Indoor Localization in IoT Networks Based on Graph Neural Networks

GUO Xiaofan

Prof. Wafa NJIMA

Table des matières

Chapitre 1 Introduction

Avec le développement rapide de l'Internet des objets (Internet of Things, IoT) et des dispositifs portables intelligents, la demande pour les services basés sur la localisation (Location-Based Services, LBS) ne cesse de croître, en particulier dans des environnements densément peuplés tels que les centres commerciaux, les parkings, les hôpitaux, les usines et les aéroports.[1]

Bien que le système de positionnement par satellites (Global Navigation Satellite System, GNSS) soient largement utilisés pour la localisation en extérieur, leur utilisation en intérieur est limitée en raison des obstacles, de l'atténuation ou de la réflexion des signaux, les rendant peu fiables. Par conséquent, des technologies telles que le Bluetooth, le Wi-Fi, l'Ultra-Wide band (UWB) et les radiofréquences sont devenues les principales alternatives pour la localisation en intérieur. [2][5]

En tant que composante essentielle de l'Internet industriel, la technologie de localisation en intérieur vise à résoudre le problème de l'impossibilité d'utiliser efficacement le Global Positioning System (GPS) en environnement intérieur en raison des blocages liés aux bâtiments. Cette technologie permet de suivre et de gérer en temps réel le personnel dans des environnements industriels réels, tout en surveillant avec précision chaque étape de la production automatisée, contribuant ainsi de manière significative à l'amélioration de l'efficacité de la production et à la digitalisation des processus industriels. [1]

Ces dernières années, les méthodes de localisation en intérieur basées sur l'apprentissage automatique sont devenues un sujet de recherche majeur.

Parmi elles, les réseaux de neurones profonds (Deep Neural Networks, DNN) se distinguent par leur capacité exceptionnelle à modéliser des relations non linéaires et leur grande précision en matière de localisation. Cependant, les DNN présentent certaines limites, notamment une forte dépendance à des ensembles de données volumineux et des difficultés à traiter des relations spatiales complexes et des caractéristiques de propagation des signaux. Pour surmonter ces défis, les réseaux de neurones graphiques (Graph Neural Networks, GNN) apparaissent comme une solution innovante. Les GNN excellent dans la modélisation des données structurées en graphes, capturant efficacement les relations spatiales entre les points d'accès Wi-Fi et les caractéristiques sous-jacentes de propagation des signaux. Ils offrent ainsi un

potentiel d'amélioration en termes de précision de localisation et d'efficacité computationnelle par rapport aux modèles DNN traditionnels. [3][4]

L'objectif de ce projet est de comparer, à travers des expériences, les performances des DNN et des GNN dans le cadre de la localisation en intérieur, en évaluant en particulier leur précision et leur complexité computationnelle. Basé sur le jeu de données public UJIIndoorLoc, ce travail vise à développer des modèles de localisation en intérieur au sein d'un même bâtiment à l'aide des DNN et des GNN, puis à analyser leurs avantages et limites respectifs et ambitionne d'explorer le potentiel des GNN dans des environnements de localisation complexes, tout en fournissant des bases théoriques et des références pratiques pour l'amélioration des technologies de localisation en intérieur dans le cadre de l'Internet industriel.

Chapitre 2 La localisation indoor

1 Contexte

Avec le développement rapide de l'IoT, il occupe une place de plus en plus importante dans notre vie quotidienne. Dans les environnements intérieurs, les besoins en technologies de localisation augmentent, avec des applications variées allant des bâtiments intelligents aux centres commerciaux, en passant par les hôpitaux, etc. Cependant, les systèmes de localisation intérieure actuels présentent encore des lacunes en termes de précision et de fiabilité, ce qui limite leur capacité à répondre à la demande croissante en matière de navigation et de LBS.[1]

Les principaux défis de la localisation intérieure incluent les aspects suivants :

- **Traitement de données multidimensionnelles** : par exemple, des signaux RSSI (Received Signal Strength Indication) comportant plus de 500 dimensions. Ces données sont complexes, leur traitement représente un défi important.
- **Complexité des caractéristiques des coordonnées géographiques** : notamment la longitude, la latitude et l'étage, qui varient souvent de manière significative selon l'environnement et nécessitent une modélisation précise.
- **Optimisation de la précision des prédictions** : il est essentiel d'améliorer la sensibilité et l'exactitude des prédictions, en particulier pour les variations subtiles des données de localisation, afin de répondre aux exigences des scénarios d'utilisation réels.

Pour relever ces défis, ce projet vise à explorer l'efficacité de deux approches d'apprentissage profond dans le cadre de la localisation intérieure :

1. **Les réseaux de neurones profonds (DNN)** : capables de traiter des données complexes et multidimensionnelles, ces réseaux permettent d'extraire efficacement des caractéristiques à partir de larges ensembles de données.
2. **Les réseaux de neurones graphiques (GNN)** : en construisant une structure en graphe, ces réseaux capturent les relations spatiales entre les points d'accès Wi-Fi, améliorant ainsi la capacité du modèle à traiter des problèmes de localisation.

Ce projet se base sur la base de données UJIIndoorLoc, largement utilisée dans la recherche sur la localisation intérieure. En comparant les performances des DNN et des GNN pour la localisation intérieure, ce projet a pour objectif de discuter des problèmes liés à la précision de la localisation intérieure et de fournir des bases solides pour l'amélioration des services de localisation.

2 L'Internet des Objets (IoT)

L'Internet des Objets (IoT) est considéré comme une avancée de l'Internet, dont l'objectif principal est de connecter le monde réel avec les technologies numériques. Grâce à l'IoT, les objets physiques peuvent être interconnectés via des réseaux de communication, permettant ainsi la génération et le partage de données. Le développement rapide de cette technologie a apporté de nombreuses opportunités dans divers domaines et a également eu un impact significatif sur les systèmes de localisation en intérieur.

Dans le domaine de la localisation en intérieur, l'IoT offre un soutien considérable. Par exemple, en déployant des capteurs et des équipements, l'IoT permet de capter en temps réel les environnements complexes à l'intérieur des bâtiments et de traiter des données multidimensionnelles.

Cependant, l'intégration de l'IoT dans la localisation en intérieur rencontre également certains défis, notamment les problèmes de confidentialité des données, la complexité de l'intégration des systèmes et les coûts élevés liés aux équipements et à leur maintenance.

1. L'IoT repose sur trois composantes principales [1]:
   - **Objets physiques** : les équipements ou capteurs qui collectent et transmettent des données.
   - **Réseaux de communication** : qui assurent l'interconnexion entre les équipements pour une transmission en temps réel des données.
   - **Réseaux de traitement des données** : qui analysent et modélisent les données collectées pour en extraire des informations pertinentes.
2. Avantages de l'IoT dans la localisation en intérieur [1]:
   - **Localisation précise** : Les besoins en localisation se divisent en position absolue (comme les coordonnées géographiques) et position relative (comme la position par rapport à des objets spécifiques dans

l'environnement). Les technologies les plus récentes permettent d'atteindre une précision de localisation inférieure à 1 mètre, ce qui est essentiel pour des tâches comme la navigation ou le suivi dans des environnements complexes.

- **Réactivité en temps réel** : L'IoT peut capter les changements de l'environnement en temps réel, soutenant les scénarios dynamiques.
- **Applications étendues** : L'IoT s'applique à divers domaines tels que la santé, les bâtiments intelligents, la surveillance de la sécurité et la maintenance industrielle.

3. Inconvénients de l'IoT dans la localisation en intérieur :
   - **Problèmes de confidentialité des données** : L'IoT implique une collecte et une transmission constantes de données, ce qui peut exposer les utilisateurs à des risques liés à la vie privée.
   - **Complexité des systèmes** : Le déploiement des équipements, la gestion des réseaux et la maintenance nécessitent un haut niveau d'expertise technique.
   - **Coûts élevés** : Les coûts associés à l'achat, à l'installation et à l'exploitation des équipements IoT peuvent constituer une barrière pour certains projets ou organisations aux ressources limitées.

À l'avenir, avec le développement et l'amélioration constants des technologies IoT, leur intégration avec la localisation en intérieur permettra d'améliorer encore davantage la précision et la réactivité tout en ouvrant la voie à des applications dans des scénarios plus larges.

3 Réseaux Neuronaux Profonds（DNN）

Les réseaux neuronaux profonds (Deep Neural Networks, DNN) sont utilisés pour traiter des données multidimensionnelles complexes, permettant ainsi une classification et des prédictions efficaces et précises[2]. DNN se distinguent par la présence de plusieurs couches cachées, capables d'appliquer des transformations non linéaires successives aux données d'entrée. Cette structure permet de progressivement extraire des caractéristiques complexes et d'apprendre des relations de haut niveau entre les données. Les DNN capturent les caractéristiques des données d'entrée, ce qui les rend adaptés à des tâches telles que la classification et la régression[3].

