNN proposé peuvent encapsuler avec précision à la fois les entités de sommet originales et les informations de structure topologique des graphes d'origine. En conséquence, le QSGCNN proposé résout les problèmes de perte d'informations apparaissant dans les modèles de réseau de neurones à convolution de graphes de pointe mentionnés ci-dessus.

Cinquièmement, comme dans le modèle DGCNN [103], la convolution de graphes spatiaux quantiques du modèle QSGCNN proposé est également liée au noyau de sous-arbre Weisfeiler-Lehman (WLSK) . Le noyau WLSK utilise plus précisément le noyau classique de Weisfeiler-Lehman (WL).) algorithme en tant que méthode de marquage canonique pour extraire les entités de sommet multi-échelles correspondant aux sous-arbres pour la classification des graphes. L'idée principale de la méthode WL est de concaténer une étiquette de sommet avec les étiquettes de ses sommets voisins, puis de trier l'étiquette concaténée lexicographiquement pour attribuer une nouvelle étiquette à chaque sommet. La procédure se répète jusqu'à une itération maximale h, et chaque libellé de sommet à une itération h correspond à un sous-arbre de hauteur t enraciné au sommet. Si les libellés concaténés de deux sommets sont identiques, les sous-arbres enracinés au niveau des deux sommets sont isomorphes, c'est-à-dire que les deux sommets partagent les mêmes caractéristiques structurelles dans le graphe. Le noyau WLSK utilise cette idée pour mesurer la similarité entre deux graphes. Il utilise la méthode WL pour mettre à jour les étiquettes de sommet, puis compte le nombre d'étiquettes de sommet identiques (c'est-à-dire en comptant le nombre maximum de sous-arbres isomorphes) jusqu'au maximum de l'itération h afin de comparer deux graphes à plusieurs échelles. Pour montrer la relation entre la convolution de graphe spatial quantique proposée définie dans l'équation (12) et le noyau WLSK, l'auteur décompose l'équation (12) de manière rangée, c'est-à-dire

$$Z_i = \mathrm{Relu}(Q_i Y) = \mathrm{Relu}(Q_{ii} Y_i + \sum_j Q_{ij}), \qquad (15)$$

Où $Y = \tilde{X}_p W$. Pour l'équation (15), Y_i peut être vu comme le libellé de sommet vectoriel à valeurs continues du ième sommet. De plus, $Q_{ii}Y_i + \sum_j Q_{ij}$ si Qij> la $Q_{ii}Y_i + \sum_j Q_{ij}$ marche quantique à partir du i-ème sommet peut visiter le j-ième sommet et la probabilité de visite est Q_{ij}. D'une manière similaire aux méthodes WL, l'équation (15) agrège l'étiquette continue Yi du ième sommet et les étiquettes continues Y_j des sommets, qui peuvent être visitées par la marche quantique à partir du ième sommet, en tant que nouveau vecteur de signature pour le i-ème sommet. La fonction Relu associe à une nouvelle étiquette vectorielle continue En conséquence, la convolution du graphe spatial quantique du modèle QSGCNN proposé peut être vue comme une version quantique de l'algorithme WL, en termes de propagation des informations de sommet quantique formulées par la marche quantique. Comme l'auteur a mentionné à la section II-A, la marche quantique peut réduire considérablement l'effet du problème de chancelement. D'autre part, la méthode classique WL souffre également d'un problème d'instabilité . En conséquence, la convolution de graphes spatiaux quantiques peut résoudre le problème de la méthode classique WL, qui est un problème de basculement, et

la convolution de graphes du modèle DGCNN est similaire à la méthode classique WL. En d'autres termes, la convolution de graphes spatiaux quantiques du modèle QSGCNN proposé permet d'apprendre de meilleures caractéristiques de vertex de graphes.

Enfin, notez que le modèle QSGCNN proposé pour chaque graphe est invariant par rapport à la permutation des sommets, indiquant que les activations d'une paire de graphes isomorphes seront les mêmes. Comme l'auteur a mentionné, le modèle QSGCNN proposé comprend trois étapes: a) la construction de la structure de grille et la couche d'entrée, b) la couche de convolution du graphe spatial quantique et c) la couche CNN traditionnelle. Pour la première couche, la construction des structures de grille repose sur les entités de vertex et la matrice d'adjacence et est invariante pour les permutations de vertex. En conséquence, les structures de grille pour une paire de graphes isomorphes sont les mêmes. Pour la deuxième couche, les structures de grille d'entrée de différents graphes partagent les mêmes pondérations de paramètres. Ainsi, les convolutions de graphes spatiaux quantiques produiront les mêmes entités de sommet extraites pour une paire de graphes isomorphes associés aux mêmes structures de grille. Par conséquent, la couche CNN classique ultérieure identifiera correctement les graphes isomorphes. En conséquence, le modèle QSGCNN proposé peut identifier correctement les paires de graphes isomorphes. Ces observations révèlent les avantages du modèle QSGCNN proposé, expliquant l'efficacité du modèle proposé. Le modèle QSGCNN proposé non seulement pallie les lacunes des méthodes de pointe existantes, mais permet également de combler les lacunes théoriques et informatiques entre ces méthodes.

IV .4 Exprériences :

Dans cette section, l'auteur compare empiriquement les performances du modèle QSGCNN proposé aux noyaux de graphes à la pointe de la technologie et aux méthodes d'apprentissage approfondi sur les problèmes de classification des graphes.

IV .4.1. Comparaisons avec les noyaux de graphes :

Jeux de données : Dans cette sous-section, l'auteur utilise neuf jeux de données graphes standard issus de la bioinformatique [109], [110], [111], [112] et des réseaux sociaux [113] pour évaluer les performances du modèle QSGCNN proposé. Ces jeux de données incluent MUTAG, PTC, NCI1, PROTÉINES, D & D, COLLAB, IMDB-B, IMDB-M et RED-B. Une sélection de statistiques de ces jeux de données est présentée dans le tableau.I.

