REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Mohamed Khider – BISKRA
Faculté des Sciences Exactes, des Sciences de la Nature et de la Vie
Département d'informatique

N° d'ordre :RTIC08/M2/2022

Mémoire

Présenté pour obtenir le diplôme de master académique en

Informatique

Parcours : **Réseaux et Technologies de l'Information et de la Communication(RTIC)**

La prédiction de flux de trafic par l'apprentissage profond hyper paramétrés dans les réseaux véhicules

---

Par :

TEBIB WISSAL

Soutenu le 28/06/2022 devant le jury composé de :

GUERROUF FAYCAL	M.A.A	Président
BITAM SALIM	PROF	Rapporteur
BOUKHLOUF DJEMAA	M.C.B	Examinateur

Année universitaire 2021-2022

Résumé

La congestion du trafic est l'un des plus grands problèmes auxquels le monde est confronté aujourd'hui, car elle menace la sécurité routière et donc les inconvénients pour les usagers de la route. Par conséquent, les problèmes croissants et le développement du monde des transports ont conduit au développement du système de transport intelligent, et l'un des éléments efficaces dans ce système pour améliorer le trafic est la prévision du trafic routier .
Faciliter les citoyens avec des prévisions de trafic précises augmente leur confort et leur sécurité sur la route, car la précision des prévisions de trafic réside dans l'optimisation ou le réglage des hyperparamètres ce qui joue un rôle majeur dans l'amélioration des performances de prévision.
Pour cela nous travaillerons dans ce mémoire de master sur l'optimisation d'un hyperparamètre dans le modèle ConvLSTM2D-Bi proposé précédemment. Où on va optimisé l'hyperparamètre **la taille du lot (Batch size)** avec la méthode de **la recherche par grille(Grid Search)** qui fonctionne pour tester les performances d'un ensemble de valeurs proposées, puis choisir la meilleure valeur qui donne les meilleures performances. Concernant le modèle proposé, il est basé sur un réseau de neurones récurrents à une mémoire convolutive à long-court terme (ConvLSTM2D) comme un encodeur afin dextraire les informations spatio-temporelles sur le flux du trafic et un LSTM bidirectionnel (Bi-LSTM) comme un décodeur pour analyser les données historiques de flux de trafic pour obtenir la caractéristique de périodicité de flux de trafic.

Dans ce travail, on utilise la base de données de Transport for Grand Manchester (TfGM) pour un ensemble d'expérimentation, il s'agit de montrer l'effet des hyperparamètres sur les performances de prédiction. Après une série d'expérimentations, les résultats obtenus ont été

satisfaisants en termes de métriques d'évaluation à savoir les métriques : la racine de lerreur quadratique moyenne (RMSE) ,erreur absolue moyenne (MAE) et le carré R (R^2).

Les mots clés : Système de transport intelligent, prédiction de flux de trafic, LSTM (Mémoire à long-court terme),la recherche par grille .

Abstract

Traffic congestion is one of the biggest problems facing the world today, as it threatens road safety and therefore inconvenience to road users. Therefore, the growing problems and development in the transportation world has led to the development of the intelligent transportation system, and one of the effective elements in this system to improve traffic is road traffic forecasting.
Facilitating citizens with accurate traffic forecast increases their comfort and safety on the road, as the accuracy of traffic forecast lies in the optimization or tuning of hyperparameters which plays a major role in improving the forecast performance .

For this we will work in this master's thesis on the optimization of a hyperparameter in the ConvLSTM2D-Bi model proposed previously. Where we will optimize **the Batch size hyperparameter** with **the Grid Search method** which works to test the performance of a set of proposed values and then choose the best value that gives the best performance .And concerning the model used, it is based on a recurrent neural network with a convolutional long-short term memory (ConvLSTM2D) as an encoder in order to extract the spatio-temporal information on the traffic flow. and a bi-directional LSTM (Bi-LSTM) as a decoder for analyzing the historical traffic flow data to obtain the traffic flow periodicity characteristic. In this work, we use the Transport for Greater Manchester (TfGM) database for a set of experiments to show the effect of hyperparameters on prediction performance.After a series of experiments, the results obtained were satisfactory in terms of evaluation metrics, namely the metrics : the root mean square error (RMSE), mean absolute error (MAE) and the square R (R^2).

Keywords : Intelligent Transport System, traffic flow prediction, LSTM (Long-Short Term Memory), grid search

الملخص

يعد الازدحام المروري من أكبر المشكلات التي تواجه العالم اليوم ، حيث إنه يهدد سلامة الطرق وبالتالي يزعج مستخدمي الطريق. لذلك ، أدت المشاكل المتزايدة والتطور في عالم النقل إلى تطوير نظام النقل الذكي ، وأحد العناصر الفعالة في هذا النظام لتحسين حركة المرور هو التنبؤ بحركة المرور على الطرق.

إن توفير توقعات مرورية دقيقة للمواطنين يزيد من راحتهم وسلامتهم على الطريق ، حيث تكمن دقة التنبؤات المرورية في تحسين أو ضبط المعلمات الفائقة التي تلعب دورًا رئيسيًا في تحسين أداء التنبؤ.

لهذا سنعمل في أطروحة الماجستير هذه على تحسين المعلمة الفائقة في نموذج ConvLSTM2D-Biالمقترح سابقًا.

حيث سنقوم بتحسين المعامل التشعبي لحجم اللوت بأسلوب البحث الشبكي الذي يعمل على اختبار أداء مجموعة من القيم المقترحة ثم اختيار أفضل قيمة تعطي أفضل أداء.

وفيما يتعلق بالنموذج المستخدم ، فهو يعتمد على شبكة عصبية متكررة للذاكرة التلافيفية طويلة المدى (ConvLSTM2D) كمشفّر لاستخراج المعلومات المكانية والزمانية حول تدفق حركة المرور. و LSTM ثنائي الاتجاه (Bi-LSTM) كمفكك تشفير لتحليل البيانات التاريخية لتدفق حركة المرور للحصول على خاصية دورية تدفق حركة المرور.

في هذا العمل ، نستخدم قاعدة بيانات النقل لمانشستر الكبرى (TfGM) لمجموعة من التجارب لإظهار تأثير المعلمات الفائقة على أداء التنبؤ.

الكلمات الرئيسية: نظام النقل الذكي ، التنبؤ بتدفق حركة المرور ، LSTM (الذاكرة طويلة المدى) ، بحث الشبكة

Remerciements

Tout d'abord,je rends grâce à ***ALLAH*** *qui ma aidé et ma donné la force ,la santé et la patience de supporter toutes les difficultés pour achever ce travail.*

Je tiens à remercie sincèrement ma profonde gratitude à mon encadreur le professeur ***Bitam salim*** *pour leffort quil a fourni avec moi lors de la préparation de la mémoire, et ses conseils, et toutes ses facilités pour moi .*

J'adresse également un merci spécial ***à tous les membres de ma famille*** *qui m'ont soutenu à tout moment et ont prié pour moi et qui m'ont aidé à atteindre ce niveau.*

Je remercie également ***Ibtissam*** *pour son aide et ses réponses à mes questions, et je lui souhaite du succès dans sa vie.*

Et je remercie aussi tous mes amis.

Je tiens également à remercier tous les professeurs du Département d'informatique de l'Université Mohamed Khider pour tous leurs efforts et leurs dons au cours des cinq années .

Et jadresse mes sincères remerciements aux membres du jury, pour avoir accepter de juger mon travail.

Dédicace

Je dédie ce travail et son succès à

***A mon cher père**, l'homme qui m'a amené ici. Tu as toujours été pour moi un exemple de père parfait ,merci pour tout ce que vous avez donné et continuez d'offrir avec amour, attention et prières. Grâce à vous, j'ai appris le sens de l'insistance sur le succès et la responsabilité. Merci beaucoup pour tout, mes mots ne vous suffiront pas. Je prie Dieu de te protéger et de prolonger ta vie.*

***A ma chère mère** qui m'a élevé et fait de moi ce que je suis Il n'y a pas de mots pour vous exprimer mon amour et ma gratitude pour votre soutien et vos prières pour moi, vous êtes mon bien le plus précieux. Que Dieu vous bénisse*

A mon cher mari "Djalal" *La personne qui m'a toujours soutenu en tout temps, m'a poussé et motivé à réussir. Je vous remercie pour tout ce que tu m'as donné et que Dieu vous garde pour moi.*

A mes frères et soeurs "Walid Toufik Widad Wissam Wafa et Wail" et leurs familles *Je vous suis reconnaissant pour les encouragements que vous m'avez donnés et pour votre confiance et votre bonheur de ma réussite.*

Et à tous ceux qui m'ont aidé de près ou de loin.

Je dédie ce travail à vous tous en remerciement du soutien que vous m'avez apporté tout au long de cette période

Wissal

Table des matières

Table des figures

Liste des tableaux

Introduction générale

Les accidents de la route tuent, chaque année près de 1,25 million de personnes et laissent environ 20 à 50 millions de blessés ou handicapés. C'est la principale cause de décès parmi les jeunes âgés de 15 à 29 ans. En raison de ces statistiques, il a été constaté que les anciens feux et les panneaux de signalisation étaient incapables ou pas assez efficaces pour maintenir la sécurité routière. Si rien n'est fait, les accidents de la route deviendront la septième cause de décès avant 2030 [9] .

En plus de ça ,il y a d'autres problèmes qui se produisent sur la route qui gênent la circulation et font perdre du temps et L'argent des usagers de la route, comme : les embouteillages. Ce dernier est le principal facteur de pollution atmosphérique.Cela menace la sécurité routière, qui est le problème le plus important au monde .

Afin de résoudre ce genre de problèmes, les recherches se sont initiées par de nouvelles technologies de communication dans le domaine des transports dont le Système de Transport Intelligent (ITS). Les ITS sont des applications qui visent à fournir des services novateurs dans différents modes de transport et de gestion du trafic. Ils permettent aux utilisateurs d'être mieux informés sur l'état de la route.Ils rendent l'utilisation des réseaux de transport plus sûre, plus coordonnée et plus intelligente [16]. Avec le le développement du ITS, où une grande quantité de données est générée chaque jour, beaucoup des approches ont été proposées pour faire face à le problème de congestion du trafic basés sur la collecte de données à partir du cas STI des systèmes routiers.La prévision des flux de trafic devient la principal élément d'amélioration de la fluidité du trafic . Cependant, la croissance rapide des données générées devient une émergence défi que les systèmes de traitement traditionnels ne sont pas en mesure pour faire face aux exigences d'analyse de données. Récemment, Deep Learning (DL), une évolution de Machine Learning (ML), qui contient plusieurs couches cachées . DL a

été présenté comme une méthode prometteuse pour améliorer la précision de la prévision du trafic qui est basés sur les réseaux de neurone (NN) et en particulier les réseaux de neurones récurrents (RNN) [25].

Cependant, il ne suffit pas de prévoir uniquement le trafic pour guider les conducteurs à circuler de manière optimale afin daméliorer la sécurité routière.Le confort des usagers de la route à laide de prédiction préciser, il doit les performances de prévision doivent être bonnes et proches à les valeurs réelles . Et pour y parvenir, nous avons besoin de ce qu'on appelle l'optimisation des ou le réglage des hyperparamètres.

Notre projet de master vise à réaliser la prédiction de flux de trafic par apprentissage profond hyper paramétrés dans les réseaux véhicules. Pour cela, pour la prédiction du flux de trafic on utilise une méthode dapprentissage profond issue de lintelligence artificielle qui est basée sur la variante du réseau de neurones récurrents RNN c'est mémoire à long-court terme (LSTM) qui se compose de deux modules : une mémoire convolutive à long-court terme (ConvLSTM2D) comme un encodeur pour extraire les caractéristiques spatiale-temporelle et une mémoire Bidirectionnelles LSTM (Bi-LSTM) comme décodeur pour extraire les caractéristiques périodiques de flux du trafic. et pour optimiser l'hyper-paramètre nous utiliserons la méthode de la recherche par grille "Grid Search" où elle permettre de tester une série de paramètres et de comparer les performances pour en déduire le meilleur paramétrage de prédiction .

Notre mémoire est organisé en quatre chapitres principaux :

- Le premier chapitre , il comprend les conception de base du réseau véhiculaire ,le système de transport intelligent, les normes relatives aux réseaux véhiculaires, la prédiction des flux de trafic pour améliorer la sécurité routière, et l'apprentissage automatique et ses types.
- Le deuxième chapitre : il comprend la définition du l'apprentissage profond pour la prédiction d les réseaux de neurones récurrents (RNN) avec ses types (Lstm et Gru). On présente aussi un état de lart sur le réglage des hyper-paramètres pour la prédiction de flux de trafic.
- le troisième chapitre : nous présenterons la conception générale et détaillé de notre système de prédiction avec la description des différentes étapes suivi dans l'étude .

— le quatrième chapitre , nous illustrerons l'implémentation de notre système, et les environnements matériels et logiciels pour le développement de notre système et les différents expérimentions et les résultats obtenus.

Et nous terminons notre mémoire avec un conclusion générale et les travaux futurs sur ce projet .

Chapitre 1

Prédiction de flux de trafic routier

1.1 Introduction

Les réseaux véhiculaires ad hoc (VANET) ne sont quune application des réseaux mobiles ad hoc (MANET). Ils constituent le noyau dun système de Transport Intelligent(STI) ayant comme objectif principal lamélioration de la sécurité routière en tirant profit de lémergence de la technologie de communication et la baisse du coût des dispositifs sans-fil.
Dans ce chapitre, nous allons avoir les notions de base nécessaires à la compréhension des réseaux véhiculaires et le système de transport intelligent, et nous présentons aussi les normes relatives aux réseaux véhiculaires (DSRC/WAVE, SAE J2735), la Prédiction de flux de trafic pour l'amélioration de la sécurité routière et l'apprentissage automatique et ses types.