1. Structure Théorique [4]:
   - **Couche d'entrée** (Input Layer) : Reçoit les données d'entrée.
   - **Couches cachées** (Hidden Layers) : Constituées de plusieurs couches entièrement connectées, chaque couche contenant un certain nombre de neurones. Ces couches réalisent une combinaison pondérée des entrées suivie de l'application d'une fonction d'activation, permettant d'extraire et d'apprendre des caractéristiques.
   - **Couche de sortie** (Output Layer) : Détermine la sortie.
   - **Fonction de perte** (Loss Function) : Utilisée pour mesurer l'écart entre les prédictions du modèle et les valeurs réelles.
2. Avantages [3] :
   - **Traitement efficace de données multidimensionnelles** : Les DNN sont capables de gérer des données complexes et à haute dimension.
   - **Extraction de caractéristiques profondes** : Grâce aux transformations non linéaires des couches cachées, les DNN peuvent extraire des caractéristiques abstraites et de haut niveau à partir des données.
   - **Forte capacité de généralisation** : Avec un support de données suffisant, les DNN offrent de bonnes performances dans des scénarios complexes.

4 Réseaux Neuronaux Graphiques（GNN）

Les réseaux neuronaux graphiques (Graph Neural Networks, GNN) sont des architectures spécialisées en apprentissage profond conçues pour traiter des données représentées sous forme de graphes. Elles exploitent la structure inhérente des graphes, ce qui les rend particulièrement adaptées aux tâches où les relations entre entités sont primordiales.

1. Architecture typique :
   - **Couches Permutation Équivariantes** (Permutation Equivariant Layers) : implémentées comme des couches de passage de messages (Message Passing).
   - **Couches de Pooling Local** (Local Pooling Layers) : simplifie le graphe en réduisant sa taille (downsampling).

- **Couches de Pooling Global** (Global Pooling Layers) : fournit une représentation de taille fixe pour l'ensemble du graphe.

2. Avantages des GNN[5] :
   - **Adaptation aux Environnements Intérieurs Complexes** : La méthode de convolution dynamique capture bien les relations locales dans les données RSSI, ce qui rend les GNN efficaces dans des environnements intérieurs complexes.
   - **Performance stable avec peu de données** : Même avec peu de données d'entraînement, les GNN gardent une grande précision et restent fiables.
   - **Applications** : Les GNN peuvent être utilisés pour la navigation, le suivi d'objets, la gestion de bâtiments et des services personnalisés.

5 Base de Données : UJIIndoorLoc

En tant que première base de données publique et la plus grande dans le domaine de la localisation et navigation en intérieur, UJIIndoorLoc permet de comparer différents algorithmes de localisation en intérieur, se distingue par ses caractéristiques suivantes : couverture multi-bâtiments et multi-étages, données diversifiées, grand volume d'échantillons, accès ouvert, et une méthode basée sur l'empreinte WLAN (WLAN fingerprinting).[6]

Dans ce projet, pour le positionnement basé sur DNN et GNN, les méthodes suivantes sont utilisées pour prétraiter les données de la base de données :

5.1 Fonctions d'Activation

ELU (Exponential Linear Unit) et ReLU (Rectified Linear Unit) sont des fonctions d'activation couramment utilisées dans les réseaux de neurones. Elles introduisent de la non-linéarité, permettant au modèle d'apprendre des caractéristiques plus complexes.

Bien que ReLU et ELU produisent une sortie identique pour les valeurs d'entrée positives, elles diffèrent pour les valeurs négatives : ReLU fixe la sortie à 0 pour $x \leq 0$, tandis qu'ELU génère des valeurs négatives. Cela rend ELU plus adaptée aux données d'entrée comprenant des valeurs négatives.

Dans les fichiers de données utilisés pour ce modèle, les valeurs RSSI sont toutes négatives. Par conséquent, la fonction d'activation ELU a été choisie pour mieux s'adapter à cette distribution, renforçant ainsi la stabilité et l'efficacité de l'apprentissage du modèle.

La fonction d'activation ELU est définie comme suit, où α est une constante positive :

$$ELU(x) = \begin{cases} x, & x > 0 \\ \alpha(e^{x} - 1), & x \leq 0 \end{cases} \qquad (2.1)$$

Cette capacité d'ELU à produire des valeurs négatives pour $x \leq 0$ permet de limiter le problème de disparition du gradient et d'améliorer la convergence du modèle. De plus, la sortie d'ELU permet aux moyennes des sorties des couches cachées de se rapprocher de 0, accélérant ainsi la vitesse d'entraînement du réseau.

5.2 Fonction de Perte

MSE (Mean Squared Error) est une fonction de perte couramment utilisée pour les tâches de régression. Elle est définie par la formule suivante :

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \qquad (2.2)$$

Ce projet implique la prédiction de la longitude et de la latitude. La MSE est utilisée pour quantifier la différence entre les valeurs prédites et les valeurs réelles.

En minimisant la MSE, le modèle est incité à réduire les erreurs importantes, ce qui améliore la précision des prédictions de la longitude, de la latitude.

5.3 Optimiseur

Adam est un optimiseur adaptatif basé sur les estimations du premier et du second moment. En ajustant dynamiquement le taux d'apprentissage, Adam permet d'accélérer la vitesse de convergence du modèle tout en réduisant les fluctuations lors de la descente de gradient.

Dans ce projet, l'optimiseur Adam a amélioré l'efficacité de l'entraînement et a réduit le temps de formation du modèle DNN.

5.4 Prétraitement des Données

1. Normalisation

MinMaxScaler est une méthode de prétraitement des données qui permet de mettre à l'échelle les caractéristiques dans l'intervalle [0,1], tout en conservant les proportions relatives des valeurs des caractéristiques. Sa formule de transformation est la suivante :

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}} \qquad (2.3)$$

Dans ce projet, MinMaxScaler est utilisé pour normaliser les données RSSI, permettant au modèle de mieux gérer les caractéristiques de différentes échelles et de réduire l'impact des différences d'ordre de grandeur entre les caractéristiques sur l'apprentissage du modèle. De plus, elle est également appliquée aux données de longitude et de latitude, en traitant les coordonnées géographiques séparément, le modèle est en mesure de prédire des valeurs de longitude et de latitude plus précises.

2. Dénormalisation

La dénormalisation permet de ramener les valeurs normalisées de sortie du modèle à leur échelle d'origine, rendant ainsi les prédictions interprétables et pertinentes dans un contexte réel. La formule de cette opération est la suivante :

$$X_{original} = X_{norm} \cdot (X_{max} - X_{min}) + X_{min} \qquad (2.4)$$

Dans ce projet, la dénormalisation est utilisée pour ramener les valeurs prédites de longitude et de latitude du modèle de l'intervalle normalisé [0,1] à leurs coordonnées géographiques d'origine. Cela permet de comparer directement les prédictions du modèle aux valeurs réelles, ce qui permet d'évaluer la précision du modèle.

Les principales méthodes utilisées uniquement dans les méthodes DNN ou uniquement dans les méthodes GNN seront présentées en détail dans les chapitres correspondants.

La méthode DNN et la méthode GNN proposées dans ce rapport sont testées sur la base de données UJIIndoorLoc. Elles traitent les données RSSI (les 519 premières colonnes de chaque fichier de données) pour prédire la longitude, la latitude et l'étage des différentes coordonnées dans un même bâtiment. Les résultats sont comparés avec les valeurs de référence (colonnes 520 à 522 de chaque fichier de données). Les modèles fournissent également le nombre d'erreurs de prédiction d'étage ainsi que la distance d'erreur en mètres pour les coordonnées de longitude et de latitude.

Chapitre 3 Méthode de Localisation Basée sur le DNN

Dans l'application de localisation en intérieur, l'architecture DNN intègre les données hiérarchiques des bâtiments, étages et positions spécifiques, offrant une performance de localisation stable et une grande extensibilité, adaptée aux environnements complexes multi-bâtiments et multi-étages[2].

1 Fonctions Clés

Pour le prétraitement des données, d'autres méthodes clés ont déjà été présentées dans la section 5 du chapitre 1, Base de Données : UJIIndoorLoc, telles que les fonctions d'activation, les fonctions de perte et les optimiseurs. Cette section met l'accent sur une technique utilisée uniquement dans le modèle DNN : la régularisation par Dropout.

1.1 Technique de Régularisation

Le Dropout est une technique de régularisation qui consiste à supprimer aléatoirement certaines sorties de neurones (en les fixant à 0) pendant l'entraînement, réduisant ainsi la dépendance entre les neurones, empêchant le surapprentissage et améliorant la capacité de généralisation du modèle.

Cette technique permet au modèle d'utiliser différentes combinaisons de neurones à chaque itération, ce qui équivaut à entraîner plusieurs sous-modèles et renforce ainsi la robustesse et la stabilité des prédictions.

2 Structure du DNN

Pour le traitement des données d'un même bâtiment (building 0), un modèle de DNN a été conçu, intégrant dans sa structure principale trois réseaux de neurones de régression. Chaque réseau est indépendant et dédié spécifiquement au traitement et à la prédiction de trois variables spatiales distinctes : la longitude, la latitude et l'étage.

2.1 Modèle d'Encodage

Dans l'apprentissage automatique, un Encoder est une structure utilisée pour extraire des caractéristiques de faible dimension à partir d'entrées de haute dimension. Il est principalement employé pour la compression des caractéristiques et l'apprentissage des modèles. En revanche, une structure non-Encoder traite directement les données d'entrée brutes et de haute dimension, ce qui limite souvent ses capacités d'extraction et de compression des caractéristiques.

2.1.1 Avantages de l'Encoder

1) Amélioration de la généralisation du modèle

L'Encoder aide à réduire les risques de surapprentissage (overfitting), améliorant ainsi la capacité du modèle à s'adapter à de nouvelles données.