Configuration expérimentale : l'auteur évalue les performances du modèle QSGCNN proposé en ce qui concerne les problèmes de classification des graphes par rapport à cinq noyaux de graphes alternatifs ultramodernes. Ces noyaux de graphes incluent 1) le noyau de q-difference de Jensen-Tsallis

TABLEAU IV .1 : INFORMATIONS SUR LES FICHIERS DE DONNÉES DE GRAPHE

Datasets	MUTAG	NCI1	PROTEINS	D&D	PTC(MR)	COLLAB	IMDB-B	IMDB-M	RED-B
Max # vertices	28	111	620	5748	109	492	136	89	3783
Mean # vertices	17.93	29.87	39.06	284.30	25.60	74.49	19.77	13.00	429.61
Mean # edges	19.79	32.30	72.82	715.65	14.69	4914.99	193.06	131.87	497.80
# graphs	188	4110	1113	1178	344	5000	1000	1500	2000
# vertex labels	7	37	61	82	19	–	–	–	–
# classes	2	2	2	2	2	3	2	3	2
Description	Chemical	Chemical	Chemical	Chemical	Chemical	Social	Social	Social	Social

TABLEAU IV .2 : PRÉCISION DU CLASSEMENT (EN% ± ERREUR STANDARD) POUR LES COMPARAISONS AVEC LES NOYAUX GRAPHES.

Datasets	MUTAG	NCI1	PROTEINS	D&D	PTC(MR)	COLLAB	IBDM-B	IBDM-M	RED-B
QSGCNN	**90.52** ± 0.95	77.50 ± 0.91	**75.90** ± 0.79	**81.70** ± 0.92	**63.37** ± 1.15	78.80 ± 0.89	**73.62** ± 1.12	**51.60** ± 1.15	**91.50** ± 0.24
JTQK	85.50 ± 0.55	**85.32** ± 0.14	72.86 ± 0.41	79.89 ± 0.32	58.50 ± 0.39	76.85 ± 0.40	72.45 ± 0.81	50.33 ± 0.49	77.60 ± 0.35
WLSK	82.88 ± 0.57	84.77 ± 0.13	73.52 ± 0.43	79.78 ± 0.36	58.26 ± 0.47	77.39 ± 0.35	71.88 ± 0.77	49.50 ± 0.49	76.56 ± 0.30
WL-OA	82.88 ± 0.57	84.77 ± 0.13	73.52 ± 0.43	79.78 ± 0.36	58.26 ± 0.47	**80.70** ± 0.10	71.88 ± 0.77	49.50 ± 0.49	76.56 ± 0.30
SPGK	83.38 ± 0.81	74.21 ± 0.30	75.10 ± 0.50	78.45 ± 0.26	55.52 ± 0.46	58.80 ± 0.2	71.26 ± 1.04	51.33 ± 0.57	84.20 ± 0.70
CORE SP	88.29 ± 1.55	73.46 ± 0.32	–	77.30 ± 0.80	59.06 ± 0.93	–	72.62 ± 0.59	49.43 ± 0.42	90.84 ± 0.14
PIGK	76.00 ± 2.69	82.54 ± 0.47	73.68 ± 0.69	78.25 ± 0.51	59.50 ± 2.44	–	–	–	–
GK	81.66 ± 2.11	62.28 ± 0.29	71.67 ± 0.55	78.45 ± 0.26	52.26 ± 1.41	72.83 ± 0.28	65.87 ± 0.98	45.42 ± 0.87	77.34 ± 0.18
RWGK	80.77 ± 0.72	63.34 ± 0.27	74.20 ± 0.40	71.70 ± 0.47	55.91 ± 0.37	–	67.94 ± 0.77	46.72 ± 0.30	–

TABLEAU IV .3 : EXACTITUDE DU CLASSEMENT (EN% ± ERREUR STANDARD) POUR LES COMPARAISONS AVEC DES RÉSEAUX DE NEURON GRAPHE CONVOLUTIONNEL

Datasets	MUTAG	NCI1	PROTEINS	D&D	PTC(MR)	COLLAB	IBDM-B	IMDB-M	RED-B
QSGCNN	**90.52** ± 0.95	77.50 ± 0.91	75.90 ± 0.79	**81.70** ± 0.92	63.37 ± 1.15	**78.80** ± 0.89	**73.62** ± 1.12	**51.60** ± 1.15	**91.50** ± 0.24
DGCNN	85.83 ± 1.66	74.44 ± 0.47	75.54 ± 0.94	9.37 ± 0.94	58.59 ± 2.47	73.76 ± 0.49	70.03 ± 0.86	47.83 ± 0.85	76.02 ± 1.73
PSGCNN	88.95 ± 4.37	76.34 ± 1.68	75.00 ± 2.51	76.27 ± 2.64	62.29 ± 5.68	72.60 ± 2.15	71.00 ± 2.29	45.23 ± 2.84	86.30 ± 1.58
DCNN	66.98	56.61 ± 1.04	61.29 ± 1.60	58.09 ± 0.53	56.60	52.11 ± 0.71	49.06 ± 1.37	33.49 ± 1.42	–
ECC	76.11	76.82	72.65	74.10	–	67.79	–	–	–
GCCNN	–	**82.72** ± 2.38	**76.40** ± 4.71	77.62 ± 4.99	**66.01** ± 5.91	77.71 ± 2.51	71.69 ± 3.40	48.50 ± 4.10	87.61 ± 2.51
DGK	82.66 ± 1.45	62.48 ± 0.25	71.68 ± 0.50	78.50 ± 0.22	57.32 ± 1.13	73.09 ± 0.25	66.96 ± 0.56	44.55 ± 0.52	78.30 ± 0.30
AWE	87.87 ± 9.76	–	–	71.51 ± 4.02	–	70.99 ± 1.49	73.13 ± 3.28	51.58 ± 4.66	82.97 ± 2.86

(JTQK) avec q = 2 [114], 2) le noyau de sous-arbre Weisfeiler-Lehman (WLSK) , 3) affectation optimale du noyau de WeisfeilerLehman (WL-OA) [115], 4) le noyau de graphe de chemin le plus court (SPGK)) [116], 5) le noyau du chemin le plus court basé sur les variantes principales (CORE SP) [117], 6) le noyau du graphe de parcours aléatoire (RWGK) [118], 7) le noyau de comptage des graphe (GK) [119], et 8) le noyau de graphe d'information propagé (PIGK) [120].

