Utilisation des Graph Neural Networks (GNN) dans le contexte de détection de fraude

Hugo Matijascic , **Thomas Fontier** ,
Polytechnique Montréal
{hugo.matijascic, thomas.fontier}@polymtl.ca

Abstract

Ce document a été rédigé dans le contexte du projet de session pour le cours INF8225: IA. techniques Probabilistes et d'Apprentissage. L'objectif de cet article est de découvrir de nouvelles méthodes et techniques d'apprentissage, ainsi que d'acquérir des connaissances supplémentaires dans divers sujets. Notre choix s'est porté sur l'utilisation des reseaux de neurones en graphe dans la détection et la prédiction de la fraude.

1 Introduction

Avec l'expansion rapide des services numériques et des interactions sociales en ligne, l'analyse des réseaux complexes est devenue essentielle pour comprendre et intervenir dans divers scénarios, tels que les communautés en ligne, les forums souterrains et les systèmes financiers.

Les méthodes traditionnelles, souvent limitées les extractions manuelles de caractéristiques, ne parviennent pas à exploiter pleinement la richesse des données relationnelles et interactionnelles. De plus, elles sont également limitées par la nécessité d'avoir des connaissances expertes dans le domaine pour le choix des caractéristiques. Les réseaux de neurones en graphe (GNN) émergent comme une solution prometteuse pour modéliser ces interactions complexes et dynamiques, offrant ainsi de nouvelles perspectives pour l'identification d'acteurs clés et la détection d'anomalies dans divers contextes.

Notre projet se penche sur l'application des GNN à deux contextes distincts mais complémentaires : la détection d'anomalies (de fraudes) et l'identification des acteurs clés dans des réseaux complexes.

Nous avons sélectionné deux articles de recherche pour guider notre exploration :

1. *A Semi-supervised Graph Attentive Network for Financial Fraud Detection* [1] introduit un modèle semi-supervisé pour améliorer la détection de fraudes dans les services financiers. Le modèle utilise un mécanisme d'attention hiérarchique. En effet, il introduit un premier niveau d'attention au niveau des nœuds (afin de mieux corréler les différents voisins dans le graphe ou les différents attributs d'un individu), puis un deuxième au niveau de la vue (afin de mieux corréler différentes vues des données).
2. *Key Player Identification in Underground Forums over Attributed Heterogeneous Information Network Embedding Framework* [2] présente un système d'automatisation de l'analyse de forums de piratage. Ce modèle vise à identifier les acteurs clés de ces sites à partir des relations et interactions entre les différents utilisateurs. Le modèle repose sur l'extraction des caractéristiques importantes des acteurs et l'utilisation d'un réseau multivue pour représenter leurs relations. Le modèle utilise un mécanisme d'attention pour réunir les différentes représentations apprises basées sur les différents graphes du réseau multivue.

En combinant les méthodologies et avancées de ces deux études, notre projet vise à explorer en profondeur les capacités des GNN à modéliser et analyser des réseaux complexes pour différentes applications. Nous commencerons par fournir une explication plus détaillée des architectures des modèles mis en place dans les deux articles, puis nous comparerons les résultats de nos différentes expériences sur les modèles. Enfin, nous apporterons une analyse critique de notre approche d'apprentissage.

2 Reseau de neurones en graphe

Les réseaux de neurones en graphe sont une classe de modèles d'apprentissage opérant sur des structures de données en graphes. Ces derniers sont particulièrement utiles pour résoudre des problèmes complexes qui représentent des relations entre divers éléments. Les GNN (Graph Neural Networks) peuvent apprendre des représentations des nœuds et des liens entre ces derniers, ce qui les rend particulièrement adaptés à des applications telles que la classification, la prédiction ou encore la recommandation dans de nombreux domaines, on peut citer les réseaux sociaux, la biologie et bien d'autres.

La principale idée derrière l'apprentissage par les GNN est de propager l'information à travers les arêtes du graphe pour agréger et mettre à jour les représentations des nœuds. Ce processus s'effectue sur plusieurs couches et plusieurs itérations et permet de capturer certaines informations à différentes échelles. Ainsi, on peut évaluer l'impact d'un nœud du graphe dans un contexte local ou global.

Il existe différents types de GNN, chacun avec ses propres architectures et mécanismes de fonctionnement. Les réseaux convolutifs sur graphes (GCN) sont inspirés des réseaux convolutifs usuels. Les GCN appliquent des opérations de convolution sur les nœuds et leurs voisins dans le graphe. Ce traitement fait ressortir les motifs principaux du réseau en graphe. Les réseaux récursifs sur graphes se basent sur des mécanismes récursifs pour propager les informations le long des arêtes (ex : Tree-LSTM). Les réseaux attentionnels sur graphes (GAT), quant à eux, utilisent le mécanisme d'attention pour pondérer l'importance des voisins sur chaque nœud, permettant une mise à jour adaptative des représentations des nœuds.