2) Extraction des caractéristiques significatives

L'Encoder peut transformer des données de haute dimension en représentations de faible dimension, réduisant les redondances et les bruits.

3) Partage des caractéristiques

Les caractéristiques extraites par l'Encoder peuvent être partagées entre plusieurs tâches, ce qui est particulièrement utile pour les modèles multitâches.

Dans ce projet, les données utilisées proviennent de signaux RSSI de haute dimension (519 dimensions). De plus, le projet vise à résoudre un problème multitâche nécessitant la prédiction simultanée de la longitude, de la latitude et de l'étage. Par conséquent, l'utilisation d'une structure Encoder est adaptée pour répondre à ces exigences.

2.1.2 Structure de base d'un Encoder

1) Couche d'entrée (Input layer)

Cette couche reçoit des données d'entrée de haute dimension, comme les 519 dimensions des signaux RSSI dans ce projet.

2) Couche d'encodage (Hidden layer)

- Une série de couches entièrement connectées (Dense layers) est utilisée pour appliquer des transformations non linéaires aux données.

- Le nombre de neurones dans chaque couche diminue progressivement pour compresser les données en une représentation de faible dimension.
- Fonctions d'activation courantes qui permettent de capturer les relations complexes dans les données.

3) Couche de sortie (Encoded output layer)

Cette couche fournit les caractéristiques compressées, qui sont ensuite utilisées pour des tâches ultérieures telles que la classification ou la régression.

Dans ce projet, le modèle comprend deux couches d'encodage et deux couches de décodage.

- Les couches d'encodage sont constituées de deux couches entièrement connectées, contenant respectivement 256 et 64 neurones, avec la fonction d'activation ELU.
- Les couches de décodage restaurent les données pour reconstruire les caractéristiques d'entrée.

Ce modèle est utilisé pour extraire des caractéristiques des données RSSI de haute dimension et pour compresser la représentation, ce qui permet de réduire le bruit et les informations redondantes pendant la phase d'encodage, améliorant ainsi la précision des prédictions de longitude, latitude et étage tout en réduisant la charge de calcul.

2.2 Réseau de Régression

2.2.1 La longitude

Ce réseau de régression est spécialement conçu pour prédire la longitude, en effectuant des calculs de régression basés sur les caractéristiques extraites par les couches d'encodage.

Il comprend deux couches entièrement connectées avec la fonction d'activation ELU, une couche Dropout, et une couche de sortie à un seul neurone pour fournir le résultat de la prédiction de la longitude.

2.2.2 La latitude

Ce réseau de régression est spécialement conçu pour prédire la latitude, en effectuant des calculs de régression basés sur les caractéristiques extraites par les couches d'encodage.

Il comprend deux couches entièrement connectées avec la fonction d'activation ELU, une couche Dropout, et une couche de sortie à un seul neurone pour fournir le résultat de la prédiction de la latitude.

2.2.3 L'étage

Ce réseau de régression est conçu pour prédire l'étage.

Il comprend deux couches entièrement connectées avec la fonction d'activation ELU, ainsi qu'une couche de sortie à un seul neurone utilisant une fonction d'activation linéaire pour prédire le numéro de l'étage.

2.3 Couche de Détection d'Erreur

La couche de détection d'erreur est utilisée pour évaluer la précision des prédictions du modèle, en calculant l'écart entre les valeurs prédites et les valeurs réelles de la longitude, de la latitude et de l'étage.

La fonction error utilise l'erreur quadratique moyenne (Root Mean Square Error, RMSE) pour mesurer les résultats de prédiction des trois paramètres. RMSE reflète efficacement l'écart moyen entre les valeurs prédites et les valeurs réelles, les grandes erreurs ayant un impact plus important sur le résultat final.

$$Longitude\ Error\ = \sqrt{\left(\hat{y}_{lng} - y_{lng}\right)^2} \qquad (3.1)$$

$$Latitude\ Error\ = \sqrt{(\hat{y}_{lat} - y_{lat})^2} \qquad (3.2)$$

$$Floor\ Error\ = \sqrt{\left(\hat{y}_f - y_f\right)^2} \qquad (3.3)$$

Pour que l'erreur d'étage ait un poids plus important dans l'erreur totale, un coefficient de pénalité floor_penalty est ajouté à l'erreur d'étage, ce qui permet au modèle d'accorder plus d'importance à la précision des prédictions d'étage.

$$Floor\ Error\ Sum\ = \sum Floor\ Error \qquad (3.4)$$

$$Coordinates\ Error\ Sum\ = \sum(Longitude\ Error + Latitude\ Error) \qquad (3.5)$$

$$Total\ Error = floor_penalty \times Floor\ Error\ Sum + Coordinates\ Error\ Sum$$

Finalement, la fonction error génère des résultats comme Average Error in meters, utilisés pour l'analyse et l'évaluation des erreurs dans les étapes ultérieures.

$$Average\ Error\ = \frac{Total\ Error}{N} \qquad (3.6)$$

En calculant séparément les erreurs de longitude, de latitude et d'étage, et en utilisant un coefficient de pénalité pour ajuster le poids de l'erreur d'étage, la

fonction error permet une évaluation plus complète de la performance du modèle en termes de localisation. Elle prend en compte la précision continue des coordonnées géographiques tout en garantissant l'exactitude des prédictions d'étage, améliorant ainsi la performance globale de localisation du modèle.

2.4 Justification des Choix de Conception

Dans ce projet, pour la localisation au sein d'un même bâtiment, les données de latitude varient dans une plage relativement restreinte, tandis que les données de longitude montrent des variations plus importantes, avec des écarts de valeurs considérables entre les deux. Si ces deux paramètres ne sont pas traités séparément, le modèle risque d'ignorer les variations plus faibles de la longitude lors de l'entraînement. Lorsque la latitude et la longitude sont traitées par un même modèle, la latitude est prédite comme une valeur fixe.

Par conséquent, ce projet a conçu des réseaux indépendants pour la longitude, la latitude et l'étage, afin de garantir que le modèle puisse pleinement apprendre les particularités de chaque paramètre et effectuer des prédictions précises.

3 Paramètres du Modèle

Dans le processus d'entraînement du DNN, il existe cinq paramètres importants pouvant être changés :

3.1 Dimension d'entrée (input)

La dimension d'entrée est directement déterminée par la structure du jeu de données.

Dans ce projet, chaque ligne de données contient 519 caractéristiques RSSI, ce qui définit la dimension d'entrée à 519 afin de garantir que le modèle puisse recevoir toutes les informations de caractéristiques.

3.2 Taux de Dropout (dropout)

Le dropout contrôle la proportion de neurones désactivés aléatoirement. Augmenter le taux de dropout peut éviter le surapprentissage, tandis que le réduire permet de conserver davantage d'informations.

Dans ce projet, le taux de dropout est défini à 0,5.

3.3 Époques (epochs)

L’epochs déterminent le nombre d'itérations sur l'ensemble de données.

Un nombre élevé d'itérations permet au modèle d’apprendre plus en profondeur les caractéristiques des données ; si un surapprentissage est détecté, il est possible de réduire cette valeur.

3.4 Taille de Lot (batch_size)

La taille des lots (batch size) détermine le nombre d'échantillons utilisés pour chaque mise à jour des poids.

Dans la limite de la mémoire disponible, il est possible d'augmenter la taille des lots pour accélérer l'entraînement ; une taille de lot plus petite est préférable pour un jeu de données plus réduit et peut aider à réduire le surapprentissage.

3.5 Coefficient de pénalisation (floor_penalty)

Le floor_penalty détermine le poids de l'erreur d'étage dans l'erreur totale. Un coefficient de pénalité plus élevé permet au modèle d'accorder plus d'importance à la précision des prédictions d'étage.

Dans ce projet, le floor_penalty est défini à 4.

4 Analyse des Paramètres et Résultats

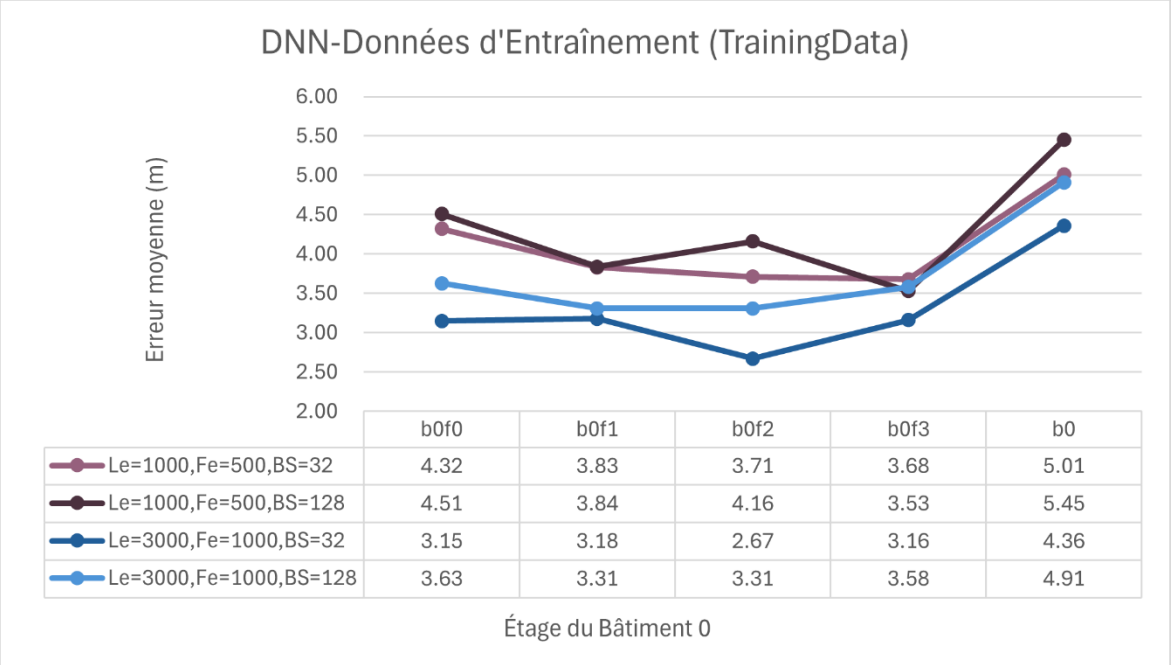
Dans ce projet, seuls les deux paramètres epochs et batch_size sont ajustés et testés.

