جامعة سعيدة د. مولاي الطاهر
كلية الرياضيات و الإعلام الآلي و الاتصالات السلكية و اللاسلكية
قسم: الإعلام الآلي

Mémoire de Master en informatique

Spécialité : Réseaux Informatiques et Systèmes Réparties

T h è m e

ESTIMATION DE LA LATENCE DANS UN RÉSEAU CELLULAIRE BASÉE SUR DES RÉSEAUX NEURONAUX

- **Présenté par :**
  Cherifi Aimene abdelkader
  Benmeddah Maamar
- **Dirigé par :**
  Dr M H.Rahmani

Année universitaire 2025-2026

Remerciements

Nous remercions tout d'abord Dieu le tout puissant de nous avoir donné la volonté et la persévérance pour réaliser ce travail.

Nous tenons à saisir cette occasion et adresser nos profonds remerciements et nos profondes reconnaissances à Dr RAHMANI MOHAMED EL HADI, notre encadrant de mémoire, pour ses précieux conseils et son orientation ficelée tout au long de notre recherche.

Nos vifs remerciements vont également aux membres du jury pour l'intérêt qu'ils ont porté à notre recherche en acceptant d'examiner notre travail et de l'enrichir par leurs propositions.
Nous souhaitons adresser nos remerciements les plus sincères aux personnes qui nous ont apporté leur aide et qui ont contribué à l'élaboration de ce mémoire .

Nous remercions tous nos enseignants pour leur dévouement, leur patience et leur contribution à notre formation.

Enfin, nous adressons nos plus sincères remerciements à tous nos proches et amis, qui nous ont toujours encouragés au cours de la réalisation de ce mémoire.

Dédicaces

Je dédie ce mémoire :

À mes très chers parents,

pour leurs dévouements et leurs encouragements. Que ce travail soit un témoignage de ma profonde gratitude envers eux.

À ma chère femme et mes enfants

qui m'a toujours soutenu et encouragé durant les moments les plus difficiles.

À mes frères et soeurs, ainsi qu'à leurs enfants.

Benmeddah Maamar

Je dédie ce mémoire :

À mes très chers parents,

pour leurs dévouements et leurs encouragements. Que ce travail soit un témoignage de ma profonde gratitude envers eux.

Cherifi Aimene

Table des matières

Liste des figures

Liste des Tableaux

Liste des Abréviations

- **IA** : Intelligence Artificielle.
- **ML** : Machine Learning (Apprentissage Automatique).
- **DL** : Deep Learning (Apprentissage Profond).
- **RN** : Réseaux de Neurones.
- **MLP** : Multi-Layer Perceptron.
- **ReLU** : Rectified Linear Unit.
- **MAE** : Mean Absolute Error.
- **RMSE** : Root Mean Square Error.
- **MSE** : Mean Squared Error.
- **Adam** : Adaptive Moment Estimation.
- **3G/4G** : 3ème/4ème Génération de réseaux mobiles.
- **5G** : 5ème Génération de réseaux mobiles.
- **ANN** : Artificiel Neural Networks.
- **API** : Application Programming Interface.
- **BatchNorm** : Batch Normalisation.
- **DNN** : Deep Neural Networks.
- **EDA** : Exploratory Data Analysis.
- **IEEE** : Institute of Electrical and Electronics Engineers.
- **QoS / QoE** : Quality of Service / Quality of Experience.
- **RNN** : Récurrent Neural Networks.
- **RSRP / RSRQ** : Reference Signal Received Power / Quality.
- **IoT** :L’Internet des objets.

Introduction générale

Les réseaux cellulaires constituent aujourd'hui une infrastructure fondamentale de la communication moderne. Leur évolution rapide, de la 2G à la 5G, a profondément transformé les usages numériques, permettant non seulement les communications vocales et les messages, mais aussi la transmission de données à très haut débit. Avec l'essor de l'Internet des objets (IoT) et des applications nécessitant un temps de réponse rapide — telles que la réalité virtuelle, les véhicules autonomes et la télémédecine — la latence des réseaux cellulaires est devenue un indicateur de performance critique pour les opérateurs et les utilisateurs finaux.

La latence, définie comme le délai nécessaire au transfert d'un paquet de données de sa source à sa destination, influence directement la qualité des services offerts. Comprendre et estimer cette latence avec précision est essentiel pour le développement de solutions réseau efficaces et robustes. Les approches traditionnelles de modélisation, basées sur des formules mathématiques ou des simulations, atteignent rapidement leurs limites face à la complexité et à la variabilité des environnements cellulaires réels.

C'est dans ce contexte que l'apprentissage automatique, et plus particulièrement les réseaux de neurones artificiels, offre une alternative prometteuse. Ces techniques permettent d'apprendre directement à partir de mesures réelles, sans nécessiter une modélisation explicite des phénomènes physiques sous-jacents. Ce mémoire présente le développement d'un système intelligent d'estimation de la latence réseau dans les réseaux cellulaires, basé sur un réseau de neurones profond entraîné sur un jeu de données réel de 16 829 mesures collectées sur des réseaux 3G, 4G, LTE et 5G.

Objectifs du travail

Les objectifs principaux de ce travail sont les suivants :

- Analyser l'évolution des réseaux cellulaires et leur impact sur la latence.
- Étudier les réseaux de neurones artificiels et leur application dans l'estimation de la latence.
- Développer une méthodologie complète pour le prétraitement et l'analyse des données.
- Implémenter et évaluer un modèle de régression par réseau de neurones sur des données réelles.

Organisation du mémoire

Ce mémoire est organisé en quatre chapitres :

Le Chapitre 1 présente les réseaux cellulaires et leur évolution, depuis la 3G jusqu'à la 5G, ainsi que les travaux existants sur l'estimation de la latence dans ce contexte.

Le Chapitre 2 introduit les fondements théoriques des réseaux de neurones artificiels, en abordant les concepts de Machine Learning, de Deep Learning, ainsi que les architectures et mécanismes d'apprentissage de ces modèles.

Le Chapitre 3 décrit la méthodologie adoptée et les outils utilisés pour le développement du système d'estimation de la latence, incluant la collecte des données, le prétraitement et la conception du modèle.

Enfin, le Chapitre 4 présente l'implémentation du système, les résultats expérimentaux obtenus et une analyse critique des performances du modèle proposé.

Chapitre I : Les Réseaux Cellulaires

I.1 Introduction

Les réseaux cellulaires constituent le fondement des communications sans fil modernes, permettant une connectivité omniprésente et un accès rapide à l'information. Ce chapitre examine l'évolution des technologies cellulaires, notamment la 3G, la 4G et la 5G, ainsi que les enjeux liés à la latence dans ces réseaux. En comprenant ces éléments, nous pouvons mieux appréhender les défis et opportunités qu'offre l'innovation dans le domaine des communications.

I.2 Évolution des Réseaux Cellulaires

Les réseaux cellulaires ont connu une progression significative au fil des décennies, avec chaque génération apportant des améliorations notables en termes de capacité, de vitesse et de latence.

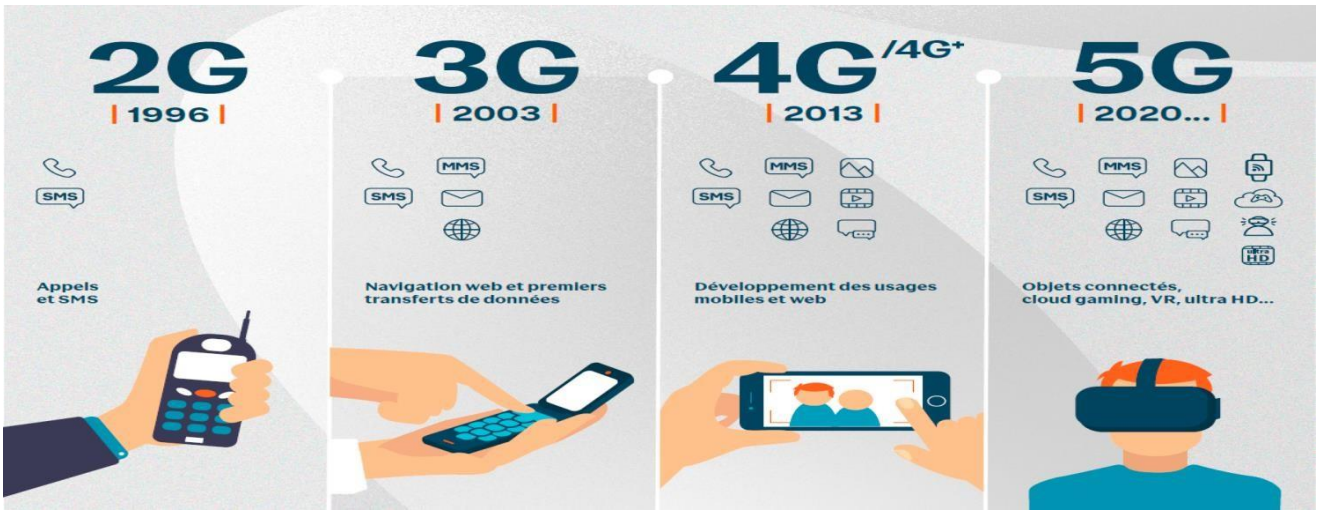


Figure I.1 : L'évolution des réseaux mobiles et de leurs usage de la 2G à la 5G[1]

I.2.1La 3G

La troisième itération de la technologie des réseaux cellulaires, communément appelée troisième génération ou 3G, a été inaugurée au début des années 2000. Cette génération a permis des débits de transmission de données nettement supérieurs à ceux de son prédécesseur, la 2G, permettant ainsi la fourniture de services multimédias, notamment le streaming vidéo et la navigation sur Internet. Le cadre 3G reposait sur des avancées telles

que le système universel de télécommunications mobiles (UMTS), qui fournissait des débits de données allant jusqu'à 2 Mbps dans des conditions idéales[2]

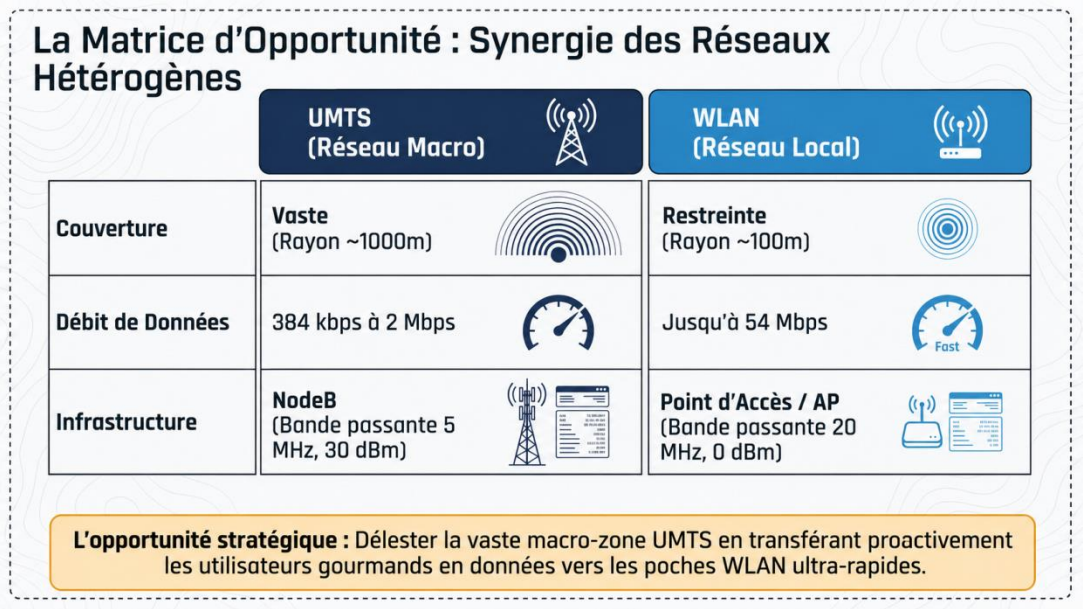


Figure I.2 : Synergie UMTS–WLAN : matrice d'opportunité dans un réseau hétérogène [3]

I.2.2La 4G / LTE

Le déploiement de la technologie 4G, en particulier celui de la technologie LTE (Long Term Evolution), a débuté en 2009. Cette avancée technologique a fondamentalement transformé le paysage des communications sans fil en permettant des taux de téléchargement pouvant atteindre 100 Mbit/s, tout en minimisant la latence à une plage généralement comprise entre 20 et 30 millisecondes[4]. En outre, cette génération a catalysé l'émergence d'applications en temps réel, y compris, mais sans s'y limiter, le streaming vidéo haute définition et les jeux en ligne interactifs.

I.2.3La 5G

L'émergence des technologies de cinquième génération (5G) a considérablement amélioré les indicateurs de performance des réseaux cellulaires. La 5G devrait fournir des taux de transfert de données atteignant plusieurs gigabits par seconde, ainsi que des latences réduites à moins d'une milliseconde, facilitant ainsi le développement d'applications révolutionnaires, notamment la réalité augmentée, les systèmes de transport autonomes et

l'Internet des objets (IoT)[5]. Cette génération repose sur une architecture réseau sophistiquée qui intègre des technologies avancées, telles que les systèmes MIMO (Massive Multiple Input Multiple Output) et le Software-Defined Networking (SDN).

,La 5G marque quant à elle une révolution dans le monde de la téléphonie mobile en introduisant l'interactivité en temps réel. Grâce à ses capacités uniques, elle ouvre la voie au divertissement sur mobile en permettant une augmentation des débits, une latence réduite et une vitesse de téléchargement jusqu'à 10 fois plus rapide que la 4G, Elle décuple ainsi les possibilités pour l'innovation dans tous les domaines : la santé avec le la télémédecine, les transports avec les voitures autonomes et l'industrie avec le développement d'usines intelligentes[6].

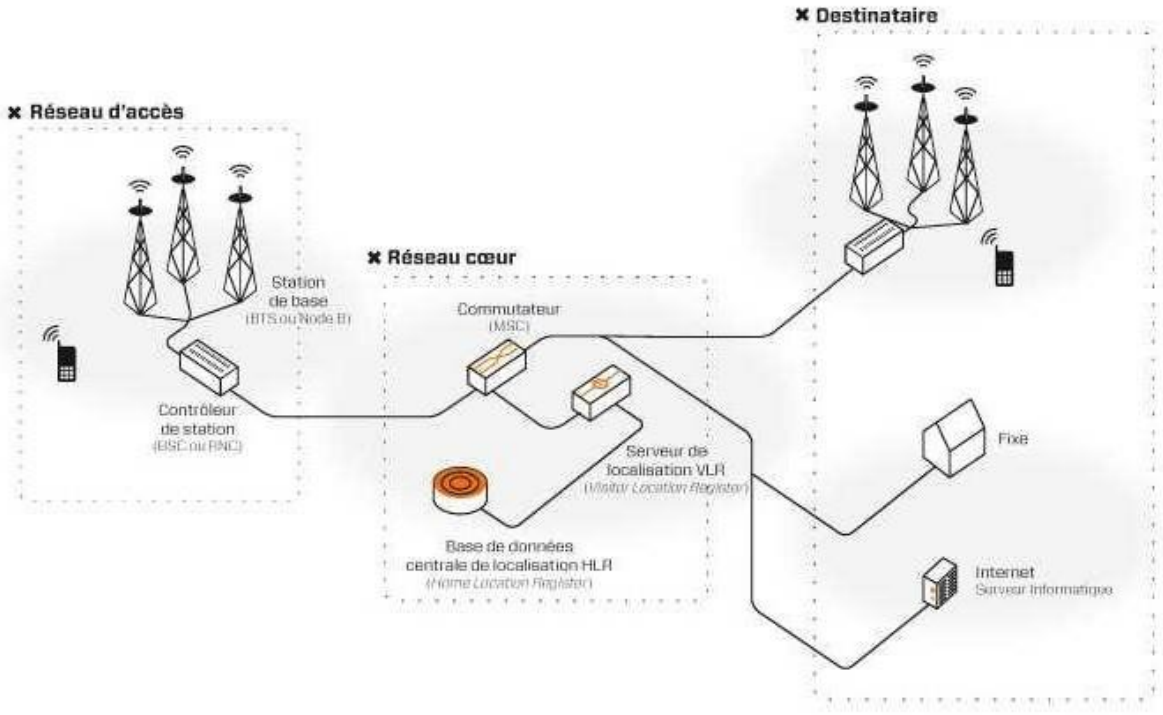


Figure I.3 : Équipements de réseau télécom[7]

Génération	Norme Principale	Débit Théorique	Principaux Services	Époque
2G	GSM / GPRS	9,6 à 171 kbps	Voix, SMS, début de la data (WAP)	Années 1990
3G	UMTS / HSPA	384 kbps à 42 Mbps	Internet mobile, visioconférence, mails	Années 2000
4G / 4G+	LTE / LTE-Advanced	100 Mbps à 1 Gbps	Vidéo HD, streaming, jeux en ligne	Années 2010
5G	NR (New Radio)	Jusqu'à 10-20 Gbps	Ultra-HD, IoT massif, latence ultra-faible	Années 2020

Tableau I.1 : Comparaison des générations de réseaux mobiles

Le tableau comparatif des générations de réseaux mobiles montre une évolution progressive des technologies cellulaires depuis 2G jusqu'aux réseaux intelligents 5G. Cette évolution se caractérise principalement par une augmentation significative du débit de transmission, une amélioration des techniques d'accès multiple et une migration vers des architectures entièrement basées sur IP. Ces améliorations permettent aujourd'hui de supporter des applications avancées telles que l'Internet des objets, la réalité virtuelle et les communications ultra-fiables à faible latence.

I.3 La Latence Réseau

Le terme « latence d'estimation » dans le contexte des réseaux cellulaires recouvre deux notions étroitement liées. D'une part, il désigne le délai nécessaire à l'estimation de paramètres critiques du réseau, tels que le délai de transmission, la qualité du canal ou la charge du trafic. D'autre part, il renvoie à l'estimation de la latence elle-même, c'est-à-dire au processus consistant à prédire, mesurer ou inférer les délais de communication entre les différents éléments du réseau cellulaire, notamment entre l'utilisateur et la station de base[8].