1.2 Réseaux de véhicules, ITS

Le réseau véhiculaire ad hoc (VANET) est une nouvelle technologie émergente qui intègre un réseau ad hoc, un réseau local sans fil (WLAN) et une technologie cellulaire pour réaliser des communications enter les véhicules intelligents et améliorer la sécurité et l'efficacité du trafic routier [18].
Les VANETs sont considérés comme un ensemble de véhicules intégrés avec des unités embarquées (On-Board Unit "OBU") et des infrastructures routières (c'est-à-dire des unités routières (Road Side Unit "RSU") et des dispositifs de contrôle appartiennent à une autorité

centrale (CA) [22].

Les unités embarquées (OBU) sont des unités de traitement regroupant un ensemble de composants matériels et logiciels de haute technologie installés dans le véhicule à savoir les GPS, les radars, les caméras, et les divers capteurs. Ces composants permettent au véhicule de se localiser et connecter directement à dautres véhicules à travers des pseudonymes et/ou RSU et CA qui sont dans sa portée de communication à l'aide d'un ensemble de programmes [16].

Les infrastructures routières (Road Side Unit - RSU-) sont des entités situées et installées au bord de la route, qui présentent des points d'accès au réseau . L'objectif d'un RSU est de transmettre des messages qui contient des informations sur les conditions météorologiques, et sur l' état de la route (vitesse maximale,autorisation de dépassement...) et qui sont destinées aux véhicules qui se trouvent dans sa zone radio [16]. Ces messages sont :

- Le message de contrôle : Sachant que chaque véhicule émet ce message chaque 100 ms pour s'afficher aux autres véhicules voisins sa destination, sa position et sa vitesse [16].
- Le message d'alerte : est envoyés lors d'une situation dangereuse (exemple : le cas d'un accident) .Ce message sert à prévenir les véhicules qui se dirigent vers la zone de l'accident, il est de petite taille pour assure sa transmission rapide[16].

La figure suivante présente la diffusion des messages entre les véhicules.

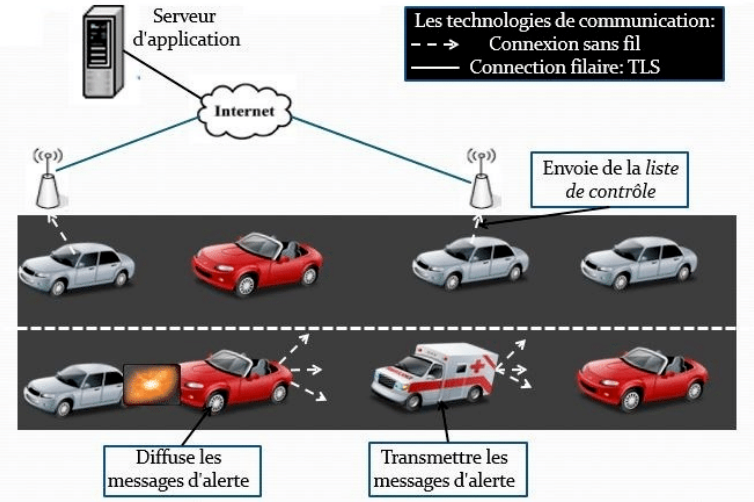

FIGURE 1.1 – La diffusion des message

La figure suivante présente Les dispositifs RSU et OBU

FIGURE 1.2 – Les dispositifs RSU et OBU

L'autorité centrale (CA) : (l'autorité de confiance) est un serveur de stockage et de transaction qui a une connexion avec toutes les entités du réseau pour assurer la sécurité des différents services tels que la délivrance des certificats et les clés de communication et les pseudonymes des véhicules [16].

Les dispositifs OBU, RSU et CA peuvent établir la connexion entre eux pour communiquer et c'est pour partager des informations utiles dans le but de faciliter la conduite.

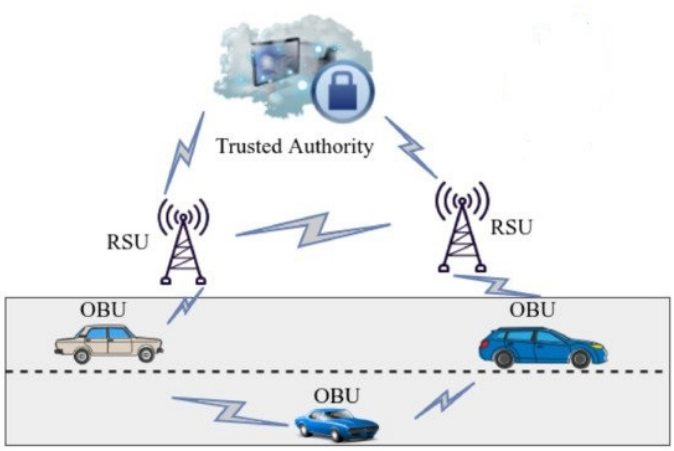

FIGURE 1.3 – Les dispositifs RSU et OBU et CA

Les réseaux de véhicules se composent principalement de deux types : les communications de véhicule à véhicule (V2V), et les communications de véhicule à infrastructure routière (V2I) [22].

La communication de véhicule à véhicule : est la communication directe entre les véhicules (les noeuds) . Cette communication est etablie à l'aide de l'OBU. Elle sert à diffuser l'information dans le réseau ou à la transporter d'un noeud vers un autre [16].

La communication de véhicule à infrastructure routière : est la communication réalisée entre les noeuds (Véhicules en utilisant l'OBU) et les entités fixes (RSU et CA). Ceci permet aux véhicules d'accéder aux différents types d'applications (Sécurité, confort, gestion ...) et aux différents types d'informations (Etat du trafic, météo ..) [16].
La figure suivante présente les types de communications dans un VANET.

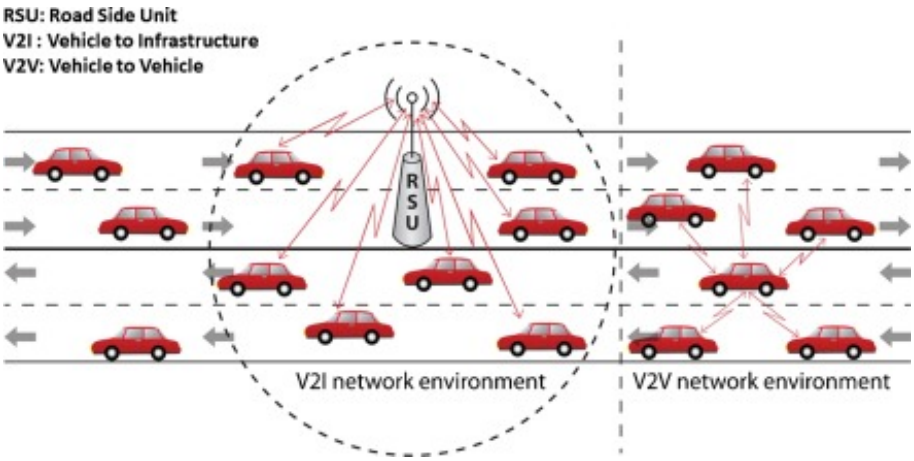

FIGURE 1.4 – Types de communications dans un VANET

L'objectif principal des réseaux de véhicules est de créer un Système de Transport Intelligent (STI) [21].

Ce système de transport intelligent (STI) comprend un large éventail d'applications qui traitent et partagent des informations qui sont obtenues à partir de sources diverses et variées telles que les cartes à puce, les GPS, les capteurs.., ceci est pour réduire les embouteillages, améliorer la gestion du trafic, accroître les avantages du transport pour les utilisateurs de la route [4].

Le STI est l'un des de plusieurs solutions fiables pour améliorer la sécurité routière, où le support de communication (par exemple, entre les véhicules et les autres composants d'un environnement STI) est généralement sans fil [4].

La figure suivante présente le Système de Transport Intelligent (STI).

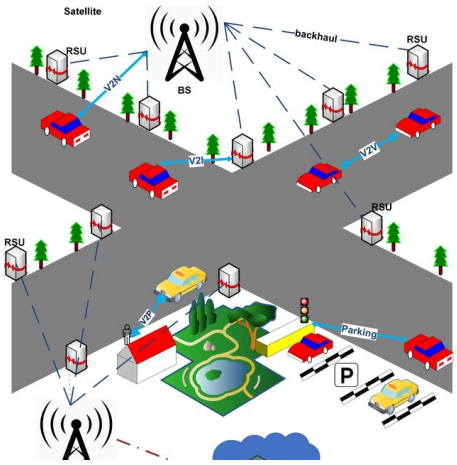

FIGURE 1.5 – Système de Transport Intelligent (STI)

1.3 Normes relatives aux réseaux véhiculaires

La grande vitesse de déplacement des véhicules et le changement dynamique du type de communication [9] obligent de faire une normalisation et de créer des standards dans le domaine de réseaux véhiculaires VANET pour mieux répondre aux exigences de ce domaine [9], et pour obtenir une excellente communication basée sur la mobilité et réduire les retards dans l'échange d'informations, par la prise en charge d'un débit de données élevé et assurer également une connexion instantanée [14].

Permis ces normes :

1.3.1 DSRC/WAVE

Communications dédiées à courte portée (DSRC) / Accès sans fil pour l'environnement des véhicules (WAVE) est une suite de protocoles standardisée destinée à être utilisée dans les communications de l'environnement des véhicules dont le mode soit v2v ou V2I) [20].

DSRC : (Dedicated Short-Range Communications) représente un ensemble de protocoles et de normes définissant la communication à courte portée. Cette norme a été créée en 2002 par l'ASTM (American Society for Testing and Materials) spécialement pour les systèmes de transport intelligents ITS [9].

WAVE : (Wireless Access in Vehicular Environment) représente un groupe de normes et

de protocoles d'accès sans fil dans un environnement véhiculaire. Il sagit dune architecture avec un ensemble de standards, de services et dinterfaces permettant de sécuriser les différents types de communications dans le système de réseau VANET [9]. On note que le protocole CSMA/CA est utilisé comme moyen de contrôle d'accès (MAC) car il prend en charge les changements fréquents dans la topologie du réseau avec une plus grande mobilité [14].

1.3.2 SAE J2735

La Société des ingénieurs de lautomobile (Society of Automotive Engineers) (SAE) a développé la norme J2735, qui spécifie un message pour prendre en charge l'interopérabilité entre les applications basées sur le DSRC [6].
La norme SAE J2735 définit environ 150 éléments de données standards et 70 trames de données standards et décrit 15 types densembles de messages de données utilisés par les applications pour échanger des données sur le DSRC/WAVE et dautres protocoles de communication [6]. De plus, le SAE J2735 comprend les catégories de messages suivantes : sécurité, géolocalisation, informations sur les voyageurs et paiement électronique [4].

1.4 Prédiction le flux de trafic pour l'amélioration de la sécurité routière

La sécurité routière est un enjeu important dans tout système de transport intelligent (STI). La sécurité routière est une question importante dans tout système de transport intelligent (STI) [3], car il tente de résoudre les problèmes qui y sont liés tels que la congestion du trafic, les émissions de carbone et les accidents de la circulation etc.
L'acquisition rapide et précise d'informations sur les flux de trafic est un élément essentiel du déploiement des STI, car elle améliore l'efficacité de la transmission des données et améliore le transport du trafic et aide à lancer de nombreux autres services ou applications, tels que la réduction des embouteillages, la réduction de la consommation de carburant et la minimisation des accidents.

L'objectif de la prévision de flux de trafic est d'analyser les informations historiques et réelles sur le flux de trafic collectées à partir de diverses sources (radar, caméras, matériel de véhicule, équipement d'infrastructure).

Il en résulte une amélioration de l'établissement des données de routage et une augmentation de la précision du calcul pour améliorer le routage des véhicules ce qui entraîne moins d'embouteillages.

La prévision de flux de trafic en général est un moyen essentiel pour aider à gérer le trafic et permettre aux conducteurs de choisir un itinéraire approprié en fonction des données qui leur sont fournies pour améliorer leur confort de voyage et améliorer la sécurité routière [15][24].

1.5 Apprentissage automatique

L'apprentissage automatique (Machine Learning -ML-) est une branche de l'intelligence artificiellen [5]. C'est un domaine d'étude qui fournit aux ordinateurs la capacité d'apprendre sans être explicitement programmés. On dit qu'un programme est en train d'apprendre lorsque sa performance pour une tâche, mesurée par un certain critère, s'améliore avec l'expérience [17].

Cet apprentissage applique systématiquement des algorithmes pour synthétiser les relations sous-jacentes entre les données et les informations [5].

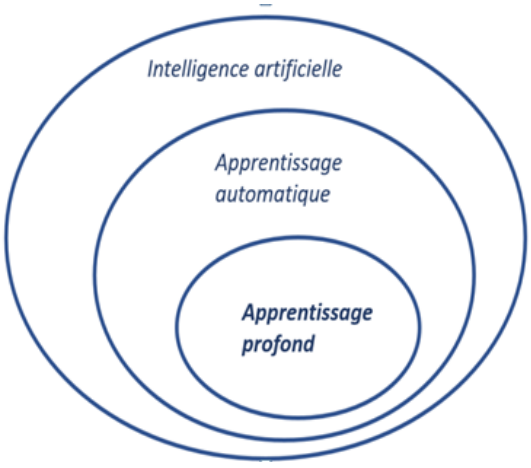

FIGURE 1.6 – l'apprentissage automatique

1.5.1 Les types d'apprentissage automatique

Lapprentissage automatique se divise en trois types :
lapprentissage supervisé, lapprentissage non supervisé et lapprentissage par renforcement. La figure ci-dessous présente les types d'apprentissage automatique.