Étant donné que le nombre de données pour la longitude et la latitude est toujours le même et que celles-ci sont plus complexes, les paramètres epochs pour la longitude et la latitude sont définis de manière égale (longitude_epochs = latitude_epochs), tandis que les floor_epochs sont fixés à une valeur plus petite.

Dans les légendes des graphiques, les abréviations suivantes sont utilisées :

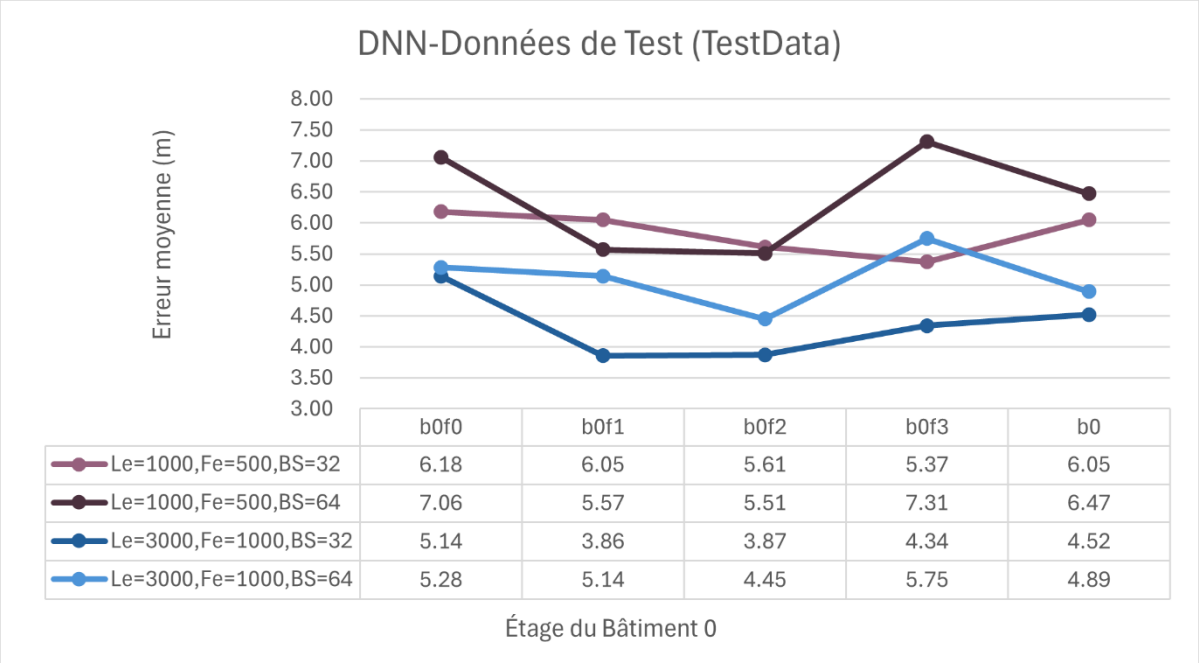
- Le : longitude et latitude epochs
- Fe: floor_epochs
- BS: batch_size

Les résultats sont les suivants :



	b0f0	b0f1	b0f2	b0f3	b0
Le=1000,Fe=500,BS=32	4.32	3.83	3.71	3.68	5.01
Le=1000,Fe=500,BS=128	4.51	3.84	4.16	3.53	5.45
Le=3000,Fe=1000,BS=32	3.15	3.18	2.67	3.16	4.36
Le=3000,Fe=1000,BS=128	3.63	3.31	3.31	3.58	4.91

Figure 1 DNN-Résultats d'erreur moyenne pour les données d'entraînement



	b0f0	b0f1	b0f2	b0f3	b0
Le=1000,Fe=500,BS=32	6.18	6.05	5.61	5.37	6.05
Le=1000,Fe=500,BS=64	7.06	5.57	5.51	7.31	6.47
Le=3000,Fe=1000,BS=32	5.14	3.86	3.87	4.34	4.52
Le=3000,Fe=1000,BS=64	5.28	5.14	4.45	5.75	4.89

Figure 2 DNN-Résultats d'erreur moyenne pour les données de test

Les conclusions suivantes peuvent être tirées des graphiques :

4.1 Effet des époques (Epochs)

De manière générale, une augmentation du nombre d'époques conduit à une diminution de l'erreur moyenne. Cela montre que le modèle est capable d'apprendre davantage les caractéristiques des données avec plus d'itérations.

Cependant, une trop grande augmentation peut entraîner un risque de surapprentissage, en particulier avec de grands lots.

Dans les données testées, avec les autres paramètres (BS) identiques, la combinaison des valeurs $Le = 3000$ et $Fe = 1000$ donne des résultats nettement meilleurs que la combinaison $Le = 1000$ et $Fe = 500$.

4.2 Impact de la taille des lots (Batch Size)

Dans la majorité des cas, des lots de petite taille ($Batch\ Size, BS = 32$) entraînent des erreurs moyennes plus faibles. Cela peut s'expliquer par le fait que de petits lots permettent une meilleure capture des variations subtiles dans les données, notamment dans des ensembles de données complexes.

En revanche, des lots plus grands ($BS = 64\ ou\ BS\ =\ 128$) offrent une stabilité mais peuvent augmenter légèrement l'erreur moyenne.

4.3 Analyse par étage

Lorsque les données sont regroupées par étage ($b0f0, b0f1, b0f2, b0f3$), les performances du modèle sont plus stables et les erreurs restent faibles.

Pour le bâtiment complet ($b0$) où les données de différents étages sont mélangées, une augmentation de l'erreur est observée, probablement due à la complexité accrue des données. Cependant, l'erreur globale reste dans une plage acceptable (moins de 5 mètres).

4.4 Meilleure combinaison de paramètres

Parmi toutes les combinaisons testées, le meilleur résultat est les paramètres : $Le = 3000, Fe = 1000, BS = 32$.

Chapitre 4 Méthode de Localisation Basée sur le GNN

Dans l'application de localisation en intérieur, l'architecture GNN modélise les données RSSI sous forme de graphe, où chaque échantillon est représenté par un nœud et les relations entre les nœuds sont définies en fonction des similitudes entre les vecteurs RSSI, calculées à l'aide de l'algorithme des k-plus proches voisins (k-NN). Cette structure permet une analyse approfondie des relations spatiales et topologiques inhérentes aux données, offrant ainsi une précision accrue dans des environnements complexes comportant plusieurs bâtiments et étages.[5]

1 Fonctions Clés

Pour le prétraitement des données, d'autres méthodes clés ont déjà été présentées dans le chapitre 1, Base de Données : UJIIndoorLoc.

Cette section met l'accent sur une technique utilisée uniquement dans le modèle GNN : la méthode des k-plus proches voisins (k-Nearest Neighbors, k-NN).

1.1 Le concept de k-NN

k-NN est un algorithme basé sur l'analyse des relations entre les données, où chaque point est classifié ou régressé en fonction de ses k voisins les plus proches dans un espace donné.

Dans le cadre des GNN, le k-NN est spécifiquement utilisé pour transformer des données non structurées en une représentation graphique, essentielle pour l'apprentissage des GNN.

1.2 Étapes principales du k-NN

1.2.1 Calcul des distances

L'objectif du k-NN est de mesurer la similitude entre les points en calculant la distance dans un espace de caractéristiques. Les distances les plus couramment utilisées sont :

- **Distance Euclidienne** (Euclidean Distance) : convient aux positions absolues et aux caractéristiques de l'espace physique

- **Distance de Manhattan** : convient aux situations où il n'y a pas d'interaction directe entre les composants de fonctionnalités et où ils sont indépendants.
- **Similarité Cosinus** (Distance Angulaire) : convient aux données clairsemées de grande dimension, en mettant l'accent sur la direction plutôt que sur l'amplitude.

Dans ce projet, la fonction *kneighbors_graph*, issue du module *sklearn.neighbors*, est utilisée pour mesurer les distances. Cette fonction utilise par défaut la distance euclidienne pour calculer les distances entre les échantillons. Par conséquent, seule cette méthode sera détaillée dans ce rapport.