Dans ce cadre, l'analyse de la latence d'estimation revêt une importance particulière pour l'optimisation des performances des réseaux cellulaires, notamment en termes de qualité de service (QoS), de gestion de la mobilité et d'allocation efficace des ressources radio.

I.3.1 Définition

La latence se réfère au délai entre l'envoi d'une demande et la réception d'une réponse. Elle est mesurée en millisecondes (ms) et est influencée par divers facteurs tels que la distance, le traitement des données et la congestion du réseau [9].

I.3.2 Facteurs influençant la latence

Plusieurs éléments peuvent influencer la latence dans les réseaux cellulaires, notamment :

Distance géographique : Plus un signal doit parcourir une distance importante, plus la latence augmente.

Congestion du réseau : Un nombre élevé d'utilisateurs actifs peut ralentir le temps de réponse.

Qualité des équipements : Les infrastructures obsolètes ou mal configurées peuvent également contribuer à une latence accrue[10].

I.3.3 Importance de la latence dans les Réseaux Cellulaires

Une latence faible est cruciale pour de nombreuses applications, en particulier celles nécessitant une interaction en temps réel, comme les jeux en ligne et les communications vidéo. Une latence élevée peut entraîner des retards perceptibles, affectant ainsi l'expérience utilisateur et l'efficacité des services [9], [10].

I.4 Travaux Existants sur l'Estimation de la Latence

De nombreuses études ont été menées pour évaluer et estimer la latence dans les réseaux cellulaires. Par exemple, des modèles d'apprentissage automatique ont été utilisés pour prédire la latence en fonction de divers paramètres réseau. Ces travaux mettent en lumière l'importance de prédire avec précision la latence réseau dans les environnements cellulaires modernes.

I.4.1 Approches par MLP et architectures légères

Zinno et al. (2025) ont exploré l'adoption d'un modèle d'apprentissage profond léger pour la prédiction du temps de round trip (RTT) dans les environnements 5G, spécifiquement conçu pour les ressources contraintes du bord du réseau. Leurs expériences démontrent qu'un perceptron multicouche (MLP) léger surpasse des modèles plus gourmands en ressources, obtenant une meilleure erreur quadratique moyenne (MSE) par rapport à plusieurs architectures de référence. Cette étude est particulièrement pertinente pour les déploiements en périphérie où les capacités de calcul sont limitées[11].

I.4.2 Approches par LSTM et réseaux récurrents

L'équipe de l'Université NOVA de Lisbonne (2025) propose une prédiction qualitative du délai de bout en bout (E2E) dans les réseaux 5G en comparant deux approches complémentaires : une méthode bayé sienne basée sur un modèle de Markov caché (HMM) et une approche d'apprentissage profond utilisant des réseaux LSTM. L'évaluation, réalisée sur le jeu de données *5G Campus*, montre que la précision dépend à la fois de la quantité de données antérieures et de l'horizon de prédiction. La méthode bayé sienne atteint une précision de 30 à 92%, mais le modèle LSTM surpasse systématiquement l'approche bayé sienne, démontrant sa supériorité pour apprendre les motifs dynamiques dans les données de latence[12]. Ce travail souligne que, bien que l'approche bayé sienne offre un cadre probabiliste interprétable, le LSTM offre des performances prédictives supérieures pour la gestion proactive des délais dans les réseaux 5G.

L'équipe de l'Université Northeastern (2025) propose une approche innovante de prévision probabiliste des délais dans les réseaux 5G en utilisant des architectures temporelles profondes (LSTM et Transformer) combinées à un réseau de densité mixte (Mixture Density Network)[13]. L'originalité de ce travail réside dans la prédiction non pas d'une valeur ponctuelle mais de la distribution complète des délais, ce qui est essentiel pour garantir des niveaux de fiabilité élevés (par exemple 99,9%). Les expériences menées sur un banc d'essai 5G Open Air Interface montrent que le modèle Transformer atteint une

log -vraisemblance négative et une MAE inférieures à celles du LSTM dans des scénarios difficiles.

I.4.3 Approches par réseaux de neurones récurrents non linéaires

Une étude récente (2025) propose l'utilisation d'un modèle autorégressif non linéaire avec entrées exogènes (NARX) basé sur un réseau de neurones récurrent (RNN) pour estimer les délais dans l'Internet des objets (IoT) et l'internet tactile [14]. Quatre fonctions d'entraînement différentes sont évaluées (Levenberg-Marquardt, gradient conjugué, rétro propagation robuste et régularisation bayé sienne). La méthode de régularisation bayé sienne (Trainbr) donne les meilleures performances en termes d'erreur quadratique moyenne (MSE), d'erreur quadratique moyenne racine (RMSE) et d'erreur absolue moyenne en pourcentage (MAPE), démontrant l'efficacité des RNN pour la prédiction de délais à plusieurs pas.

I.4.4 Approches par apprentissage contrastif et architectures avancées

Une recherche publiée dans *Computer Networks* (Elsevier, 2024) propose CCSND (*Contrastive Convolutional Structure for Network Delay*), un cadre d'apprentissage contrastif adaptatif pour la prédiction de latence dans les scénarios 5G URLLC[15] Ce modèle intègre un module d'extraction de représentation temporelle pour capturer la périodicité temporelle et les caractéristiques statistiques, ainsi que des méthodes d'augmentation de données stratégiques (segmentation temporelle et masquage des délais). Les résultats expérimentaux montrent que CCSND surpasse le modèle Informer de pointe de 38,7% et le modèle LSTM traditionnel de 64,5% en termes d'erreur quadratique moyenne (MSE), établissant une nouvelle référence dans le domaine.

I.4.5 Approches hybrides et allocation de ressources

Une étude comparative (2025) évalue plusieurs modèles d'apprentissage automatique et profond pour la prédiction de la demande en ressources dans les réseaux 5G, en utilisant des indicateurs clés de performance (KPI) réels tels que la puissance du signal, la latence et la bande passante. Un modèle hybride combinant XGBoost, GRU et mécanisme d'attention atteint une précision de 99,50 %, démontrant la supériorité des approches

hybrides pour modéliser les interactions temporelles et les dépendances entre caractéristiques[16].

Référence	Année	Architecture utilisée	Application principale	Performance clé
Zinno et al	2025	MLP léger	Prédiction RTT en environnement 5G	Meilleure MSE que les modèles plus complexes
Équipe NOVA Lisbonne	2025	LSTM vs HMM bayésien	Prédiction délai E2E 5G	LSTM surpasse l'approche bayé sienne
Mostafavi et al.	2025	LSTM, Transformer + MDN	Prévision probabiliste des délais 5G	Transformer meilleur que LSTM
Équipe Harbin Engineering	2024	CCSND (contrastif + convolutif)	Prédiction latence 5G URLLC	MSE améliorée de 38,7 % vs Informer, 64,5 % vs LSTM
Havolli & Fetaji	2025	XGBoost-GRU-Attention	Allocation ressources 5G	Précision de 99,50 %
Recherche NARX	2025	NARX-RNN	Prédiction délais IoT / tactile internet	Meilleure performance avec régularisation bayé sienne

Tableau I.2 : Tableau Récapitulatif des Études Récentes

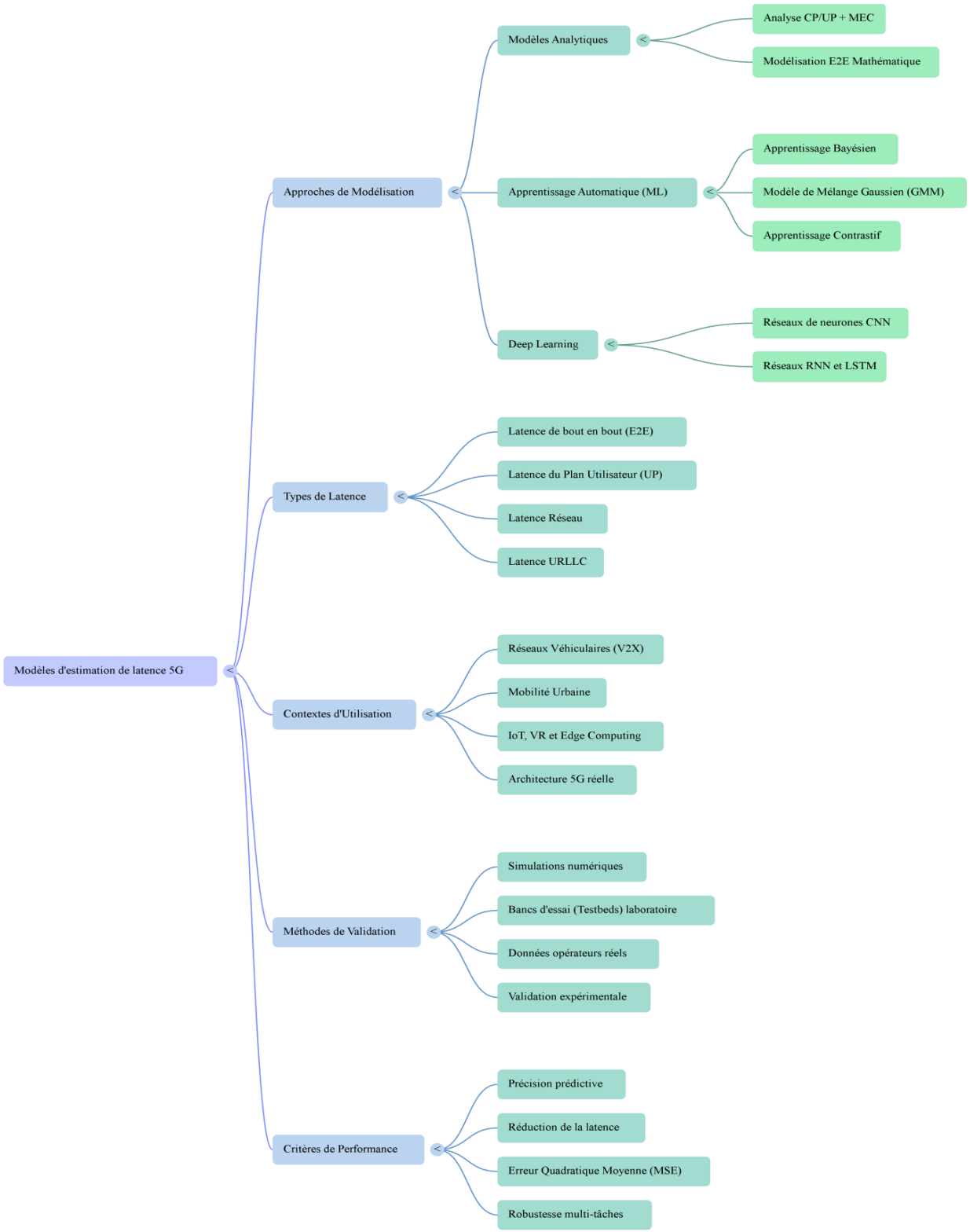


Figure I.4 : modèle d'estimation de latence 5G[17]

I.5 Conclusion

Ce chapitre a présenté l'évolution des réseaux cellulaires, en mettant l'accent sur les générations 3G, 4G et 5G. Il a également abordé la notion de latence, ses facteurs influents et son importance dans le contexte des communications modernes. Les travaux existants sur l'estimation de la latence soulignent la nécessité d'approches innovantes pour améliorer l'expérience utilisateur dans un paysage technologique en constante évolution.

Chapitre II : Les Réseaux De Neurones

II.1 Introduction

Les réseaux de neurones représentent une avancée majeure dans le domaine de l'intelligence artificielle, offrant des solutions puissantes pour une variété de problèmes complexes. Ce chapitre explore les fondements des réseaux de neurones, y compris leur définition, les différents types d'apprentissage, les architectures, et les processus d'apprentissage. Nous examinerons également les méthodes de régression et d'évaluation associées aux réseaux de neurones.

II.2 L'Intelligence Artificielle

L'intelligence artificielle (IA) englobe un large éventail de technologies qui permettent aux machines d'effectuer des tâches qui nécessitent généralement une intelligence humaine. Parmi ces technologies, les réseaux de neurones sont particulièrement bien adaptés pour traiter des données complexes et non structurées.

II.2.1 Définition

L'intelligence artificielle se définit comme la capacité d'un système à interpréter des données, apprendre de ces données, et prendre des décisions autonomes. Elle se divise en plusieurs sous-domaines, dont le Machine Learning et le Deep Learning [18].

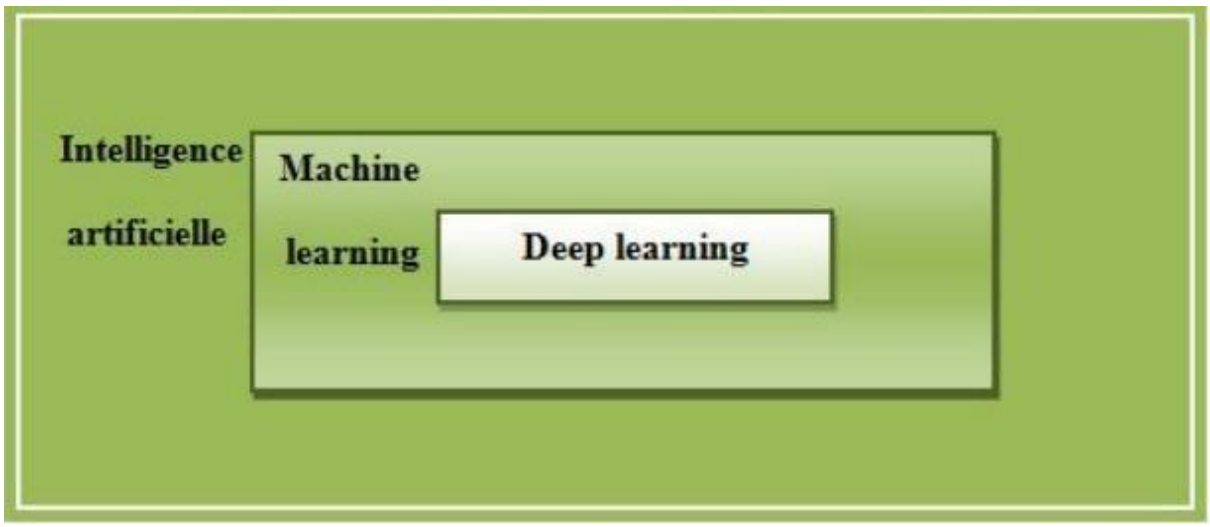


Figure II.1 : Intelligence artificielle, Machine Learning et Deep Learning[19]

II.2.2 Machine Learning

Machine Learning (apprentissage automatique) est une branche de l'IA qui permet aux systèmes d'apprendre à partir de données sans être explicitement programmés. Les algorithmes de Machine Learning s'améliorent avec l'expérience, rendant ce domaine essentiel pour l'analyse de grandes quantités de données[18],[20] Selon Bishop l'apprentissage automatique peut être divisé en trois grandes catégories : l'apprentissage supervisé (données étiquetées), non supervisé (données non étiquetées) et par renforcement (apprentissage par essais erreurs)[21].

II.2.3 Deep Learning

Le Deep Learning est une sous-catégorie du Machine Learning qui utilise des architectures de réseaux de neurones profonds, c'est-à-dire comportant plusieurs couches cachées. Ces réseaux permettent d'extraire automatiquement des caractéristiques complexes à partir des données brutes, rendant possible des avancées significatives dans des domaines tels que la vision par ordinateur, le traitement du langage naturel et la prédiction de séries temporelles[22]. Goodfellow et al. soulignent que la profondeur des architectures permet d'apprendre des représentations hiérarchiques, chaque couche transformant la représentation de la couche précédente en une représentation plus abstraite[23].

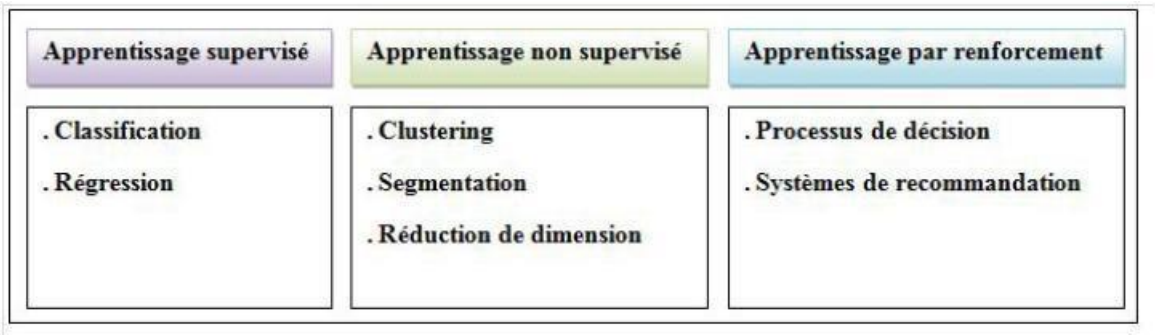


Figure II.2 : Types d'apprentissage [24]

II.3 Le Neurone Artificiel

Un neurone artificiel est l'unité fondamentale de calcul d'un réseau de neurones. Il imite le fonctionnement d'un neurone biologique en recevant plusieurs entrées, en les pondérant, en les sommant et en appliquant une fonction d'activation pour produire une sortie[25].

II.3.1 Modèle de McCulloch-Pitts

Le modèle de McCulloch-Pitts, proposé en 1943, est l'un des premiers modèles mathématiques de neurones artificiels [27]. Il simule le comportement des neurones biologiques à l'aide d'une fonction de transfert binaire : le neurone s'active (sortie = 1) lorsque la somme pondérée des entrées dépasse un certain seuil, et reste inactif (sortie = 0) dans le cas contraire. Formellement, la sortie y s'exprime :

$$y = \left\{1 \; si \sum_{i=1}^{n} \text{wixi} \geq \theta\right\}$$

$$= 0 \text{ sinon}$$

$$y = f\left(\sum_{i=1}^{n} \text{wixi} + \text{b}\right) pour\ le\ perceptron$$

avec *f* une fonction seuil, wi les poids synaptiques, xi les entrées et *b* le biais[28]

II.3.2 Fonctions d'activation (ReLU, Sigmoïde)

Les fonctions d'activation déterminent la sortie d'un neurone en fonction de son entrée nette. Elles introduisent la non linéarité indispensable pour que le réseau puisse apprendre des relations complexes.