Ces différentes architectures de GNN offrent des approches variées pour modéliser et traiter des données structurées sous forme de graphes, chacune présentant ses propres avantages et limitations selon le contexte d'application. Dans notre cas de figure, les GNN sont utilisés pour comprendre les relations complexes entre les entités d'un réseau et prédire la présence de fraude.

3 A Semi-supervised Graph Attentive Network for Financial Fraud Detection

3.1 Contexte

L'objectif de cet article est de proposer une avancée dans les modèles utilisés pour faire de la détection de fraude financière. Lors d'une transaction financière, un premier filtre va être responsable de vérifier des informations générales sur le compte débitant. Ces informations sont directement accessibles par l'organisme de vérification, il ne représente donc pas de défi. Si la transaction ne passe pas le filtre, elle sera refusée, cependant, ne pas être bloqué par le filtre ne signifie pas pour autant que la transaction est légitime. Pour déterminer cela un deuxième filtre est responsable de prédire la légitimité de celle-ci. Ce deuxième filtre va devoir apprendre de données de transactions pour pouvoir ensuite faire une prédiction sur des transactions. Si une transaction est épinglée comme frauduleuse par ce deuxième filtre, alors une investigation humaine aura lieu, celle-ci rendra son verdict qui bloquera ou autorisera la transaction. Finalement, le résultat pourra être fourni au modèle pour lui permettre de s'améliorer.

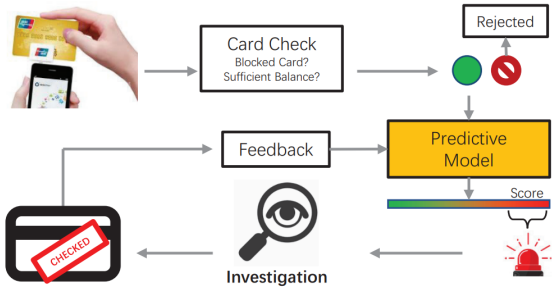

Figure 1: Representation d'une procédure de détection de fraude financière.

Entraîner un modèle de détection de fraude financière est composé de plusieurs défis :

1. Données non-labelisées : Un grand nombre de transactions financières sont effectuées en permanence. Il est donc théoriquement possible de construire d'énormes jeux de données. Cependant, il est complexe et long d'étiqueter ces jeux de données. En effet, il faut des connaissances expertes pour être capable de classifier un jeux de données de transactions. C'est pourquoi le modèle présenté dans l'article est un modèle semi-supervisé, afin de ne pas être limité aux données labelisées mais également profiter des informations contenues dans les données non labelisées.
2. Hétérogénéité des graphes : Il existe deux types de graphes : les graphes hétérogènes et les graphes homogènes. Cette distinction est basée sur les types des noeuds et des arêtes d'un graphe.

   Lorsque tous les noeuds et toutes les arêtes sont du même type alors il s'agit d'un graphe homogène. Par exemple, dans un graphe de représentation d'un réseau social, où les noeuds représenteraient les individus et les arêtes les amitiés entre les individus. Alors tous les noeuds seraient du même type (individus), de même pour les arêtes (amitiés).

   Lorsque les noeuds ou les arêtes peuvent être de différents types, alors il s'agit de graphes hétérogènes. Ces graphes sont plus complexes à traiter en raison de la diversités des types des noeuds et relations.

   Dans le contexte de la fraude, si on souhaite ajouter des informations temporelles, spatiales, etc. alors nous sommes dans un contexte d'hétérogénéité. Le modèle présenté va présenter une solution à ce problème.

3.2 Modèle proposé

Le modèle proposé implémente un mécanisme d'attention hiérarchique. Une attention au niveau des noeuds et une attention au niveau des vus. L'attention au niveau des noeuds va permettre d'apprendre le poids des voisins d'un noeud (source ou réception de la transaction) pour chaque vue. L'attention au niveau des vues va permettre de fournir au modèle les données sous différentes vues, tout en lui permettant d'appendre comment les interpréter.

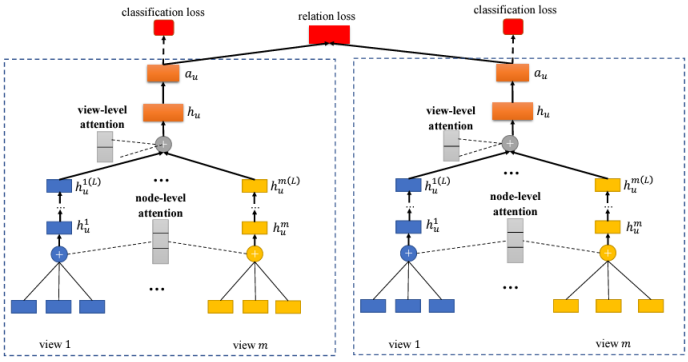

Figure 2: Representation de l'architecture du modèle proposé.