Pour l'évaluation, le modèle QSGCNN proposé utilise la même structure de réseau pour tous les jeux de données graphes. Spécifiquement, l'auteur fixe le nombre de représentations de prototype à M = 64, le nombre de couches de convolution de graphe spatial quantique à 5 (notez que, y compris les structures de grille d'entrée d'origine, la convolution de graphe spatial produit 6 sorties concaténées) et les canaux. de chaque convolution de graphe spatial quantique égale à 32. Après chaque sortie concaténée après les couches de convolution de graphe quantique, il ajoute une couche CNN traditionnelle dotée de l'architecture C64-P2-C64-P2-C64-F64 pour apprendre les motifs extraits, où Ck désigne une couche convolutive traditionnelle avec k canaux, P k désigne une couche classique MaxPooling de taille et foulée k, et FCk désigne une couche entièrement connectée constituée de k unités cachées. La taille du filtre et la foulée de chaque Ck sont toutes égales à 5 et 1. Avec les six ensembles de motifs extraits après les couches CNN, il les concaténe et ajoutons un nouveau calque entièrement connecté suivi d'un calque Softmax avec un taux de perte de 0 : 5. il utilise les unités linéaires rectifiées (ReLU) dans la convolution du graphe ou dans la couche de convolution traditionnelle. Le taux d'apprentissage du modèle proposé est de 0: 00005 pour tous les jeux de données. Le seul hyperparamètre que il a optimisé est le nombre d'époques et la taille de lot pour l'algorithme décent à gradient de mini-batch. Pour optimiser le modèle QSGCNN proposé, l'auteur utilise la descente de gradient stochastique avec les règles de mise à jour d'Adam.

Enfin, notez que le modèle QSGCNN proposé doit construire les représentations de prototype pour identifier les informations d'alignement de sommet transitif sur tous les graphes. Les représentations de prototype peuvent être calculées à partir des graphes d'apprentissage ou des graphes d'apprentissage et de test. l'auteur observe que le modèle proposé associé aux deux variantes dose n'influence pas la performance finale. Ainsi, dans notre évaluation, il a proposé de calculer les représentations du prototype à partir des graphes de formation et de test. En ce sens, notre modèle peut être vu comme une instance d'apprentissage par transduction [121], où tous les graphes sont utilisés pour calculer les représentations du prototype, et les étiquettes de classe des graphes de test ne sont pas observées pendant la phase de formation. Pour le modèle QSGCNN proposé, il effectue une validation croisée de 10 fois afin de calculer les précisions de classification, avec neuf plis pour la formation et un pli pour les tests. Pour chaque jeu de données, l'auteur répéte l'expérience 10 fois et signalons les exactitudes de classification moyennes et les erreurs types dans le tableau.II.

l'auteur définisse les paramètres contrôlant la hauteur maximale des sous-arbres pour le test d'isomorphisme de Weisfeiler-Lehman (noyau WLSK) et pour la méthode d'indexage (noyau JTQK) sur 10. C'est basé sur les études empiriques précédentes de Shervashidze et al. et Bai et al. [41] Pour chaque noyau de graphe, il effectue une validation croisée de 10 fois à l'aide de l'implémentation LIBSVM des machines à vecteurs de support C (C-SVM) et l'auteur calcule la précision de la classification. il effectue une validation croisée sur les données d'apprentissage pour sélectionner les paramètres optimaux pour chaque noyau et repli. il répéte l'expérience 10 fois pour chaque noyau et ensemble de données et l'auteur rapporte les exactitudes de classification moyennes et les erreurs types dans le tableau.II. Notez que pour certains noyaux, l'auteur rapporte directement les meilleurs résultats des papiers correspondants originaux, car le l'évaluation de ces noyaux a suivi le même réglage que le nôtre.

Résultats expérimentaux et discussion : Le tableau II montre que le modèle QSGCNN proposé surpasse de manière significative les noyaux graphes de pointe proposés dans cette étude. Bien que le modèle proposé ne puisse pas atteindre la meilleure précision de

classification sur les jeux de données NCI1 et COLLAB, il reste compétitif et la précision sur le jeu de données COLLAB n'est que légèrement inférieure à celle du noyau WL-OA. D'autre part, la précision du modèle proposé sur le jeu de données NCI1 est toujours supérieure à celle des noyaux SPGK, CORE SP, GK et RWGK. Les raisons de l'efficacité sont doubles. Premièrement, les noyaux de graphes de pointe pour la comparaison sont des exemples typiques de noyaux de R-convolution. Spécifiquement, ces noyaux sont basés sur la mesure d'isomorphisme entre toute paire de sous-structures, en ignorant les informations de correspondance de structure entre les sous-structures. En revanche, la structure de grille de sommets alignée associée pour le modèle QSGCNN proposé incorpore les informations d'alignement transitif entre les sommets sur tous les graphes. Ainsi, le modèle proposé peut mieux refléter les caractéristiques précises des graphes. Deuxièmement, le classifieur C-SVM associé aux noyaux de graphes ne peut être considéré que comme un cadre d'apprentissage superficiel [122]. En revanche, le modèle QSGCNN proposé peut fournir une architecture d'apprentissage approfondi de bout en bout pour la classification des graphes et mieux connaître les caractéristiques des graphes. Les expériences démontrent les avantages du modèle QSGCNN proposé par rapport au cadre d'apprentissage peu profond. Troisièmement, certains noyaux alternatifs sont liés à la méthode Weisfeiler-Lehman. Comme l'auteur a indiqué à la section III-D, les noyaux basés sur la méthode de Weisfeiler-Lehman peuvent souffrir du problème de la bascule. En revanche, le modèle proposé basé sur la marche quantique peut réduire considérablement l'effet de la marche en titubant. Les expériences démontrent également l'efficacité .

IV .4.2. Comparaisons avec les méthodes d'apprentissage en profondeur :

Jeux de données : Dans cette sous-section, l'auteur compare davantage les performances du modèle QSGCNN proposé avec des méthodes d'apprentissage en profondeur de pointe pour les classifications de graphes. Les jeux de données pour les évaluations comprennent les cinq jeux de données bioinformatiques mentionnés, ainsi que trois jeux de données de réseaux sociaux. Les jeux de données de réseaux sociaux comprennent COLLAB, IMDB-B et IMDB-M. Vous trouverez des détails sur ces jeux de données de réseaux sociaux dans le tableau.I.