- **ReLU (Rectified Linear Unit)** : définie par ReLU(x)=max(0,x). Elle active le neurone si l'entrée est positive, sinon elle renvoie zéro. ReLU est largement utilisée car elle évite le problème du gradient vanishing et accélère la convergence[29].
- **Sigmoïde** : Elle produit une sortie comprise entre 0 et 1, ce qui la rend adaptée aux problèmes de classification binaire, mais elle souffre du problème du gradient vanishing pour des valeurs extrêmes $\sigma(x) = \frac{1}{1+e^{-x}}$ [30].
- **Tangente hyperbolique (tanh)** : variante de la sigmoïde produisant une sortie entre -1 et 1.

$$\tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \quad [31].$$

II.4 Architectures des Réseaux de Neurones

Les réseaux de neurones se distinguent par leur architecture, c'est-à-dire l'organisation des neurones en couches et les connexions entre ces couches. On distingue principalement les réseaux feedforward (unidirectionnels), les réseaux récurrents (avec boucles de rétroaction) et les réseaux convolutifs (adaptés aux données spatiales).

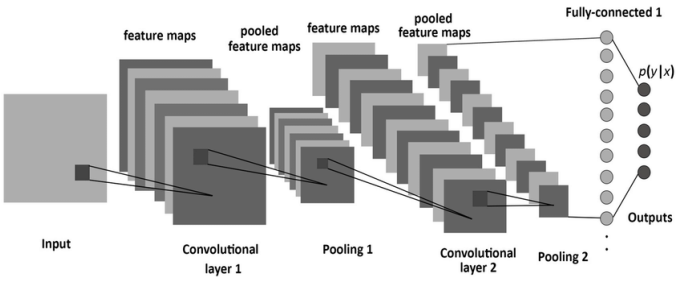


Figure II.3 : Architecture classique d'un réseau de neurones convolutif [32]

II.4.1 Réseau Feedforward

Le réseau feedforward est l'architecture la plus simple : les données circulent dans une seule direction, de la couche d'entrée vers la couche de sortie, sans boucle de rétroaction. Chaque neurone d'une couche est connecté à tous les neurones de la couche suivante. Ce type de réseau est principalement utilisé pour des tâches de classification et de régression statique[33].

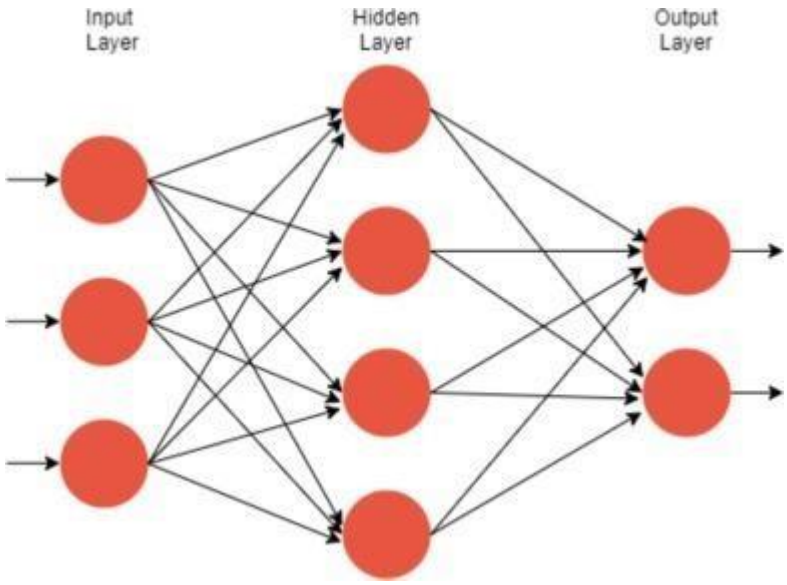


Figure II.4 : Architecture d'un réseau de neurones artificiels (ANN) [33].

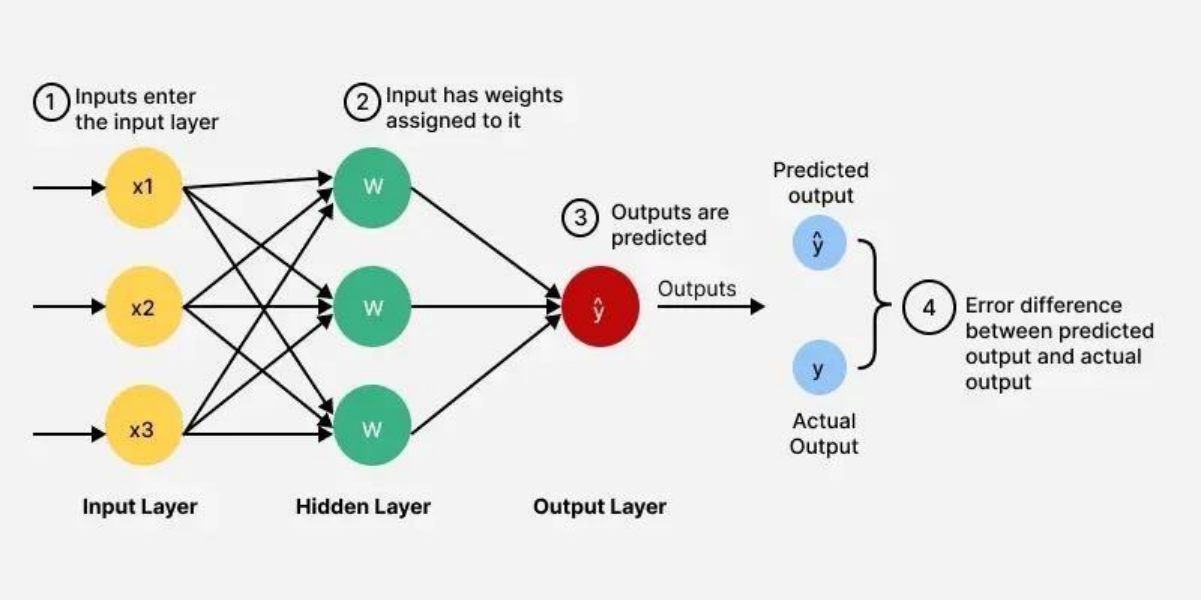


Figure II.5 : Mécanisme de pondération et de calcul de l'erreur[34].

II.4.2 Perceptron multicouche (MLP)

Le perceptron multicouche (MLP) est une forme avancée de réseau feedforward comportant au moins une couche cachée entre la couche d'entrée et la couche de sortie[35]. Les MLP sont capables de modéliser des relations non linéaires complexes grâce à leur profondeur et à l'utilisation de fonctions d'activation non linéaires. Selon Haykin [36]. un MLP avec une seule couche cachée suffisante en nombre de neurones est un approximateur universel, capable d'approcher toute fonction continue sur un intervalle compact.

La sortie de la couche cachée I dans un réseau de neurones de type perceptron

Multicouche (MLP) est donnée par [37] :

$$h^{[l]} = f^{[l]}(W^{[l]}h^{[l-1]} + b^{[l]})$$

h(l) : vecteur de sortie (activations) de la couche l

W(l) : matrice des poids de la couche l

b(l) : vecteur des biais

h(l−1) : sortie de la couche précédente

f(l) : fonction d'activation (ReLU, sigmoïde, tanh)

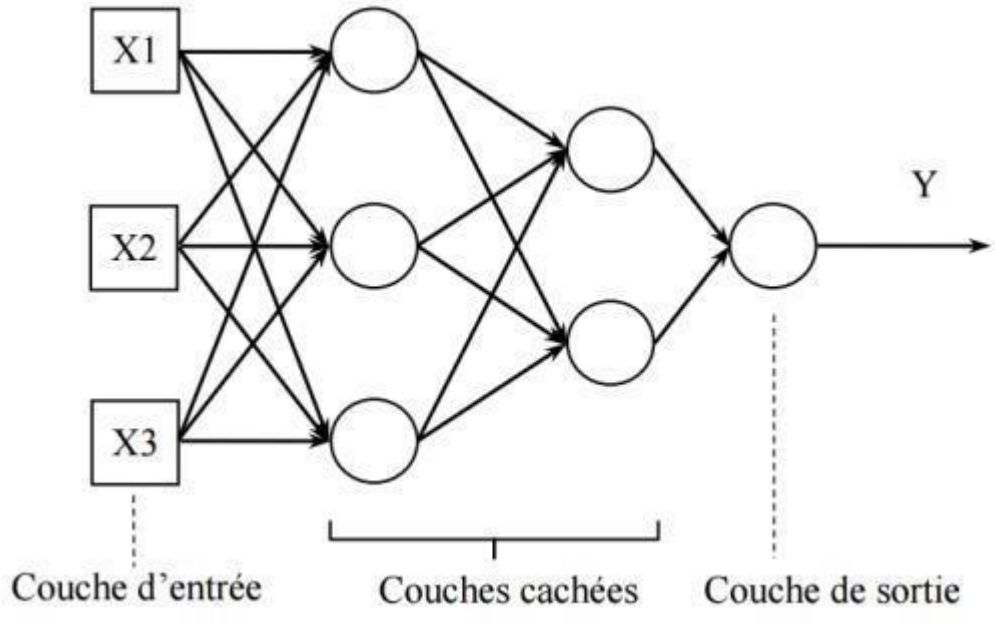


Figure II.6 : Architecture d'un réseau de neurones multicouches[38]

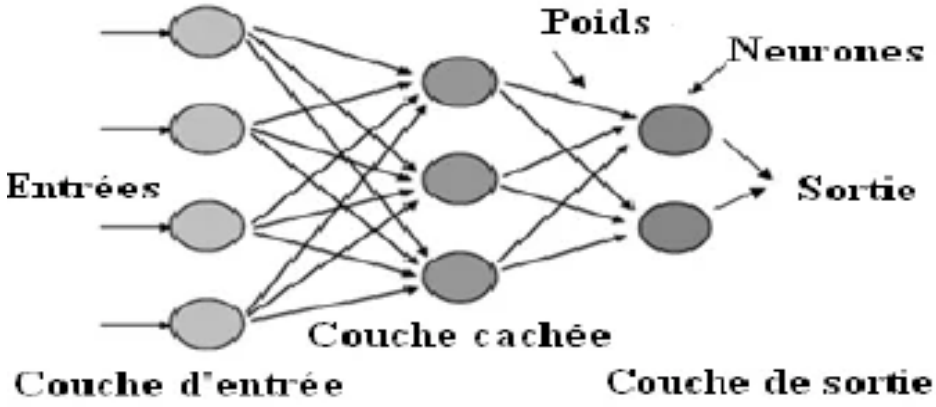


Figure II.7 : Structure des Couches de Traitement et de Pondération[38]

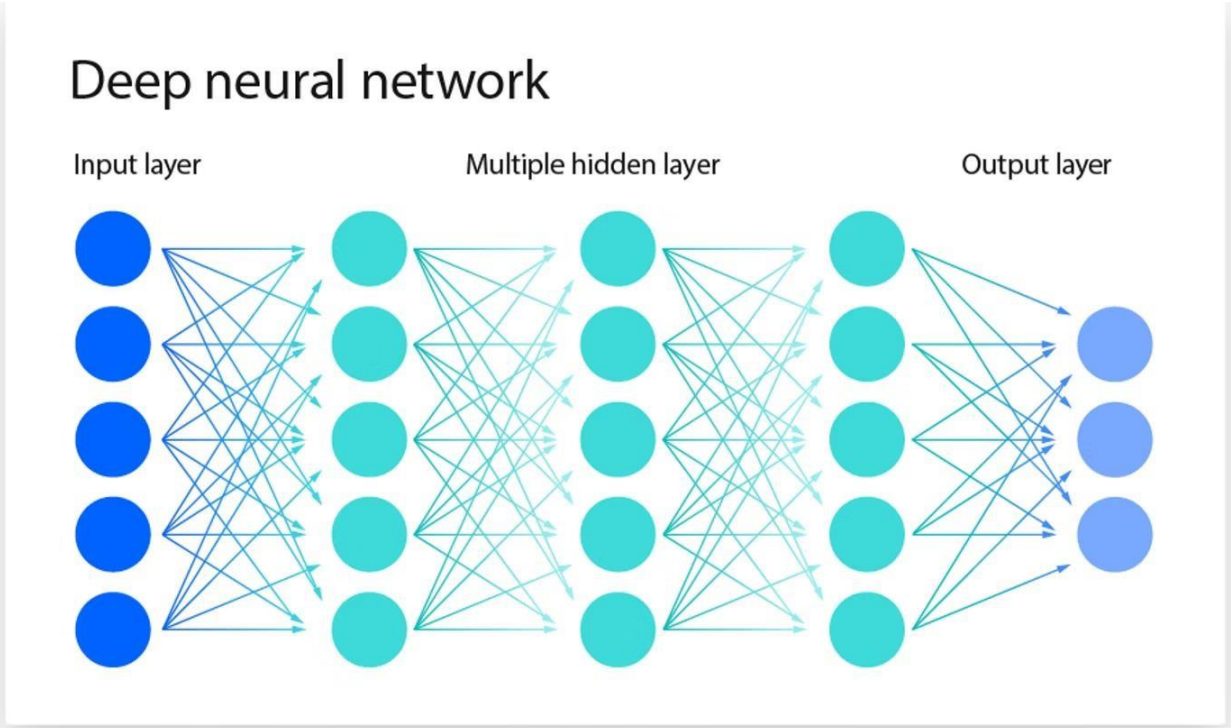


Figure II.8 : Structure fonctionnelle du Perceptron Multicouche (MLP) [39]

II.5 Processus d'apprentissage

II.5.1 Rétropropagation

Les poids sont ensuite mis à jour par descente de gradient :

$$w_{ij} \leftarrow w_{ij} - \eta \frac{\partial L}{\partial w_{ij}}$$

où η est le taux d'apprentissage et L la fonction de perte. Cet algorithme, combiné à la descente de gradient stochastique, permet aux réseaux profonds d'« apprendre » à partir des données[40].

II.5.2 Optimiseur Adam

Adam (Adaptive Moment Estimation) est un optimiseur largement utilisé qui combine les avantages de deux autres méthodes : AdaGrad (qui adapte le taux d'apprentissage pour chaque paramètre) et RMSProp (qui utilise une moyenne mobile des gradients au carré) [41], [42]. Adam maintient deux estimateurs :

- m_t : moyenne mobile exponentielle des gradients (premier moment)
- v_t : moyenne mobile exponentielle des carrés des gradients (second moment)

Les paramètres sont mis à jour selon :

$$\theta t + 1 = \theta t - \eta \frac{m\hat{}t}{\sqrt{\hat{}}t + \epsilon}$$

Adam offre un taux de convergence rapide, même pour les jeux de données bruités ou clairsemés.Ses hyperparamètres par défaut $\eta = 0.001, \quad \beta_1 = 0.9, \quad \beta_2 = 0.999, \quad \epsilon = 10^{-8}$

fonctionnent bien dans la plupart des cas

II.5.3 Régularisation (Dropout, BatchNorm, L2)

La régularisation est essentielle pour éviter le surapprentissage (overfitting), c'est-à-dire que le modèle mémorise les données d'entraînement sans généraliser correctement sur des données nouvelles .

- **Dropout** : proposé par Srivastava et al. , cette technique désactive aléatoirement une fraction des neurones à chaque itération d'entraînement. Cela force le réseau à apprendre des représentations redondantes et robustes. Le taux de dropout (typiquement entre 0,2 et 0,5) est un hyper paramètre à ajuster $\tilde{h}_i = \frac{r_i}{p} h_i$[43]
- **Batch Normalisation** : introduite par Ioffe et Szegedy , cette technique normalise les activations de chaque couche pour qu'elles aient une moyenne nulle et un écart type unitaire. Elle stabilise l'apprentissage, accélère la convergence et permet d'utiliser des taux d'apprentissage plus élevés $\mathbf{y_i} = \boldsymbol{\gamma} \frac{((\mathbf{x_i} - \boldsymbol{\mu}\_\mathbf{B})}{\sqrt{(\boldsymbol{\sigma}\_\mathbf{B}^2 + \boldsymbol{\varepsilon}))}} + \boldsymbol{\beta}$ [44]

II.6 Régression par Réseaux de Neurones

Les réseaux de neurones peuvent être utilisés pour des tâches de régression, en plus des classifications.

II.6.1 Classification vs Régression

La classification et la régression sont deux types de problèmes en apprentissage supervisé

Classification : la variable cible est discrète (catégorielle). Le réseau apprend à assigner chaque entrée à une classe (exemple : reconnaissance d'image, détection de spam). La

fonction d'activation de sortie est généralement soft max (multi classes) ou sigmoïde (binaire)[45].

- **Régression** : la variable cible est continue. Le réseau apprend à prédire une valeur numérique (exemple : estimation de latence, prédiction de prix). La couche de sortie comporte un seul neurone avec activation linéaire.

II.6.2 Métriques d'Évaluation (MAE, RMSE, R²)

Pour évaluer les performances des modèles de régression, trois métriques principales sont utilisées

- **MAE (Mean Absolute Error)** : erreur absolue moyenne. Elle mesure en moyenne l'écart absolu entre les valeurs prédites y^iy^i et les valeurs réelles yiyi :

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i|$$

La MAE est intuitive et robuste aux outliers.

- **RMSE (Root Mean Square Error)** : racine de l'erreur quadratique moyenne. Elle pénalise plus lourdement les grandes erreurs que la MAE :

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}$$

- **R² (Coefficient de détermination)** : proportion de la variance de la variable cible expliquée par le modèle. Un R² proche de 1 indique un excellent ajustement[46] :

$$R^2 = 1 - \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2}$$

II.7 Conclusion

Ce chapitre a présenté les fondements théoriques des réseaux de neurones artificiels. Après avoir situé le Deep Learning dans le contexte plus large de l'intelligence artificielle et de la Machine Learning, nous avons détaillé le fonctionnement du neurone artificiel, les principales fonctions d'activation et les architectures de réseaux (feed forward, MLP). Les mécanismes d'apprentissage (rétro propagation, optimiseur Adam) et les techniques de régularisation (Dropout, BatchNorm, L2) ont été expliqués, car ils sont essentiels pour entraîner des modèles performants sans sur apprentissage. Enfin, les spécificités des problèmes de régression et les métriques d'évaluation (MAE, RMSE, R²) ont été

introduites. Ces concepts seront mis en œuvre dans les chapitres suivants pour l'estimation de la latence dans les réseaux cellulaires.