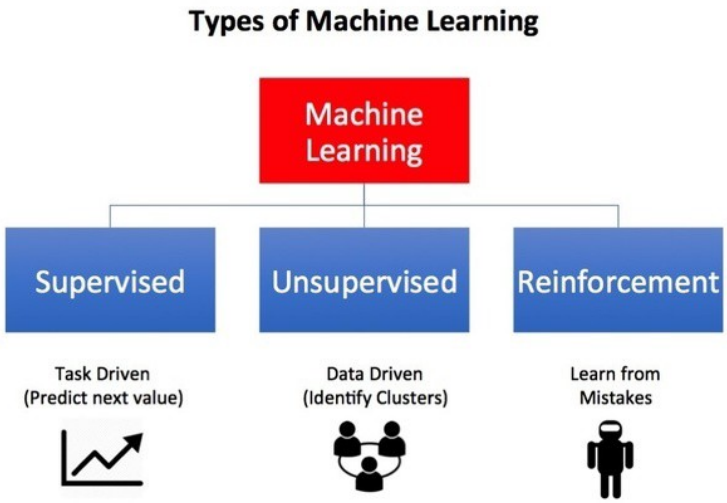

FIGURE 1.7 – les types d'apprentissage automatique

1.5.1.1 L'apprentissage supervisé

Ce sont des techniques d'apprentissage qui extraient des associations entre attributs indépendants et un attribut dépendant désigné (l'étiquette). Cet apprentissage utilise un ensemble de données d'apprentissage (training data) pour développer un modèle de prédiction à l'aide de données d'entrée et de valeurs de sortie. Le modèle peut alors faire des prédictions de la valeur de sortie pour un nouveau jeu de données. Les performances des modèles développés à l'aide de l'apprentissage dépend de la taille et de la variance de l'ensemble de données d'apprentissage pour atteindre une meilleure généralisation et un plus grand pouvoir prédictif pour les nouveaux ensembles de données [5]. Il existe deux types principaux de problèmes qui sont traités par lapprentissage supervisé : la classification :où la sortie de la prédiction est un étiquette de classe (valeurs discrètes). La régression : où la sortie est une étiquette numérique (valeurs réelles continue) [15].

1.5.1.2 L'apprentissage non supervisé

Il décrit une classe de problèmes qui implique l'utilisation d'un modèle pour décrire ou extraire des relations entre les données et pour apprendre des techniques qui regroupent des instances sans attribut dépendant prédéfini. Comparé à l'apprentissage supervisé, l'apprentissage non supervisé fonctionne uniquement sur les données d'entrée sans sorties ni variables cibles. En tant que tel, l'apprentissage non supervisé n'a pas d'enseignant corrigeant le modèle, comme dans le cas de l'apprentissage supervisé [2]. Il existe de nombreux types d'apprentissage non supervisé :

— Clustering : problème d'apprentissage non supervisé qui consiste à trouver des groupes dans les données.

— Estimation de la densité : problème d'apprentissage non supervisé qui consiste à résumer la distribution des données[2].

1.5.1.3 L'apprentissage par renforcement

Il décrit une classe de problèmes où un agent opère dans un environnement et doit apprendre à fonctionner en utilisant la rétroaction reçue par les interactions avec un environnement externe. L'utilisation d'un environnement signifie qu'il n'y a pas d'ensemble de données d'apprentissage fixe, mais plutôt un objectif ou un ensemble d'objectifs qu'un agent doit atteindre c'est-à-dire ne se contente pas de faire l'expérience d'un ensemble de données fixe, sachant qu'il existe une boucle de rétroaction entre le système d'apprentissage et ses expériences [2], c'est ce qui fait la différence entre l'apprentissage par renforcement et les deux autres types d'apprentissage (supervisé et non supervisé).
L'apprentissage par renforcement est fréquemment utilisé pour la robotique, les jeux, etc.

1.6 Conclusion

La prédiction de flux de trafic est cruciale et a été considérée comme un problème clé du système de transports intelligents, en raison de son rôle important pour faire face aux problèmes qui surviennent sur la route afin daméliorer la sécurité routière et doffrir des services aux conducteurs pour leur confort.

Dans ce chapitre, on a présenté les notions générales et de base pour notre étude tels que les VANET, l'apprentissage machine, etc.

Dans le chapitre suivant, nous allons présenter les modèles de base d'apprentissage pour la prédiction de flux de trafic à savoir les modèles "LSTM" et "GRU".

Chapitre 2

Les modèles à base d'apprentissage pour la prédiction de flux de trafic : état de l'art

2.1 Introduction

La prédiction de trafic a toujours été une tâche difficile en raison de ses dépendances spatiales et temporelles complexes, le réseau de neurones récurrent (RNN) est connu comme un modèle très populaire pour le traitement des données de séquence, utilisé par la plupart des chercheurs comme une méthode hybride qui se combine avec des méthode traditionnelle ou des méthodes d'apprentissage profond (deep learning), pour donner une précision de prédiction des flux de trafic très élevée.

Dans ce chapitre, nous allons présenter l'apprentissage profond et les réseaux de neurones récurrents (RNN) avec ses types (LSTM et GRU). En suite, on va exposer l'état de l'art du réglage les hyperparamétres pour la prédiction de flux de trafic routier.

2.2 L'apprentissage profond pour la prédiction de flux de trafic routier :

L'apprentissage en profondeur est un type d'apprentissage automatique qui a été appliqué et a donné des résultats impressionnants dans les tâches de classification, de traitement du langage naturel, de réduction de dimensionnalité, de détection d'objets, de modélisation de mouvement .. . Cela a suscité beaucoup d'intérêt dans le milieu universitaire et l'industrie.

Les algorithmes d'apprentissage profond utilisent des architectures multicouches ou des architectures profondes pour extraire les caractéristiques inhérentes aux données du niveau le plus bas au niveau le plus élevé. Ces architecture les permet de découvrir d'énormes quantités de structure dans les données. Pour cela , ils sont utilisés pour la prédiction de flux de trafic car ce processus est de nature compliquée et ces algorithmes peuvent représenter des caractéristiques de trafic sans connaissances préalables, ce qui présente de bonnes performances pour la prédiction des flux de trafic[19].

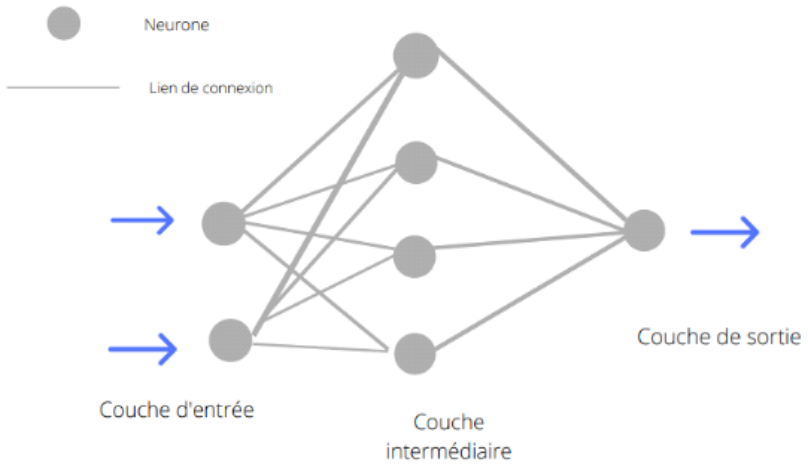

FIGURE 2.1 – Architecture des couches dapprentissage profond

L'apprentissage en profondeur a de nombreuses méthodes, certaines de ces méthodes sont utilisées pour la prédiction de flus de trafic routier comme les réseaux de neurones récurrents (RNN) et ces variantes la méthode Mémoire à long terme et à court terme LSTM et la

méthode Unité récurrente à porte GRU.

2.2.1 les réseaux de neurones récurrents (RNN)

Les modèles neuronaux profonds basés sur les données sont populaires pour prédire les flux de trafic. Les réseaux de neurones récurrents (RNN) appartiennent à un type de modèle d'apprentissage en profondeur qui peut mémoriser les données d'entrée historiques et les utiliser pour améliorer la précision des prédictions.

RNN peut mémoriser des caractéristiques historiques dans des données séquentielles temporelles . Cependant, cela conduit le problème de gradient de fuite et perd le souvenir de séquences plus longues au fil du temps [13].

La figure suivante représente la structure du réseaux de neurones récurrents (RNN).

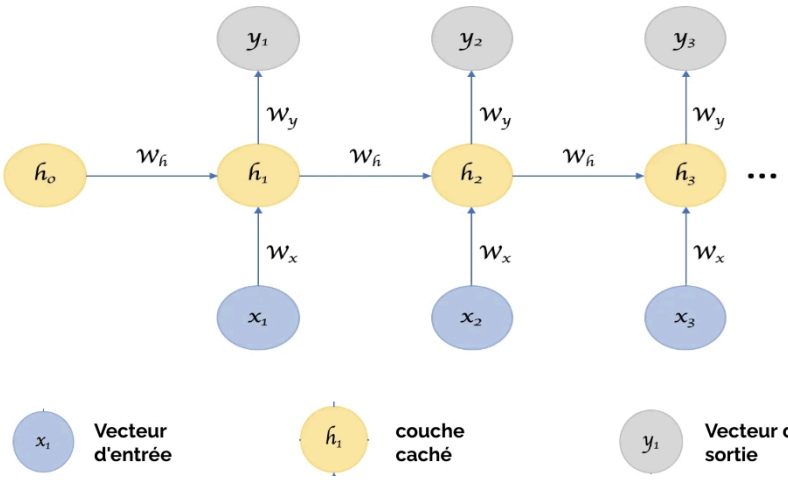

FIGURE 2.2 – Structure dun réseaux de neurones récurrents (RNN)

Pour résoudre ce problème, les chercheurs ont proposé des variations de RNN : La mémoire à long terme (LSTM) et l'unité récurrente fermée (GRU) sont deux variantes largement utilisées du réseau récurrent.

La version améliorée de RNN,le réseaux Long Short-Term Memory (LSTM), dans de nombreux cas, fonctionnent bien mieux que la version traditionnelle de RNN. Ces dernières

années,le variante Gated Recurrent Unit (GRU), a démontré des performances comparables dans de nombreuses tâches dans le domaine du traitement du langage naturel, mais avec une structure plus simple et une vitesse de fonctionnement plus élevée [12].

2.2.2 Mémoire à long terme et à court terme (LSTM)

Les réseaux de neurones LSTM ont une structure en forme de chaîne similaire à celle du RNN traditionnel, mais les opérations internes dans les cellules LSTM sont plus complexes, ce qui permet à LSTM d'apprendre les dépendances à long terme [12]. Pour résoudre les problèmes de gradient d'explosion/disparition qui surviennent généralement lors de l'apprentissage de dépendances à long terme, même lorsque les décalages temporels minimaux sont très longs[23]. Par rapport au RNN traditionnel, les réseaux LSTM ont un état de cellule supplémentaire dans lequel les informations peuvent être stockées [12].

La cellule LSTM a trois portes ajustant son état de cellule et son état caché comprenant une porte d'oubli, une porte d'entrée et une porte de sortie. Ces trois portes agissent comme des filtres, servant des objectifs différents.

- La porte d'oubli : détermine quelles informations seront éloignées de l'état de la cellule.
- La porte d'entrée : détermine quelles nouvelles informations vont être stockées dans l'état de la cellule.
- La porte de sortie : spécifie quelles informations de l'état de la cellule sont utilisées comme sortie.

Via les fonctions des trois portes, les cellules LSTM peuvent capturer les corrélations complexes dans les séries chronologiques à court et à long terme, ce qui représente une amélioration significative par rapport au RNN traditionnel [12].

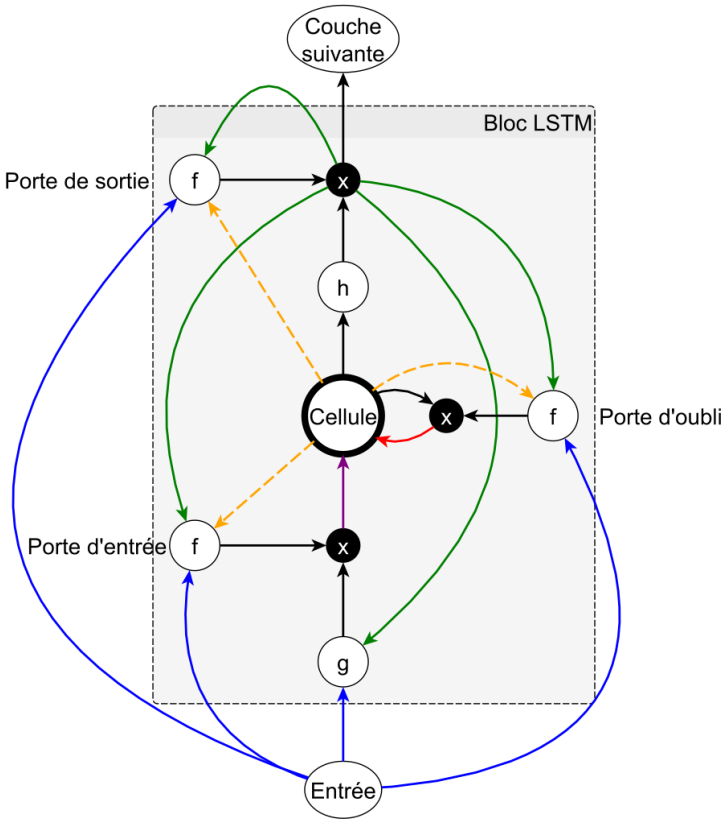

FIGURE 2.3 – La structure dun LSTM

En bref, l'architecture LSTM consiste en un ensemble de sous-réseaux connectés de manière récurrente, appelés blocs de mémoire. L'idée derrière le bloc mémoire est de maintenir son état dans le temps et de réguler le flux d'informations à travers les unités de déclenchement non linéaires [23].