Configuration expérimentale : l'auteur évalue les performances du modèle QSGCNN proposé en ce qui concerne les problèmes de classification des graphes par rapport à cinq méthodes alternatives d'apprentissage en profondeur dernier cri pour les graphes. Ces méthodes comprennent 1) le réseau de neurones convolutionnels à graphes profonds (DGCNN) [103], 2) le réseau neuronal convolutionnel à bases de PATCHY-SAN (PSGCNN) [102], 3) le réseau neuronal convolutionnel à diffusion (DCNN) [101]. , 4) les réseaux convolutionnels conditionnés par les bords (ECC) [123], 5) le noyau de graphe profond (DGK) [124], 6) le réseau neuronal convolutionnel à capsule de graphe (GCCNN) [125], et 7) Pour le modèle QSGCNN proposé, il utilise les mêmes configurations expérimentales lorsque il compare le modèle proposé aux noyaux de graphes. Pour les modèles PSGCNN, ECC et DGK, il présente les meilleurs résultats des documents originaux [102], [123], [124]. Notez que ces méthodes suivent le même paramètre avec le modèle QSGCNN proposé. Pour le modèle DCNN, il rapporte les meilleurs résultats des travaux de Zhang et al., [103], suivant le même paramètre que l'auteur. Pour le modèle AWE, il rapporte l'exactitude de la classification de AWE axée sur les fonctionnalités, car l'auteur a indiqué que ce type de modèle AWE peut atteindre des

performances compétitives sur les jeux de données d'étiquettes. Enfin, les modèles PSCN et ECC peuvent exploiter des fonctionnalités de périphérie supplémentaires. Comme la plupart des jeux de données graphes et toutes les méthodes alternatives à utiliser pour les comparaisons ne tirent pas parti des entités marginales, il ne présente pas dans ce travail les résultats associés aux entités marginales. Les précisions de classification et les erreurs types pour chaque méthode d'apprentissage en profondeur sont indiquées dans le tableau.III.

Résultats expérimentaux et discussion : le tableau III indique que le modèle QSGCNN proposé surpasse de manière significative les méthodes d'apprentissage en profondeur les plus avancées pour les classifications de graphes, sur les jeux de données MUTAG, D & D, COLLAB, IBDM-B, IBDM-M et RET-B. D'autre part, seules la précision du modèle GCCNN sur les jeux de données NCI1 et PTC et celle du modèle DGCNN sur le jeu de données PROTEINS sont supérieures à celles du modèle QSGCNN proposé. Mais le QSGCNN proposé reste compétitif et surpasse les méthodes restantes sur les trois jeux de données. Les raisons de l'efficacité sont cinq fois.

Premièrement, comme dans les noyaux de graphes à la pointe de la technologie, toutes les méthodes alternatives d'apprentissage en profondeur (à savoir, les modèles DGCNN, PSGCNN, DCNN, ECC, GCCNN, DGK et AWE) ne peuvent pas intégrer les informations de correspondance entre graphes. dans l'architecture d'apprentissage. En particulier, les modèles PSGCNN, DGCNN et ECC doivent réorganiser les sommets, mais ces méthodes reposent sur une heuristique simple mais inexacte pour aligner les sommets des graphes, c'est-à-dire qu'elles trient les ordres de sommet en fonction du descripteur de structure locale de chaque graphe et ignorer les informations de correspondance de sommet entre différents graphes. Ainsi, seul le modèle QSDCNN proposé peut refléter avec précision les caractéristiques du graphe grâce à l'apprentissage par couches.

Deuxièmement, les modèles PSGCNN et DGCNN doivent former une structure de grille de vertex de taille fixe pour chaque graphe. Étant donné que les numéros de sommet de différents graphes sont différents, la formation de telles structures de grille de taille six signifie que certains sommets de chaque graphe peuvent être ignorés, entraînant une perte d'informations. En revanche, comme l'auteur a mentionné aux sections II et III, les structures de grille de sommets alignées associées peuvent conserver complètement les informations des graphes originaux. De ce fait, seul le modèle QSGCNN proposé peut intégrer complètement les caractéristiques du graphe d'origine dans le processus d'apprentissage.

Troisièmement, contrairement au modèle proposé, le modèle DCNN doit résumer les entités de sommet de niveau local extraites de l'opération de convolution en tant qu'entités de graphe de niveau global via une couche SumPooling. Ainsi, seul le modèle QSGCNN peut apprendre les informations topologiques du graphe par le biais des entités de sommet locales.

Quatrièmement, contrairement au PSGCNN, DGCNN, GCCNN et ECC Pour les modèles basés sur la matrice de communication de vertex d'origine pour formuler les informations de connexion de vertex de l'opération de convolution de graphe, l'opération de convolution de graphe du modèle QSGCNN proposé formule les informations de connexion de vertex en termes de matrice de mélange moyenne de la marche quantique continue. Comme l'auteur a indiqué à la section II, la marche quantique n'est pas dominée par la basse fréquence du spectre laplacien et peut mieux distinguer les différentes structures de graphes. Ainsi, le modèle QSDCNN proposé a une meilleure capacité à identifier la différence entre différents graphes.

Cinquièmement, similaire aux modèles DGCNN, PSGCNN et DGK, le modèle QSGCNN proposé est également lié à la méthode classique de Weisfeiler-Lehman (WL). Comme la méthode classique WL souffre d'un problème de basculement, les modèles DGCNN, PSGCNN et DGK associés traitent également le même inconvénient. En revanche, l'opération de convolution entre graphes du modèle QSGCNN proposé peut être considérée comme la version quantique de l'algorithme classique WL. Étant donné que la marche quantique peut réduire le problème du problème de la bascule, le modèle QSGCNN proposé pallie le problème posé par le problème de la bascule dans les modèles DGCNN, PSGCNN et DGK. Sixièmement, le modèle AWE est basé sur la marche aléatoire classique. En revanche, le modèle QSGCNN proposé est basé sur la marche aléatoire quantique, qui s'est révélée puissante pour mieux distinguer les différentes structures de graphes. L'évaluation démontre les avantages du modèle QSGCNN proposé par rapport aux méthodes d'apprentissage en profondeur les plus modernes.