Comme on peut le voir sur la figure 2, chaque vue est étudiée indépendamment des autres. On obtient l'embedding de chaque noeud en assemblant l'embedding de ses voisins. L'importance $\alpha_{u_i}^v$ d'une paire de noeud (u, i), dans une vue spécifique v et en considérant une matrice d'attention H^v, ainsi que l'embedding pondéré du noeud i comme e_{ui}^v, sera calculée en suivant la formule suivante :

$$\alpha_{ui}^v = \frac{\exp(e_{ui}^v \cdot H_{ui}^v)}{\sum_{k \in N_u^v} \exp(e_{uk}^v \cdot H_{uk}^v)} \tag{1}$$

On obtient ainsi l'embedding d'un noeud dans une vue spécifique de la manière suivante :

$$h_u^v = \sum_{k \in N_u^{(A)}} \alpha_{uk} e_{uk} \tag{2}$$

Comme exposé dans les défis des GNN, combiner les embeddings d'un noeud provenant des différentes vues représente un défi de par l'hétérogénéité du problème. Une solution proposée dans l'article est d'utiliser des MLP, via la formule suivante :

$$h_u^{v(l)} = Relu(h_u^{v(l-1)} W_l^v + b_l^v) \tag{3}$$

Le mécanisme d'attention au niveau des vues permet d'apprendre l'importance relative de chaque vues et ainsi de mieux comprendre comment associer les différentes vues.

Voici le pseudo-code qui résume le fonctionnement de l'algorithme SemiGNN

Algorithm 1 Training Algorithm for SemiGNN

Input: The multiview graph: $G_v = \{U \cup S^v, E^v\}, v \in \{1, \ldots, m\}$. Balancing weight α. Regularizer weight λ.
Output: The model parameters Θ.

```
Randomly initialize the model parameters Θ, and attention parameters H^v and φ^v.
Generate random walk paths according to the relation graph G^(U) and construct the user paired set S.
for each (u, v) ∈ S do
    for each k ∈ {1, ..., m} do
        Obtain low-level view-specific user embeddings h_u^k and h_v^k by Eq. 2.
        Obtain high-level view-specific user embeddings h_u^k(L) and h_v^k(L) by Eq. 3.
        Obtain view preference vectors a_u^k and a_v^k.
    end for
    Combine embeddings to obtain a_u and a_v.
    Compute loss L_SemiGNN.
    Perform backpropagation and update model parameters: Θ_new = Θ_old − λ · ∂L_SemiGNN / ∂Θ_old.
end for
return Θ
```

4 Key Player Identification in Underground Forums over Attributed Heterogeneous Information Network Embedding Framework

Cet article se concentre sur la mise en lumiere d'acteurs cles dans les forums de piratage. Les acteurs cles sont definis comme etant les utilsateurs qui jouent un role crucial dans le marche des logiciels malveillants et des techniques de piratage.

Le systeme presente est compose de plusieurs blocs de traitement utilises pour faire la collecte de donnees et l'inference sur les categories d'acteurs.

1. Pre-traitement et collecte de donnees : Des outils de scrapping sont utilises afin de recoltes des donnees pertinentes sur les forums cibles.
2. Extraction de caracteristiques : Un module est utilise pour nettoyer le jeu de donnees recoltes et extraire des caracteristiques importantes
3. Modelistation des relations : Un algorithme est utilise pour modeliser les relations entre les donnees
4. Representation multivue : A partir des relations etablies, un reseau en graphe multivue est genere. Ce reseau se compose de plusieurs sous graphes qui representent une vue du graphe du point de vue d'un utilisateur.
5. Traitement par un GCN : Les differents sous-graphe alimentent des reseaux convolutifs en graphe qui apprenent leurs structures. Un mecanisme d'attention est utilise pour regrouper les differents resultats.
6. Representation dans un espace latent : Enfin, la representation vectorielle finale est un espace latent de noeuds utilises pour alimenter un classificateur. Ce dernier va alors etablir la correspondance avec les classes recherchees : acteurs cle et acteurs secondaires.