Chapitre III : Méthodologie et Outils

III.1 Introduction

Ce chapitre présente la méthodologie adoptée ainsi que l'ensemble des outils utilisés pour la conception, le développement et le déploiement du système d'estimation de la latence réseau dans les réseaux cellulaires. Après avoir exposé dans les chapitres précédents les fondements théoriques relatifs aux réseaux cellulaires et aux réseaux de neurones artificiels, il s'agit ici de décrire la chaîne de traitement mise en œuvre, depuis l'exploration des données brutes jusqu'à la mise à disposition d'une interface interactive de prédiction.

Le système proposé repose sur un pipeline structuré en six étapes successives, chacune implémentée dans un fichier Python indépendant, assurant ainsi une modularité et une traçabilité optimales du processus de développement. Les données exploitées proviennent d'un jeu de 16 829 mesures réelles collectées sur des réseaux 3G, 4G, LTE et 5G, et le modèle de prédiction repose sur un réseau de neurones profond entraîné et évalué selon des protocoles rigoureux.

III.2 Présentation du système proposé

Le système développé dans le cadre de ce mémoire a pour objectif principal d'estimer la latence réseau, exprimée en millisecondes, à partir de caractéristiques mesurables du signal cellulaire. L'architecture globale du système s'articule autour d'un pipeline de traitement séquentiel composé de six modules fonctionnels, allant de l'exploration des données brutes jusqu'à la présentation des résultats via une interface web interactive.

Le jeu de données source, dénommé signal_metrics.csv, contient 16 829 mesures réelles de réseaux cellulaires, réparties sur quatre types de réseaux (3G, 4G, LTE et 5G) et collectées en plusieurs localisations géographiques.

Le système proposé vise à estimer la latence (Latence (ms)) à partir d'indicateurs clés de performance (KPI) relevés sur le terrain. Le jeu de données comprend des mesures temporelles (timestamp), des informations géographiques (localité, latitude, longitude), des métriques de qualité radio (puissance du signal, débit), le type de réseau (3G, 4G, LTE, 5G) ainsi que trois mesures de puissance issues de dispositifs matériels distincts (BB60C, srsRAN, BladeRFxA9). La latence, variable cible, présente une forte variabilité selon les conditions de propagation et la technologie d'accès.

L'architecture de réseaux de neurones suivante est mise en œuvre :

- MLP (Perceptron Multicouche) : réseau Feedforward entièrement connecté, adapté à l'apprentissage de relations statiques entre les caractéristiques d'entrée et la latence. Il traite chaque échantillon de manière indépendante.

Ce modèle est implémenté, entraîné et évalué suivant un pipeline rigoureux décrit ci-après.

III.3 Outils et Environnement de Développement

III.3.1 Python

L'ensemble du développement a été réalisé en langage Python dans sa version 3.10, au sein d'un environnement virtuel dédié nommé tf_env. L'utilisation d'un environnement virtuel permet d'isoler les dépendances du projet et d'assurer la reproductibilité des expérimentations[47]. L'environnement de développement intégré (IDE) utilisé est Visual Studio Code, couplé à un système de gestion de versions Git hébergé sur un dépôt public GitHub, facilitant ainsi le suivi des modifications et la collaboration. L'autre L'environnement de développement utilisé est Jupyter Notebook, qui facilite l'exploration interactive des données et l'itération sur les modèles.

III.3.2 TensorFlow / Keras

La bibliothèque TensorFlow en version 2.20.0, accessible via son API de haut niveau Keras, constitue le Framework principal pour la construction et l'entraînement du réseau de neurones[48]. Keras offre une interface simple et modulaire permettant de définir des architectures de réseaux de neurones sous forme de modèles séquentiels, de compiler le modèle avec un optimiseur et une fonction de perte, ainsi que de gérer l'entraînement à l'aide de callbacks pour le contrôle du processus d'apprentissage. L'optimiseur Adam, développé par Kingma et Ba[49], a été retenu pour sa capacité à adapter le taux d'apprentissage par paramètre et à accélérer la convergence.

III.3.3 Scikit-learn

La bibliothèque Scikit-learn est utilisée pour plusieurs opérations essentielles du pipeline de traitement, notamment la division des données en ensembles d'entraînement, de validation et de test, la normalisation des caractéristiques numériques par la méthode StandardScaler, ainsi que l'implémentation de deux modèles de référence : la régression linéaire (LinearRegression) et la forêt aléatoire (RandomForestRegressor). Elle fournit également les métriques d'évaluation nécessaires au calcul de l'erreur absolue moyenne, de la racine de l'erreur quadratique moyenne et du coefficient de détermination R^2[50].

III.3.4 Pandas et NumPy

Pandas est la bibliothèque de référence pour le chargement et la manipulation du jeu de données tabulaires. Elle permet la lecture du fichier CSV, l'inspection des types de données, le traitement des valeurs manquantes, l'extraction de caractéristiques temporelles à partir du time stamp, ainsi que l'encodage des variables catégorielles. NumPy, quant à elle, assure la gestion des tableaux numériques et les opérations mathématiques sous-jacentes, notamment lors des transformations de normalisation et des calculs de prédiction[51]

III.3.5 Matplotlib

La bibliothèque Matplotlib est employée pour la génération de l'ensemble des visualisations graphiques produites au cours du projet, incluant les courbes d'évolution de la fonction de perte et des métriques au fil des époques d'entraînement, ainsi que les graphiques de comparaison des performances entre le réseau de neurones et les modèles de référence[52]. Ces représentations visuelles sont essentielles pour l'analyse qualitative du comportement du modèle et la communication des résultats.

III.3.6 Streamlit

Streamlit est un Framework Python open source dédié à la création rapide d'applications web interactives orientées data science. Il a été utilisé pour développer l'interface utilisateur du système de prédiction, permettant à un utilisateur de sélectionner un type de réseau, de saisir les mesures de signal disponibles, d'obtenir une estimation de la latence en temps réel accompagnée d'une indication qualitative de la qualité de connexion, et de comparer les prédictions selon les différents types de réseau[53].

III.4 Pipeline du système (6 étapes)

Le pipeline de traitement se décompose en six étapes séquentielles, chacune encapsulée dans un fichier Python indépendant. Cette modularité garantit une séparation claire des

responsabilités et facilite la maintenance ainsi que la réutilisation des différentes composantes.

III.4.1 Exploration des données

La première étape du pipeline consiste en une exploration systématique du jeu de données brut. Le fichier explore_data.py charge le fichier signal_metrics.csv à l'aide de Pandas et affiche les caractéristiques générales du jeu de données : 16 829 lignes et 12 colonnes. L'analyse des types de données révèle un mélange de variables numériques et catégorielles. Le décompte des valeurs manquantes indique l'absence totale de données absentes, ce qui constitue un atout majeur pour la qualité du jeu de données.

Les statistiques descriptives calculées sur l'ensemble des variables numériques permettent d'appréhender la distribution des valeurs et de détecter d'éventuelles anomalies. L'analyse des valeurs uniques par colonne met en évidence un résultat déterminant : la colonne Signal Quality (%) présente une valeur unique de zéro pour l'ensemble des 16 829 enregistrements, ce qui la rend totalement non informative pour la tâche de prédiction. Par ailleurs, l'analyse de la répartition de la variable Network Type montre une distribution équilibrée entre les quatre classes, chacune comptant environ 4 200 occurrences, ce qui est favorable à l'apprentissage d'un modèle généralisable.

III.4.2 Prétraitement

La seconde étape, implémentée dans le fichier prepare_data.py, concerne le prétraitement et la préparation des données en vue de l'entraînement du modèle. La première opération consiste à supprimer la colonne Signal Quality (%) en raison de son absence de variance. Ensuite, la colonne Time stamp fait l'objet d'un traitement permettant d'extraire deux nouvelles caractéristiques temporelles : l'heure de la mesure (hour) et le jour de la semaine (day_of_week), susceptibles de capturer des patterns de congestion réseau liés à l'activité des utilisateurs.

La variable catégorielle Network Type fait l'objet d'un encodage one-hot à l'aide de la fonction pd.get_dummies(), générant quatre colonnes binaires distinctes : net_3G, net_4G, net_5G et net_LTE. Au terme de ces opérations, le jeu de données final comporte onze caractéristiques prédictives : Signal Strength (dBm), Data Throughput (Mbps), BB60C Measurement (dBm), srsRAN Measurement (dBm), BladeRFxA9 Measurement (dBm), hour, day_of_week, net_3G, net_4G, net_5G et net_LTE.

La division des données s'effectue selon les proportions suivantes : 70% pour l'entraînement, soit 11 786 échantillons, 15% pour la validation, soit 2 518 échantillons, et 15% pour le test, soit 2 525 échantillons. Cette répartition stratifiée assure que chaque sous-ensemble est représentatif de la distribution globale des données. La normalisation des caractéristiques numériques est réalisée à l'aide de StandardScaler, qui soustrait la moyenne et divise par l'écart-type. Il est impératif de noter que le scaler est ajusté uniquement sur l'ensemble d'entraînement, puis appliqué aux ensembles de validation et de test, afin d'éviter toute fuite d'information (data leakage). Les données préparées sont sauvegardées sous forme de fichiers CSV (train.csv, val.csv, test.csv), et le scaler ajusté est sérialisé dans le fichier scaler.pkl à l'aide de Joblib pour une réutilisation ultérieure. Un fichier feature_names.json conserve également la liste des noms des caractéristiques.

III.4.3 Architecture du modèle

Le fichier model.py définit l'architecture du réseau de neurones profond chargé de prédire la latence. Il s'agit d'un modèle séquentiel de type perceptron multicouches (feedforward), conçu pour une tâche de régression. La couche d'entrée reçoit les onze caractéristiques prédictives. L'architecture se compose ensuite de trois couches cachées suivies d'une couche de sortie.

La première couche cachée comporte 128 neurones, suivie d'une couche de normalisation par lots (Batch Normalization) d'une activation ReLU, d'un dropout de taux 0,3 et d'une régularisation L2 sur les poids. La seconde couche cachée comporte 64 neurones, également suivie de Batch Normalization, d'une activation ReLU, d'un dropout de taux 0,2 et d'une régularisation L2. La troisième couche cachée comporte 32 neurones avec activation ReLU. La couche de sortie est constituée d'un seul neurone sans fonction d'activation, produisant une valeur scalaire continue représentant la latence estimée en millisecondes. Le nombre total de paramètres entraînables du modèle est de 12 673.

Le modèle est compilé avec l'optimiseur Adam configuré avec un taux d'apprentissage de 0,001 La fonction de perte utilisée est l'erreur quadratique moyenne (Mean Squared Error), adaptée aux problèmes de régression. Les métriques suivies pendant l'entraînement sont l'erreur absolue moyenne (MAE) et la racine de l'erreur quadratique moyenne (RMSE).

III.4.4 Entraînement

Le fichier train.py gère la phase d'entraînement du réseau de neurones. Il charge les fichiers train.csv et val.csv, puis lance l'entraînement avec une taille de lot (batch size) de 32 et un nombre maximal d'époques fixé à 100. Quatre mécanismes de callback sont mis en place pour contrôler et optimiser le processus d'apprentissage.

Le callback EarlyStopping surveille la perte de validation (val_loss) avec une patience de 10 époques et restaure les poids du meilleur modèle observé, prévenant ainsi le sur apprentissage. Le callback Reduce LROnPlateau réduit le taux d'apprentissage d'un facteur 0,5 lorsque la perte de validation stagne pendant 5 époques consécutives, avec une borne inférieure de 1e-6. Le callback Model Checkpoint sauvegarde automatiquement le modèle présentant la meilleure perte de validation dans le fichier best_model.keras. Enfin, le callback Tensor Board journalise l'ensemble des métriques dans un répertoire logs/ pour une visualisation interactive ultérieure.

En pratique, l'entraînement s'est arrêté à la 22 époque grâce au mécanisme d'arrêt précoce, le meilleur modèle ayant été identifié à la 12 époque.

III.4.5 Évaluation

Le fichier evaluate.py a pour mission d'évaluer les performances du modèle entraîné sur l'ensemble de test et de les comparer à celles de deux modèles de référence. Il charge le fichier best_model.keras ainsi que l'ensemble de test (test.csv), puis calcule les trois métriques principales : MAE, RMSE et R^2.

Parallèlement, deux modèles de référence sont entraînés sur les mêmes données d'entraînement et évalués sur le même ensemble de test : une régression linéaire (LinearRegression) et une forêt aléatoire (RandomForestRegressor avec 100 estimateurs), tous deux issus de la bibliothèque Scikit-learn.

L'analyse de ces résultats révèle que les trois modèles affichent des performances globalement comparables sur ce jeu de données.

Le fichier evaluate.py génère également deux visualisations : un graphique composé de six panneaux (evaluation_results.png) présentant les nuages de points des valeurs prédites versus réelles, les distributions d'erreur et les diagrammes comparatifs des métriques pour chaque modèle, ainsi qu'un tableau coloré des prédictions sur 5 échantillons par type de réseau (sample_predictions_table.png).

III.4.6 Prédiction

La dernière étape du pipeline concerne la mise à disposition du modèle entraîné pour des prédictions en conditions réelles, via deux composantes complémentaires. Le fichier predict.py implémente la fonction de prédiction autonome. Il charge le modèle best_model.keras et le scaler scaler.pkl, reçoit les entrées utilisateur sous forme de Data Frame Pandas dont les noms de colonnes correspondent exactement aux onze caractéristiques attendues, applique la transformation de normalisation via scaler.transform(), puis exécute la prédiction par model.predict(). Le résultat retourné est la latence estimée en millisecondes (ms).

Le fichier app.py constitue l'interface utilisateur du système. Il s'agit d'une application web interactive développée avec Streamlit qui offre plusieurs fonctionnalités. L'utilisateur peut sélectionner un type de réseau parmi les quatre disponibles (3G, 4G, LTE, 5G) et saisir les mesures de signal correspondantes. En cliquant sur le bouton de prédiction, l'application affiche la latence estimée accompagnée d'un label qualitatif indiquant la qualité de connexion (Excellent, Good, Fair ou Poor), ainsi que d'une barre de progression visuelle. La barre latérale de l'application présente les informations sur les performances du modèle, et un bouton "Compare All Network Types" permet d'afficher les prédictions pour l'ensemble des types de réseau avec les mêmes paramètres de signal.

III.5 Conclusion

Ce chapitre a présenté de manière détaillée la méthodologie et les outils utilisés pour le développement du système d'estimation de la latence réseau. L'architecture modulaire en six étapes successives, chacune implémentée dans un fichier Python dédié, a permis de structurer le processus de développement de manière claire et reproductible, depuis l'exploration des données brutes jusqu'à la mise à disposition d'une interface interactive de prédiction.

Le choix des outils s'est porté sur un écosystème Python mature et largement adopté dans la communauté scientifique : TensorFlow et Keras pour la modélisation par réseaux de neurones Scikit-learn pour le prétraitement et les modèles de référence Pandas et NumPy pour la manipulation des données , Matplotlib pour la visualisation, et Streamlit pour le développement de l'interface utilisateur.

Le chapitre suivant présentera l'implémentation concrète du système, les résultats expérimentaux détaillés ainsi qu'une analyse critique des performances du modèle proposé.

Chapitre IV : Implémentation et Résultats

IV.1 Introduction

Après avoir posé le cadre méthodologique dans le chapitre précédent, nous passons désormais à la partie concrète de notre travail. Ce chapitre relate fidèlement comment nous avons implémenté le système d'estimation de la latence réseau, quelles données réelles nous avons utilisées, et quels résultats nous avons obtenus.

Nous commencerons par une présentation minutieuse du jeu de données collecté sur le terrain, puis nous décrirons toutes les étapes de prétraitement qui l'ont rendu exploitable. Ensuite, nous détaillerons l'architecture du réseau de neurones que nous avons conçu, les paramètres d'entraînement choisis, et les mécanismes d'évaluation mis en place. Les résultats chiffrés seront analysés, comparés à d'autres modèles, et discutés avec honnêteté. Enfin, nous présenterons l'interface utilisateur développée avec Streamlit pour rendre notre outil accessible au plus grand nombre.

L'objectif est ici de montrer non seulement ce qui fonctionne, mais aussi d'expliquer nos choix et d'interpréter les écarts observés.

IV.2 Présentation du jeu de données

IV.2.1 Source et description

Le jeu de données que nous avons exploité s'intitule signal_metrics.csv. Il a été publié sur la plateforme Kaggle par l'utilisateur Suraj sous le titre "Cellular Network Analysis Dataset"

Concrètement, il s'agit d'un ensemble de 16 829 mesures réelles de signaux cellulaires, collectées directement sur le terrain dans 20 localités différentes de l'État du Bihâr, en Inde. Parmi ces localités, on trouve par exemple Kankarbagh, Rajendra Nagar ou encore Boring Road.

Ces mesures n'ont pas été générées artificiellement. Elles proviennent d'un équipement matériel professionnel comprenant un analyseur de spectre BB60C, le logiciel de radio définie par logiciel srsRAN, et un périphérique SDR BladeRFxA9. L'ensemble était piloté

depuis une console Valve Steam Deck équipée de DragonOS. Cette chaîne d'acquisition garantit l'authenticité des données et leur représentativité des conditions réelles d'utilisation.

Chaque ligne du fichier contient plusieurs informations : un horodatage (avec un intervalle de 10 minutes entre les mesures), la localisation géographique (localité, latitude, longitude), la puissance du signal reçu exprimée en dBm (qui varie de -116 dBm à -74 dBm), la qualité du signal (en pourcentage, mais nous verrons qu'elle pose problème), le débit de données en Mbps, et enfin la latence en millisecondes — notre variable cible. Cette latence s'étend de 10 ms à 200 ms, avec une moyenne d'environ 101 ms. Le type de réseau est également précisé : 3G, 4G, 5G ou LTE, avec une répartition remarquablement équilibrée (entre 4 178 et 4 224 échantillons par classe). Enfin, trois colonnes supplémentaires enregistrent les mesures issues respectivement des trois capteurs (BB60C, srsRAN et BladeRFxA9).