L'état de la cellule et l'état caché de la cellule LSTM sont calculés comme suit :

$$f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f)$$

$$i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i)$$

$$o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o)$$

$$\tilde{c}_t = \sigma_c(W_c x_t + U_c h_{t-1} + b_c)$$

$$c_t = f_t \circ c_{t-1} + i_t \circ \tilde{c}_t$$

$$h_t = o_t \circ \sigma_h(c_t)$$

Où les 'it', 'ot', 'ft' et 'ct' représentent respectivement la sortie de la porte dentrée, de la porte de sortie, de la porte doubli et létat de la cellule de mémoire,les données dentrée de la couche cachée à lintervalle t représentent par 'xt', où 'g' signifie la fonction dactivation sigmoïde, 'c' et 'h' sont les fonctions tangentes hyperboliques, 'W' et 'U' signifie les matrices de pondération et 'b' signifie les vecteurs de biais, lopérateur 'o' désigne le produit de Hadamard (produit par éléments)[11].
LSTM a une meilleure fonction de mémoire, ce qui évite efficacement les problèmes de disparition de gradient et d'explosion, et il est donc plus apte à effectuer des tâches de prédiction[8].

2.2.3 Unité récurrente à porte (GRU)

Une architecture GRU a été proposée comme une variante de RNN par Cho et al. GRU vise à résoudre le problème du gradient de fuite. GRU ont été appliqués avec succès au problème de transport, en particulier l'estimation des flux de trafic en quelques étapes à venir . Le modèle GRU est très similaire à LSTM et les deux peuvent produire des résultats tout aussi excellents. Il peut affronter le gradient de fuite et le sur-ajustement considérés comme des problèmes bien connus avec les RNN traditionnels.
GRU structure interne est plus simple et plus rapide à former que LSTM[13].

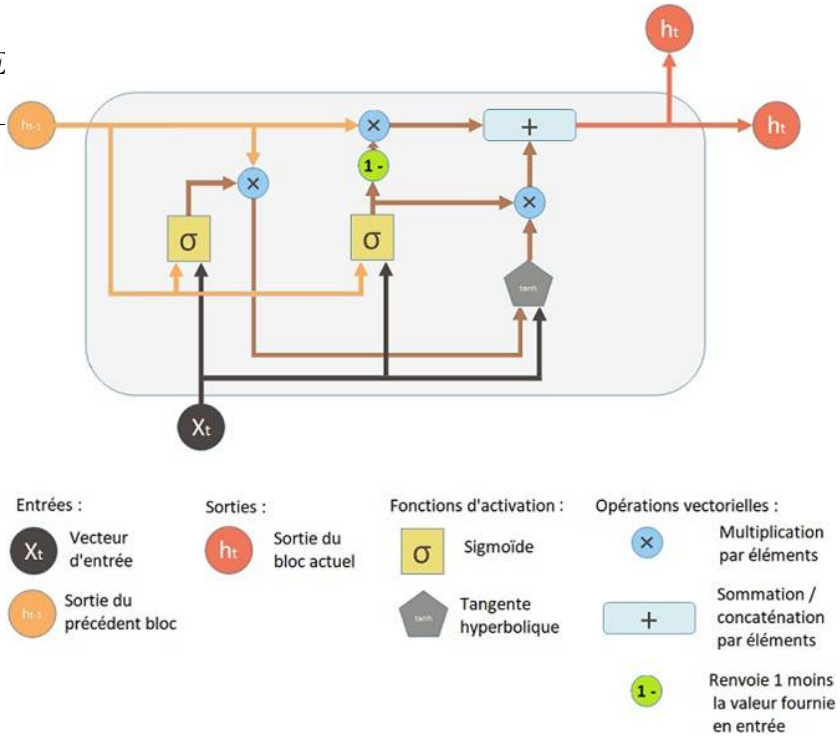

FIGURE 2.4 – La structure dun GRU

Il dispose de deux portes de contrôle (porte de réinitialisation et de mise à jour) pour surmonter le problème du gradient de fuite dans RNN. Par conséquent, GRU nécessite moins d'étapes de calcul pour mettre à jour son état latent.

— La porte de mise à jour : aide à déterminer le passé informations de la série temporelle qui doivent être transmises au avenir. La conservation de telles informations permet d'éliminer le risque d'explosion évanescente.

— La porte de réinitialisation : détermine à décider de la quantité d'informations passées à oublier [13].

GRU peut être exprimé par les équations suivantesă:

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z) \quad (1)$$

$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r) \quad (2)$$

$$c_t = \tanh(W_h x_t + r \cdot U_h h_{t-1} + b_h) \quad (3)$$

$$h_t = (1 - z_t) \cdot h_{t-1} + z_t \cdot c_t \quad (4)$$

où 'zt', 'rt' et 'h't indiquent la porte de mise à jour, la porte de réinitialisation et l'état caché actuel de GRU. 'Wz', 'Wr', 'Wh', 'Uz', 'Ur', 'Uh', 'bz', 'br' et 'bh' représentent respectivement les poids et les vecteurs de biais de deux portes et l'état caché actuel. '' est une fonction

sigmoïde non linéaire. Pour minimiser le coût, les paramètres correspondants sont mis à jour[26].

2.3 les modèles de réglage des hyperparamètres :

2.3.1 Apprentissage efficace des Hyperparamètres en ligne pour la prédiction des flux de trafic [27] :

2.3.1.1 Principe

Dans l'article [27], les auteurs proposent un algorithme d'apprentissage hyperparamétrique en ligne et pour les modèles de prédiction de trafic basés sur le noyau. Ce algorithme consiste à mettre à jour de manière adaptative les hyperparamètres du modèle avec des données en continu et pour diminuer la charge de calcul de lapplication de réglage dhyperparamèteres basée sur les gradients.

Algorithme d'apprentissage d'hyperparamètres en ligne calculer les gradients d'hyperparamètres à la volée lorsqu'une nouvelle donnée est observée, en suite, il calculer la moyenne de lhyper-gradients historiques pour faire une mise à jour des hyperparamètres .
à chaque fois, on fait une re-sélection d'hyperparamètres on supprime les hyperparamètres précédent et on les remplace par une procédure d'apprentissage incrémentale.

2.3.1.2 Discussion

- **Les avantages :**
  - Cet algorithme propose d'éviter l'optimisation périodique des hyperparamètres, il les mettre à jour de manière adaptative.
    l'algorithme proposé est 7 fois plus rapide en réglage de Hyperparamètres par rapport aux autres algorithmes de réglage.
  - il ne nécessite pas de backtesting avec les données de validation à chaque fois.
- **inconvénients :**
  - coût de calcul élevé car il necessite un adapation périodique des hyperparameters.

2.3.2 Prédiction intelligente de flux de trafic à l'aide du modèle GRU optimisé [13]

2.3.2.1 Principe

Dans l'étude publiée dans [13],les auteurs proposent un mécanisme de recherche de réseau neuronal récurrent (modèle GRU) qui permet d'obtenir des hyperparamètres de réseau, d'optimiser le nombre d'étapes de fenêtre glissante à venir et d'améliorer la précision de la prédiction en réduisant l'erreur de prédiction du flux de trafic.

L'algorithme proposé est appliqué pour obtenir un ensemble supérieur de taux d'apprentissage, de longueur de fenêtre glissante, de nombre de neurones et de technique d'optimisation. Le cadre de ce méthode proposé est : tout d'abord, ils extraient les données de flux de trafic de la base de données PeMS (Caltrans Performance Measurements Systems), qui offre une base de données historique des flux de trafic en Californie) . PeMS contient des séquences temporelles des données de trafic enregistrées par le capteur routier installé sur les autoroutes.Ces données sont observées toutes les 5 minutes d'intervalle. L'ensemble de données est traité et nettoyées pour obtenir des séquences de flux de trafic univariées et pour obtenir des détails au niveau horaire pour simplifier et l'idée est de réduire les coûts de traitement. La formation est ensuite effectuée à l'aide de l'algorithme proposé pour l'optimisation GRU, le réglage du réseau est effectué afin d'obtenir un meilleur ensemble à chaque itération et après avoir obtenu de meilleurs hyperparamètres ils ont effectué une prédiction en utilisant ensemble de mesures de performance pour comparer la précision.

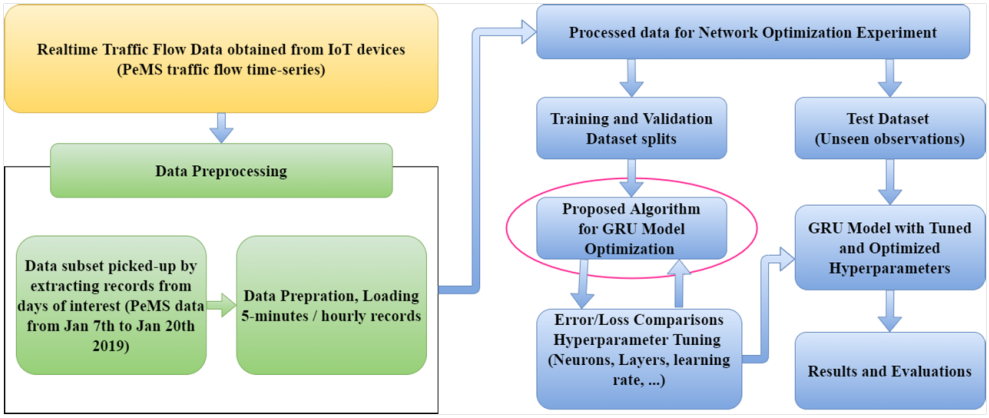

FIGURE 2.5 – Cadre d'optimisation de la formation et de la validation des flux de trafic pour la détection des flux de trafic avec le réseau de neurones GRU.[13]

L' algorithme est proposé pour remplir un double objectif, c'est-à-dire ajuster les hyperparamètres (taux d'apprentissage) et rechercher le pas de temps de la fenêtre glissante pour une meilleure validation et prédiction. L'entrée de l'algorithme est une séquence originale de séries chronologiques de flux de trafic, un ensemble de fenêtres glissantes de longueur L, taux d'apprentissage à partir d'un ensemble fini. Ils initialisent diverses valeurs de départ pour le nombre de neurones de la couche d'entrée, la fonction d'activation, la taille de la fenêtre de lot, le nombre d'époques, la fonction d'optimisation et les fonctions de perte pour le réseau GRU.

L'objectif est d'obtenir un ensemble optimisé de paramètres et d'hyperparamètres de réseau de neurones.

2.3.2.2 Discussion

— **Les avantages :**

— la méthode proposée réduit les métriques d'erreur RMSE (l'erreur quadratique moyenne), MAPE (l'erreur absolue moyenne en pourcentage) et MAE (l'erreur absolue moyenne) et assure un gain moyen de performance du réseau optimisé par

rapport au modèle normal

— **Les inconvénients :**
  — la méthode proposée nexplore pas les caractéristiques spatiales (Lors de la prédiction de flux de trafic sur un segment de route, il faut prendre en considération les caractéristiques des autres segments routiers proches pour une prédiction précise) et temporelles ; il sagit d'une dépendance ou une relation lors de la prédiction de flux de trafic actuels ou futures avec le flux de trafic des instants passés.
  — la configuration des réseaux de neurones reste un problème difficile dans lequel ils explorent de différentes configurations pour fournir un modèle approprié.

2.3.3 Un modèle d'apprentissage profond automatisé basé sur la recherche d'hyperparamètres pour la prévision du trafic routier [25]

2.3.3.1 Principe

Dans ce article [25], les auteurs concentrent sur l'application de l'apprentissage automatique pour le réglage des hyperparamètres afin d'apprendre les ensembles de données de trafic dans les principales régions des réseau routier. Dans ce modèle, les auteurs proposent un cadre automatisé pour le réglage des hyperparamètres afin d'apprendre les ensembles de données de trafic dans un écosystème en termes de réduction des tâches, ce qui prend beaucoup de temps.
Ce cadre HyperNet, utilisant des techniques avancées de science d'analyse des données pour accélérer considérablement le processus de recherche d'hyperparamètres.
les techniques utilisées sont :

— l'approche bayésienne pour la recherche automatisée des hyperparamètres et qui permet de prendre moins d'étapes de formation et d'obtenir un résultat comparable avec un nombre suffisamment élevé d'expériences.
— le méta-apprentissage : dans les réseaux routiers, les schémas de circulation (ex. heures de pointe et heures creuses) sont assez similaires à plusieurs endroits précis. Pour cela, ils utilisent le méta-apprentissage afin de réduire considérablement le temps consacré

à l'apprentissage des ensembles de données de trafic sur les réseaux routiers, dans lesquels des algorithmes du ML sont développés pour apprendre des données sans être explicitement programmés (c'est-à-dire pour automatiser le processus d'apprentissage et réduire le temps du tâches).

Ce système est basé sur trois principaux processus :

- Traitement des données : proposé pour comprendre les données de trafic collectées sur l'autoroute . Il est basé sur trois fonctionnalités
  - Prétraitement des données
  - Sélection des caractéristiques
  - Extraction de méta-caractéristiques
- Modèle profond LSTM-RNN basé sur le cadre HyperNet : ils appliquent un réseau LSTM et ils ont proposé un cadre d'entraînement du modèle LSTM dans lequel les hyperparamètres sont automatiquement activés pour améliorer les performances du modèle d'apprentissage en termes de précision et de complexité temporelle.
  les résultats sont les valeurs de prédiction du TPI (indice de performance du trafic) pour le prochain intervalle de temps dans chaque région considérée.
- Analyse prédictive : ce processus démontre les performances du modèle proposé en termes de minimisation de l'erreur carrée . Ensuite, ils comparent les résultats prédits avec les données de test (pour évaluer l'efficacité de l'approche proposée, ils comparent l'optimisation bayésienne à la recherche aléatoire en termes de précision et de tâches chronophages.)