Les caracteristiques relationnelles considerees sont les suivantes :

- **user-post-thread**: Une matrice **P** ou chaque element $p_{i,j} \in 0, 1$ indique si l'utilisateur i a cree un fil (*thread*) j
- **user-write-reply**: Une matrice **W** ou chaque element $w_{i,j} \in 0, 1$ indique si l'utilisateur i ecrit une reponse j
- **user-make-comment**: Une matrice **M** ou chaque element $m_{i,j} \in 0, 1$ indique si l'utilisateur i ecrit une commentaire j
- **thread-belongto-section**: Une matrice **B** ou chaque element $b_{i,j} \in 0, 1$ indique si le *thread* i est dans la section j
- **reply-echo-thread**: Une matrice **E** ou chaque element $e_{i,j} \in 0, 1$ indique si la reponse i fait reference au *thread* j
- **comment-referto-user**: Une matrice **R** ou chaque element $r_{i,j} \in 0, 1$ indique si un commentaire i fait reference a l'utilisateur j

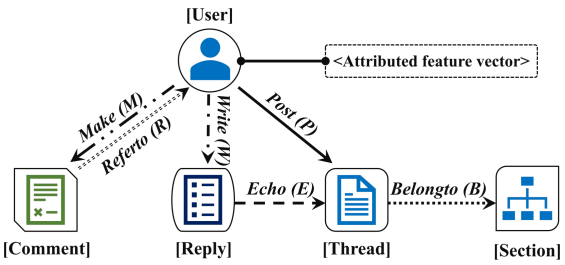

Figure 3: Representation d'un utilisateur et de ses caracteristiques.

A partir de ces caracteristiques, on peut alors representer le forum comme un reseau ou chaque noeud est l'identifient d'un utilisateur. Ces derniers sont connectes par plusieurs types d'aretes representant les differentes caracteristiques relationnelles enoncees plus haut. En partant d'un certain utilisateur, on peut se deplacer sur les aretes et retrouver un autre utilisateur.

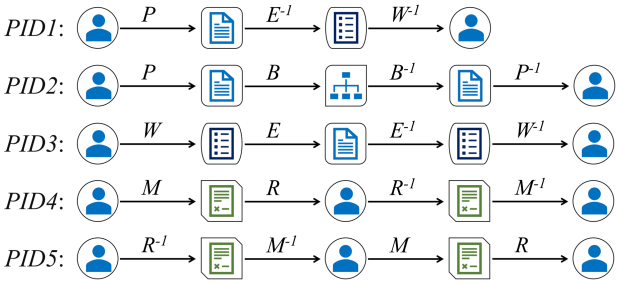

Figure 4: Relations entre les utilisateurs.

Ces differentes caracteristiques sont representees par un reseau multivue ou chaque sous-graphe represente la perspective d'un seul utilisateur du reseau.

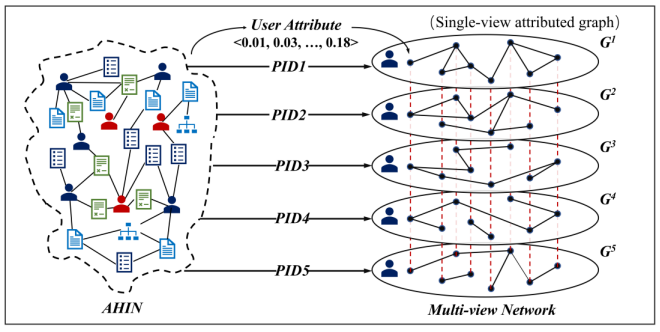

Figure 5: Reseau multivue construit.

Ces reseaux alimentent plusieurs GNN afin d'extraire les representations importantes de chaque sous-graphe. Les resultats sont regroupes grace a l'attention et fournis a un classificateur.

5 Experimentations

5.1 Implementations des modeles des articles

Le dataset utilise est le meme que nous avons adapte aux modeles des articles. Nous avons décidé d'utiliser le dataset S-FFSD accessible ici $https : //github.com/AI4Risk/antifraud/tree/main/data$ [4]. Ce dataset comporte une serie de transactions financieres entre des acteurs sources $Source$ et des acteurs cibles $Target$ avec plusieurs informations comme l'index de transaction $Time$, la valeur echangee $Amount$, l'endroit ou la transaction a lieu $Location$ et le type de transaction $Type$. Il y a aussi les labels decrivant si la transaction represente une fraude ou non. Le nombre D represente le nombre de caracteristiques differentes. Ce qui fait que chaque transaction x est represente comme un vecteur a D dimensions pour chaque valeur.