Conclusion :

Dans ce chapitre, Nous avons expliqué les mesures prises par l'auteur pour développer un nouveau modèle de graphe spatial de réseau neuronal à convolution (QSGCNN), dans lequel l'auteur a détaillé les caractéristiques des graphes et le rôle de l'efficacité du modèle QSGCNN proposé dans les problèmes de classification des graphes. il a également utilisé la même structure de réseau pour tous les jeux de données. Dans les travaux futurs, l'auteur vise à connaître la structure optimale de chaque jeu de données, ce qui améliorera les performances.

Conclusion général :

Dans cette recherche, notre objectif était d'utiliser la science moderne pour représenter des données sous forme de diagrammes et nous avons essayé de soumettre des diagrammes et de les appliquer à des réseaux de neurones artificiels. Apprentissage automatique et apprentissage en profondeur Nous avons mentionné les domaines d'apprentissage en profondeur les plus couramment utilisés. Nous avons également abordé la présentation d'importants concepts de graphes et de leurs caractéristiques, en particulier l'utilisation répétée de graphes en expliquant le travail de l'algorithme (FSM) afin d'obtenir les graphes les plus fréquents. Lorsqu'il existe plusieurs structures de méthodes d'apprentissage basées sur des graphes profonds (GNN, GCN, GAE), nous avons également mentionné les équations de base appliquées aux graphes dans chaque type de réseau de neurones (GNN, GCN et GAE). Nous avons expliqué la définition du nouveau modèle de réseau neuronal par l'auteur convolution spatiale (QSGCNN), qui est le dernier modèle dans lequel l'auteur a mentionné le rôle de l'efficacité du modèle QSGCNN proposé dans les problèmes de classification des graphes. l'auteur a essayé d'utiliser la même structure de réseau pour un ensemble de données dans lequel les scientifiques et les programmeurs sont toujours à la recherche constante pour trouver la structure optimale de chaque ensemble de données, ce qui améliorera les performances.

REFERENCES

[1] Y. . LeCun, Y. Bengio, and G. Hinton, "Deep learning," Nature, vol. 521, no. 7553, p. 436, 2015.

[2] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath et al., "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," IEEE Signal Processing Magazine, vol. 29, no. 6, pp. 82–97, 2012.

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in Advances in Neural Information Processing Systems, 2012, pp. 1097–1105

[4] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in Proceedings of the 4th International Conference on Learning Representations, 2015

[5] A.-L. Barabasi, Network science. Cambridge university press, 2016.

[6] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Van dergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," IEEE Signal Processing Magazine, 2013.

[7] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, "Geometric deep learning: going beyond euclidean data," IEEE Sig Processing Magazine, vol. 34, no. 4, pp. 18–42, 2017

[8] C. Zang, P. Cui, and C. Faloutsos, "Beyond sigmoids: The nettide model for social network growth, and its applications," in Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 2016, pp. 2015–2024

[9] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zam baldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner ,C. Gulcehre, F. Song, A. Ballard, J. Gilmer, G. Dahl, A. Vaswani, K. Allen, C. Nash, V. Langston, C. Dyer, N. Heess, D. Wierstra, P. Kohli, M. Botvinick, O. Vinyals, Y. Li, and R. Pascanu, "Relational inductive biases, deep learning, and graph networks," arXiv preprint arXiv:1806.01261, 2018.

[10] J. B. Lee, R. A. Rossi, S. Kim, N. K. Ahmed, and E. Koh, "Attention models in graphs: A survey," arXiv preprint arXiv:1807.07984, 2018.

[11] S. Yan, D. Xu, B. Zhang, H.-J. Zhang, Q. Yang, and S. Lin, "Graph

embedding and extensions: A general framework for dimensionality reduction," IEEE Transactions on Pattern Analysis and Machine . intelligence, vol. 29, no. 1, pp. 40–51, 2007.

[12] W. L. Hamilton, R. Ying, and J. Leskovec, "Representation learning on graphs: Methods and applications," arXiv preprint arXiv:1709.05584.

[13] P. Cui, X. Wang, J. Pei, and W. Zhu, "A survey on network embedding," IEEE Transactions on Knowledge and Data Engineering, 2018.

[14] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," Nature, 1986.

[15] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in Proceedings of the 3rd International Conference on Learning Representations, 2014.

[16] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," Journal of Machine Learning Research, vol. 15, no. 1, pp. 1929–1958, 2014.

[17] X. Wang, P. Cui, J. Wang, J. Pei, W. Zhu, and S. Yang, "Community preserving network embedding," in Proceedings of the 31st AAAI Conference on Artificial Intelligence, 2017.

[18] J. Leskovec and J. J. Mcauley, "Learning to discover social circles in ego networks," in Advances in Neural Information Processing Systems, 2012, pp. 539–547.

[19] M. Gori, G. Monfardini, and F. Scarselli, "A new model for learning in graph domains," in IEEE International Joint Conference on Neural Networks Proceedings, vol. 2. IEEE, 2005, pp. 729–734.

[20] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," IEEE Transactions on Neural Networks, vol. 20, no. 1, pp. 61–80, 2009.

[21] P. Frasconi, M. Gori, and A. Sperduti, "A general framework for adaptive processing of data structures," IEEE transactions on Neural Networks, vol. 9, no. 5, pp. 768–786, 1998.

[22] M. J. Powell, "An efficient method for finding the minimum of a function of several variables without calculating derivatives," The computer journal, vol. 7, no. 2, pp. 155–162, 1964.

[23] L. B. Almeida, "A learning rule for asynchronous perceptrons with feedback in a combinatorial environment." in Proceedings, 1st First International Conference on Neural Networks. IEEE, 1987.

[24] F. J. Pineda, "Generalization of back-propagation to recurrent neural networks," Physical Review Letters, vol. 59, no. 19, p. 2229, 1987

[25] M. A. Khamsi and W. A. Kirk, An introduction to metric spaces and fixed point theory. John Wiley & Sons, 2011, vol. 53.