	Timestamp	Locality	Latitude	Longitude	Signal Strength (dBm)	Signal Quality (%)	Data Throughput (Mbps)	Latence (ms)	Network Type	BB60C Measurement (dBm)	srsRAN Measurement (dBm)	BladeRFxA9 Measurement (dBm)
0	2023-05-05 12:50:40.000000	Anisabad	25.599109	85.137355	-84.274113	**0.0**	1.863890	129.122914	3G	0.000000	0.000000	0.000000
1	2023-05-05 12:53:47.210173	Fraser Road	25.433286	85.070053	-97.653121	**0.0**	5.132296	54.883606	4G	-95.810791	-105.452359	-99.920892
2	2023-05-05 12:56:54.420346	Boring Canal Road	25.498809	85.211371	-87.046134	**0.0**	1.176985	119.598286	LTE	-91.593861	-95.419482	-87.714070
3	2023-05-05 13:00:01.630519	Danapur	25.735138	85.208400	-94.143159	**0.0**	68.596932	46.598387	5G	-90.642773	-101.895905	-96.570698
...	...	...	...	...	...	**0.0**	...	...	...	...	...	...
16826	2023-06-10 23:50:38.370898	Boring Road	25.574020	85.030036	-90.451396	**0.0**	72.870842	32.556578	5G	-91.098875	-97.447725	-87.752628
16827	2023-06-10 23:53:45.581071	Boring Road	25.619325	85.183155	-85.661814	**0.0**	2.482843	144.007572	3G	0.000000	0.000000	0.000000

	Timestamp	Locality	Latitude	Longitude	Signal Strength (dBm)	Signal Quality (%)	Data Throughput (Mbps)	Latence (ms)	Network Type	BB60C Measurement (dBm)	srsRAN Measurement (dBm)	BladeRFxA9 Measurement (dBm)
16828	2023-06-10 23:56:52.791244	Rajendra Nagar	25.682516	85.264655	-93.228967	**0.0**	1.638291	123.234570	LTE	-95.598301	-99.438645	-94.268015

Tableau IV.1 : Description des données brutes

16829 rows × 12 columns

IV.2.2 Variables et caractéristiques

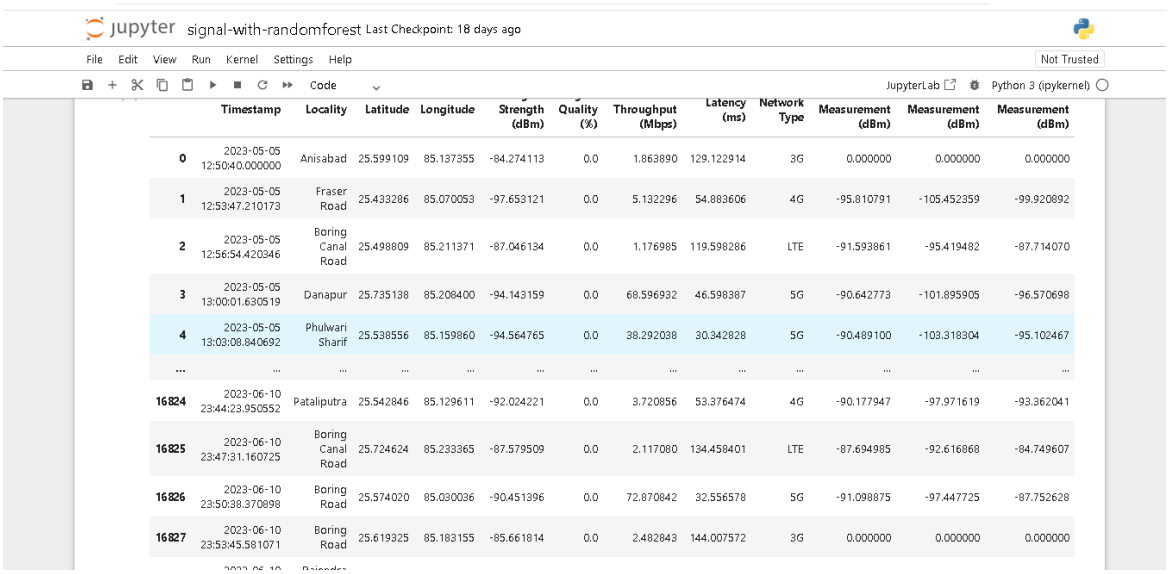
Avant toute manipulation, nous avons procédé à une analyse minutieuse du contenu. Cette exploration nous a livré plusieurs renseignements importants.

Le premier constat, et non des moindres, est qu'il n'y a aucune valeur manquante dans l'ensemble du jeu de données. Sur les 16 829 enregistrements et les 12 colonnes, toutes les cases sont remplies. C'est un atout considérable car cela nous évite d'avoir à recourir à des techniques d'imputation qui auraient pu introduire des biais.

Le deuxième constat est plus problématique : la colonne Signal Quality (%) contient exclusivement des zéros. Cela révèle manifestement une erreur systématique au niveau du capteur ou du protocole de collecte. Nous avons donc logiquement décidé d'écarter complètement cette variable de notre analyse, car elle n'apporterait aucune information utile.

Le troisième constat est rassurant : la variable Network Type est remarquablement équilibrée. Les quatre catégories (3G, 4G, 5G, LTE) comptent chacune entre 4 178 et 4 224 échantillons. Cet équilibre nous préserve d'un éventuel biais de prédiction en faveur d'une classe majoritaire et facilite l'apprentissage d'un modèle véritablement généralisable.

Enfin, quatrième constat : l'analyse de la latence selon le type de réseau montre des profils très différenciés et parfaitement cohérents avec nos attentes technologiques. Les réseaux 5G affichent une latence moyenne d'environ 30 ms, les réseaux 4G une latence d'environ 75 ms, tandis que les réseaux 3G et LTE tournent autour de 149-150 ms. Cette hiérarchie est exactement celle que l'on attend d'un point de vue technique, ce qui confirme la pertinence du type de réseau comme prédicteur.


jupyter signal-with-randomforest Last Checkpoint: 18 days ago

File Edit View Run Kernel Settings Help

Not Trusted

Code

JupyterLab Python 3 (ipykernel)

	Timestamp	Locality	Latitude	Longitude	Strength (dBm)	Quality (%)	Throughput (Mbps)	Latency (ms)	Network Type	Measurement (dBm)	Measurement (dBm)	Measurement (dBm)
0	2023-05-05 12:50:40.000000	Anisabad	25.599109	85.137355	-84.274113	0.0	1.863890	129.122914	3G	0.000000	0.000000	0.000000
1	2023-05-05 12:53:47.210173	Fraser Road	25.433286	85.070053	-97.653121	0.0	5.132296	54.883606	4G	-95.810791	-105.452359	-99.920892
2	2023-05-05 12:56:54.420346	Boring Canal Road	25.498809	85.211371	-87.046134	0.0	1.176985	119.598286	LTE	-91.593861	-95.419482	-87.714070
3	2023-05-05 13:00:01.630519	Danapur	25.735138	85.208400	-94.143159	0.0	68.596932	46.598387	5G	-90.642773	-101.895905	-96.570698
4	2023-05-05 13:03:08.840692	Phulwari Sharif	25.538556	85.159860	-94.564765	0.0	38.292038	30.342828	5G	-90.489100	-103.318304	-95.102467
...	...	...	...	...	...	...	...	...	...	...	...	...
16824	2023-06-10 23:44:23.950552	Pataliputra	25.542846	85.129611	-92.024221	0.0	3.720856	53.376474	4G	-90.177947	-97.971619	-93.362041
16825	2023-06-10 23:47:31.160725	Boring Canal Road	25.724624	85.233365	-87.579509	0.0	2.117080	134.458401	LTE	-87.694985	-92.616868	-84.749607
16826	2023-06-10 23:50:38.370898	Boring Road	25.574020	85.030036	-90.451396	0.0	72.870842	32.556578	5G	-91.098875	-97.447725	-87.752628
16827	2023-06-10 23:53:45.581071	Boring Road	25.619325	85.183155	-85.661814	0.0	2.482843	144.007572	3G	0.000000	0.000000	0.000000

Figure IV.1 : échantillon des données constatées

IV.2.3 Statistiques descriptives

Nous avons ensuite calculé les corrélations entre les différentes variables numériques et la latence. Cette analyse révèle des relations très parlantes.

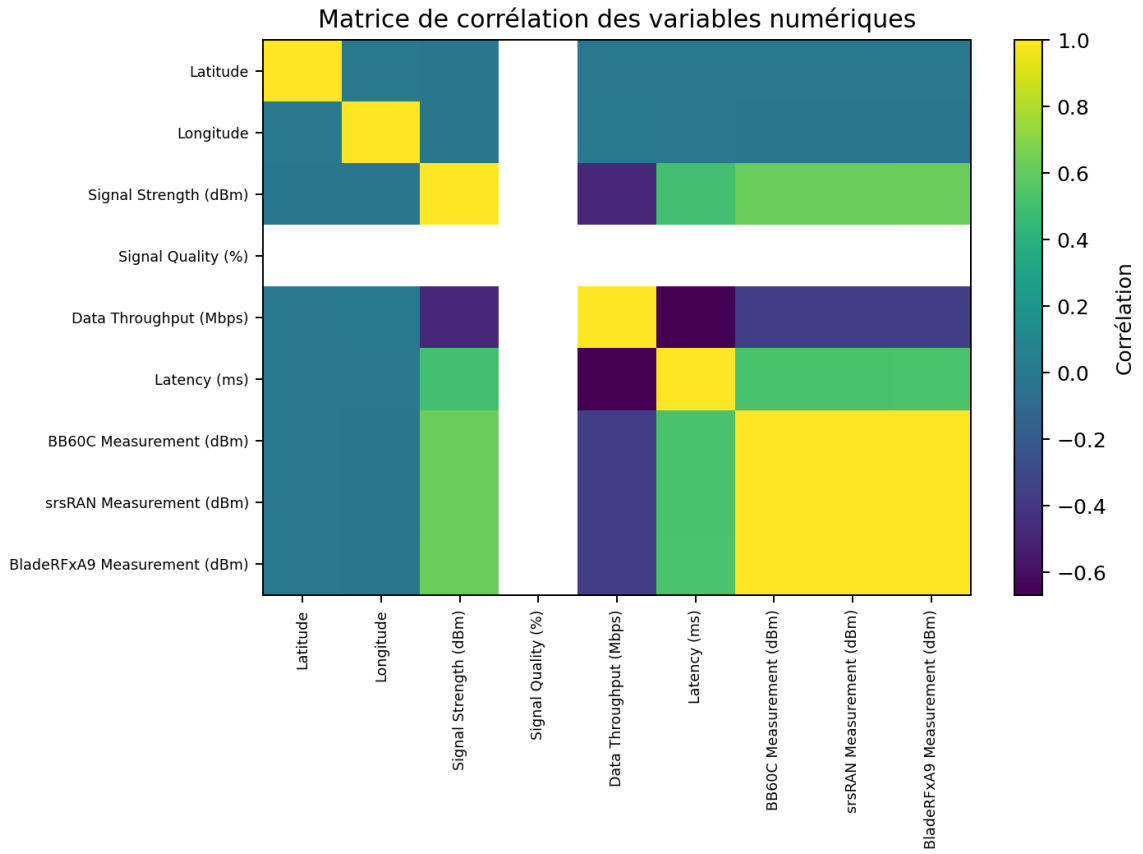


Figure IV.2 : Matrice de corrélation des variables numériques

La variable Data Throughput (le débit) présente la corrélation la plus forte avec la latence : -0,668. Ce coefficient négatif indique que plus le débit est élevé, plus la latence est faible. C'est tout à fait logique : les réseaux les plus performants, comme la 5G, offrent à la fois des débits élevés et des latences réduites.

Les trois mesures issues des capteurs (BB60C, srsRAN et BladeRFxA9) présentent quant à elles des corrélations modérément positives, d'environ 0,52 chacune. Le fait que les trois coefficients soient très proches suggère que ces trois capteurs capturent une information partiellement redondante mais néanmoins pertinente pour la prédiction de la latence.

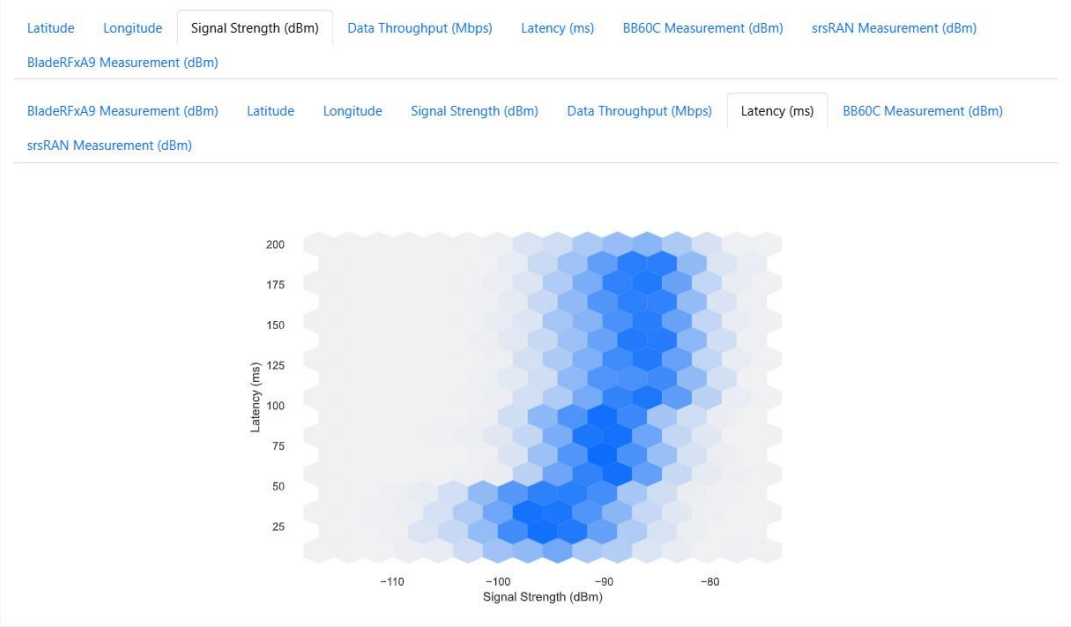


Figure IV.3 : Distribution de la latence selon signal strength

IV.3 Prétraitement des données

IV.3.1 Suppression des colonnes inutiles

Avant toute modélisation, nous avons procédé à un nettoyage méthodique des données. La première étape a consisté à supprimer la colonne Signal Quality (%), dont nous avons déjà expliqué qu'elle était totalement inutilisable en raison de l'absence totale de variance.

Nous avons également écarté les colonnes géographiques : Locality, Latitude et Longitude. Pourquoi ? Parce que notre objectif est de construire un modèle qui se base sur les propriétés physiques du signal, et non sur la localisation précise. Si nous avions conservé ces informations, le modèle risquerait de "mémoriser" des particularités locales et serait moins performant lorsqu'il serait appliqué à d'autres régions. Nous privilégions donc la généralisation.

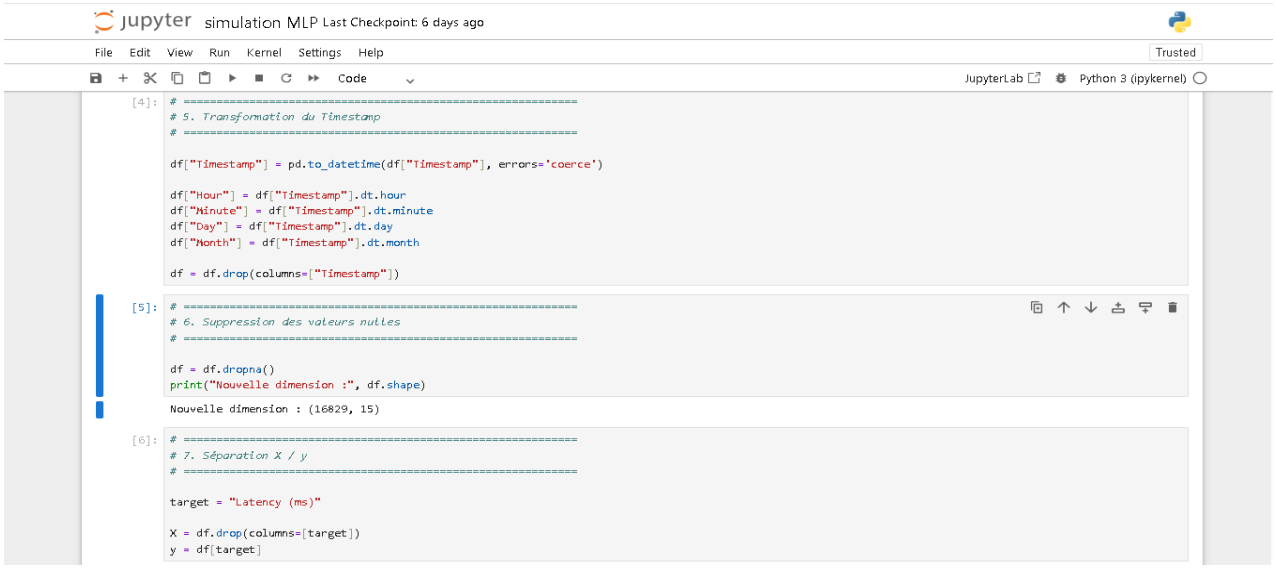


Figure IV.4 : suppression des colonnes inutiles et les valeurs manquantes

IV.3.2 Encodage one-hot (Type de Réseau)

La colonne Network Type est une variable catégorielle. Pour qu'elle puisse être exploitée par notre réseau de neurones, nous l'avons transformée selon la méthode dite one-hot encoding. Concrètement, cela consiste à créer quatre nouvelles colonnes binaires :

- net_3G (qui vaut 1 si le réseau est 3G, 0 sinon)
- net_4G (1 si 4G, 0 sinon)
- net_5G (1 si 5G, 0 sinon)
- net_LTE (1 si LTE, 0 sinon)

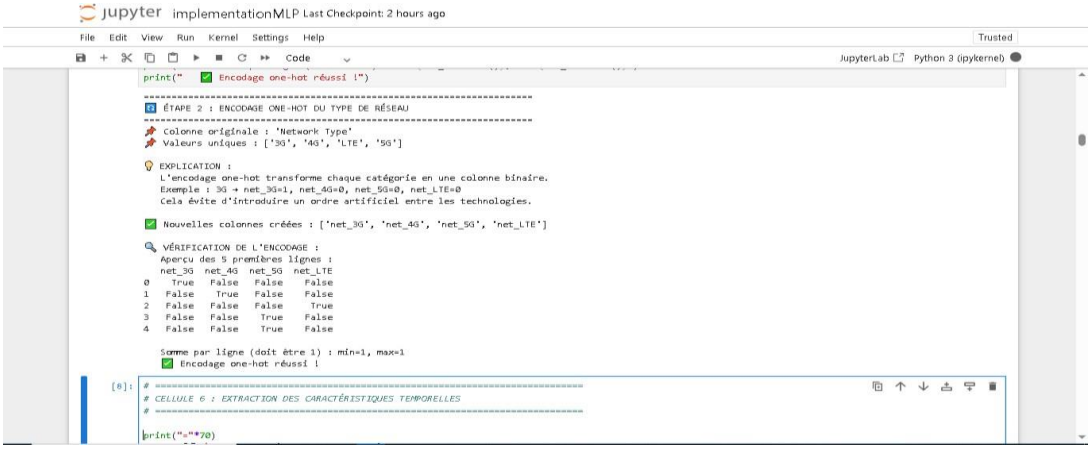


Figure IV.5 : Encodage one-hot (type de réseau)

Cette transformation est préférable à un simple codage numérique (par exemple 1,2,3,4) qui aurait artificiellement introduit une notion d'ordre entre les technologies, ce qui n'a aucun sens dans la réalité.