1) Définition de la Distance Euclidienne

La distance euclidienne est l'une des métriques les plus couramment utilisées pour mesurer la distance entre deux points dans un espace euclidien. Elle est définie comme suit :

$$d(i,j) = \sqrt{\sum_{k=1}^{n} \left(x_{i,k} - x_{j,k}\right)^2} \qquad (4.1)$$

- $d(i,j)$: La distance euclidienne entre les points i et j .
- $x_{i,k}$: La valeur du point i sur la k-ième dimension.
- $x_{j,k}$: La valeur du point j sur la k-ième dimension.
- n : Le nombre de dimensions des caractéristiques.

2) Interprétation géométrique

La distance euclidienne correspond à la longueur de la ligne droite qui relie deux points dans un espace 2D ou 3D.

- Dans un espace 2D, la distance euclidienne est la longueur du segment reliant les deux points : (x_1, y_1) et (x_2, y_2).

$$d(i,j) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \qquad (4.2)$$

- Dans un espace 3D, la distance euclidienne prend également en compte la troisième dimension : (x_1, y_1, z_1) et (x_2, y_2, z_2).

$$d(i,j) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2} \qquad (4.3)$$

3) Logique principale pour le calcul des distances

- **Représentation des données** : Les points sont représentés sous la forme d'une matrice de caractéristiques X. X a une forme de $[N, F]$,

où N est le nombre d'échantillons et F est le nombre de caractéristiques.

- **Calcul des distances entre tous les points** : La distance euclidienne est calculée en appliquant la formule.
- **Sélection des k-plus proches voisins** : Chaque point est relié à ses k-plus proches voisins en fonction des distances calculées.
- **Construction du graphe** : Les relations entre les points sont utilisées pour construire une matrice d'adjacence représentant les connexions.

1.3 Utilisation du k-NN dans le GNN

Dans les GNN, le k-NN est utilisé pour construire un graphe à partir des données non structurées. Chaque échantillon est représenté comme un nœud, et les relations entre les échantillons sont définies à l'aide des k-plus proches voisins.

Pour chaque point, les k voisins les plus proches sont déterminés en fonction des distances calculées. La valeur de k doit être choisie avec soin :

- k-petit : sensible au bruit.
- k-grand : perte des détails locaux.

Étapes principales :

- Représenter chaque point comme un nœud.
- Relier chaque nœud à ses k voisins les plus proches en fonction des distances calculées.
- Générer une matrice d'adjacence où $A_{ij} = 1$ si j est parmi les k-plus proches voisins de i.

1.4 Avantages du k-NN

1) Simple et intuitif.
2) Capable de capturer les relations locales dans les données.
3) Facile à appliquer sur des données de haute dimension.

2 Structure du GNN

Le réseau neuronal de graphe (Graph Neural Network, GNN) est un modèle d'apprentissage profond conçu pour traiter des données structurées sous forme de graphe.

2.1 Concepts de Base d'un Graphe

1) Graphe G

Un graphe G est composé V et E :

- **V** : Ensemble de nœuds.
- **E** : Ensemble d'arêtes, représentant les relations ou connexions entre les nœuds.

2) Matrice d'adjacence A
   - Représente les relations entre les nœuds.
   - Si les nœuds i et j sont connectés, $A_{ij} = 1$; sinon $A_{ij} = 0$.
3) Matrice des caractéristiques des nœuds X
   - Chaque nœud peut avoir des caractéristiques.
4) Matrice des degrés D
   - La matrice des degrés est une matrice diagonale utilisée pour représenter les degrés des nœuds dans un graphe.
   - Chaque élément diagonal D_{ii} correspond au degré du nœud i soit le nombre d'arêtes connectées à ce nœud.

2.1.1 Comment fonctionne GNN

Le GNN utilise un mécanisme appelé passage de messages (Message Passing) pour apprendre les représentations des nœuds. Les étapes clés sont les suivantes :

- **Passage de messages** : Chaque nœud échange des informations avec ses voisins.
- **Agrégation des caractéristiques** : Chaque nœud agrège les caractéristiques de ses voisins et les combine avec ses propres caractéristiques.

- **Mise à jour des nœuds** : Les caractéristiques agrégées sont passées dans un réseau neuronal (comme un MLP) pour obtenir une nouvelle représentation des nœuds.

GNN est capable de capturer les relations de haut niveau entre les nœuds et d'apprendre des représentations riches et significatives.

2.1.2 Représentation mathématique de GNN[7]

Soit $G = (V, E, \sigma, \varphi, \epsilon)$ un graphe, met à jour de manière itérative l'étiquetage des sommets dans G. Ce processus comporte deux étapes :

- AGRÉGER (AGGREGATE) : cette étape agrège les informations des sommets voisins pour chaque sommet ;
- COMBINER (COMBINE) : cette étape met à jour l'étiquette du sommet en combinant son étiquette actuelle avec celles de ses voisins.

À une itération k, la formule de mise à jour des nœuds de GNN est :

$$a_v^{(k)} = AGGREGATE^{(k)}\left(h_u^{(k-1)} : u \in N(v)\right) \qquad (4.4)$$

$$h_v^{(k)} = COMBINE^{(k)}\left(h_v^{(k-1)}, a_v^{(k)}\right) \qquad (4.5)$$

- σ : fonction d'activation non linéaire (comme ReLU ou ELU).
- φ : un étiquetage des sommets.
- ϵ : un étiquetage des arêtes.
- $N(v)$: Ensemble des voisins du nœud v.

La représentation vectorielle du graphe entier peut être obtenue à l'aide d'une procédure READOUT qui agrège les caractéristiques des sommets à l'itération finale ($k = K$) :

$$h_G = READOUT\left(\{h_u^{(K)} \middle| v \in G\}\right) \qquad (4.6)$$

2.2 Couches de Convolution Graphique (GCN)

Le Graph Convolutional Network (GCN) est une variante classique des réseaux de neurones de graphe (GNN), proposée par Thomas Kipf et Max Welling en 2016.[8] Le GCN étend l'opération de convolution aux données structurées sous forme de graphe, permettant d'apprendre efficacement les représentations des nœuds.[9]

2.2.1 Formule principale du GCN[9]

$$X^{(l+1)} = \sigma\left(\hat{A}X^{(l)}W^{(l)}\right) \qquad (4.7)$$

- $X^{(l)}$: Représentations des nœuds à la couche l.
- $\hat{A}$: Matrice d'adjacence normalisée, incluant les boucles auto-relationnelles.
- $W^{(l)}$: Matrice de poids (paramètres apprenables) pour la couche l.
- σ : Fonction d'activation.

2.2.2 Processus de fonctionnement du GCN

1) **Entrée** : Matrice des caractéristiques des nœuds X et matrice d'adjacence A.
2) **Ajout de boucles et normalisation** : Ajouter des auto-relations à A et la normaliser pour obtenir $\hat{A}$.
3) **Agrégation des caractéristiques** : Chaque couche multiplie les caractéristiques des nœuds $X^{(l)}$ avec $\hat{A}$, réalisant l'agrégation des voisins.
4) **Transformation linéaire et activation** : Appliquer une transformation via $W^{(l)}$ et une activation non linéaire σ .
5) **Sortie** : Représentations finales des nœuds $X^{(l+1)}$.

2.2.3 Impact du nombre de couches

1) **1 couche GCN** : Agrège les informations des voisins de premier ordre.
2) **2 couches GCN** : Agrège les informations des voisins de second ordre.
3) **Couches multiples** : Capture des relations de voisinage plus larges, mais risque de lissage excessif (Over-smoothing) où les représentations des nœuds deviennent similaires.

3 Paramètres du Modèle

Certains paramètres du processus d'entraînement, tels que la dimension d'entrée (input =519), le nombre d'époques (epochs), le coefficient de pénalisation d'étage (floor_penalty = 4) et le Droupout (droupout = 0.1), ont déjà été présentés dans la section dédiée au DNN. Ces paramètres restent inchangés dans le cadre du GNN car

ils sont directement liés à la structure des données utilisées ou à l'évaluation des erreurs de prédiction. Pour plus de détails, voir la section 3 du deuxième chapitre : Paramètres du Modèle.

Cependant, le GNN introduit des paramètres spécifiques qui nécessitent une attention particulière :

3.1 Nombre de Voisins k

Le paramètre k définit le nombre de voisins considérés pour chaque nœud lors de la construction du graphe à l'aide de l'algorithme des k-plus proches voisins (k-NN). Ce paramètre est crucial car il influence directement la structure du graphe et la qualité des informations locales capturées.

Dans ce projet, des expériences ont été menées avec $k = 5$ et $k = 15$.