Nous avons effectué un prétraitement afin d'optmiser les performances sur nos modèles. En effet, le dataset était grandement déséquilibré en faveur des transactions non-frauduleuses

- **over-sampling**: Nous avons commencé par un over-sampling sur les transactions financières frauduleuses avec un ratio de 1/2. Cela a permit de réduire le déséquilibre sans pour autant créer un trop grand nombre de données(*thread*) j
- **under-sampling**: Afin de réduire d'avantage ce déséquilibre nous avons effectué un under-sampling sur les transactions non-frauduleuses avec un ratio de 1.0. Cela a permis d'avoir un équilibre total entre les transactions frauduleuses et non-frauduleuses.(*thread*) j
- **matrices d'adjacences**: Les GNNs sont des réseaux de neurones de graphes, il faut donc fournir des informations au modèle sous format de graphe. Pour cela nous pouvons fournir des matrices d'adjacence. Cela correspond à une matrice qui représente les intéractions entre les différents noeuds. Dans le cas des transactions financières, nous avons réalisé une matrice M dans laquelle $M[i][j]$ indique s'il existe une transaction entre le noeud i et le noeud j dans le jeux de données.(*thread*) j
- **matrices d'adjacences pondérées**: Comme expliqué dans le fonctionnement de Semi-GNN, il est possible de lui fournir plusieurs vues pour lui permettre de mieux apprendre. Nous avons donc créé des matrices d'adjacences pondérées par le montant des transactions. Ainsi une matrice M' dans laquelle $M'[i][j]$ indique le montant des transactions entre le noeud i et le noeud j.(*thread*) j

5.2 Recherche d'hyperparametres

Nous avons evalue les performances des modeles en jouant avec certains hyperparametres.

SemiGNN

Pour le modèle de SemiGNN nous avons décidé d'étudier les impacts des paramètres : de node embedding et de nombre d'epoch

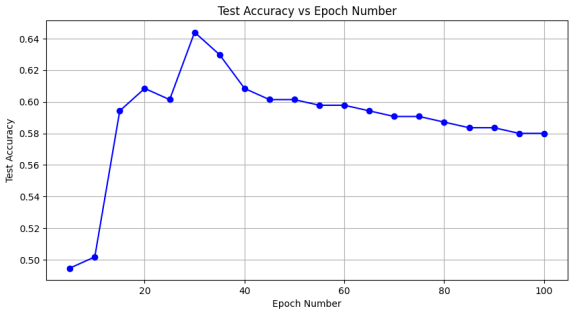

Figure 6: Evolution de l'accuracy en fonction de nombre d'epoch.

Comme on peut le voir, la meilleure valeur de nombre d'epoch sur nos données est de 30, au delà le modèle va faire du sur-apprentissage sur les données d'entraînement et donc moins bien performer sur les données de test, d'ou la baisse progressive de l'accuracy de test au delà de 30.

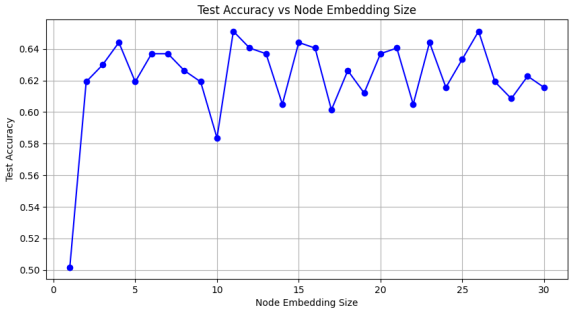

Figure 7: Evolution de l'accuracy en fonction de la taille d'embedding des noeuds.

On remarque qu'une fois une valeur minimale de 5 atteinte, l'accuracy de test ne varie plus, elle reste entre 0.6 et 0.64. Cela s'explique par le fait qu'après une certaine dimension d'embedding, la dimension devient suffisante pour stocker les informations pertinentes d'embedding de noeuds.

Player2Vec

Nous avons aussi evalue le modele Player2Vec sur son propre dataset. Nous avons donc joue avec certains hyperparametres du modele.

Nous pouvons observer que l'accuracy du modele atteint un maximum aux alentours de 40 epochs et reste assez constante apres. On constate une legere diminution au fur et a mesure, certainement du au sur-apprentissage du jeu de donnees.

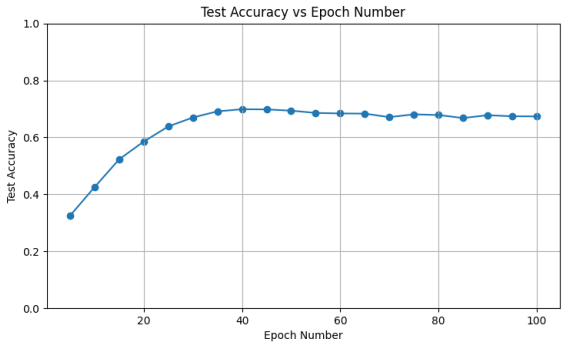

Figure 8: Evolution de l'accuracy du modele sur un jeu de donnees de test selon le nombre d'epochs.