IV.3.3 Normalisation (Standard Scaler)

Les caractéristiques numériques du jeu de données n'évoluent pas toutes dans les mêmes plages de valeurs. Par exemple, le débit peut atteindre plusieurs centaines de Mbps alors que la puissance du signal s'exprime en dBm avec des valeurs négatives autour de -80 à -100. Si l'on alimentait le réseau de neurones avec des données non normalisées, les variables avec de grandes amplitudes domineraient artificiellement l'apprentissage.

Pour éviter ce problème, nous avons appliqué la méthode Standard Scaler de la bibliothèque Scikit-learn [2]. Cette technique centre chaque variable autour de sa moyenne (soustraction de la moyenne) et réduit son écart-type à l'unité (division par l'écart-type). Une règle d'or a été scrupuleusement respectée : le scaler n'a été ajusté que sur l'ensemble d'entraînement, puis appliqué aux ensembles de validation et de test. Cela garantit l'absence de toute fuite d'information.

IV.3.4 Division Train / Validation / Test (70/15/15)

Nous avons ensuite fractionné notre jeu de données en trois sous-ensembles distincts :

- Ensemble d'entraînement (70%) : 11 786 échantillons, utilisés pour apprendre les poids du réseau.
- Ensemble de validation (15%) : 2 518 échantillons, utilisés pour ajuster les hyperparamètres et surveiller l'apparition du surapprentissage.
- Ensemble de test (15%) : 2 525 échantillons, gardés précieusement de côté pour l'évaluation finale, objective et impartiale.

Cette répartition 70/15/15 est classique et équilibrée : elle offre suffisamment de données pour l'apprentissage tout en conservant des volumes respectables pour la validation et le test.

IV.4 Architecture du réseau de neurones

IV.4.1 Structure des couches

Le modèle que nous avons implémenté est un perceptron multicouches (MLP) de type séquentiel, construit avec l'API Keras de TensorFlow Il reçoit en entrée onze caractéristiques (celles que nous avons préparées) et produit en sortie une valeur scalaire qui correspond à la latence estimée, en millisecondes.

Le tableau ci-dessous détaille l'architecture couche par couche. Le nombre total de paramètres entraînables s'élève à 12 673, ce qui est modeste et garantit un entraînement rapide tout en offrant une capacité suffisante pour modéliser les relations entre les variables.

Model: "LatencyEstimator"

Layer (type)	Output Shape	Param #
dense_1 (Dense)	(None, 128)	1,536
bn_1 (BatchNormalization)	(None, 128)	512
relu_1 (Activation)	(None, 128)	0
dropout_1 (Dropout)	(None, 128)	0
dense_2 (Dense)	(None, 64)	8,256
bn_2 (BatchNormalization)	(None, 64)	256
relu_2 (Activation)	(None, 64)	0
dropout_2 (Dropout)	(None, 64)	0
dense_3 (Dense)	(None, 32)	2,080
output (Dense)	(None, 1)	33

Total params: 12,673 (49.50 KB)
Trainable params: 12,289 (48.00 KB)
Non-trainable params: 384 (1.50 KB)

Tableau IV.2 : Architecture couche par couche

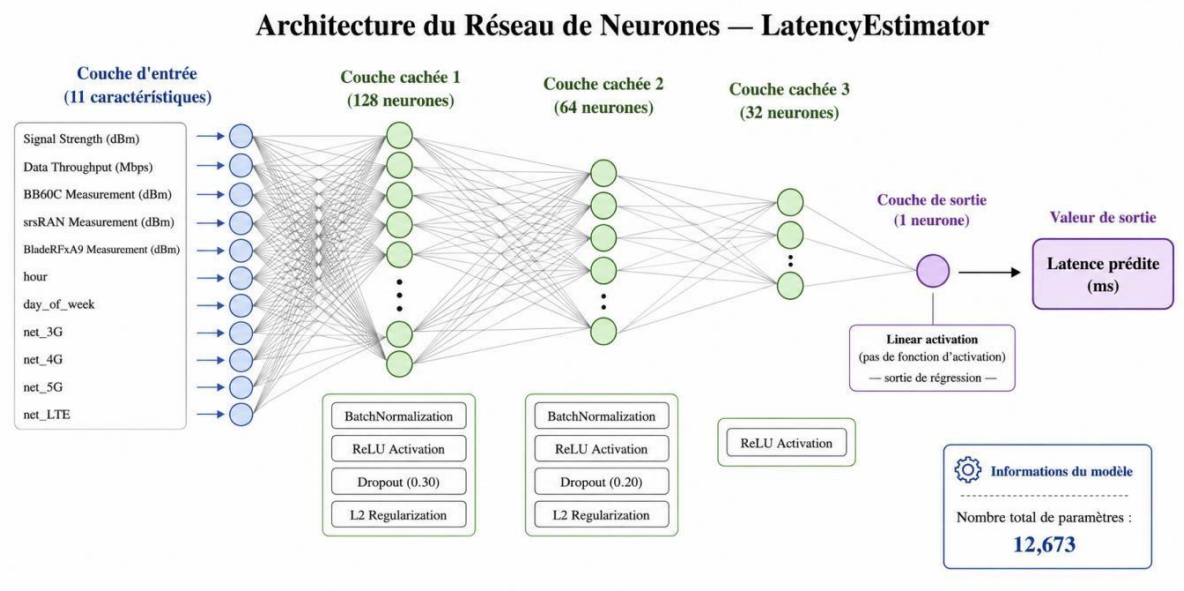


***Figure IV.6 : Architecture du* modèle**

IV.4.2 Régularisation et activation

Pour éviter que le modèle ne "mémorise" simplement les données d'entraînement sans être capable de généraliser – phénomène connu sous le nom de sur apprentissage – nous avons intégré plusieurs mécanismes de régularisation.

D'abord, nous avons placé des couches de BatchNormalization après les deux premières couches denses. Cette technique normalise les activations intermédiaires, ce qui accélère la convergence et stabilise l'apprentissage.

Ensuite, nous avons appliqué une régularisation L2 (coefficient 1e-4) sur les poids des deux premières couches denses. Cette pénalité mathématique encourage les poids à rester petits, ce qui favorise des solutions plus simples et plus généralisables.

Nous avons également inséré des couches Dropout après les deux premiers blocs denses, avec des taux respectifs de 0,3 et 0,2. Pendant l'entraînement, ces couches désactivent aléatoirement une fraction des neurones à chaque itération. Cela contraint le réseau à développer des représentations redondantes et empêche une dépendance excessive à certains neurones particuliers.

Enfin, la fonction d'activation choisie pour les couches cachées est ReLU (Rectified Linear Unit), appréciée pour sa simplicité et son efficacité. La couche de sortie, quant à elle, n'a

pas de fonction d'activation, ce qui permet au modèle de produire n'importe quelle valeur réelle – ce dont nous avons besoin pour une tâche de régression.

IV.4.3 Compilation — optimiseur Adam

Pour l'entraînement, nous avons compilé notre modèle avec l'optimiseur Adam en utilisant un taux d'apprentissage de 0,001. L'optimiseur Adam est un choix très répandu car il combine les avantages d'AdaGrad (qui adapte le taux d'apprentissage à chaque paramètre) et de RMSProp (qui traite le problème des gradients qui s'évaporent ou explosent).

La fonction de perte que nous minimisons est l'erreur quadratique moyenne (MSE) , qui est la plus courante pour les problèmes de régression. En complément, nous suivons deux métriques supplémentaires : l'erreur absolue moyenne (MAE) et la racine de l'erreur quadratique moyenne (RMSE).

IV.5 Entraînement du modèle

IV.5.1 Paramètres d'entraînement

L'entraînement a été lancé sur l'ensemble d'entraînement de 11 786 échantillons, avec les réglages suivants :

- Taille de lot (batch size) : 32 échantillons par mise à jour des poids
- Nombre maximal d'époques : 100 (nous nous arrêterons plus tôt si nécessaire)
- Ensemble de validation : 2 518 échantillons pour le suivi en cours d'apprentissage

IV.5.2 Callbacks utilisés

Pour garder le contrôle sur le processus d'apprentissage et éviter de gaspiller des ressources, nous avons configuré quatre callbacks :

EarlyStopping : Ce mécanisme surveille la perte de validation. Si celle-ci ne s'améliore pas pendant 10 époques consécutives, l'entraînement s'arrête automatiquement. De plus, il restaure les poids du meilleur modèle observé. C'est une manière élégante d'éviter le sur apprentissage sans avoir à deviner le nombre idéal d'époques à l'avance.

ReduceLROnPlateau : Parfois, le taux d'apprentissage initial devient trop grand ou trop petit au fil de l'entraînement. Ce callback réduit le taux d'apprentissage d'un facteur 0,5 lorsque la perte de validation stagne pendant 5 époques, avec une borne inférieure à 1e-6.

ModelCheckpoint : Ce callback sauvegarde automatiquement le meilleur modèle (celui avec la perte de validation la plus faible) dans un fichier nommé best_model.keras.

TensorBoard : Enfin, nous avons journalisé toutes les métriques pour pouvoir les visualiser et les analyser a posteriori.

L'entraînement s'est effectivement arrêté à la 22e époque, déclenché par le critère d'arrêt précoce. Le meilleur modèle a été identifié à la 12e époque, avec des performances de validation de 18,41 ms en MAE et 22,85 ms en RMSE.

IV.5.3 Courbes d'entraînement

Les courbes d'évolution des métriques au fil des époques nous permettent de visualiser la progression de l'apprentissage.

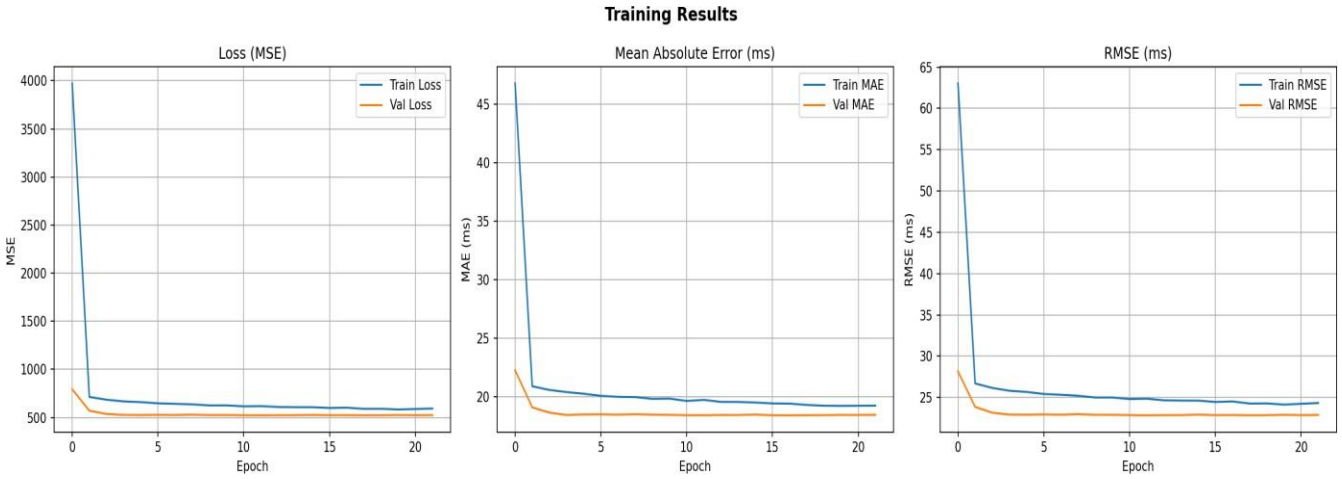


Figure IV.7 : Courbes d'entraînement

- Évolution de la perte MSE sur les ensembles d'entraînement et de validation*
- Évolution de l'erreur absolue moyenne (MAE) en millisecondes*
- Évolution de la racine de l'erreur quadratique moyenne (RMSE) en millisecondes*

Sur ces courbes on observe une décroissance rapide des métriques dès les premières époques, suivie d'une stabilisation. L'écart entre les courbes d'entraînement et de validation reste faible, signe que le modèle ne souffre pas d'un sur apprentissage sévère.

IV.6 Résultats et évaluation

IV.6.1 Résultats sur l'ensemble de test

Après avoir entraîné et validé notre modèle, nous l'avons enfin évalué sur l'ensemble de test, celui que nous avions précieusement mis de côté et que le modèle n'a jamais vu. C'est cette évaluation qui donne la véritable mesure de la performance généralisable.

Voici les chiffres obtenus sur les 2 525 échantillons de test :

Métrique	Valeur
MAE (Erreur Absolue Moyenne)	**18,00 ms**
RMSE (Racine de l'Erreur Quadratique Moyenne)	**22,30 ms**
R^2 (Coefficient de détermination)	**0,843**

Tableau IV.3 : Résultats sur l'ensemble de test

Interprétons ces résultats. Un R^2 de 0,843 signifie que notre modèle explique 84,3% de la variance de la latence à partir des onze caractéristiques d'entrée. C'est une performance solide pour des données réelles de réseau, où le bruit est inévitable.

L'erreur absolue moyenne de 18 ms signifie qu'en moyenne, notre prédiction s'écarte de moins de 18 millisecondes de la valeur réelle. Si l'on considère que la latence varie entre 10 et 200 ms, cette erreur représente environ 9% de l'amplitude totale – un niveau d'erreur très acceptable pour de nombreuses applications de planification ou de gestion de la qualité de service.

IV.6.2 Comparaison des modèles (NN vs LR vs RF)

Pour évaluer la pertinence de notre réseau de neurones, nous avons entraîné deux modèles de référence sur les exactement mêmes données (mêmes partitions, mêmes caractéristiques) :

- Une régression linéaire (Linear Regression)
- Une forêt aléatoire (Random Forest) avec 100 arbres de décision

Ces deux modèles ont été implémentés avec Scikit-learn. Le tableau ci-dessous résume les performances comparées sur l'ensemble de test.

Modèle	MAE (ms)	RMSE (ms)	R^2
Réseau de neurones	18,00	22,30	0,843
Régression linéaire	17,94	22,22	0,844
Forêt aléatoire	18,47	23,10	0,832

Tableau IV.4 : comparaison des modèles d'apprentissage

Les résultats sont frappants : les trois modèles affichent des performances très proches. La régression linéaire devance notre réseau de neurones de seulement 0,06 ms en MAE et 0,001 en R^2. L'écart est négligeable, bien inférieur à la précision de mesure des outils de diagnostic réseau classiques.

La forêt aléatoire, quant à elle, est légèrement moins performante avec une MAE de 18,47 ms.

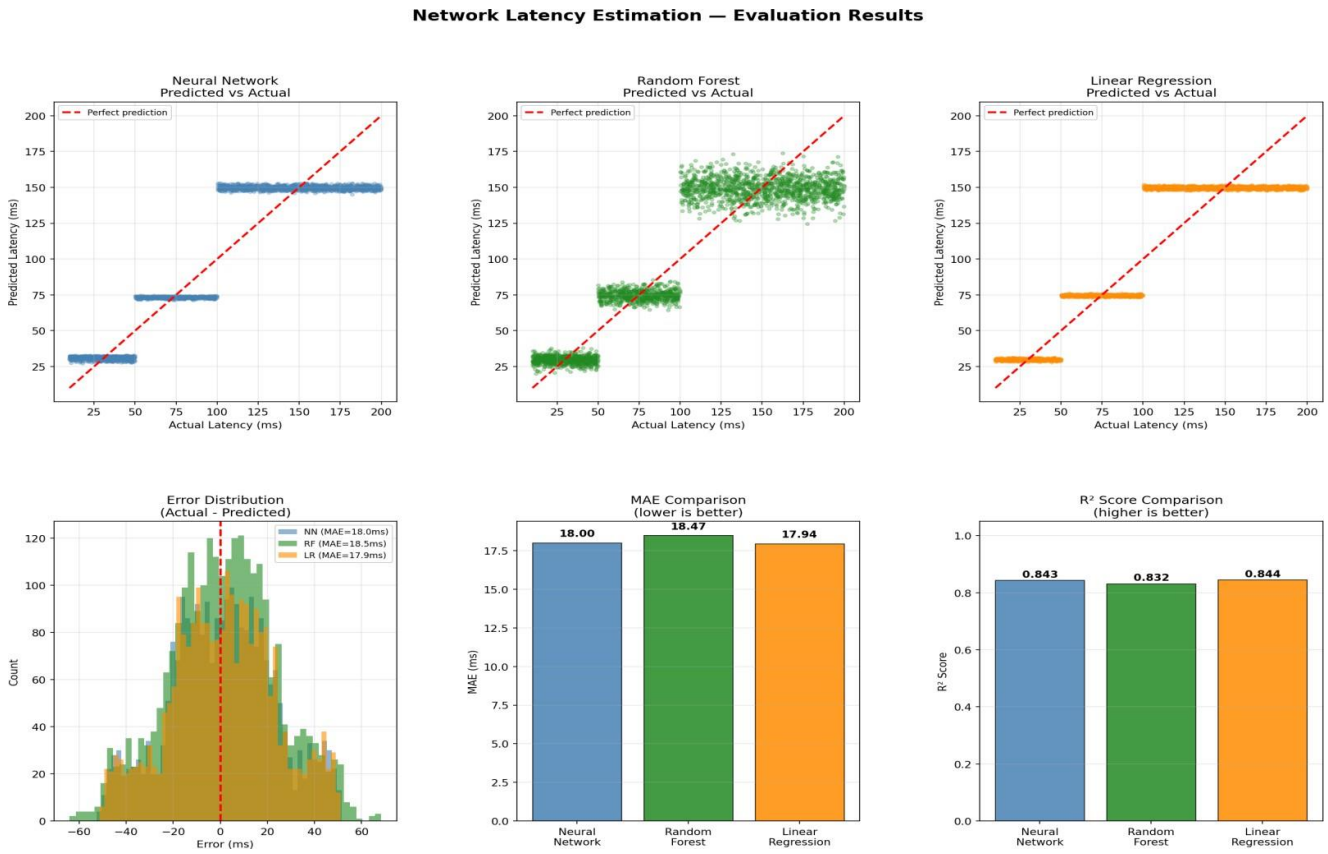


Figure IV.8 : Comparaison des prédictions (NN, Régression Linéaire, Forêt Aléatoire) : valeurs prédites vs réelles*

IV.6.3 Tableau prédictions vs valeurs réelles

Pour donner un aperçu plus concret et plus visuel de la qualité des prédictions, nous avons extrait quelques exemples. Le tableau ci-dessous présente, pour chaque type de réseau, cinq prédictions tirées de l'ensemble de test.