4 Analyse des Paramètres et Résultats

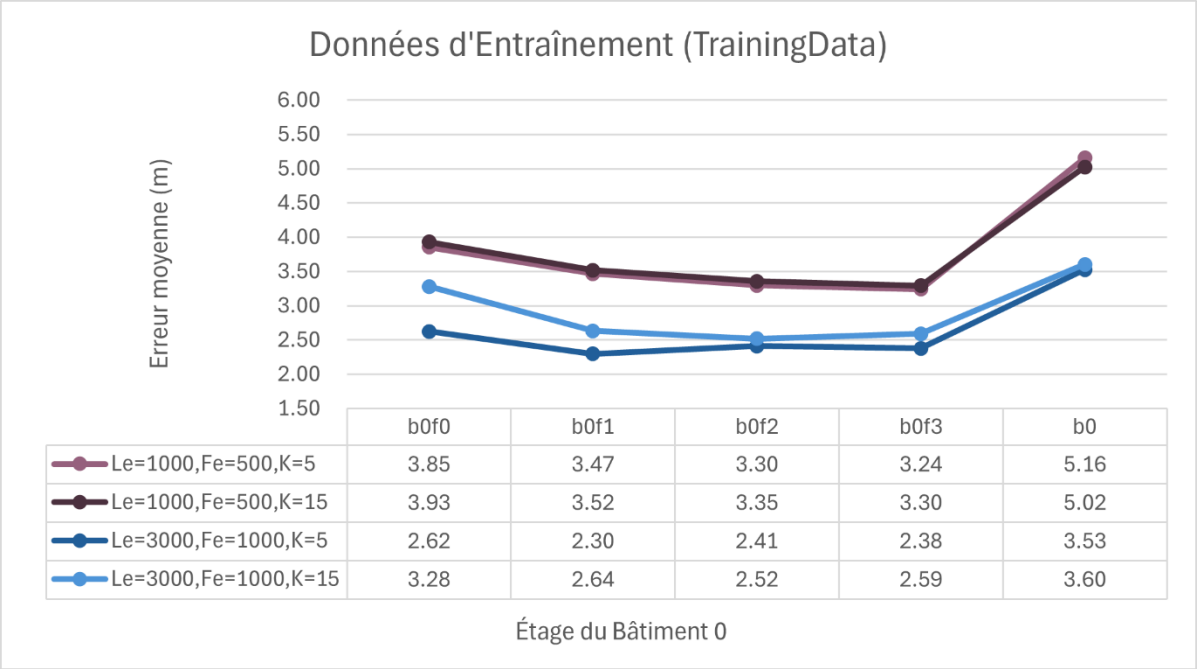
Dans ce projet, seuls les deux paramètres epochs et batch_size sont ajustés et testés.

Les paramètres epochs pour la longitude et la latitude sont définis de manière égale, tandis que les floor_epochs sont fixés à une petite valeur.

Dans les légendes des graphiques, les abréviations suivantes sont utilisées :

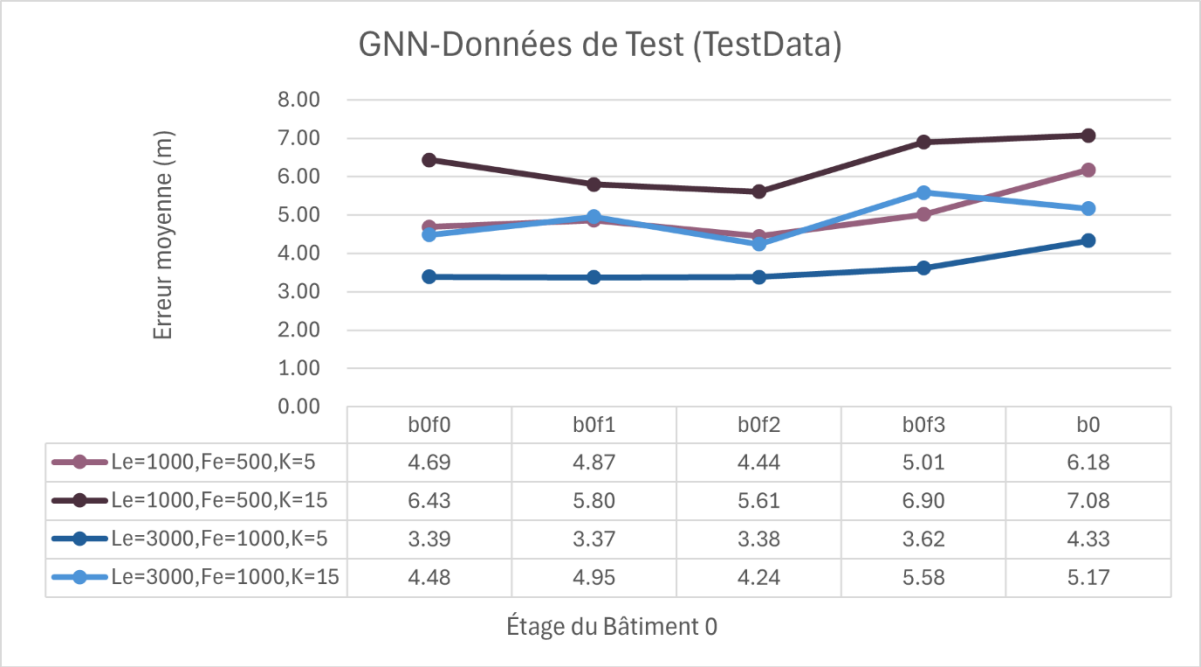
- Le : longitude et latitude epochs
- Fe : floor_epochs
- K : Nombre de voisins utilisés pour construire le graphe.

Les résultats sont les suivants :



	b0f0	b0f1	b0f2	b0f3	b0
Le=1000,Fe=500,K=5	3.85	3.47	3.30	3.24	5.16
Le=1000,Fe=500,K=15	3.93	3.52	3.35	3.30	5.02
Le=3000,Fe=1000,K=5	2.62	2.30	2.41	2.38	3.53
Le=3000,Fe=1000,K=15	3.28	2.64	2.52	2.59	3.60

Figure 3 GNN-Résultats d'erreur moyenne pour les données d'entraînement



	b0f0	b0f1	b0f2	b0f3	b0
Le=1000,Fe=500,K=5	4.69	4.87	4.44	5.01	6.18
Le=1000,Fe=500,K=15	6.43	5.80	5.61	6.90	7.08
Le=3000,Fe=1000,K=5	3.39	3.37	3.38	3.62	4.33
Le=3000,Fe=1000,K=15	4.48	4.95	4.24	5.58	5.17

Figure 4 GNN-Résultats d'erreur moyenne pour les données de test

4.1 Effet des Époques

L'augmentation du nombre d'epochs entraîne une diminution de l'erreur moyenne. Cela indique que le modèle est capable d'apprendre davantage de caractéristiques des données grâce à un plus grand nombre d'itérations.

Dans les données testées, avec les autres paramètres (BS) identiques, la combinaison des valeurs $Le = 3000$ et $Fe = 1000$ donne des résultats nettement meilleurs que la combinaison $Le = 1000$ et $Fe = 500$.

4.2 Impact du Paramètre k

Lorsque les autres valeurs des paramètres sont les mêmes (Le & Fe), $K = 5$ donne de meilleurs résultats que $K = 15$. Les raisons possibles sont les suivantes :

1) Une valeur de k trop grande introduit du bruit

Avec une valeur de k plus grande, chaque nœud est connecté à davantage de voisins. Cette augmentation inclut potentiellement des nœuds avec peu ou pas de corrélation avec les caractéristiques du nœud cible, ce qui introduit davantage de bruit.

2) L'équilibre entre caractéristiques locales et globales

Dans les données RSSI (intensité de signal), les caractéristiques locales (par exemple, la distribution du signal des nœuds voisins) sont souvent cruciales pour les tâches de modélisation.

- Petit k $(K = 5)$: Capture principalement les caractéristiques locales, mieux adaptées aux scénarios comme la localisation en intérieur, en reflétant fidèlement les relations de voisinage.
- Grand k $(K = 15)$: Capture davantage de caractéristiques globales, mais ces informations globales peuvent être inutiles, voire interférer avec l'apprentissage des caractéristiques locales.

3) L'impact du paramètre k sur les graphes clairsemés

Dans un graphe clairsemé, le nombre d'arêtes entre les nœuds est limité. Une augmentation de la valeur de k rend chaque nœud plus densément connecté.

4) La propriété de lissage des GCN

Les GCN ont une propriété de lissage des caractéristiques où, à mesure que les couches augmentent, les caractéristiques des nœuds deviennent de plus en plus similaires (phénomène dit d'Over-smoothing). Avec un k grand, cela accélère ce phénomène, car davantage de caractéristiques des voisins sont agrégées.

4.3 Analyse par Étages

Lorsque les données sont regroupées par étage ($b0f0, b0f1, b0f2, b0f3$), les performances du modèle sont plus stables et les erreurs restent moins de 3 mètres.

Pour le bâtiment complet ($b0$) où les données de différents étages sont mélangées, une augmentation de l'erreur est observée. Cependant, l'erreur globale reste moins de 4 mètres.