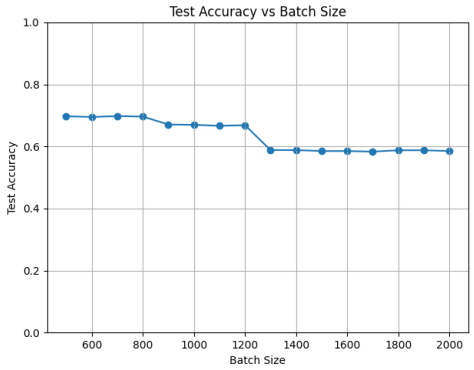

Figure 9: Evolution de la precision du modele sur un jeu de donnees de test selon la taille des mini-lots.

L'impact de la taille des mini-lots est tres importante. On peut voir qu'a partir de 1200, l'accuracy descend subitement de 10%. Nous avons emis une hypothese sur ce comportement et pensons que la taille trop importante des batchs empeche le modele d'apprendre suffisamment a chaque iteration. Les donnees sont trop distinctes entre elles pour que le modele converge efficacement.

5.3 Utilisation d'autres techniques d'apprentissage

Afin de verifier l'importance de l'utilisation d'un GNN pour traiter des donnees issues d'un reseau d'acteurs, nous avons decide de comparer les modeles issus des articles avec d'autres types de modeles d'apprentissage. Pour ce faire, nous avons utilise deux modeles : un modele de perceptron a plusieurs couche et un modele de melange gaussien.

Melange Gaussien [7]

Pour le melange gaussien, nous voulions voir si la detection de fraude pouvait etre reduite a un probleme d'apprentissage non-supervise. Afin de repondre a nos questions, nous avons decide de modeliser le dataset par un melange gaussien compose de deux distributions gaussiennes multivariees representees par les populations de donnees de non-fraude et de fraude. Les deux distributions sont definies comme [8]:

$$p(x|\mu,\Sigma) = \mathcal{N}(\mu,\Sigma) = \frac{1}{\sqrt{(2\pi)^D|\Sigma|}} e^{-(x-\mu_i)^T\Sigma_i^{-1}(x-\mu_i)} \tag{4}$$

avec μ le vecteur des moyennes dans chaque dimension et Σ la matrice de covariance.

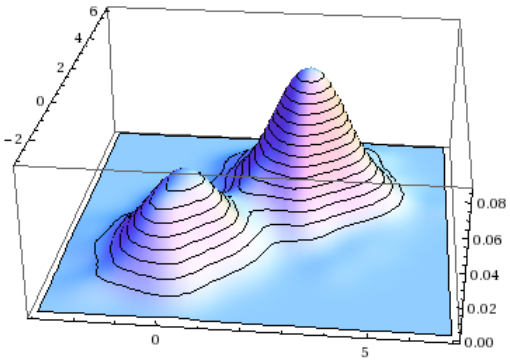

Figure 10: Representation quelconque d'un melange gaussien.

Dans notre cas de figure, afin de modeliser le jeu de donnees nous avons decide de representer notre modele gaussien avec 2 loi multivariees.

Perceptron multi couches

Le modele de perceptron utilise est un modele de reseau de neurones basique qui peut effectuer des taches de classification, par exemple une evaluation MNIST.

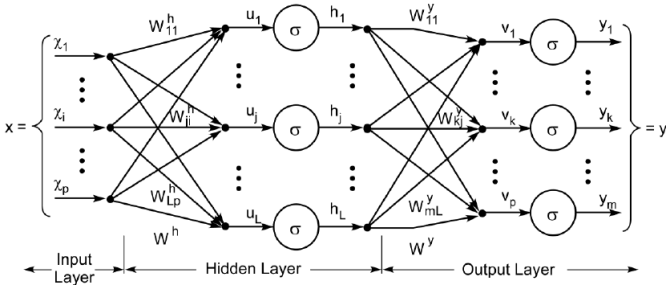

Figure 11: Representation quelconque d'un perceptron multicouche. [6]

Le reseau comporte 2 couches cachees de largeur 10 avec une fonction d'activation $ReLU$. La couche de sortie utilise la fonction $softmax$ pour retourner les probabilites associees aux classes a predire [5]:

$$\hat{y} = softmax(W^{n+1}(ReLU(W^n(...ReLU(W^0(\tilde{x}))) \tag{5}$$

La fonction objective utilisee est l'entropie croisee :

$$L_{CE} = -\Sigma_{i=1}^{n} \tilde{y} \log(\hat{y}) \tag{6}$$

Reseau de neurones en graphe

Nous avons essaye de construire nous-meme un modele de GNN afin de voir si on pouvait reussir a detecter des fraude parmi le meme dataset que celui utilise pour le SemiGNN. Apres quelques recherches, nous avons determine qu'il y avait 2 approches possibles [9] .