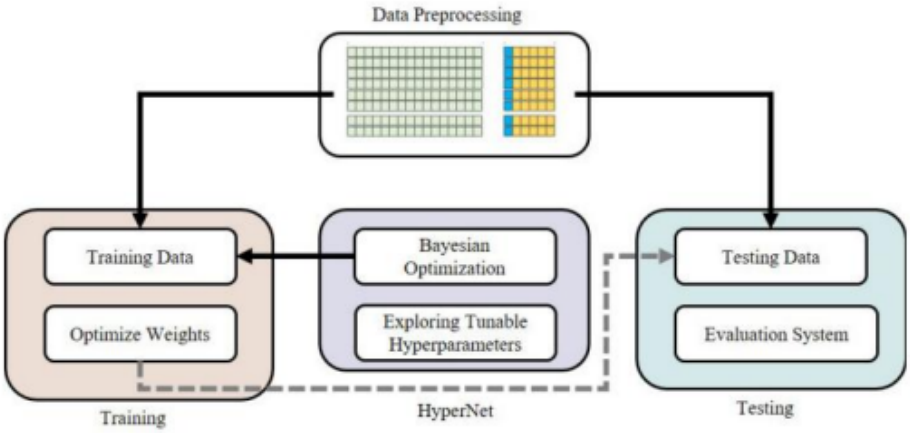

FIGURE 2.6 – Le système proposé pour la prévision du trafic utilisant le cadre HyperNet[25].

2.3.3.2 Discussion

— **Les avantages :**

— L'optimisation des hyperparamètres sur plusieurs ensembles de données en utilisant le méta-apprentissage pour le système de circulation routière pour éviter le problème de chronophage.

— **Les inconvénients :**

— l'utilisation d'optimisation bayésienne des hyperparamètres nécessite un nombre important d'évaluations pour détecter les régions à haute performance lors du démarrage d'un nouveau problème d'optimisation. En particulier, dans le cas des domaines de transport, où plusieurs ensembles de données ont été collectés et traités à partir de diverses sources dans le système (par exemple, des capteurs sur la route)

— l'entraînement des modèles d'apprentissage profond nécessite des travaux coûteux (une expertise chronophage et la détermination des configurations d'hyperparamètres dans les modèles.)

2.3.4 Modèle de prédiction des flux de trafic basé sur le réseau de croyances et l'algorithme génétique[28]

2.3.4.1 Principe :

Dans cet article [28], les auteurs proposent un modèle basé sur les réseaux de croyances profondes (DBN) pour prédire le flux de trafic. De plus, ils utilisent l'algorithme de gradient conjugué de Fletcher-Reeves pour optimiser le réglage fin des paramètres du modèle. Étant donné que le flux de trafic présente diverses caractéristiques à différents moments, tels que le jour de la semaine, le week-end, le jour et la nuit, les hyper-paramètres du modèle doivent s'adapter à l'heure. Par conséquent, ils utilisent l'algorithme génétique pour trouver les hyper-paramètres optimaux des modèles DBN à différents moments.

L'idée principale de DBN est de former de grandes quantités de données non étiquetées pour apprendre les caractéristiques typiques des données de manière non supervisée. Un DBN est formé en empilant des machines Boltzmann restreintes (RBM). Le RBM est un modèle graphique non orienté, et il n'a qu'une couche visible v et une couche cachée h. Sachant que après l'apprentissage du premier RBM, les données de sortie du premier RBM sont utilisées comme données d'entrée du second RBM.
Pour utiliser le modèle DBN pour la prédiction des flux de trafic, ils ajoutent une couche de régression sur la couche supérieure pour la formation supervisée. De cette manière, le DBN, aura la capacité d'extraire des caractéristiques de données à partir des données d'apprentissage non étiquetées

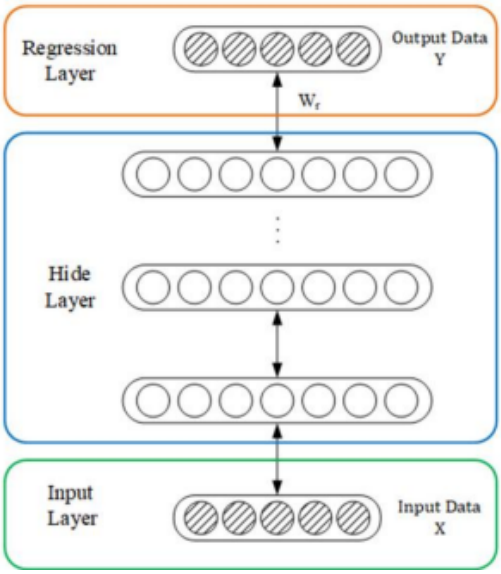

FIGURE 2.7 – La structure du réseau DBN. [28]

Pour obtenir une meilleure précision de prédiction, les auteurs utilisent un algorithme d'apprentissage non supervisé gourmand par couches pour pré-entraîner le DBN. Ensuite, ils utilisent la méthode de décente FR-CG est utilisée pour affiner l'ensemble des paramètres du réseau. La clé de cet algorithme est d'utiliser la direction du gradient des paramètres pour construire un ensemble de directions conjuguées. Les paramètres sont mis à jour dans le sens du gradient conjugué.
Et pour diminuer le coût de recherche, ils utilisent l'algorithme génétique (GA)pour trouver les hyperparamètres les plus appropriés. GA opère sur le codage des paramètres plutôt que sur les paramètres eux-mêmes. De plus, GA peut effectuer efficacement une recherche heuristique dans l'espace des solutions au lieu de la recherche aléatoire.

2.3.4.2 Discussion

— **Les avantages :**
  — L'utilisation du la méthode de réglage fin des modèles est optimisé par l'algorithme FR-CG qui accélère la convergence de les poids des modèles et ils utilisent GA pour trouver les meilleurs hyper-paramètres de modèles pou période différente.
— **Les inconvénients :**

— Le réseau DBN nécessite une pré-formation, le temps de formation de DBN est un peu plus long par rapport les réseaux de neurones récurrents (RNN).
— cette proposition n'a pas tenu en compte des informations sur l'emplacement des stations. C'est-à-dire, elle ne concentre pas sur l'impact de la relation spatiale entre les stations sur prévision des flux de trafic.

2.4 Conclusion

Dans ce chapitre, nous avons présenté lapprentissage profond pour la prédiction de flux de trafic routier, les réseaux de neurones récurrents RNN et ses types de LSTM et GRU. Ensuite, un état de lart sur les méthodes utilisées pour le réglage des hyperparamètres pour la prédiction de flux de trafic routier a été présenté.
Sur la base de cette étude, nous avons constaté que ces approches n'ont pas pris en compte un ensemble de critères tels que les corrélations spatiales ou temporelles et elles sont appliquées sur un ensemble de données réduit qui pourra non représentatif.

Par conséquent, nous allons proposer, dans le chapitre suivant un modèle de réglage des hyperparamètres pour la prédiction du flux de trafic qui améliore la précision.

Chapitre 3

Conception du système de prédiction hyper paramétrée de flux de trafic

3.1 Introduction

Dans les chapitres précédents, nous avons présenté la partie théorique qui explique les notions de base de notre projet et quelques travaux précédents qui ont étudiés le réglage des hyper paramétres, ce qui nous ont permis d'utiliser un méthode de réglage des hyper paramètres dans la conception et la réalisation de notre système .
Dans ce chapitre, nous allons présenter la tâche la plus importante dans ce travail, c'est la tâche de conception du système proposé. Tout d'abord, nous présentons larchitecture générale de notre système de prédiction de flux de trafic (hyper paramétrage), ensuite, nous allons définir les différentes étapes d'architecture et puis, nous détaillons chacune de ces étapes en précisant leurs fonctionnements.

3.2 La conception du système

La sécurité routière est considérée comme l'un des sujets les plus importants dans le monde . Un des facteurs les plus importants qui y sont parvenus est la prévision du flux de trafic sur les routes c'est l'un des rôles les plus importants du système de transport intelligent.
Dans notre système, nous optimisons les hyperparamètres pour améliorer la prédiction du

flux de trafic à partir des modules d'apprentissage profond pour assurer la sécurité et le confort sur les routes.

3.2.1 La conception générale

Notre système commence par un ensemble de données réelles de flux du trafic routier (dataset) suivi par un processus d'apprentissage profond qui contient un traitement initial des données (découpage, standardisation, ...), et par la suite, on a la détermination des hyper-paramètres par la méthode de la rechercha par grille (Grid Search). Après plusieurs itérations, on obtient le résultat final et optimal.
l'architecture générale de notre système est comme suit :

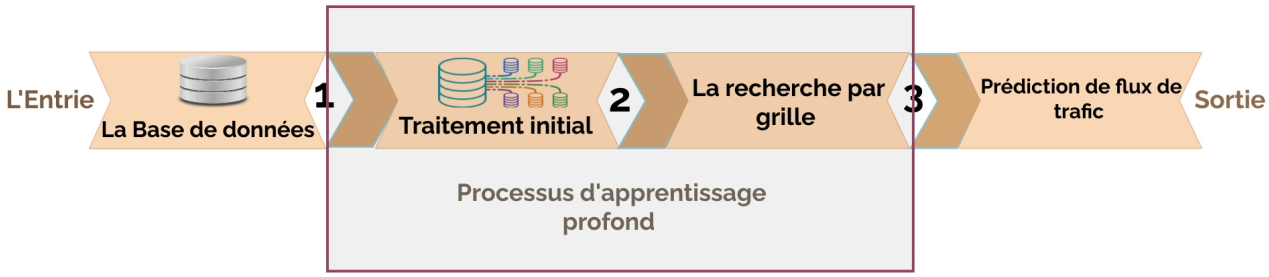

FIGURE 3.1 – L'architecture générale du système .

3.2.2 Conception détaillée

Dans cette partie nous allons présenter et décrire les étapes détaillées que notre système va suivre qui sont clarifiées dans la figure suivante :

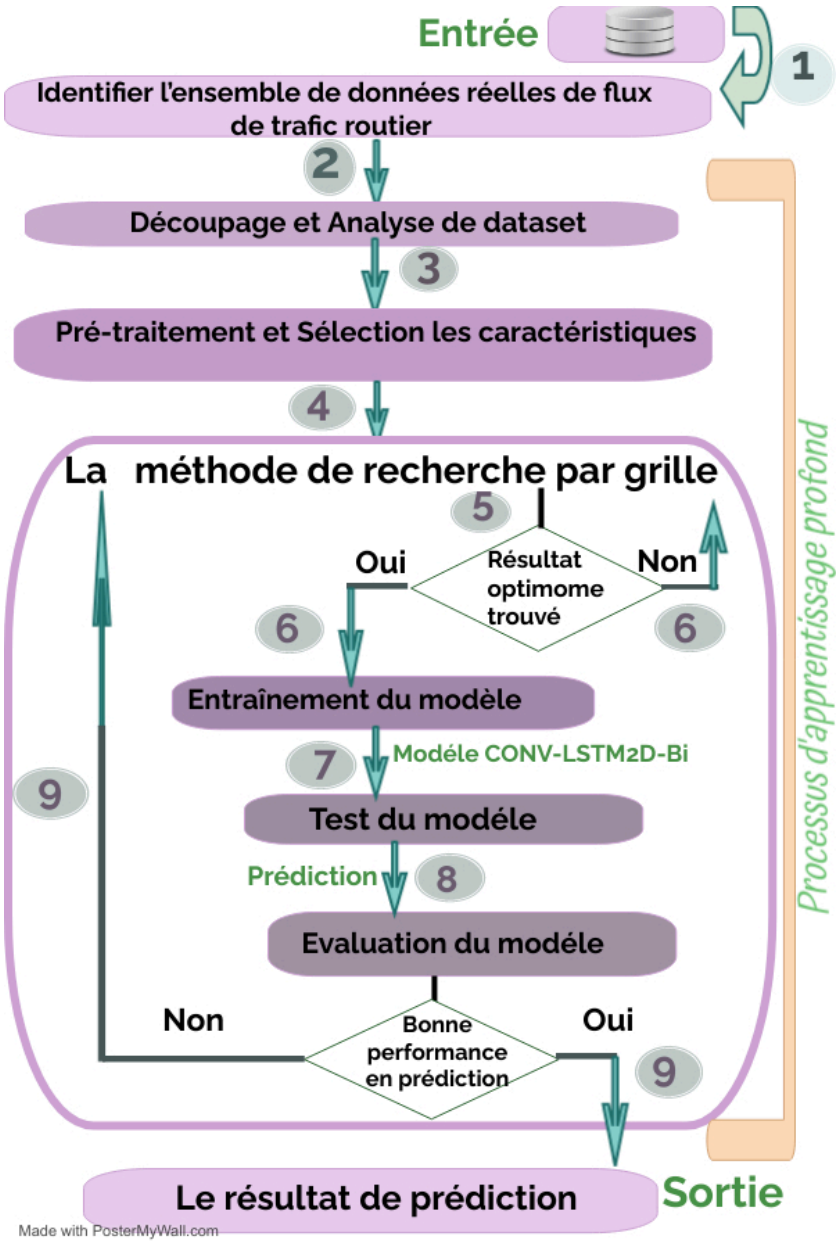

FIGURE 3.2 – L'architecture détaillée du système .