[26] . Aridhi S. Distributed Frequent Subgraph Mining in the Cloud. Engineering

Doctoral School of Clermont Ferrand; 2013.
[27] . Müller DE. Lesematerial | Big Data Analytics | openHPI [Internet]. [cited 2018 May 25]. Available from: https://open.hpi.de/courses/bigdata2017/items/4TO9c3oDe1AhP9tecEUDp5
[28] . Washio T, Motoda H. State of the art of graph-based data mining. ACM SIGKDD Explor Newsl [Internet]. 2003;5(1):59. Available from: http://portal.acm.org/citation.cfm?doid=959242.959249
[29] . Holder LB, Cook. Substructure Discovery in the SUBDUE System. J Chem Inf Model. 2013;53(9):1689–99.
[30] . Xifeng Yan, Jiawei Han. gSpan: graph-based substructure pattern mining. EEE Int Conf Data Min [Internet]. 2002;1(d):721–4. Available from: http://ieeexplore.ieee.org/document/1184038/
[31] . Inokuchi A, Washio T, Motoda H. An Apriori-Based Algorithm for Mining Frequent Substructures from Graph Data. 2000;13–23. Available from: http://link.springer.com/10.1007/3-540-45372-5_2
[32] . Ozaki T, Ohkawa T. Mining correlated subgraphs in graph databases. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining. 2008.
[33] . Yan X, Cheng H, Han J, Yu PS. Mining significant graph patterns by leap search. Proc 2008 ACM SIGMOD Int Conf Manag data - SIGMOD '08 [Internet]. 2008;433. Available from: http://portal.acm.org/citation.cfm?doid=1376616.1376662
[34] . Chen C, Yan X, Zhu F, Han J. gApprox: Mining frequent approximate patterns from a massive network. Proc - IEEE Int Conf Data Mining, ICDM. 2007;445–50.
[35] . Yan X, Cheng H, Han J, Xin D. Summarizing itemset patterns. Proceeding Elev ACM SIGKDD Int Conf Knowl Discov data Min - KDD '05 [Internet]. 2005;314. Available from: http://portal.acm.org/citation.cfm?doid=1081870.1081907
[36] . Huan J, Wang W, Prins J, Yang J. Spin: mining maximal frequent subgraphs from graph databases. In: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining. 2004. p. 581–6.
[37] . Flake GW, Tarjan RE, Tsioutsiouliklis K. Graph clustering and minimum cut trees. Internet Math. 2004;1(4):385–408.
[38] . Huang Y, Niu B, Gao Y, Fu L, Li W. CD-HIT Suite: a web server for clustering and comparing biological sequences. Bioinformatics. 2010;26(5):680–2.
[39] . Yan X, Yu PS, Han J. Graph indexing: a frequent structure-based approach. In: Proceedings of the 2004 ACM SIGMOD international conference on Management of data. 2004. p. 335–46.
[40] . Yan X, Yu PS, Han J. Substructure similarity search in graph databases. In: Proceedings of the 2005 ACM SIGMOD international conference on Management of data. 2005. p. 766–77.
[41] . Shasha D, Wang JTL, Giugno R. Algorithmics and applications of tree and graph searching. In: Proceedings of the twenty-first ACM SIGMODSIGACT-SIGART symposium on Principles of database systems.

[42] . Gärtner T, Flach P, Wrobel S. On graph kernels: Hardness results and efficient alternatives. In: Learning Theory and Kernel Machines. Springer; 2003. p. 129–43.
[43] Kashima H, Tsuda K, Inokuchi A. Marginalized kernels between labeled graphs. In: Proceedings of the 20th international conference on machine learning (ICML-03). 2003. p. 321–8.
[44] . Chakrabarti S, Dom BE, Kumar SR, Raghavan P, Rajagopalan S, Tomkins A, et al. Mining the Web's link structure. Computer (Long Beach Calif). 1999;32(8):60–7.
[45] . Kosala R, Blockeel H. Web mining research: A survey. ACM Sigkdd Explor Newsl. 2000;2(1):1–15.
[46] . Getoor L, Diehl CP. Link mining: a survey. Acm Sigkdd Explor Newsl. 2005;7(2):3–12.
[47] . Kleinberg JM. Authoritative sources in a hyperlinked environment. In: In Proceedings of the ACM-SIAM Symposium on Discrete Algorithms. 1998.
[48] . Brin S, Page L. The anatomy of a large-scale hypertextual web search engine. Comput networks ISDN Syst. 1998;30(1–7):107–17.
[49] Greco G, Guzzo A, Manco G, Sacca D. Mining and reasoning on workflows. IEEE Trans Knowl Data Eng. 2005;17(4):519–34.
[50] . Hu H, Yan X, Huang Y, Han J, Zhou XJ. Mining coherent dense subgraphs across massive biological networks for functional discovery. Bioinformatics. 2005;21(suppl_1):i213--i221.
[51] . Yan X. Mining, indexing and similarity search in large graph data sets. 2006; Available from: https://www.ideals.illinois.edu/handle/2142/11256
[52] . Kavitha D, Rao BVM, Babu VK. A Survey on Assorted Approaches to Graph Data Mining. Int J Comput Appl. 2011;14(1):43–6.
[53] . Trlnajstic N. CHEMICAL GRAPH THEORY. Randie M, Klein DJ, Mezey P 0., Trinajstic N, editors. 323 p.
[54] . Aldrich H, Bates T, Kuhn T. Open HPI Mooc Courses. 2001. p. 1–5.
[55] . Chuntao Jiang FC and MZ. A Survey of Frequent Subgraph Mining Algorithms. Knowl Eng Rev. 2004;00(January):1–24.
[56] . Gutman I, Polansky OE. Mathematical concepts in organic chemistry. Springer Science & Business Media; 2012. 28 p.
[57] . Agrawal R, Srikant R, others. Fast algorithms for mining association rules. In: Proc 20th int conf very large data bases, VLDB. 1994. p. 487–99.
[58] . Agrawal R, Srikant R. Fast Algorithms for Mining Association Rules in Large Databases. J Comput Sci Technol [Internet]. 1994;15(6):487–99. Available from: http://portal.acm.org/citation.cfm?id=645920.672836
[59] . Garey MR, Johnson DS. Computers and intractability: a guide to NPcompleteness. . WH Freeman and Company, San Francisco; 1979.
[60] . Ullmann JR. An algorithm for subgraph isomorphism. J ACM. 1976;23(1):31–42.

[61] . Schmidt DC, Druffel LE. A fast backtracking algorithm to test directed graphs for isomorphism using distance matrices. J ACM. 1976;23(3):433–45.