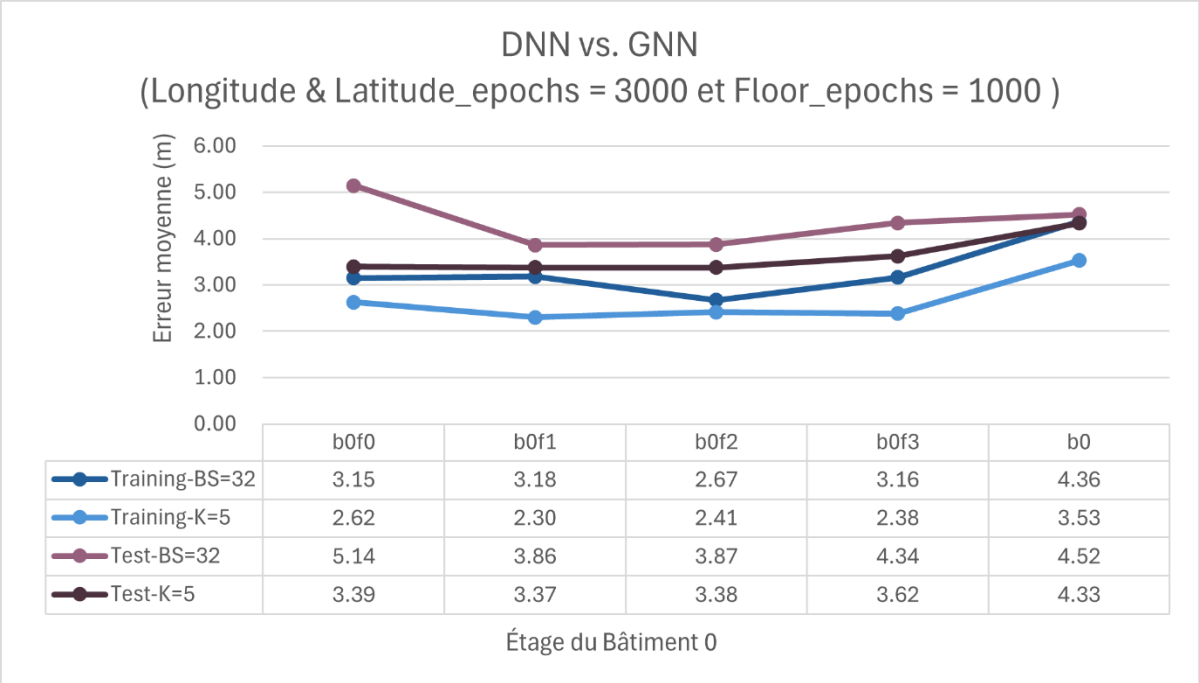
4.4 Meilleure Combinaison de Paramètres

Parmi toutes les combinaisons testées, le meilleur résultat est les paramètres : $Le = 3000, Fe = 1000, K = 5$.

Chapitre 5 Comparaison entre les Méthodes DNN et GNN

1 Comparaison de Performance

Sur la base des résultats expérimentaux précédents, les performances optimales obtenues avec les meilleurs paramètres (epochs pour la longitude et la latitude = 3000, epochs pour l'étage = 1000 ; pour le DNN, BS=32 ; pour le GNN, K=5) ont été comparées.



	b0f0	b0f1	b0f2	b0f3	b0
Training-BS=32	3.15	3.18	2.67	3.16	4.36
Training-K=5	2.62	2.30	2.41	2.38	3.53
Test-BS=32	5.14	3.86	3.87	4.34	4.52
Test-K=5	3.39	3.37	3.38	3.62	4.33

Figure 5 Comparaison des résultats sous paramètres optimaux (erreur de prédiction/mètre)

Les résultats montrent que, d'après la Figure 5, les prédictions obtenues avec le GNN sont systématiquement et nettement supérieures à celles obtenues avec le DNN.

Dans le cadre de cette étude, il a été observé que le DNN produit des résultats de prédiction plus précis sur l'ensemble d'entraînement, tandis que le GNN excelle sur l'ensemble de test. Cette divergence peut s'expliquer par plusieurs facteurs :

1. **Absence de prise en compte des relations structurées dans le DNN**

   Contrairement au GNN, le DNN ne modélise pas les relations entre les échantillons, ce qui le rend plus vulnérable aux variations de distribution entre les ensembles d'entraînement et de test.

2. Utilisation de la structure graphique par le GNN

Le GNN exploite les relations structurelles entre les échantillons de données grâce à une structure de graphe, ce qui renforce sa robustesse et améliore ses performances, notamment sur les données de test.

3. Adaptabilité du GNN aux différences de distribution

Le GNN atténue l'impact des divergences de distribution entre les données d'entraînement et de test en modélisant efficacement les relations grâce à la structure graphique.

2 Analyse de la Complexité Computationnelle

La complexité des programmes est un indicateur clé pour évaluer la difficulté et la maintenabilité d'un système logiciel, elle inclut principalement la complexité temporelle et la complexité spatiale.

La complexité temporelle mesure la rapidité et l'efficacité d'un algorithme, tandis que la complexité spatiale évalue les ressources mémoire utilisées par l'algorithme. L'analyse de la complexité implique l'étude du nombre d'opérations effectuées par un algorithme ou la taille de la mémoire nécessaire, en fonction de la situation spécifique.

Ce chapitre présente une analyse comparative de la complexité des modèles DNN et GNN, en tenant compte de leurs spécificités et de l'impact de leur complexité sur les performances des modèles.

2.1 La complexité du modèle DNN

Pour le DNN, la complexité est principalement déterminée par le nombre de paramètres dans les couches entièrement connectées (Dense Layer) et les étapes de calcul associées :

1) Formule

$$Complexité\ des\ paramètres = \sum_{i=1}^{L}(d_i \times d_{i+1} + d_{i+1}) \qquad (5.1)$$

$$FLOPs = \sum_{i=1}^{L}(d_i \times d_{i+1} + d_{i+1}) \quad (5.2)$$

- d_i : Dimension d'entrée (sortie de la couche précédente).
- d_{i+1} : Dimension de sortie (couche actuelle).
- L : Nombre total de couches, y compris l'entrée, les couches cachées et la sortie.

2) Étapes de calcul

- Calculer la complexité des paramètres de l'entrée à la première couche cachée.
- Calculer la complexité des paramètres entre les couches cachées.
- Calculer la complexité des paramètres de la dernière couche cachée à la sortie.
- Additionner toutes les couches pour obtenir la complexité totale des paramètres et des calculs.

3) Résultat

```
--- Model Complexity ---
Encoder Model: Parameters = 299591, FLOPs = 299591
Longitude Model: Parameters = 223937, FLOPs = 223937
Latitude Model: Parameters = 223937, FLOPs = 223937
Floor Model: Parameters = 199233, FLOPs = 199233

Overall Model Complexity: Parameters = 946698, FLOPs = 946698
DNN Complexity: Parameters = 946698, FLOPs = 946698
Finish
```

Figure 6 Complexité du modèle DNN

- Le modèle DNN a un total de 946,698 paramètres et FLOPs, répartis entre l'encodeur et les sous-modèles pour les prédictions de longitude, latitude et étage.

2.2 La complexité du modèle GNN

Pour le modèle GNN, la complexité est évaluée en termes de complexité des paramètres et du nombre d'opérations en virgule flottante par passage avant (FLOPs). Les FLOPs quantifient le coût de calcul pour chaque traitement de données. Cette complexité met en évidence le coût associé à la modélisation des relations structurelles dans les données par le GNN, tout en illustrant l'efficacité du GNN pour traiter des données relationnelles.

1) Formule

$$Complexité\ des\ paramètres\ = d_{hidden}^2 + d_{hidden} \qquad (5.3)$$

$$FLOPs = 2 * NumEdges \times d_{hidden} + NumNodes * (d_{hidden}^2 + d_{hidden}) \qquad (5.4)$$

- d_{hidden} : Dimension de la couche cachée.
- $NumEdges$: Nombre d'arêtes
- $NumNodes$: Nombre de nœuds

2) Étapes de calcul

- Calculer la complexité des paramètres de la matrice de poids des couches cachées.
- Calculer la complexité de propagation.

3) Résultat

```
--- Model Complexity ---
Longitude Model: Parameters = 264961, FLOPs = 5530625
Latitude Model: Parameters = 264961, FLOPs = 5530625
Floor Model: Parameters = 264961, FLOPs = 5530625

Overall Model Complexity: Parameters = 794883, FLOPs = 16591875
Finish.
```

Figure 7 Complexité du modèle GNN

- Le GNN, avec 794,883 paramètres par modèle, présente une structure plus compacte grâce à l'utilisation partagée des relations de graphe.
- Toutefois, le coût de calcul est plus élevé, avec 16,591,875 FLOPs par forward pass, en raison des opérations de convolution sur les graphes.
- Malgré cette complexité, les résultats montrent que le GNN utilise plus efficacement les relations locales et globales pour réduire les erreurs de prédiction.

2.3 La complexité du modèle DNN vs. GNN

Les modèles DNN (Deep Neural Network) et GNN (Graph Neural Network) présentent des différences fondamentales en termes de complexité, reflétant leurs approches respectives pour traiter les données. Les GNN se distinguent par leur capacité à exploiter efficacement les relations structurelles dans les données, ce qui les rend particulièrement adaptés aux données connectées, comme les graphes ou les réseaux. Bien que les GNN présentent un nombre de paramètres inférieur, leur coût en FLOPs (Floating Point Operations Per Second) est plus élevé, en raison des calculs intensifs nécessaires pour agréger les informations des nœuds voisins.

Cette capacité des GNN à capturer des relations complexes permet de modéliser avec précision des structures interconnectées, ce qui constitue un avantage considérable pour des tâches comme la classification de nœuds, la prédiction de liens ou l'apprentissage sur des graphes dynamiques. Grâce à cette conception optimisée pour les interactions structurées, les GNN nécessitent moins

de paramètres à ajuster, réduisant ainsi les risques de surapprentissage tout en conservant des performances élevées.

En revanche, les DNN sont plus adaptés aux données non structurées où chaque entrée peut être traitée indépendamment. Leur simplicité structurelle entraîne une consommation de FLOPs plus faible, mais cela se fait au prix d'un nombre de paramètres significativement plus élevé. Cette approche rend les DNN moins performants pour capturer des dépendances complexes dans des données structurées.