3.2.3 Identification de l'ensemble de données :

Dans notre projet, le dataset choisi est un ensemble de données de trafic sur une zone d'étude choisie dans le Grand Manchester, au Royaume-Uni. Les données de trafic utilisées dans cette étude a été fourni par le Transport for Greater Manchester (TfGM). La base de données de trafic fournie comprenait des observations par minute des caractéristiques du flux

de trafic (vitesse, débit, densité), collecté à l'aide de capteurs à boucle inductive. La zone d'étude comprenait 10 capteurs de mesure du trafic, chacun étant distant de 0,3 mille sur le route artérielle. La zone d'étude est une artère urbaine (Chester Road - A56) à Stretford, le Grand Manchester, Royaume-Uni, entre les coordonnées de longitude et de latitude entre (53.46281, -2.28398) et (53.43822, -2.31394) .[10]

Les points de repère autour sont le stade de football de Manchester United - Old Trafford - en plus d'autres points de loisirs tels que les centres commerciaux (Stretford Mall), les clubs, les restaurants, etc. Bien qu'il soit une route "A", ce qui implique qu'il devrait s'agir d'une autoroute , le tronçon considéré a une limite de vitesse réduite de 30ămph en raison du fait qu'il s'agit d'un segment très fréquenté, ayant de nombreux passages pour piétons, des lieux d'affaires et des magasins. Les données météorologiques obtenues au cours de la période d'étude comprenaient des observations horaires de température (Celsius) et précipitations (mesurées en millimètres). Les données météorologiques a été obtenu auprès du Centre for Atmospheric Studies (CAS) de l'Université de Manchester. Les stations météorologiques sont situées dans un rayon de 3 milles de la zone d'étude.[10]

La figure suivante représente la zone d'étude dans le Grand Manchester.

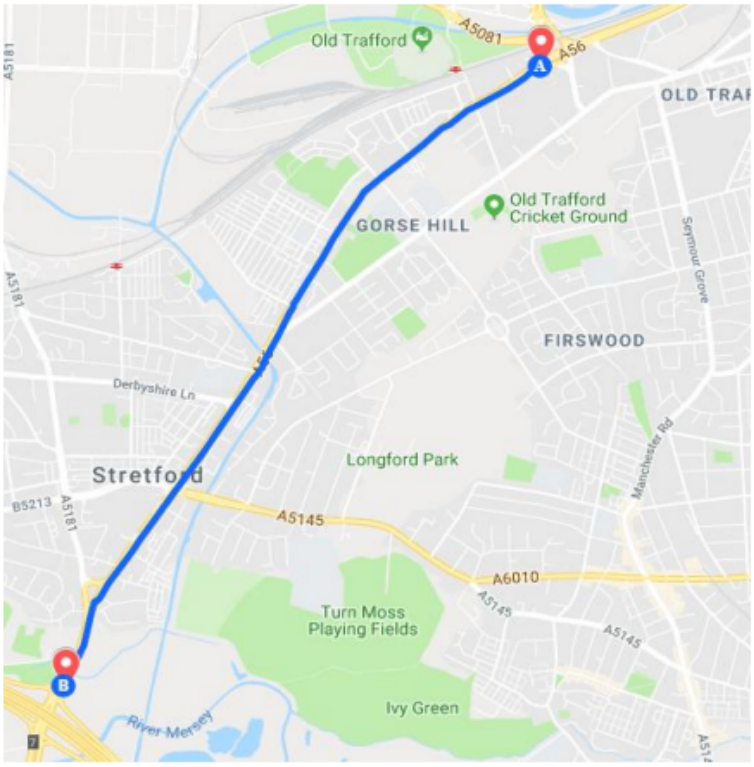

FIGURE 3.3 – La Zone d'étude.

Notre scénario de ce dataset modélise un accident de la route historique le long d'un segment de la zone d'étude qui a entraîné la fermeture de voies et une grave congestion. L'accident s'est produit le 2 mai 2014, à 13h04. Les données sur les accidents ont été obtenues à partir d'une base de données en ligne [1] , qui a mis à disposition un rapport contenant des détails sur l'accident, qui inclut la date et l'heure, les coordonnées géographiques de la route concernée, la cause de l'incident, la ou les voies fermées, etc. .[10]
la figure suivante montre un extrait de l'emplacement spécifique de l'accident le long du segment de route.

FIGURE 3.4 – Zone d'étude indiquant le lieu de l'incident .

Pour ce scénario, les données de trafic ont été agrégées en intervalles de 5 minutes pour correspondre aux exigences du trafic scientifique des données. En outre, l'ensemble de données contenait 245 376 observations des cinq (5) variables - vitesse, débit, densité, précipitations et température, soit 852 jours. de données d'entrée.[10]
Pour cela, nous avons choisi ce dataset, parce que il est volumineuse et multivariée (cest-à-dire il contient des données sur le flux de trafic et la météo) afin davoir un apprentissage plus précis.

3.2.4 Découpage et analyse de la base de donnée :

3.2.4.1 Découpage du dataset

C'est une étape importante, qu'elle permet d'évaluer efficacement notre modèle, sachant qu'elle sépare les données d'entrée en sous-ensembles d'entraînement, de validation et de test pour éviter que notre modèle ne soit trop ajusté.
La division Train-Valid-Test est une technique permettant d'évaluer les performances de

modèle d'apprentissage, qu'il s'agisse de classification ou de régression. On prend un ensemble de données donné et on le divise en trois sous-ensembles.

— **Ensemble de données d'entraînement :** ensemble de données utilisées pour l'apprentissage (par le modèle), c'est-à-dire pour ajuster les paramètres au modèle d'apprentissage automatique.

— **Ensemble de données de validation :** ensemble de données utilisé pour fournir une évaluation impartiale d'un modèle adapté à l'ensemble de données d'apprentissage lors du réglage et détermine les meilleurs hyperparamètres du modèle. Il joue également un rôle dans d'autres formes de préparation de modèle, telles que la sélection de caractéristiques.

— **Ensemble de données de test :** ensemble de données utilisé pour fournir une évaluation impartiale d'un modèle final adapté à l'ensemble de données d'apprentissage et pour mesurer la performance du modèles.

Dans notre étude, on sépare le dataset tels que la plupart des données sont utilisées pour lentraînement (90%), et une plus petite partie des données est utilisée pour le test (10%), et également (20%) de donnés dentraînement utiliser pour la validation afin dévaluer le modèle.

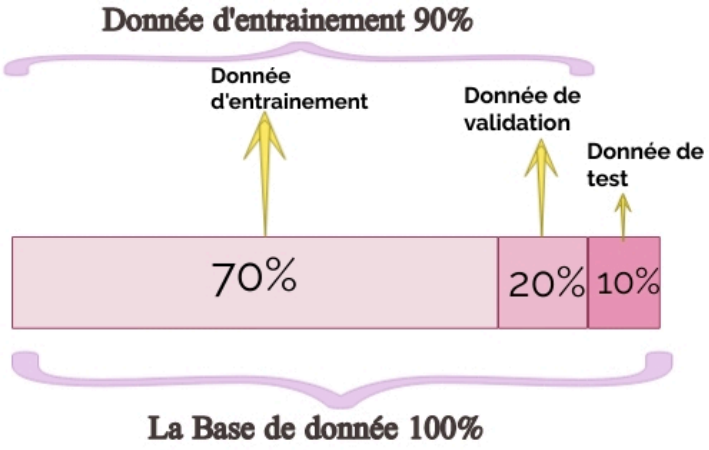

FIGURE 3.5 – La division du dataset

3.2.4.2 Analyse du dataset

Cette étape permet dobtenir une bonne idée de ce que contient lensemble de données à partir de préparer les données afin de les rendre exploitables par le modèle ,et étudier les donnée dans le but de la comprendre, de voir comment elle évolue, sa qualité, comprendre les particularités de la série. c'est-à-dire :

- Elle détermine :
  - Le type de donnée.
  - Le nombre dobservations(lignes).
  - Nombre de variables (colonnes).
  - les variables connues, les variables à prédire.
- Nettoyage des données : trouvez les "Null ", les valeurs manquantes et les données dupliquées. Il faut remplacer les "Null" et les valeurs manquantes par dautres valeurs (ou les supprimer) et sassurer de ne pas avoir de doublons.

A la fin de cette étape, nous avons préparé notre base de données pour les prochaines étapes .

3.2.5 Pré-traitement et sélection les caractéristiques

3.2.5.1 le Pré-traitement du base de donnée :

Le pré-taritement est un ensemble d'étapes préparant les données pour la phase dapprentissage. Il s'agit de transformer des données brutes en un ensemble de données structurées, prêtes à être traitées par un modèle . Dans notre travail on a consiste deux étapes :

- **Standardisation :** c'est la normalisation de la gamme de caractéristiques de lensemble de données dentrée à partir de re-dimensionner la distribution des valeurs observées de manière à ce que la moyenne μ soit égale à 0 et lécart type σ à 1 à partir de la formule suivante :

$$Z = (x\text{-}\mu\ )/\ \sigma \quad (3.1)$$

sachant que :

— X : l'observation (les données dentrée (features) originale).

— Mu(μ) : la moyenne.

— Sigma(σ) : l'écart type.

Dans notre travail, le dataset utilisé est volumineux et multivarié qui contient 5 variables dentrée (vitesse, débit, densité, précipitations et température), avec des unités différentes (mph, veh/h, veh/mi, mm, řc) C'est ce qui montre que les variables ont des échelles différentes pour cela on a besoin de la standardisation pour éviter les problèmes qui fait face au réseau neuronal durant lentraînement et réduire la difficulté du problème modélisé, causés par la mise à jour du gradient.

— **Conversion des données de séries temporelles multivariées aux séries temporelles supervisée :**

Une série temporelle multivariée contient plus d'une variable dépendante du temps, et chaque variable dépend non seulement de ses valeurs précédentes, mais également d'autres variables. Pour cela, il existe des problèmes qui entravent le travail sur des séries temporelles multivariées, tels que : la difficulté de traitement et la difficulté de modélisation, et les méthodes classiques d'analyse des séries temporelles ne donnent pas de bons résultats, donc pour résoudre ces problèmes et faciliter le travail sur la série multivariée nous transformons la série temporelle multivariée en une série temporelle supervisée en utilisant la méthode de la fenêtre glissante

Étant donné une séquence de nombres pour un ensemble de données de série temporelle multivariée, Sliding window restructurer les données pour ressembler à un problème d'apprentissage supervisé. elle utilise les pas de temps précédents comme variables d'entrée (X) et en utilisant le pas de temps suivant comme variable de sortie (y) .L'ordre entre les observations est préservé et doit continuer à être préservé lors de l'utilisation de cet ensemble de données pour former un modèle supervisé. Si il ya au-

cune valeur précédente que elle utiliser pour prédire la première valeur de la séquence, elle supprime cette ligne car n'est pas l'utiliser. Et si il n'a pas une valeur suivante connue à prédire pour la dernière valeur de la séquence, supprime cette valeur lors de l'entraînement de notre modèle supervisé. C'est-a-dire il utilise de pas de temps antérieurs pour prédire le prochain pas de temps.

Cette fenêtre glissante est donc le moyen de conversion n'importe quel ensemble de données de séries chronologiques en un problème d'apprentissage supervisé.

3.2.5.2 Sélection des caractéristiques :

Cette étape est venue après l'étape de pré-traitement car nous avons besoin de la base de données pour être prêt pour l'entraînement et le test. A cette étape, on choisit les colonnes sur lesquelles on va utiliser comme caractéristiques dentrée pour lentraînement du modèle.

3.2.6 La méthode de recherche par grille (Grid Search)

Les modèles d'apprentissage automatique ont des hyperparamètres qu'il faut définir afin de personnaliser le modèle en fonction de notre jeu de données.

Souvent, les effets généraux des hyperparamètres sur un modèle sont connus, mais la meilleure façon de définir un hyperparamètre et des combinaisons d'hyperparamètres en interaction pour un ensemble de données donné est difficile. Il existe souvent des heuristiques générales ou des règles empiriques pour configurer les hyperparamètres.

Une meilleure approche consiste à rechercher objectivement différentes valeurs pour les hyperparamètres du modèle et à choisir un sous-ensemble qui aboutit à un modèle qui atteint les meilleures performances sur un ensemble de données donné. C'est ce qu'on appelle l'optimisation des hyperparamètres ou le réglage des hyperparamètres.

Et l'un des meilleures méthodes de l'optimisation des hyperparamètres ou le réglage des hyperparamètres est **la méthode de la recherche par grille (Grid Search)** .

Grid Search la stratégie de réglage des hyperparamètres la plus largement utilisée sachant que elle permettre de tester une série de paramètres et de comparer les performances pour en déduire le meilleur paramétrage.

Étant donné un ensemble de validation "V" et un ensemble de formation "S" à partir des données historiques. Elle énumère une liste de paramètres d'hyperparamètres C-T-D pour chaque paramètre,l'utilisateur détermine un ensemble de valeurs que lon souhaite teste ,en suite elle croise simplement chacune de ces hypothèses (créé une grille de paramètres) et va créer un modèle pour chaque combinaison de paramètres . La méthode consiste à découper le data set en k échantillons. On sélectionne x échantillons pour constituer léchantillon dapprentissage. Les k-x échantillons restants permettront dévaluer la performance du modèl ,Une fois que chaque modèle a pu être entrainé et évalué, il ne reste plus quà comparer la performance pour choisir le meilleur prédiction .

Dans notre système, on va optimiser un seul hyperparamètre avec cette méthode pour améloirer le perfermance du modèle pré-proposé du prédiction de flux de trafic routier.

3.2.7 Entraînement du modèle

3.2.7.1 Modélisation :

Le modèle qu'ils ont proposé dans [15] est ConvLSTM2D-Bi et sur lequel nous continuerons à travailler , il est composé de deux modules :

- ConvLSTM2D comme un encodeur : est un LSTM convolutif utilise les convolutions directement dans le cadre de la lecture des entrées dans les unités LSTM elles-mêmes, il peut être utiliser pour extraire les données spatio-temporelles de deux dimensions des données de flux de trafic.
- Le LSTM Bidirectionnel "Bi-LSTM" (deux LSTM unidirectionnels) comme un décodeur : est utilisé pour agréger les informations dentrée dans le passé et le futur dun pas de temps spécifique dans les modèles LSTM, il est également adopté pour analyser les données historiques de flux de trafic du point de prédiction pour obtenir la caractéristique de périodicité de flux de trafic [15].