Type réseau	Latence réelle (ms)	Latence prédite par NN (ms)	Erreur (ms)	Erreur relative
3G	168,53	149,52	-19,01	11,3%
3G	177,70	148,92	-28,78	16,2%
3G	143,65	149,69	+6,04	4,2%
3G	176,05	148,18	-27,87	15,8%
3G	149,42	150,02	+0,60	0,4%
4G	72,10	73,67	+1,57	2,2%
4G	69,44	74,31	+4,87	7,0%
4G	62,27	72,54	+10,27	16,5%
4G	98,65	72,87	-25,78	26,1%
4G	77,95	72,50	-5,45	7,0%
5G	23,14	30,37	+7,23	31,2%
5G	20,33	31,00	+10,67	52,5%
5G	31,26	32,42	+1,16	3,7%
5G	35,17	31,30	-3,87	11,0%

Type réseau	Latence réelle (ms)	Latence prédite par NN (ms)	Erreur (ms)	Erreur relative
5G	42,16	31,63	-10,53	25,0%
LTE	119,10	148,91	+29,81	25,0%
LTE	148,34	149,17	+0,83	0,6%
LTE	100,38	152,10	+51,72	51,5%
LTE	152,25	149,62	-2,63	1,7%
LTE	103,19	151,49	+48,30	46,8%

Tableau IV.5 : Prédictions vs valeurs réelles

Que constate-t-on ? Les prédictions sont excellentes pour la 5G et la 4G dans la plupart des cas, avec des erreurs faibles. On observe cependant quelques erreurs ponctuelles plus importantes, notamment sur le LTE (parfois plus de 50 ms d'écart) et sur certaines mesures 4G. Ces écarts méritent analyse, mais ils restent marginaux au regard du volume total de données.

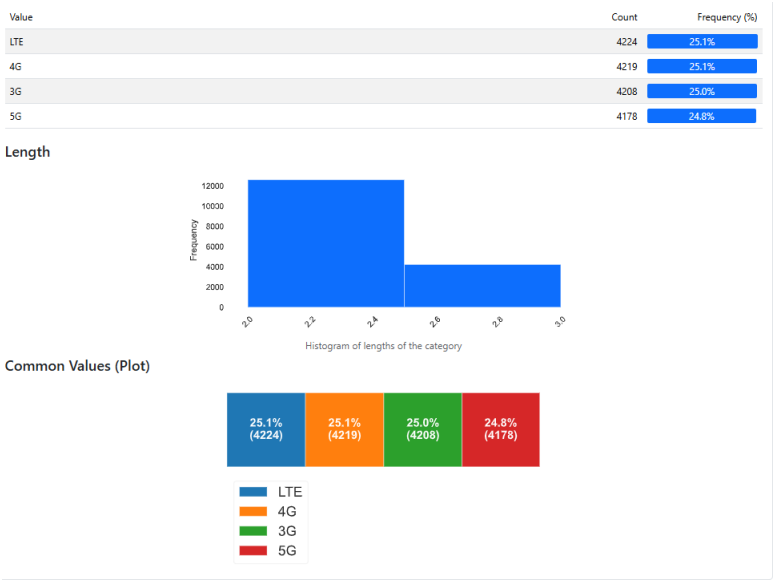


Figure IV.9 : Équilibre des classes des types de réseaux

La figure Equilibre des classe des types réseaux confirme par ailleurs l'excellent équilibre des classes, ce qui nous rassure sur l'absence de biais lié à une surreprésentation.

IV.6.4 Discussion des résultats

La proximité quasi-parfaite des performances entre notre réseau de neurones et la simple régression linéaire est à la fois surprenante et instructive. Pourquoi un modèle aussi sophistiqué ne bat-il pas nettement un modèle aussi simple ?

L'explication la plus probable est que, sur ce jeu de données particulier, la relation entre les caractéristiques et la latence est essentiellement linéaire. La forte corrélation entre le débit et la latence (-0,668) semble être le principal moteur de la prédiction. Cette relation linéaire est parfaitement capturée par une régression linéaire, qui n'a d'ailleurs besoin d'aucun réglage complexe.

Mais alors, pourquoi avons-nous choisi un réseau de neurones ?

Trois raisons principales justifient ce choix :

1. La capacité à modéliser la non-linéarité – Même si ce jeu de données ne présente pas de fortes non-linéarités, d'autres jeux de données en contiendront. En choisissant un réseau de neurones, nous nous dotons d'un outil capable de s'adapter à des relations plus complexes sans avoir à changer d'architecture.

2. La flexibilité – L'architecture neuronale est beaucoup plus facile à enrichir. Si demain nous souhaitons ajouter de nouvelles variables (par exemple, la charge du réseau, le nombre d'utilisateurs connectés, ou des indicateurs de qualité de service plus fins), nous pourrons le faire sans révolutionner notre approche.
3. La robustesse – Les mécanismes de régularisation intégrés (BatchNormalization, Dropout, L2) rendent le réseau de neurones naturellement plus robuste au bruit de mesure et aux variations des conditions réelles d'exploitation.

En définitive, un écart de 0,06 ms en MAE est ridiculement faible. Dans la pratique opérationnelle, un ingénieur réseau ne fera pas la différence entre les deux modèles. Nous avons donc choisi le réseau de neurones non pas pour un gain de performance immédiat, mais pour ses potentialités d'évolution et sa capacité à s'adapter à des données futures plus complexes.

IV.7 Interface utilisateur (Streamlit)

Un modèle de Machine Learning, aussi performant soit-il, n'a de valeur que s'il est accessible et utilisable. C'est pourquoi nous avons développé une application web interactive avec le Framework Streamlit, implémentée dans le fichier app.py.

L'interface est conçue pour être intuitive, même pour un utilisateur non spécialiste. Voici comment elle fonctionne.

L'utilisateur commence par sélectionner le type de réseau dans un menu déroulant (3G, 4G, LTE ou 5G). Il renseigne ensuite cinq mesures physiques :

- La puissance du signal (Signal Strength, en dBm)
- Le débit (Data Throughput, en Mbps)
- Les trois mesures issues des capteurs (BB60C, srsRAN, BladeRFxA9, toutes en dBm)

Deux champs optionnels permettent également d'indiquer l'heure de la journée et le jour de la semaine, si ces informations sont disponibles – elles aident à capturer les effets de congestion.

Lorsque l'utilisateur clique sur le bouton "Predict Latence", l'application déclenche tout le pipeline en arrière-plan : les entrées sont formatées dans un Data Frame Pandas, la normalisation est appliquée à l'aide du scaler que nous avons sauvegardé pendant l'entraînement, et enfin le modèle chargé produit une prédiction.

Le résultat s'affiche alors clairement :

- La latence estimée en millisecondes
- Un label qualitatif : Excellent (moins de 40 ms), Good (moins de 80 ms), Fair (moins de 130 ms) ou Poor (130 ms ou plus)
- Une barre de progression visuelle qui positionne la latence estimée sur une échelle de 0 à 200 ms

Dans la barre latérale, l'application rappelle en permanence les performances du modèle (MAE = 18 ms, RMSE = 22,30 ms, R^2 = 0,843). Un bouton supplémentaire "Compare All Network Types" permet de générer en un seul clic les quatre prédictions pour les quatre technologies avec les mêmes paramètres de signal – très pratique pour évaluer le gain potentiel d'un passage à la 5G par exemple.

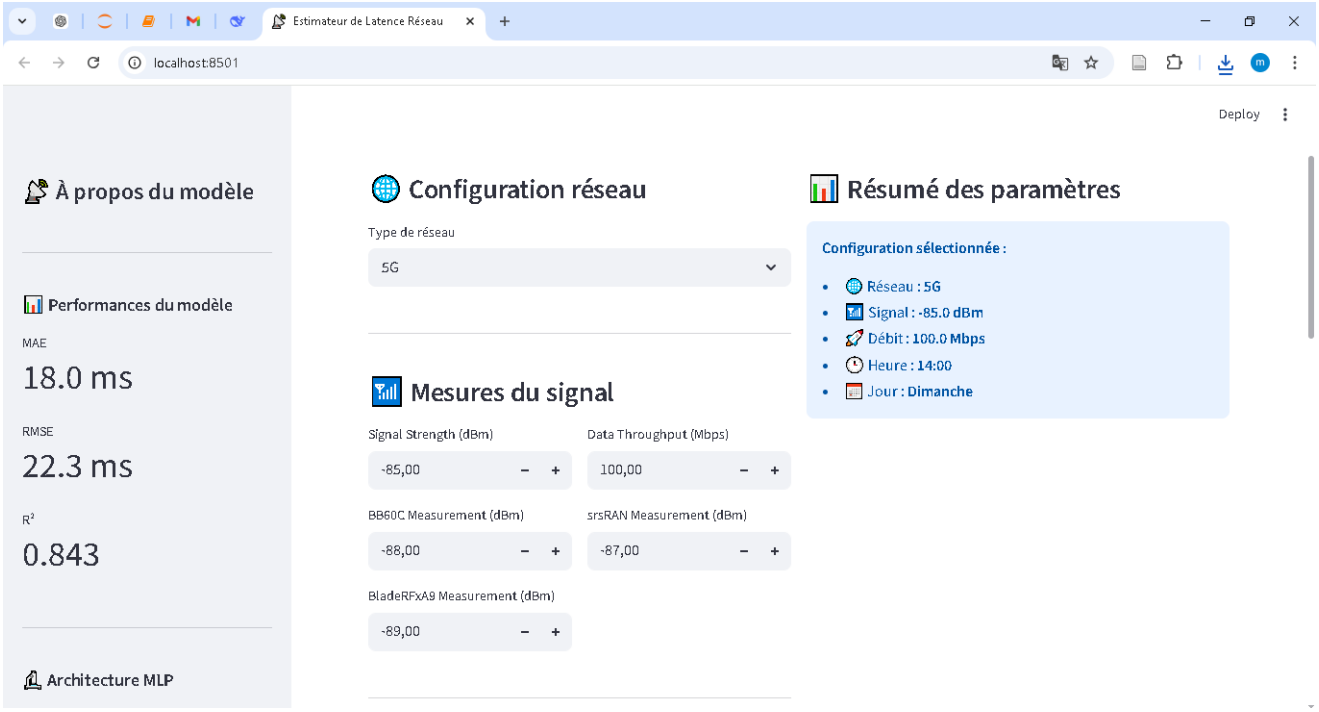


Figure IV.10 : Interface Streamlit — vue principale avec sélection du type de réseau

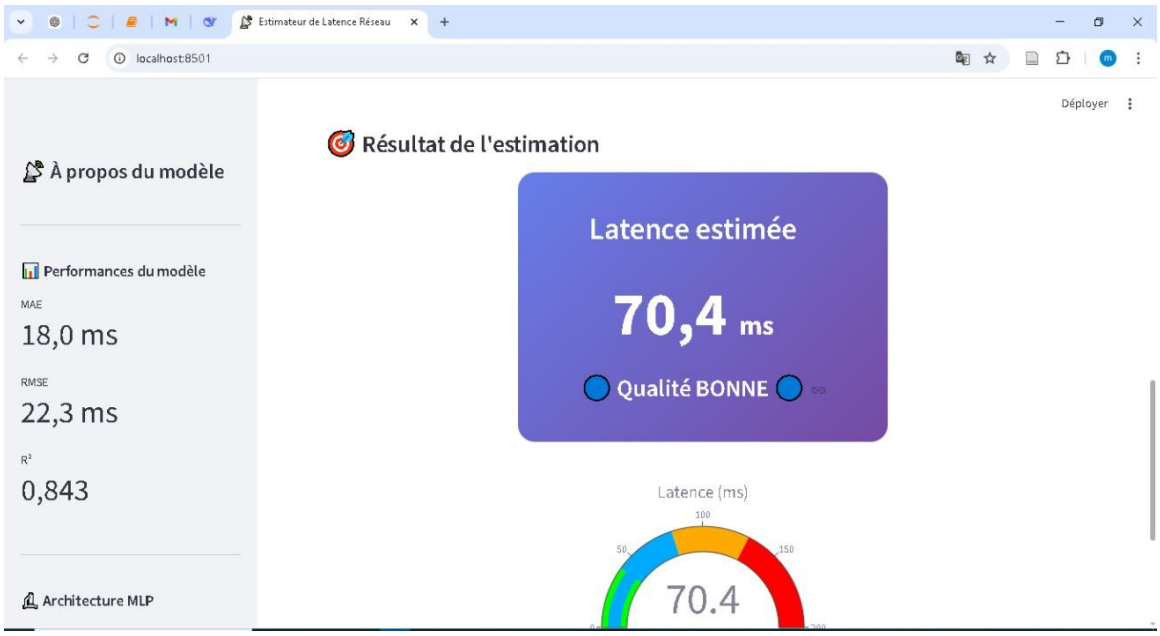


Figure IV.11 : Interface Streamlit — vue estimation de latence (bonne)

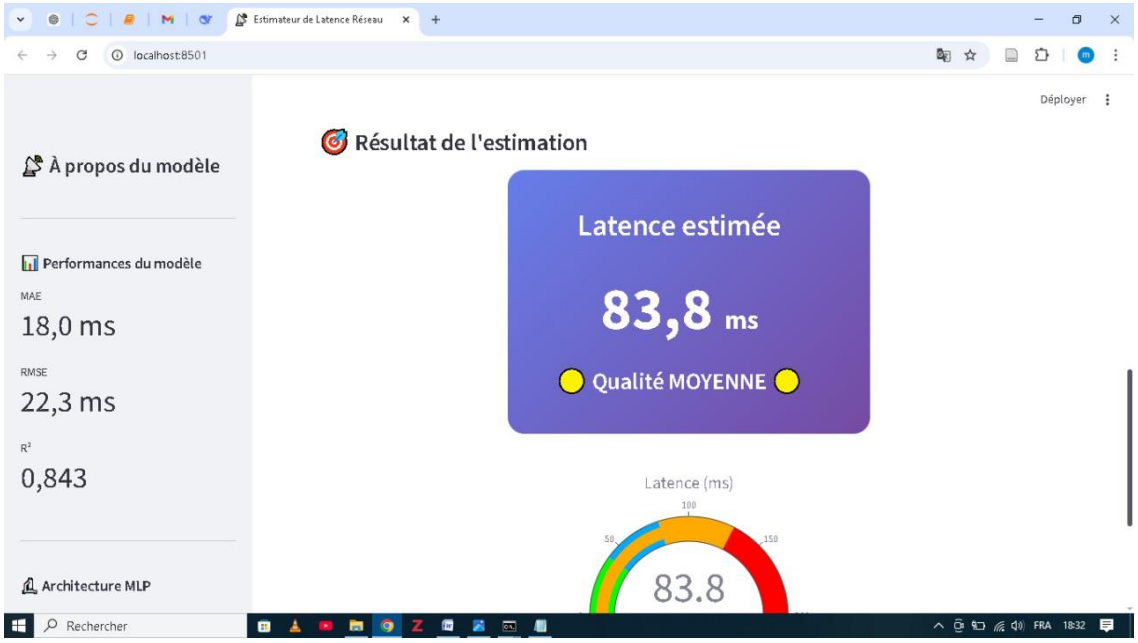


Figure IV.12 : Interface Streamlit : vue Résultat de l'estimation(moyenne)

IV.8 Conclusion

Ce chapitre nous a permis de mettre en lumière l'ensemble du processus, depuis les données brutes collectées sur le terrain jusqu'à l'interface utilisateur opérationnelle.

Nous avons exploité un jeu de données réel de 16 829 mesures de signal, collectées dans l'État du Bihâr en Inde. Après un nettoyage minutieux (suppression de colonnes inutilisables, encodage des variables catégorielles, extraction de caractéristiques temporelles et normalisation), nous avons conçu un réseau de neurones de 12 673 paramètres, régularisé par Batch Normalization, Dropout et pénalité L2.

L'entraînement, contrôlé par des callbacks intelligents (arrêt précoce, réduction dynamique du taux d'apprentissage), s'est arrêté à la 22e époque. Le modèle atteint sur l'ensemble de test une erreur absolue moyenne de 18,00 ms et un coefficient de détermination R^2 de 0,843, ce qui signifie qu'il explique 84,3% de la variance de la latence.

La comparaison avec une régression linéaire et une forêt aléatoire a révélé des performances très proches, suggérant une relation globalement linéaire entre les caractéristiques du signal et la latence sur ce jeu de données. L'écart infime (0,06 ms) nous conforte dans l'idée que le réseau de neurones n'apporte pas de gain immédiat, mais sa flexibilité et sa capacité à modéliser des relations non linéaires futures constituent des atouts décisifs pour la pérennité du système.