Ainsi, le choix entre GNN et DNN dépend de la nature des données. Les GNN offrent un avantage décisif lorsque les relations entre les entités jouent un rôle essentiel.

3 Avantages et Inconvénients

3.1 DNN (Deep Neural Networks)

3.1.1 Les Avantages

1) **Simplicité d'implémentation** : Leur architecture standard (couches denses, convolutions, etc.) permet une mise en œuvre relativement directe avec des frameworks bien établis.
2) **Flexibilité** : Les DNN peuvent être facilement modifiés pour différents cas d'utilisation en ajustant les couches, les neurones et les fonctions d'activation.
3) **Efficacité computationnelle** : Moins de surcharge liée aux relations complexes entre les échantillons, ce qui les rend rapides pour les données simples et non structurées.

3.1.2 Les Inconvénients

1) **Incapacité à modéliser les relations entre échantillons** : Les DNN ne capturent pas les relations intrinsèques entre les données, ce qui limite leur application à des problèmes où les connexions inter-échantillons sont cruciales.

2) **Sensibilité aux différences de distribution** : Les DNN peuvent mal généraliser lorsque les distributions des ensembles d'entraînement et de test diffèrent.
3) **Risques de surapprentissage** : Avec un nombre élevé de paramètres, les DNN peuvent facilement surajuster les données d'entraînement, surtout avec des ensembles de données limités.

3.2 GNN (Graph Neural Networks)

3.2.1 Les Avantages

1) **Modélisation des relations complexes** : Les GNN exploitent les structures en graphe pour capturer les relations locales et globales entre les échantillons.
2) **Généralisation robuste** : Grâce à l'intégration des structures graphiques, les GNN sont moins sensibles aux différences de distribution entre les ensembles d'entraînement et de test.
3) **Adaptés aux données structurées** : Les GNN sont particulièrement performants pour des tâches impliquant des graphes.
4) **Flexibilité dans les représentations des données** : Capables de traiter des graphes de différentes tailles et topologies sans nécessiter de redimensionnement des entrées.

3.2.2 Les Inconvénients

1) **Complexité computationnelle élevée** : Le calcul des matrices d'adjacence et des FLOPs rend les GNN plus coûteux en termes de temps et de mémoire, surtout avec des graphes de grande taille.
2) **Dépendance à la qualité des graphes** : La performance des GNN dépend fortement de la précision et de la pertinence des relations entre les nœuds dans le graphe.
3) **Moins adaptés aux données non structurées** : Pour des données massives et non structurées, les GNN sont moins efficaces que les DNN en raison de la surcharge liée au traitement graphique.
4) **Risque de sur-lissage (over-smoothing)** : Avec l'augmentation du nombre de couches de propagation ou de la valeur de k dans K-NN, les

représentations des nœuds peuvent devenir trop similaires à celles de leurs voisins, rendant les nœuds difficilement différenciables et réduisant ainsi la capacité du modèle à capturer des distinctions locales.

4 Synthèse des Résultats

En résumé, bien que le DNN présente des avantages tels qu'une structure simple et une rapidité d'entraînement, ses limitations, notamment l'incapacité à capturer les relations structurelles entre les données, réduisent son efficacité pour les tâches basées sur des données RSSI.

En revanche, le GNN, grâce à sa capacité à exploiter la structure des graphes et à modéliser les relations entre les échantillons, offre une meilleure efficacité computationnelle et des résultats significativement plus précis pour les données RSSI complexes. Cela fait du GNN un choix supérieur pour les scénarios nécessitant une compréhension approfondie des relations topologiques et spatiales.

Chapitre 6 Conclusion

Ce projet a exploré et comparé deux approches majeures pour la localisation en intérieur à partir des données RSSI : les réseaux neuronaux profonds (DNN) et les réseaux neuronaux graphiques (GNN). Ces analyses ont été menées sur le jeu de données UJIIndoorLoc, permettant de mettre en lumière les avantages, les limites et les performances respectives des deux modèles.

Les DNN présentent une architecture standard basée sur des couches entièrement connectées, caractérisée par une simplicité d'implémentation et une efficacité computationnelle adaptée aux données non structurées. Leur capacité à traiter rapidement de grands volumes de données en fait une option intéressante pour des applications simples. Cependant, leur principal inconvénient réside dans leur incapacité à modéliser les relations complexes entre les échantillons. Cette limitation réduit leur efficacité pour les tâches où les structures relationnelles jouent un rôle clé, comme la localisation en intérieur. De plus, les DNN sont sensibles aux différences de distribution entre les ensembles d'entraînement et de test, ce qui peut limiter leur capacité de généralisation. Enfin, leur tendance au surapprentissage, particulièrement lorsque les données d'entraînement sont limitées, constitue un défi important.

Les GNN, en revanche, se distinguent par leur capacité à exploiter les relations complexes dans les données structurées, grâce à l'utilisation de graphes. Ces modèles sont particulièrement performants dans des scénarios nécessitant une compréhension approfondie des relations topologiques et spatiales, comme la localisation en intérieur. En intégrant les relations entre les nœuds, les GNN offrent une robustesse accrue face aux variations de distribution entre les ensembles d'entraînement et de test. Toutefois, cette capacité s'accompagne d'un coût computationnel plus élevé, en raison des calculs liés aux matrices d'adjacence et à la propagation des informations au sein des graphes. De plus, les GNN sont fortement dépendants de la qualité et de la précision des graphes, et l'effet de sur-lissage (over-smoothing) peut limiter leur performance lorsque les graphes deviennent trop denses ou profonds.

L'analyse de la complexité computationnelle a mis en évidence des différences fondamentales entre les deux modèles. Pour les DNN, la complexité est principalement influencée par le nombre de paramètres dans les couches entièrement connectées, ce qui impacte directement les besoins en stockage et le temps de calcul. En revanche, pour les GNN, la complexité combine le nombre de paramètres avec le

coût de propagation des informations à travers les graphes, rendant les GNN plus coûteux en calcul, mais leur permettant d'exploiter efficacement les structures complexes dans les données.

En conclusion, bien que les DNN soient adaptés aux données simples ou non structurées grâce à leur rapidité et leur simplicité, les GNN offrent des performances nettement supérieures pour les tâches nécessitant une compréhension des relations complexes, comme la localisation en intérieur basée sur les données RSSI. Les GNN, en tirant parti des structures graphiques, apportent une meilleure précision et une robustesse accrue, faisant de cette approche un choix optimal pour des scénarios où les relations topologiques et spatiales sont essentielles.

Ces résultats soulignent l'importance d'une sélection rigoureuse des modèles en fonction des caractéristiques des données et des exigences de l'application.

À l'avenir, les pistes d'amélioration incluent l'optimisation des combinaisons de paramètres des deux modèles, l'exploration d'architectures hybrides DNN-GNN et l'intégration de données complémentaires, telles que celles issues de capteurs inertiels, afin d'améliorer encore davantage les performances de localisation.

Références

[1] Njima, Wafa. Méthodes de localisation de capteurs dans le contexte de l'Internet des Objets. Traitement des images [eess.IV]. Conservatoire national des arts et metiers – CNAM ; École supérieure des communications de Tunis (Tunisie), 2019, pp.22-23. NNT : 2019CNAM1264. URL : https://tel.archives-ouvertes.fr/tel-02484757

[2] Kim, K.S., Lee, S., & Huang, K. (2018). A scalable deep neural network architecture for multi-building and multi-floor indoor localization based on Wi-Fi fingerprinting. Big Data Analytics, 3(4). https://doi.org/10.1186/s41044-018-0031-2

[3] H. Asadollahi, N. Mokari, H. Saeedi, H. Yanikomeroglu, "Enhancing Indoor Localization Accuracy Using a Hybrid Deep Learning Algorithm: A DNN-CNN Approach," 2023.

[4] W. Njima, A. Bazzi, M. Chafii, "DNN-Based Indoor Localization Under Limited Dataset Using GANs and Semi-Supervised Learning," IEEE Access, vol. 10, pp. 69896–69910, 2022. DOI : 10.1109/ACCESS.2022.3187837

[5] Vishwakarma, R., Joshi, R.B., & Mishra, S. (2023). IndoorGNN: A Graph Neural Network Based Approach for Indoor Localization using WiFi RSSI.

[6] J. Torres-Sospedra, R. Montoliu, A. Martínez-Usó, et al., "UJIIndoorLoc : A new multi-building and multi-floor database for WLAN fingerprint-based indoor localization problems," IPIN 2014, 2014. DOI : 10.1109/IPIN.2014.7275492

[7] T. Dash, A. Srinivasan, and A. Baskar, "Inclusion of domain-knowledge into GNNs using mode-directed inverse entailment," Machine Learning, vol. 111, pp. 575–623, 2022. DOI: 10.1007/s10994-021-06090-8.

[8] T. N. Kipf and M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks," International Conference on Learning Representations (ICLR), 2017. [Online]. Available: https://doi.org/10.48550/arXiv.1609.02907

[9] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A Comprehensive Survey on Graph Neural Networks," arXiv preprint arXiv:1901.00596v4, Aug. 2019. [Online]. Available: https://arxiv.org/abs/1901.00596