La figure suivante représente le modèle ConvLSTM2D-Bi.

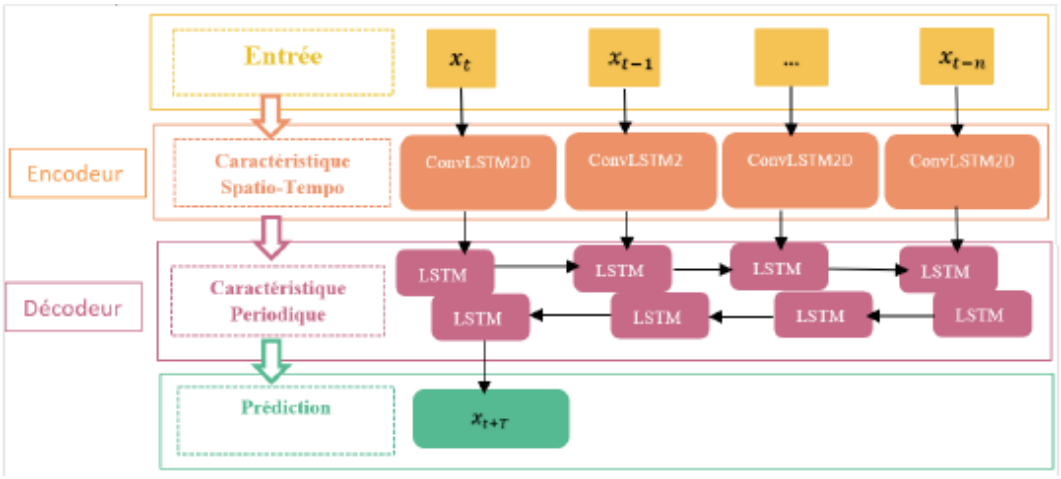

FIGURE 3.6 – Le modèle ConvLSTM2D-Bi[15].

Ce modèle contient cinq éléments principaux : lencodeur, le vecteur de répétition, laplatissement (Flatten), le décodeur et les couches entièrement connectées (FullyConnected FC). Les couches de ce modéles sont :

- La première couche : est la couche dentrée (le vecteur dentrée) qui contient les données relatives au trafic(débit,vitesse) et à la météo (température, précipitations et densité) sous la forme dun vecteur bidimensionnel "m Œ n", sachant que "m" est représente le nombre déchantillons dans lensemble de données dapprentissage, et "n" est le nombre de caractéristiques (les colonnes).
- Le deuxième couche Composé de 3 couches de ConvLSTM2D qui constituent lensemble de la couche encodeur, qui lit la séquence de vecteurs en entrée et qui faire lecture de la dernière séquence pour extraire les caractéristiques spatio-temporelles par le filtre .
- la troisème couche :c'est la couche daplatissement, Couche d'aplatissement, dont le rôle est de la transformer en un vecteur unidimensionnel.
- le quatrième couche : C'est une couche de répétition de vecteurs ,elle répète la séquence de vecteurspour les reproduire par les couches encodeurs.
- le cinquième couche : La couche décodeur contient un couche LSTM bidirectionnel qui reprend la séquence de la couche de répétition de vecteurs et produit une prédiction sous la forme dune séquence de vecteurs à une seule ligne .
- la sixième couche : c'est un ensemble des couches entièrement connectées (FC) qui

prédit la séquence cible après réception d'une série de vecteurs décodeur .

Entre chaque couche, il y a une couche dexclusion (dropout) pour éviter le sur-apprentissage (overfitting) et daccélérer le processus dapprentissage [15].

3.2.7.2 Entraînement :

Après avoir initialisé les paramètres et optimisé les hyper-paramètres du modèle, ce qui a un rôle important dans l'amélioration des performances du modèle du prédiction .
Dans cette étape, notre modèle apprend à travers les données dentraînement, Afin d' obtenir un meilleur modèle entraîné pour d'améliorer sa capacité prédictive et donne de meilleurs résultats . Et pour vérifier la capacité du modèle, il sera ensuite soumis à des tests via les données du test. lalgorithme de rétro-propagation par le temps (BPTT) est l'utilisé pour entraîner les paramètres de la modèle .

3.2.8 Test et évaluation du modèle

À ce stade et après avoir obtenu nos résultats de prédiction, nous devons évaluer les performances notre système a partir les données de test .Et puisque le modèle appartient aux problèmes de régression on va appliquer les métriques spécifiées à ce type du problèmes . les métriques utilisé sont :

3.2.8.1 Walk-Forward validation :

la validation croisée k-fold : c'est une méthode d'évaluation des modèles d'apprentissage automatique .Elle fournit une estimation robuste de la performance d'un modèle sur des données invisibles. Pour ce faire, il divise l'ensemble de données d'apprentissage en k sous-ensembles et forme à tour de rôle des modèles sur tous les sous-ensembles sauf un qui est retenu, et évalue les performances du modèle sur l'ensemble de données de validation retenu. Le processus est répété jusqu'à ce que tous les sous-ensembles aient la possibilité d'être l'ensemble de validation retenu. La mesure de performance est ensuite moyennée sur tous les modèles créés.

3.2.8.2 la racine de lerreur quadratique moyenne (RMSE) :

est une mesure d'erreur absolue qui élève au carré de toutes les erreurs et les écarts pour empêcher les écarts positifs et annuler les uns les autres. Cette mesure tend également à exagérer les erreurs importantes, ce qui peut vous aider à éliminer les méthodes comportant des erreurs importantes.

l'équation du RMSE est comme suite :

$$RMSE = \sqrt{\frac{1}{n}\Sigma_{i=1}^{n}\Big(yp - yr\Big)^2} \tag{3.2}$$

Où

— yp :la valeur prédite

— yr :la valeur réelle de base

— n : le nombre de toutes les valeurs prédites

3.2.8.3 Erreur absolue moyenne (MAE) :

c'est une mesure d'erreur populaire pour les problèmes de régression, elle est simplement défini comme la moyenne de la différence absolue entre la sortie prédite et la sortie réelle. L'erreur quadratique est couramment utilisée car elle est indépendante du fait que la prédiction était trop élevée ou trop faible, elle signale simplement que la prédiction était incorrecte. l'équation du MAE est comme suite :

$$MAE = \frac{1}{n}\Sigma_{i=1}^{n}|yp - yr| \tag{3.3}$$

Où

— yp :la valeur prédite

— yr :la valeur réelle de base

— n : le nombre de toutes les valeurs prédites

Plus la valeur du critère (MAE, RSME) est petite, plus le modèle est proche de la valeur réelle.

3.2.8.4 le carré r (R^2) :

est une mesure statistique dajustement pour tester lajustement du modèle,représente la proportion de variance dans le résultat que notre modèle est capable de prédire en fonction de ses caractéristiques .Une valeur de 0 à 1 est considérée comme un pourcentage et plus la valeur est élevée, plus amélioration du modèle l'équation du Rš est comme suit :

$$R^2 = 1 - \Sigma_{i=1}^{n} \frac{(yr - yp)^2}{(yp - yi)^2} \tag{3.4}$$

Où yi :la moyenne des mesures .

3.2.9 Résultat de prédiction

Le résultat obtenu à partir ce modèle est un vecteur de sortie de 12 pas de temps de chaque 5 min à lavance (c'est la prédiction de flux de trafic de l'heure suivante) .

3.3 Conclusion

Dans ce chapitre, nous avons présenté la conception générale et détaillée de la prédiction Hyper-paramétré et qui comprend : l'identification de l'ensemble de données, le découpage et l'analyse du dataset, le pré-traitement et la sélection des caractéristique, la définition du méthode de la recherche par grille et l'apprentissage du modèle jusquau résultat de prédiction. Dans le chapitre suivant, on va présenter l'étude expérimentale du notre étude et on présente aussi l'implémentation de notre système, les outils matériels et logiciels et le langage de développement de modèle. A la fin de chapitre, nous présenterons les résultats obtenus et nous les comparerons pour détermine la valeur optimale du hyper paramètre.

Chapitre 4

Etude expérimentale et résultats

4.1 Introduction

Dans le chapitre précédent, nous avons présenté la conception du système de réglage des hype-paramètres pour la prédition du flux de trafic routier, comme nous l'avons expliqué en détail les différentes étapes de notre système.

Dans ce chapitre, nous allons présenter lenvironnement de travail, le langage de programmation et les outils que nous avons utilisés pour réaliser nos projet. Par la suite, nous expliquerons toutes les expérimentations que nous avons menés ainsi que nous discuterons les résultats obtenus.

4.2 outils et langage de développement

4.2.1 outils et matériel

Nous avons réalisé notre système en utilisant :

— la plateforme de Calcul intensif (HAUT Performance Computing) IBNKHALDOUN du l'université mohamed khider biskra avec processeur : Intel Xeon (R) E5-2660 v3 @ de 2.60 GHz x 20, memoire : 64 GB de RAM, disque : 2 TB HDD, Système d'exploitation : Red Hat Enterprise Linux Server 7.2 Type d'SE : 64 bitsProcesseur : Intel Xeon (R) E5-2660 v3 @ de 2.60 GHz x 20.

FIGURE 4.1 – Hpc IBNKHALDOUN du l'université mohamed khider biskra

— Collaboration Google (google colab)

4.2.2 Outils logiciel

4.2.2.1 Plateform :

— **Anaconda :**

est une distribution gratuite et open-source des langages de programmation Python et R pour le calcul scientifique (science des données, applications d'apprentissage automatique, traitement de données à grande échelle, analyse prédictive, etc.) qu'est développé et maintenu par Anaconda, Inc., qui a été fondée par Peter Wang et Travis Oliphant en 2012. La distribution est livrée avec l'interpréteur Python et divers packages liés à l'apprentissage automatique et à la science des données.

FIGURE 4.2 – Logo du anaconda

4.2.2.2 Langage de programmation :

Python :

est un langage de programmation open source, l'un des langages de programmation les plus intéressants du moment car il est puissant et facile à apprendre. Il dispose de structures de données de haut niveau et permet une approche simple mais efficace de la programmation orientée objet. il est aussi un langage interprété (c'est-à-dire qu'il n'est pas nécessaire de le compiler avant de l'exécuter) . Python est un langage idéal pour l'écriture de scripts de vaçon très simples et le développement rapide d'applications dans de nombreux domaines et sur la plupart des plateformes ,tous ça grâce à ses nombreuses bibliothèques.

FIGURE 4.3 – Logo du python

4.2.2.3 Bibliothèques

Dans notre programme, on a utilisé un ensemble de librairies pour utiliser des fonctions prédéfinies. Ils sont appelées au début de programme.

— **Keras :**

est une bibliothèque open source écrite en python qu'est exécutée sur la plate-forme d'apprentissage automatique TensorFlow et aussi elle est facile à utiliser pour développer et évaluer des modèles dapprentissage profond car Elle permet de créer très facilement des layers pour les réseaux de neurones ou de mettre en place des architectures complexes.

— **TensorFlow :**

est une bibliothèque open source de l'apprentissage automatique , créée par Google (la deuxième génération du système de Google Brain) permettant de développer et dexécuter des applications de l'apprentissage automatique et profond. il sagit dune boîte à outils permettant de résoudre des problèmes mathématiques extrêmement

complexes et le calcul numérique haute performance avec aisance.Son architecture flexible permet de déployer facilement le calcul sur une variété de plateformes (CPU, GPU, TPU).

— **Scikit-learn "Sklearn" :**

est une importante bibliothèque d'outils dédiés au l'apprentissage automatique et à le sciences des données dans l'univers Python.

— **Numpy :**

Le terme NumPy est en fait labréviation de Numerical Python . Il sagit dune bibliothèque Open Source en langage Python. On utilise cet outil pour la programmation scientifique en Python, et notamment pour la programmation en sciences des données sachant que elle propose un grand nombre de routines pour un accès rapide aux ces données . Elle est destinée à manipuler des matrices ou tableaux multidimensionnels (array) ainsi que des fonctions mathématiques opérant sur ces tableaux.

— **Matplotlib :**

Une bibliothèque Python, destinée à tracer et visualiser des données sous formes de graphiques.Elle peut être combinée avec les bibliothèques python de calcul scientifique NumPy et SciPy et elle est notamment utilisée sur des serveurs dapplication web, des shells et des scripts Python.

— **Pandas :**

Le nom Pandas est en fait la contraction du terme Panel Data , désignant les ensembles de données incluant des observations sur de multiples périodes temporelles.
La bibliothèque logicielle open-source Pandas est spécifiquement conçue pour la manipulation et lanalyse de données en langage Python ,elle propose en particulier des structures de données et des opérations de manipulation de tableaux numériques et de séries temporelles pour rendre le travail avec les données plus facile.

4.3 Implémentation

la présentation de l'implémentation des étapes de notre systèm est comme suit :

4.3.1 Identification de l'ensemble de données (dataset) :

Le dataset utilisé dans notre système est une série temporelle qui contient une suite dobservations chiffrées ordonnées dans le temps, et considérer comme une collection déléments variés. (comme il est défini dans le chapitre président) .

Il est sous la forme dun fichier d'extension .csv et il contient 245376 lignes et 6 colonnes (date et temp,débit,température précipitations, densité et vitesse).