[62] D. E. Rumelhart, G. E. Hinton, and R. J. Williams
. "Learning representations by back- propagating errors," Nature, 1986.
[63] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in .
. Proceedings of the 3rd International Conference on Learning Representations, 2014.
[64] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, .
. "Dropout: a simple way to prevent neural networks from overfitting," Journal of .
. Machine Learning Research, vol. 15, no. 1, pp. 1929– 1958, 2014.
[65] X. Wang, P. Cui, J. Wang, J. Pei, W. Zhu, and S. Yang, "Community preserving network embedding," in Proceedings of the 31st AAAI Conference on Artificial Intelligence, 2017.
[66] J. Leskovec and J. J. Mcauley, "Learning to discover social circles in ego networks," in Advances in Neural Information Processing Systems, 2012, pp. 539–547.

[67] M. J. Powell, "An efficient method for finding the minimum of a function of .
. variables without calculating derivatives," The computer journal, vol. 7, no. 2, pp. 155–162, 1964.

[68] M. A. Khamsi and W. A. Kirk, An introduction to metric spaces and fixed point theory. John Wiley & Sons, 2011, vol. 53.
[69] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel, "Gated graph sequence neural networks," in Proceedings of the 5th International Conference on Learning Representations, 2016.
[70] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder–decoder for statistical machine translation," in Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, 2014, pp. 1724–1734.
[71] M. Brockschmidt, Y. Chen, B. Cook, P. Kohli, and D. Tarlow, "Learning to decipher the heap for program verification," in Workshop on Constructive . . . .
. Machine Learning at the International Conference on Machine Learning, 2015.
[72] S. Sukhbaatar, R. Fergus et al., "Learning multiagent communication with backpropagation," in Advances in Neural Information Processing Systems, 2016, pp. 2244–2252.
[73] P. W. Battaglia, R. Pascanu, M. Lai, D. Rezende, and K. Kavukcuoglu, "Interaction networks for learning about objects, relations and physics," in Advances in Neural Information Processing Systems, 2016.
[74] Y. Hoshen, "Vain: Attentional multi-agent predictive modeling," in Advances in Neural Information Processing Systems, 2017.
[75] A. Santoro, D. Raposo, D. G. T. Barrett, M. Malinowski, R. Pascanu, P. Battaglia, and T. Lillicrap, "A simple neural network module for

relational reasoning," in Advances in Neural Information Processing Systems, 2017.
[76] J. Bruna, W. Zaremba, A. Szlam, and Y. Lecun, "Spectral networks and locally connected networks on graphs," in Proceedings of the 3rd International Conference on Learning Representations, 2014.
[77] M. Henaff, J. Bruna, and Y. LeCun, "Deep convolutional networks on graph-structured data," arXiv preprint arXiv:1506.05163, 2015.
[78] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in Advances in Neural Information Processing Systems, 2016, pp. 3844–3852.
[79] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in Proceedings of the 6th International Conference on Learning Representations, 2017.
[80] D. K. Duvenaud, D. Maclaurin, J. Iparraguirre, R. Bombarell, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams, "Convolutional networks on graphs for learning molecular fingerprints," in Advances in Neural Information Processing Systems, 2015, pp. 2224–2232.
[81] M. Niepert, M. Ahmed, and K. Kutzkov, "Learning convolutional neural networks for graphs," in International Conference on Machine Learning, 2016, pp. 2014–2023.
[82] J. Atwood and D. Towsley, "Diffusion-convolutional neural networks," in Advances in Neural Information Processing Systems, 2016.
[83] C. Zhuang and Q. Ma, "Dual graph convolutional networks for graphbased semi-supervised classification," in Proceedings of the 2018 World Wide Web Conference, 2018, pp. 499–508.
[84] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in International Conference on Machine Learning, 2017, pp. 1263–1272.
[85] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in Advances in Neural Information Processing Systems, 2017, pp. 1024–1034.
[86] F. Monti, D. Boscaini, J. Masci, E. Rodola, J. Svoboda, and M. M. Bronstein, "Geometric deep learning on graphs and manifolds using mixture model cnns," in Proceedings of Computer Vision and Pattern Recognition, vol. 1, no. 2, 2017, p. 3.
[87] R. Ying, J. You, C. Morris, X. Ren, W. L. Hamilton, and J. Leskovec, "Hierarchical graph representation learning with differentiable pooling," in Advances in Neural Information Processing Systems, 2018.
[88] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," in Proceedings of the 7th International Conference on Learning Representations, 2018.
[89] T. Pham, T. Tran, D. Q. Phung, and S. Venkatesh, "Column networks for collective classification." in Proceedings of the 31st AAAI Conference on Artificial Intelligence, 2017, pp. 2485–2491.
[90] K. Xu, C. Li, Y. Tian, T. Sonobe, K.-i. Kawarabayashi, and S. Jegelka, "Representation learning on graphs with jumping knowledge networks," in International Conference on Machine Learning, 2018.

[91] M. Simonovsky and N. Komodakis, “Dynamic edgeconditioned filters in convolutional neural networks on graphs,” in 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.

[92] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. V. D. Berg, I. Titov, and M. Welling, “Modeling relational data with graph convolutional networks,” pp. 593–607, 2018.

[93] S. Kearnes, K. McCloskey, M. Berndl, V. Pande, and P. Riley, “Molecular graph convolutions: moving beyond fingerprints,” Journal of Computer-Aided Molecular Design, vol. 30, no. 8, pp. 595–608, 2016.

[94] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, “Graph convolutional neural networks for web-scale recommender systems,” in Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2018.

[95] J. Chen, T. Ma, and C. Xiao, “Fastgcn: fast learning with graph convolutional networks via importance sampling,” in Proceedings of the 7th International Conference on Learning Representations, 2018.

[96] J. Chen, J. Zhu, and L. Song, “Stochastic training of graph convolutional networks with variance reduction,” in International Conference on Machine Learning, 2018, pp. 941–949.

[97] M. Belkin and P. Niyogi, “Laplacian eigenmaps and spectral techniques for embedding and clustering,” in Advances in neural information processing systems, 2002, pp. 585–591.

[98] D. K. Hammond, P. Vandergheynst, and R. Gribonval, “Wavelets on graphs via spectral graph theory,” Applied and Computational Harmonic Analysis, vol. 30, no. 2, pp. 129–150, 2011.