Parmi les GNN, on trouve les Graph Convolutional Networks (GCN) et les Graph Attention Networks (GAT).

- Graph Convolutional Network: Les GCN sont un type de GNN qui utilise le mecanisme de convolution sur les graphes [10].

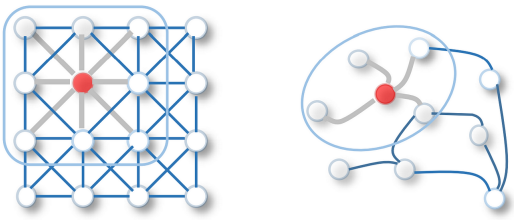

Figure 12: Convolution sur un noeud et ses noeuds adjacents.

Pour un ensemble de N donnees a F caracteristiques d'entree et pour F' classes de prediction:

$$h = \{\vec{h}_1, \vec{h}_2, ..., \vec{h}_N\}, \vec{h}_i \in \mathbb{R}^F \tag{7}$$

La propagation vers la couche k est de la forme :

$$\vec{h}_i^{(k)} = \sigma(\mathbf{W}^{(k)} \sum_{j \in \mathcal{N}(i) \cup \{i\}} \frac{h_j}{\sqrt{|\mathcal{N}(i)|\mathcal{N}(j)}}) \tag{8}$$

avec :

- Matrice de poids : $W \in \mathbb{R}^{F' \times F}$
- Le voisinage du noeud i : $\mathcal{N}_i$

- Graph Attention Network: Un GAT utilise le mecanisme d'attention pour capturer les relations entre les noeuds du graphe. [11]

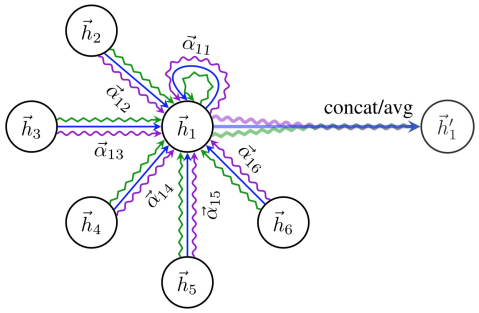

Figure 13: Calcul d'attention sur graphe.

Le calcul d'attention est :

$$\vec{h}_i^{'} = \sigma(\sum_{j \in \mathcal{N}_i} \alpha_{ij} \mathbf{W} \vec{h}_j^{'}) \tag{9}$$

avec les coefficients d'attention obtenus par la formule :

$$e_{ij} = \vec{a}^T[\mathbf{W}\vec{h}_i \oplus \mathbf{W}\vec{h}_j] \quad (10)$$

$$a_{ij} = softmax_j(e_{ij}) = \frac{\exp(e_{ij})}{\sum_{k\in\mathcal{N}_i}\exp(e_{ik})} \quad (11)$$

Afin d'experimenter avec les GNN, nous avons decide de modeliser 3 architectures de GNN:

1. GCN: Un reseau en graphe avec 3 couches cachees de convolution.
2. GAT: UN reseau attentionnel en graphe avec une couche cachee d'attention
3. GNN compose: Un reseau en graphe avec 2 couches de convolutions entrelacees de 2 couches d'attention.

La fonction objective utilisee dans chaque modele est l'entropie croisee et l'optimiseur est Adam. Le taux d'apprentissage est de 0.02.

6 Resultats

Modele	Accuracy
SemiGNN	0.644
SemiGNN$_{multi-vue}$	1
GCN	0.50
GAT	0.63
GNN$_{compos}$	0.75
Multilayer Perceptron	0.89
Gaussian Mixture	0.41

Table 1: Comparaison des acccuracy obtenues sur un ensemble de donnees de test sur le dataset S-FFSD.

Nous avons ainsi comparé les performances de 7 modèles. Nous avons utilisé l'accuracy pour les comparer car notre jeux de données, suite au pré-traitement expliqué, était équilibré entre les transactions frauduleuses et non-frauduleuses. Le modèle n'était donc pas influencé à prédire une classe plus qu'une autre et la mesure de l'accuracy reste donc pertinente. SemiGNN correspond au modèle présenté auquel nous avons uniquement fournis une vue, soit la matrice d'adjacence des transactions entre les différentes sources et les différentes targets. SemiGNN$_{multi-vue}$ correspond au même modèle, mais auquel nous avons également fournis une matrice d'adjacence pondérée grâce au montant des transactions afin de profiter du mécanisme d'attention entre les vues.