L'interface Streamlit, simple et intuitive, rend le modèle accessible à tous. Elle transforme un résultat de laboratoire en un outil concret, capable d'estimer la latence en temps réel à partir de quelques mesures de signal. Cette démonstration prouve la faisabilité d'un déploiement opérationnel pour la gestion de la qualité de service dans les réseaux cellulaires.

Enfin, plusieurs perspectives d'amélioration s'offrent à nous : enrichir le jeu de données avec des mesures provenant d'autres régions géographiques, intégrer des variables contextuelles supplémentaires (charge du réseau, nombre d'utilisateurs actifs), ou encore expérimenter des architectures plus avancées comme les réseaux de neurones récurrents pour modéliser les séries temporelles de latence.

Conclusion générale

Au terme de ce travail de recherche consacré à l'estimation de la latence dans les réseaux cellulaires à l'aide des réseaux de neurones artificiels, plusieurs contributions scientifiques et techniques peuvent être mises en évidence.

Dans un premier temps, ce mémoire a permis de mettre en lumière l'évolution rapide des réseaux cellulaires, depuis les premières générations jusqu'aux architectures avancées de la 5G. Cette évolution s'accompagne d'exigences accrues en termes de performance, notamment en matière de latence, devenue un paramètre critique pour garantir la qualité de service (QoS) et la qualité d'expérience utilisateur (QoE), en particulier pour les applications temps réel.

Dans un second temps, l'étude des approches existantes a montré les limites des modèles analytiques traditionnels face à la complexité et à la variabilité des environnements cellulaires modernes. Cette constatation justifie pleinement le recours aux techniques d'intelligence artificielle, et plus particulièrement aux réseaux de neurones, capables de modéliser des relations non linéaires complexes à partir de données réelles.

Sur le plan méthodologique, une approche complète a été proposée, intégrant l'ensemble des étapes du processus d'apprentissage automatique, depuis l'exploration et le prétraitement des données jusqu'à la conception, l'entraînement et l'évaluation du modèle. Le choix d'un réseau de neurones de type MLP s'est avéré pertinent pour la modélisation du problème de régression étudié.

Les résultats expérimentaux obtenus démontrent l'efficacité de l'approche proposée. Le modèle développé est capable d'estimer la latence avec une précision satisfaisante, atteignant un coefficient de détermination élevé ($R^2 \approx 0,84$) et une erreur moyenne acceptable. La comparaison avec d'autres modèles tels que la régression linéaire et les forêts aléatoires a permis de confirmer la robustesse globale de la solution, tout en mettant en évidence que certaines relations dans les données restent partiellement linéaires.

Cependant, ce travail présente certaines limites. En effet, la qualité des prédictions dépend fortement de la richesse du jeu de données utilisé. L'absence de certaines variables importantes, telles que la charge du réseau, la congestion, la distance à la station de base ou les événements de handover, peut limiter la capacité du modèle à capturer l'ensemble des dynamiques réelles du réseau.

Dans cette perspective, plusieurs pistes d'amélioration peuvent être envisagées. Il serait intéressant d'intégrer des données plus riches et plus diversifiées, d'explorer des architectures de réseaux de neurones plus avancées (telles que les LSTM ou les modèles hybrides), et d'optimiser davantage les hyper paramètres du modèle. De plus, l'extension du système vers une intégration en temps réel dans des environnements opérationnels constitue une perspective prometteuse pour l'optimisation intelligente des réseaux cellulaires.

En conclusion, ce travail confirme que l'utilisation des techniques de Machine Learning, et en particulier des réseaux de neurones, représente une solution efficace et prometteuse pour l'estimation de la latence dans les réseaux cellulaires. Il s'inscrit ainsi dans une dynamique de transformation des réseaux vers des systèmes intelligents, adaptatifs et autonomes, capables de répondre aux exigences des technologies futures.

Bibliographie

[1] Evolution de la 2G… à la 5G," Bouygues Telecom. Accessed: Jun. 19, 2026. [Online]. Available: https://mag.bouyguestelecom.fr/technologie/reseau-mobile-4g-5g/evolution-des-usages-mobiles-de-la-2g-a-la-5g.

[2] S. Tarapiah, K. Aziz, et S. Atalla, « Third Generation (3G) Mobile Network Planning Process and Methodology - Case Study », Int. J. Comput. Appl., vol. 143, no 12, p. 1-6, juin 2016.

[3] N. Vulic, S. Heemstra De Groot, et I. Niemegeers, « A Framework for Integration of Different WLAN Technologies at UMTS Radio Access Level », in Fourth Annual IEEE International.

[4] Conference on Pervasive Computing and Communications Workshops (PERCOMW'06), Pisa, Italy: IEEE, 2006, p. 436-441.

[5] M. S. Iqbal et al., « Mobile communication (2G, 3G & 4G) and future interest of 5G in Pakistan: a review », Indones. J. Electr. Eng. Comput. Sci., vol. 22, no 2, p. 1061, mai 2021.

[6] A. El-Shorbagy, « 5G Technology and the Future of Architecture », Procedia Comput. Sci., vol. 182, p. 121-131, 2021.

[7] LAGRANGE, Xavier, GODLEWSKI, Philippe et TABBANE, Sami. Réseaux GSM : des principes à la norme. 4e édition. Paris : Éditions Hermès Science Publications, 2000.

[8] S. Z. Asif, 5G mobile communications: concepts and technologies, First edition. Boca Raton, FL: CRC PressTaylor and Francis Group, 2019.

[9] Z. Wen et M. Rabinovich, « Dynamic landmark triangles: A simple and efficient mechanism for inter-host latency estimation », Comput. Netw., vol. 55, no 8, p. 1864-1879, juin 2011.

[10] I. Semenkov, N. Fedosov, I. Makarov, et A. Ossadtchi, « Real-time low latency estimation of brain rhythms with deep neural networks », J. Neural Eng., vol. 20, no 5, p. 056008, oct. 2023

[11] K. Matsuda, « Advanced Information Technology for Production Sites with Private 5G », Commun. Ind., vol. 4, no 0/5.0, p. 54, 2024.

[12] Y. Deng et S. Manoharan, « A review of network latency optimization techniques », in 2013 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM, août 2013, p. 20-25.

[13] D. L. Amenyo Fiase, K. Opoku Attah, S. Lartey, D. Adjin, et M. M.- Lisa Allotey, « Evaluating the Impact of Network Latency on user Experience in Telecommunication Industries in Ghana », Int. J. Innov. Sci. Res. Technol. IJISRT, p. 2861-2869, sept. 2024.

[14] S. Zinno, A. Navarro, S. Rotbei, N. Pasquino, A. Botta, et G. Ventre, « A Lightweight Learning Approach for Latency Prediction in 5G and Beyond », in 2025 21st International Conference on Network and Service Management (CNSM, 2025, p. 1-6.

[15] D. Fadhil et R. Oliveira, « Qualitative Prediction of End-to-End Delay in 5G Networks », IEEE Access, vol. 13, p. 179406-179418, 2025, doi: 10.1109/ACCESS.2025.3622329.

[16] S. Mostafavi, G. P. Sharma, A. Traboulsi, et J. Gross, « Probabilistic Delay Forecasting in 5G Using Recurrent and Attention-Based Architectures », 2025.

[17] J. B, K. Aggarwal, K. Malarvizhi, S. Z. Beevi, M. Malathy, et D. Suganthi, « Machine Learning Algorithms for Delay Prediction in IoT and Tactile Internet », in 2024 International Conference on Communication, Computing and Energy Efficient Technologies (I3CEET), Gautam Buddha Nagar, India: IEEE, sept. 2024, p. 1575-1579.

[18] A. M. Alwakeel, A. Ghaleb, and S. A. Alqahtani, *"A Deep Learning Framework for End-to-End Latency Prediction in 5G-Advanced Networks,"* IEEE Trans. Netw. Serv. Manag**.**, vol. 21, no. 2, pp. 1450–1465, Apr. 2024.

[19] Y. Cai, W. Li, X. Meng, W. Zheng, C. Chen, et Z. Liang, « Adaptive contrastive learning based network latency prediction in 5G URLLC scenarios », Comput. Netw., vol. 240, p. 110185, 2024.

[20] M. Vakalopoulou al, "Deep Learning: Basics and Convolutional Neural Networks," in

Handbook of Medical Image Computing and Computer Assisted Intervention, K. K. Zhou, H. Greenspan, and D. Shen, Eds., Cambridge, MA, USA: Academic Press, 2020, ch. 5, pp. 105–126.
[21] A. Havolli et M. Fetaji, « Performance evaluation of Machine Learning and Learning models for 5G resource allocation », J. Comput. Sci. Inform., vol. 12, no 3, 2025.
[22] S. J. Russell, P. Norvig, et E. Davis, Artificial intelligence: a modern approach, 3rd ed. in Prentice Hall series in artificial intelligence. Upper Saddle River: Prentice Hall, 2010.
[23] C. Touzet, « LES RESEAUX DE NEURONES ARTIFICIELS ».
[24] T. M. Mitchell, Machine learning, Nachdr. in McGraw-Hill series in Computer Science. New York: McGraw-Hill, 2013.
[25] I. Antonopoulos et al., « Artificial intelligence and machine learning approaches to energy demand-side response: A systematic review », Renew. Sustain. Energy Rev., vol. 130, p. 109899, sept. 2020.
[26] I. Goodfellow, Y. Bengio, et A. Courville, Deep learning. in Adaptive computation and machine learning. Cambridge, Mass: The MIT press, 2016.
[27] Y. Goldberg, « A Primer on Neural Network Models for Natural Language Processing », 2 octobre 2015, arXiv: arXiv:1510.00726.
[28] F. Rosenblatt, « The perceptron: A probabilistic model for information storage and organization in the brain », Psychol. Rev., vol. 65, no 6, p. 386-408, 1958.
[29] W. S. McCulloch et W. Pitts, « A logical calculus of the ideas immanent in nervous activity », Bull. Math. Biophys., vol. 5, no 4, p. 115-133, déc. 1943.
[30] V. Nair et G. E. Hinton, « Rectified Linear Units Improve Restricted Boltzmann Machines », in Proc. 27th Int. Conf. Machine Learning (ICML, Haifa, Israel, 2010, p. 807-814.
[31] L. Ven et J. Lederer, « Regularization and Reparameterization Avoid Vanishing Gradients in Sigmoid-Type Networks ». juin 2021.
[32] S. Haykin, Neural Networks and Learning Machines, 3rd ed. Upper Saddle River, NJ, USA: Prentice Hall, 2009, p. 47
[33] P. Blanc-Durand, "Réseaux de neurones convolutifs en médecine nucléaire : Applications à la segmentation automatique des tumeurs gliales et à la correction d'atténuation en TEP/IRM.," 2018.
[34] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. Cambridge, MA, USA: MIT Press, 2016, ch. 6
[35] G. Arulampalam et A. Bouzerdoum, « A generalized feedforward neural network architecture for classification and regression », Neural Netw., vol. 16, no 5-6, p. 561-568, 2003.
[36] X. Glorot et Y. Bengio, « Understanding the difficulty of training deep feedforward neural networks », in Proc. Int. Conf. Artificial Intelligence and Statistics (AISTATS, 2010, p. 249-256.
[37] S. Haykin, Neural Networks and Learning Machines. Hamilton: Third Edition, Pearson Education, Inc., McMaster University, 2009.
[38] A. Sabry, J. Benhra, and A. Bacha, Utilisation des réseaux de neurones pour le tuning des Algorithmes d'optimisation par colonies de fourmis : Application aux chaines logistiques. [S.l.] : [s.n.], Dec. 2, 2015.
[39] D. E. Rumelhart, G. E. Hinton, et R. J. Williams, « Learning representations by back-propagating errors », Nature, vol. 323, no 6088, p. 533-536, oct. 1986.
[40] R. Rajan and S. N. Kumar, "IoT based optical coherence tomography retinal images classification using OCT Deep Net2," Meas. Sens., vol. 25, p. 100652, 2022
[41] D. E. Rumelhart, G. E. Hinton, et R. J. Williams, « Learning representations by back-propagating errors », Nature, vol. 323, p. 533-536, 1986.
[42] D. P. Kingma et J. Ba, « Adam: A method for stochastic optimization », in Proc. 3rd Int. Conf. on Learning Representations (ICLR, 2015.
[43] S. Ruder, « An overview of gradient descent optimization algorithms », 2016.
[44] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, et R. Salakhutdinov, « Dropout: A

simple way to prevent neural networks from overfitting », J. Mach. Learn. Res., vol. 15, no 56, p. 1929-1958, 2014.
[45] S. Ioffe et C. Szegedy, « Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift », 2015.
[46] T. Hastie, R. Tibshirani, et J. Friedman, The Elements of Statistical Learning. in Springer Series in Statistics. New York, NY: Springer New York, 2009.
[47] P. Virtanen, R. Gommers, T. E. Oliphant, et al., "SciPy 1.0: fundamental algorithms for scientific computing in Python," Nat. Methods, vol. 17, no. 3, pp. 261–272, Mar. 2020.
[48] C. M. Bishop, Pattern recognition and machine learning. in Information science and statistics. New York: Springer, 2006.
[49] M. Abadi et al., « TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems », 2016.
[50] D. P. Kingma et J. Ba, « Adam: A Method for Stochastic Optimization », 2014.
[51] C. R. Harris et al., "NumPy v1.26.0," *Zenodo*, Oct. 2023,
[52] W. McKinney, « Data Structures for Statistical Computing in Python », présenté à Python in Science Conference, Austin, Texas, 2010, p. 56-61.
[53] S. Inc, « Streamlit — The fastest way to build data apps ». 2019. [En ligne]. Disponible sur: https://streamlit.io

Annexe : Code Python utilisé

le code complet sur le lien :

https://github.com/AimenDev/Network-Latency-Estimation-in-Cellular-Networks

ملخص

يأتي هذا العمل في إطار التطور السريع لشبكات الاتصالات الخلوية والأهمية المتزايدة لزمن التأخير كمؤشر أساسي لقياس الأداء .ومع ظهور تقنيات الجيل الرابع والخامس، بالإضافة إلى التطبيقات ذات الزمن الحقيقي مثل إنترنت الأشياء والواقع الافتراضي والأنظمة ألذكية أصبح تقدير زمن التأخير بدقة أمرا ضرورياً.

يهدف هذا البحث إلى اقتراح مقاربة تعتمد على تقنيات التعلم ألآلي وبشكل خاص الشبكات العصبية الاصطناعية من أجل التنبؤ بزمن التأخير في الشبكات الخلوية وقد تم اعتماد منهجية متكاملة تشمل تحليل البيانان المعالجة مسبقة التصميم من خلال تصميم النموذج وتقييمه

يعتمد النموذج المقترح على شبكة عصبية متعددة الطبقات تم تدريبها على مجموعة بيانات حقيقة تضم أكثر من 16000ملاحظة مأخوذة من شبكات الجيل الثالث و الرابع و الجيل الخامس وقد أظهرت النتائج التجريبية أن النموذج يحقق أداءً جيداً بمعامل تحديد (R^2)يقارب 0.84وخطأ متوسط مقبول. كما تم إجراء مقارنة مع نماذج أخرى مثل الانحدار الخطي وغابات القرار، مما أكد فعالية النموذج المقترح الأخير،و تم اقتراح آفاق مستقبلية لتحسين الأداء من خلال إدماج متغيرات إضافية واستخدام نماذج أكثر تقدماً.

الكلمات المفتاحية :الشبكات الخلوية، زمن التأخير، التعلم الآلي، التعلم العميق، الشبكة العصبية متعددة الطبقات

Abstract

This thesis is conducted within the context of the rapid evolution of cellular networks and the increasing importance of latency as a key performance indicator. With the emergence of 4G and 5G technologies, as well as real-time applications such as the Internet of Things, virtual reality, and intelligent systems, accurate latency estimation has become a critical challenge.

The main objective of this work is to propose a Machine Learning-based approach, particularly using neural networks, to estimate latency in cellular networks. A comprehensive methodologie has been adopted, including data exploration, preprocessing, model design, and evaluation.

The proposed model is based on a Multi-Layer Perceptron (MLP) trained on a real dataset containing more than 16,000 observations collected from 3G, 4G, and 5G networks. Experimental results demonstrate that the model achieves good performance, with a coefficient of determination (R^2) of approximately 0.84 and an acceptable mean error.

A comparison with other regression models, such as Linear Regression and Random Forest, validates the effectiveness of the proposed approach. Finally, several improvement perspectives are discussed, including the integration of additional features and the use of more advanced models.

Keywords: Cellular networks, Estimation latency, Machine Learning, Deep Learning, Neural networks, MLP.

Résumé

Ce mémoire s'inscrit dans le contexte de l'évolution rapide des réseaux cellulaires et de l'importance croissante de la latence en tant qu'indicateur clé de performance. Avec l'émergence des technologies 4G et 5G, ainsi que le développement des applications temps réel telles que l'Internet des objets, la réalité virtuelle et les systèmes intelligents, la maîtrise et l'estimation précise de la latence réseau deviennent essentielles.

L'objectif principal de ce travail est de proposer une approche basée sur les techniques d'apprentissage automatique, en particulier les réseaux de neurones artificiels, pour prédire la latence dans les réseaux cellulaires. Une méthodologie complète a été adoptée, incluant l'exploration des données, le prétraitement, la conception du modèle et son évaluation.

Le modèle proposé repose sur un réseau de neurones multicouches (MLP) entraîné sur un jeu de données réel comportant plus de 16 000 observations issues de réseaux 3G, 4G et 5G. Les résultats expérimentaux montrent que le modèle atteint une bonne performance avec un coefficient de détermination R^2 d'environ 0,84 et une erreur moyenne acceptable.

Une comparaison avec d'autres modèles de régression, tels que la régression linéaire et les forêts aléatoires, a permis de valider la pertinence de l'approche proposée. Enfin, des perspectives d'amélioration sont proposées, notamment l'intégration de nouvelles variables et l'utilisation de modèles plus avancés.

Mots-clés : Réseaux cellulaires, Estimation de la latence, Machine Learning,Deep Learning, réseaux de neurones, MLP.