[99] Z. Zhang, P. Cui, X. Wang, J. Pei, X. Yao, and W. Zhu, “Arbitrary-order proximity preserved network embedding,” in Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. ACM, 2018, pp. 2778–2786.

[100] D. K. Duvenaud, D. Maclaurin, J. Aguilera-Iparraguirre, R. Gomez-´ Bombarelli, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams, “Convolutional . . . . . networks on graphs for learning molecular fingerprints,” in Proceedings of NIPS, 2015, pp. 2224–2232.

[101] J. Atwood and D. Towsley, “Diffusion-convolutional neural networks,” in Proceedings of NIPS, 2016, pp. 1993–2001.

[102] M. Niepert, M. Ahmed, and K. Kutzkov, “Learning convolutional neural networks for graphs,” in Proceedings of ICML, 2016, pp. 2014–2023.

[103] M. Zhang, Z. Cui, M. Neumann, and Y. Chen, “An end-to-end deep learning architecture for graph classification,” in Proceedings of AAAI, 2018.

[104] E. Farhi and S. Gutmann, “Quantum computation and decision trees,” Physical Review A, vol. 58, p. 915, 1998.

[105] C. Godsil, “Average mixing of continuous quantum walks,” Journal of Combinatorial Theory, Series A, vol. 120, no. 7, pp. 1649–1662, 2013.

[106] I. H. Witten, E. Frank, and M. A. Hall, Data Mining: Practical Machine Learning Tools and Techniques. Morgan Kaufmann, 2011.

[107] L. Bai, L. Cui, L. Rossi, L. Xu, and E. Hancock, "Local-global nested graph kernels using nested complexity traces," Pattern Recognition Letters, To appear.

[108] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," CoRR, vol. abs/1609.02907, 2016. [Online]. Available: http://arxiv.org/abs/1609.02907

[109] I. Schomburg, A. Chang, C. Ebeling, M. Gremse, C. Heldt, G. Huhn, and D. Schomburg, "Brenda, the enzyme database: updates and major new developments," Nucleic Acids Research, vol. 32, no. Database-Issue, pp. 431–433, 2004.

[110] A. K. Debnath, R. L. Lopez de Compadre, G. Debnath, A. J. Shusterman, and C. Hansch, "Structure-activity relationship of mutagenic aromaticand . heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity," Journal of Medicinal Chemistry, vol. 34, no. 2, pp. 786–797, 1991.

[111] P. D. Dobson and A. J. Doig, "Distinguishing enzyme structures from non-enzymes without alignments," J. Mol. Biol., vol. 330, no. 4, p. 771C783, 2003.

[112] N. Wale, I. A. Watson, and G. Karypis, "Comparison of descriptor spaces for chemical compound retrieval and classification," Knowl. Inf. Syst., vol. 14, no. 3, pp. 347–375, 2008.

[113] K. Kersting, N. M. Kriege, C. Morris, P. Mutzel, and M. Neumann, "Benchmark data sets for graph kernels," 2008. [Online]. Available: http://graphkernels.cs.tu-dortmund

[114] L. Bai, L. Rossi, H. Bunke, and E. R. Hancock, "Attributed graph kernels using the jensen-tsallis q-differences," in Proceedings of ECML-PKDD, 2014, pp. 99–114.

[115] N. M. Kriege, P. Giscard, and R. C. Wilson, "On valid optimal assignment . . kernels and applications to graph classification," in Proceedings of NIPS, 2016, pp. 1615–1623.

[116] K. M. Borgwardt and H.-P. Kriegel, "Shortest-path kernels on graphs," in Proceedings of the IEEE International Conference on Data Mining, 2005, pp. 74–81.

[117] G. Nikolentzos, P. Meladianos, S. Limnios, and M. Vazirgiannis, "A degeneracy framework for graph similarity," in Proceedings of IJCAI, 2018, pp. 2595–2601.

[118] H. Kashima, K. Tsuda, and A. Inokuchi, "Marginalized kernels between labeled graphs," in Proceedings of ICML, 2003, pp. 321–328.

[119] N. Shervashidze, S. Vishwanathan, K. M. T. Petri, and K. M. Borgwardt, "Efficient graphlet kernels for large graph comparison," Journal of Machine Learning Research, vol. 5, pp. 488–495, 2009.

[120] M. Neumann, R. Garnett, C. Bauckhage, and K. Kersting, "Propagation kernels: efficient graph kernels from propagated information," Machine Learning, vol. 102, no. 2, pp. 209–245, 2016.

[121] A. Gammerman, K. S. Azoury, and V. Vapnik, "Learning by transduction," in

. . Proceedings of UAI, 1998, pp. 148–155.
[122] S. Zhang, C. Liu, K. Yao, and Y. Gong, “Deep neural support vector machines for speech recognition,” in Proceedings of ICASSP, 2015, pp. 4275–4279.
[123] M. Simonovsky and N. Komodakis, “Dynamic edge-conditioned filters in convolutional neural networks on graphs,” in Proceedings of CVPR, 2017, pp. 29–38.
[124] P. Yanardag and S. V. N. Vishwanathan, “Deep graph kernels,” in Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, NSW, Australia, August 10-13, 2015, 2015, pp. 1365–1374.
[125] S. Verma and Z. Zhang, “Graph capsule convolutional neural networks,” CoRR, vol. abs/1805.08090, 2018. [Online]. Available: http://arxiv.org/abs/1805.08090
[126] S. Ivanov and E. Burnaev, “Anonymous walk embeddings,” in Proceedings of . . ICML, 2018, pp. 2191–2200.
[127] L. Bai, P. Ren, L. Rossi, and E. R. Hancock, “An edge-based matching kernel through discrete-time quantum walks,” in Processing of ICIAP.
[128] L. Bai, L. Rossi, L. Cui, Z. Zhang, P. Ren, X. Bai, and E. R. Hancock, “Quantum kernels for unattributed graphs using discrete-time quantum walks,” Pattern Recognition Letters, vol. 87, pp. 96–103, 2017.
[129] L. Bai, F. Escolano, and E. R. Hancock, “Depth-based hypergraph complexity . traces from directed line graphs,” Pattern Recognition, vol. 54, pp. 229–240, 2016.
[130] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, “How powerful are graph neural networks?” CoRR, vol. abs/1810.00826, 2018. [Online]. Available: http://arxiv.org