On remarque que le SemiGNN$_{multi-vue}$ performe mieux que le SemiGNN classique. Cela s'explique notamment par le fait qu'il obtient des informations plus riches sur les données, nous lui fournissons plus d'informations sur les liens qui existent entre les différents noeuds du graphe.

Les modeles de GNN semble assez bien performer. On voit que le GCN performe moins bien que le GAT, ce a quoi nous nous attendions. Le GNN compose de couches de convolution et d'attention a la meilleure accuracy parmi les 3. Pour autres les modeles, nous sommes assez surpris par la performance du MLP. Le melange gaussien a une performance tres mauvaise qui montre qu'il est difficile d'utiliser de l'apprentissage non-supervise pour la detection de fraude.

7 Critique de l'approche utilisee

Le domaine de la détection de fraude financière est un domaine très varié et en pleine expansion. De plus, il s'agissait d'un sujet très intéressant tant par les modèles utilisés que par le but de détecter les fraudes financières. Cependant, il s'agit également d'un domaine dans lequel les données ne sont pas en libre accès. En effet, les données bancaires sont sensibles et confidentielles. Ce qui nous a posé un défi significatif. Il a été très compliqué de trouver un jeux de données exploitable et compréhensible (certains jeux de données de transactions financières accessibles sont le résultat d'une décomposition en composantes principales ou autres traitement, ce qui rend les données exploitables mais non compréhensibles).

Également, nous étions intéressé par les deux aspects de détection des GNNs, détecter les intrus/fraudes et détecter les acteurs principaux. C'est pourquoi nous nous sommes orientés vers les deux articles présentés dans ce rapport. Cependant, en nous focalisant sur un seul aspect de la détection, nous aurions pu allouer nos ressources différemment. Étudier les deux aspects simultanément a demandé une charge de travail beaucoup plus importante.

8 Conclusion

Nous avons présenté la détection de fraudes dans un contexte moderne, nous avons fait un résumé des travaux antérieurs dans la littérature. Nous avons ensuite expliqué l'intérêt des GNN dans ces contextes relationnels/interactionnels, notamment comment les GNN sont capables d'utiliser les informations de ces interactions, contrairement aux autres méthodes. Nous avons par la suite présenté deux modèles d'articles de recherches. Nous avons expliqué les expérimentations que nous avons mis en place. Finalement nous avons présenté les résultats et avons apporté une analyse critique de l'approche que nous avons utlisé.

References

[1] Wang, D., Lin, J., Cui, P., Jia, Q., Wang, Z. (2019). A semi-supervised graph attentive network for financial fraud detection. In IEEE International Conference on Data Engineering (ICDE) https://arxiv.org/pdf/2003.01171.pdf.

[2] Zhang, Y., Fan, Y., Ye, Y., Zhao, L., Shi, C. (2019). Key player identification in underground forums over attributed heterogeneous information network embedding framework. Proceedings of the 28th ACM International Conference on Information and Knowledge Management (CIKM) https://mason.gmu.edu/ lzhao9/materials/papers/lp0110-zhangA.pdf.

[3] safe-graph. (n.d.). DGFraud: [A Python package for fraud detection with deep learning]. Retrieved in April 2024, from https://github.com/safe-graph/DGFraud.

[4] AI4Risk. (n.d.). AntiFraud data. Retrieved in April 2024, from https://github.com/AI4Risk/antifraud/tree/main/data

[5] GLAZERadr. 2023. Multi Layer Perceptron Using Pytorch from https://github.com/GLAZERadr/Multi-Layer-Perceptron-Pytorch

[6] AILEPHANT. (n.d.). Multilayer Perceptron from https://ailephant.com/glossary/multilayer-perceptron/

[7] Carrasco, O. 2024. Gaussian Mixture Model Explained from https://builtin.com/articles/gaussian-mixture-model:˜:text=A%20Gaussian%20mixture%20model%20is%20a%20soft%20clustering%20technique%20used,clusters%20in%20a%20data%20set.

[8] Looney, O. 2019. ML From Scratch, Part 5: Gaussian Mixture Models from https://www.oranlooney.com/post/ml-from-scratch-part-5-gmm/

[9] OMS1996. 2022. Graph Classification with Graph Neural Networks. from https://github.com/OMS1996/pytorch-geometric/blob/main/3_Graph_Classification.ipynb

[10] (n.d.). The Graph Neural Network Model. from https://www.cs.mcgill.ca/ wlh/grl_book/files/GRL_Book-Chapter_5-GNNs.pdf

[11] Salehi Y. (n.d.). Graph Attention Networks from https://cs.mcgill.ca/ wlh/comp766/files/yasmin_salehi.pdf