La figure suivante représente le dataset

```
Mounted at /content/drive
             datDateTime   Flow  Temp      Rain  Density  Speed
0       01/01/2014 00:00   72.0  5.98  0.001583    16.13   19.9
1       01/01/2014 00:05   27.0  5.89  0.000067    20.63   20.8
2       01/01/2014 00:10   57.0  5.63  0.000000    17.44   26.2
3       01/01/2014 00:15   27.0  5.70  0.000000    23.71   17.0
4       01/01/2014 00:20   39.0  6.03  0.044550    21.02   18.6
...                  ...    ...   ...       ...      ...    ...
245372  01/05/2016 23:40  111.0  4.50  0.000000    16.23   22.3
245373  01/05/2016 23:45  138.0  3.76  0.000000    13.07   21.5
245374  01/05/2016 23:50  168.0  2.90  0.000000    10.66   19.8
245375  01/05/2016 23:55   93.0  2.56  0.000000    13.09   23.0
245376  02/05/2016 00:00   87.0  2.42  0.000000     7.03   28.3

[245377 rows x 6 columns]
```

FIGURE 4.4 – Le Dataset utilisé .

4.3.2 Découpage du dataset :

Pour former notre modèle d'apprentissage, nous avons divisé l'ensemble de données en données d'entraînement et en données de test. Comme nous avons mentionné dans la section 3.2.2 du chapitre précédant on a divisé le dataset on 90% pour l'entraînement c'est-à-dire 220838 observation (de 01/01/2014 à 00 :00 jusqu'à 06/02/2016 à 19 :05) (20% pour le validation) et 10% pour le test c'est-à-dire 24538 observation (de 06/02/2016 à 19 :10 jusqu'à 01/05/2016 à 23 :55) .

voir les figures suivantes :

	datDateTime	Flow	Temp	Rain	Density	Speed
0	01/01/2014 00:00	72.0	5.98	0.001583	16.13	19.9
1	01/01/2014 00:05	27.0	5.89	0.000067	20.63	20.8
2	01/01/2014 00:10	57.0	5.63	0.000000	17.44	26.2
3	01/01/2014 00:15	27.0	5.70	0.000000	23.71	17.0
4	01/01/2014 00:20	39.0	6.03	0.044550	21.02	18.6
...	...	...	...	...	...	...
220833	06/02/2016 18:45	177.0	7.94	0.016250	67.69	19.5
220834	06/02/2016 18:50	222.0	8.39	0.204583	38.54	22.6
220835	06/02/2016 18:55	243.0	8.02	2.156117	46.10	21.0
220836	06/02/2016 19:00	252.0	9.09	0.110717	50.16	18.5
220837	06/02/2016 19:05	297.0	10.30	0.144933	45.23	21.8

[220838 rows x 6 columns]

FIGURE 4.5 – Les données d'entraînement.

	datDateTime	Flow	Temp	Rain	Density	Speed
220838	06/02/2016 19:10	189.0	9.76	0.208600	53.59	18.4
220839	06/02/2016 19:15	219.0	9.83	0.061583	50.91	21.9
220840	06/02/2016 19:20	201.0	10.93	0.004100	52.11	23.2
220841	06/02/2016 19:25	219.0	10.77	0.001550	72.79	18.3
220842	06/02/2016 19:30	195.0	10.84	0.000233	44.07	23.1
...	...	...	...	...	...	...
245371	01/05/2016 23:35	108.0	4.96	0.000000	22.27	20.7
245372	01/05/2016 23:40	111.0	4.50	0.000000	16.23	22.3
245373	01/05/2016 23:45	138.0	3.76	0.000000	13.07	21.5
245374	01/05/2016 23:50	168.0	2.90	0.000000	10.66	19.8
245375	01/05/2016 23:55	93.0	2.56	0.000000	13.09	23.0

[24538 rows x 6 columns]

FIGURE 4.6 – Les données de test.

Ce découpage à été réalisé par la fonction **Split-dataset()**.

voici le code :

```
#Diviser l'ensemble de données (dataset) en ensembles d'entraînement et de test
def split_dataset(data):
    # divisé le dataset  en ensembles d'entraînement / test
    train, test = data[:-24538, :], data[-24538:, :]
```

FIGURE 4.7 – Le découpage du dataset.

En suite on a besoin de structuré les donné en heure car on a faire la prédiction de chaque heure . Pour cela nous avons le structuré avec 12 valeurs de 5 min avec la fonction **Split()**.

Sachant que cette fonction est prise comme une entrée des données de notre dataset et les organise en heure standard et en la renvoyant dans la sortie.

voici le code :

```
# restructurer le dataset  en d'heures standard (12 valeur de 5 min )
train = np.array(np.split(train, len(train) / 12))
test = np.array(np.split(test, len(test) / 12))
return train,
```

FIGURE 4.8 – Structuré le dataset.

Aprés ces étapes notre dataset est prét pour l'étape de pré-traitement .

4.3.3 Pré-traitement et sélection des caractéristiques

4.3.3.1 Pré-traitment du dataset :

— **Standardisation :**

Dans ce étape, nous changeons la forme de l'ensemble de données en données de petites valeurs (entre 0 et 1) pour éviter les problèmes du réseau neuronal durant lentraînement, comme le ralentissement dapprentissage.

Ce changement se fait par les instructions suivantes :

```
# standardiser (train/test)
mu = np.mean(train) #la Moyenne de donnes de l'entrainement
sig = np.std(train) #l'ecart type  de donnes de l'entrainement

dataTrainStandardized = (train - mu) / sig #datatrain standardiser
```

FIGURE 4.9 – Standardisation de données d'entraînement .

Le résultat obtenu par la standardisation du données d'entraînement est comme suit :

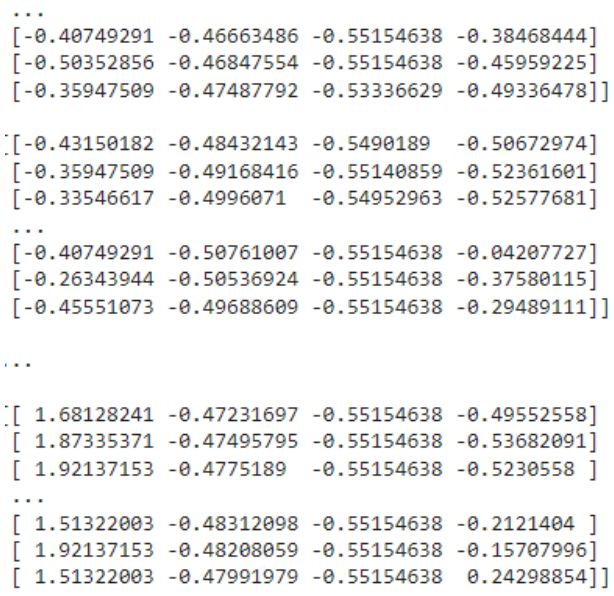

FIGURE 4.10 – Partie de données dentraînement standardisé

— **Conversion des données de séries temporelles multivariées aux séries temporelles supervisée :**

Comme nous l'avons mentionné précédemment, pour travailler avec une séries temporelles multivariées, nous devons la convertir en séries temporelles supervisée afin de faciliter le travail et de donner de meilleurs résultats. Et parce que nous travaillons avec une dataset temporelles et multivariées, nous devons utiliser la méthode de la fenêtre glissante "Sliding window " à l'aide de la fonction **to-supervised()** pour faire la conversion.

la figure suivante représenta la fonction **to-supervised()**.

```python
# convertir les données historiques  en entrées et sorties par 'Sliding window'
def to_supervised(train, n_input, n_out=12):
    train = np.array(train)
    # aplatir les données (flatten)
    data = train.reshape((train.shape[0] * train.shape[1], train.shape[2]))
    X, y = list(), list()
    in_start = 0
    # parcourir toute l'historique , une heure après l'autre
    for _ in range(len(data)):
        #  définir la fin de la séquence d'entrée
        in_end = in_start + n_input
        out_end = in_end + n_out #la fin de la séquence de sortie
        # determiner les données d'entrée et de sortie en cas (out_end <= len(data))
        if out_end <= len(data):
            # x_input = data[in_start:in_end, 0]
            # x_input = x_input.reshape((len(x_input), 1))
            X.append(data[in_start:in_end, :]) #prend la sequence avec tout les features comme en entrée
            # X.append(x_input)
            y.append(data[in_end:out_end, 0]) #la sortie specifie a le flux de trafic

        # se déplacer d'un pas de temps pour parcourir tout l'historique
        in_start += 1
    return np.array(X), np.array(y)
```

FIGURE 4.11 – La fonction de conversion de séries temporelle multivariée au série temporelle supervisée

Où le format de données traitées est transformé aux formes X = [220813, 12, 5] et y=[220813, 12].

4.3.3.2 Sélection des caractéristiques :

Comme nous l'avons mentionné précédemment, le dataset contient un ensemble de colonnes que nous considérons comme des caractéristiques (débit, température,précipitations,densité et le vitesse) que nous choisirons de manuellement dans chaque expérimentation et qui contribuent le plus à notre variable de prédiction ou la sortie .
Nous faisons ce choix à chaque fois pour évaluer notre modèle.

4.3.4 La méthode de la recherche par grille (Grid Search)

Grid Search est la méthode de réglage ou d'optimiser des hyperparamètres afin de déterminer les valeurs optimales pour un modèle, comme nous avons déja mentionné précédemment.

Le travail de cette méthode dépend de la définition de l'espace de recherche qu'est défini comme un dictionnaire où les noms sont les hyperparamètres du modèle et les valeurs sont des valeurs discrètes sont les valeurs proposées.
où prendre en entrée pour faire la recherche :

- estimator : l'instance de modèle.
- space : l'objet dictionnaire qui contient les valeur proposés pour le batch size .
- scoring : le métrique d'évaluation " l'erreur absolue moyenne"(MAE).
- n-jobs : le nombre de processus que vous souhaitez exécuter en parallèle pour cette tâche
- cv : le nombre de validations croisées (on utilise 10 plis et trois répétitions).

  Une fois la recherche est effectuée ,elle appelant la fonction fit () pour faire l'entraînement sur un partie du s données d'entraînement et l'autre partie elle l'utilise pour évaluer la performance d'hyperparamètre dans le modèle à l'aide de la validation croisée.

  L'implémentation du méthode est illustrée dans la partie du code suivant :

```
 # gridsearch:
model = KerasClassifier(build_fn=model,verbose=0) #definir un instance du modèle
cv = RepeatedKFold(n_splits=10, n_repeats=3, random_state=1) #spécifier le nombre de splits et repeats
#definir l'espace de recherche (dictionnaire)
space = dict()
space['batch_size'] = [32,100,128,256,420,500,512,1024,1500,2048]

search = GridSearchCV(model, space, scoring='neg_mean_absolute_error'
                          , n_jobs=1, cv=cv)#faire la recherche
result = search.fit(train, trainy) # executer la recherche sur l'ensemble du valeurs prposés
```

FIGURE 4.12 – La méthode du grid search

Après la fin de cette étape, cette méthode donne le meilleur résultat suggéré qui donne la meilleure performance.

4.3.5 Entraînement du modèle :

L'entraînement du modèle est considéré comme l'étape la plus importante pour améliorer les performances .Pour cela avant entraîner notre modèle nous devons d'abord construire le modèle (ce qui était fait auparavant dans [15] et puis nous déterminerons lhyperparamètre optimisé par la méthode du recherche par grille (Grid Search) qui peut nous aider à obtenir les meilleurs résultats.

4.3.5.1 Construire le modèle ConvLSTM2d-Bi

Ce modèle a été crée par la fonction **"Build-model()"**.

```
# Construire  le modèle
def build_model(train, n_steps, n_length, n_input):
```

FIGURE 4.13 – Construire du modèle

Voici la partie du code qui implémente le modèle ConvLSTM2D-Bi.[15]

```
# le modèle
model = Sequential()
#couche 1
model.add(ConvLSTM2D(filters=128, kernel_size=(1, 4), activation='relu', return_sequences=True,
                     input_shape=(n_steps, 1, n_length, n_features)))
model.add(Dropout(0.2))
#couche 2
model.add(ConvLSTM2D(filters=128, kernel_size=(1, 4), activation='relu', return_sequences=True))
model.add(Dropout(0.2))
#couche3
model.add(ConvLSTM2D(filters=128, kernel_size=(1, 4), activation='relu', return_sequences=True))
model.add(Dropout(0.2))
model.add(Flatten())
model.add(RepeatVector(n_outputs))
#couche4
model.add(Bidirectional(LSTM(512, activation='relu', return_sequences=True),
                        input_shape=(n_steps, 1, n_length, n_features)))
model.add(Dropout(0.2))
#couche 5
model.add(TimeDistributed(Dense(100, activation='relu')))
#couche 6
model.add(TimeDistributed(Dense(1)))
model.compile(loss='mse', optimizer= tf.keras.optimizers.Adam(lr=0.0001))
es = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=15)
checkpointer = ModelCheckpoint(filepath="best_weightsconvBi.hdf5",
                               monitor='val_loss',
                               verbose=2,
                               save_best_only=True)
callbacks_list = [checkpointer, es]  # early
model.summary()
plot_model(model, to_file='model_plotconv.png', show_shapes=True, show_layer_names=True)
```

FIGURE 4.14 – L'implémentation du modèle proposer dans [15]

4.3.5.2 Les hyperparamètres :

Les hyperparamètres affectent grandement les performances du modèle, nous avons donc choisi les hyperparamètres suivants :

TABLEAU 4.1 – Les hyperparamètres utilisé

Hyperparamètres	Valeurs
époques	200.
Taux d'apprentissage	0.0001
Nombre de couche caché	6
Nombre dunités cachées (bi-LSTM, Dense, Dense)	512,100,1
Pas de temp (Timestamps)	12

Et concernant **la taille du lot "Batch size"** , c'est l'hyperparamètre que nous allons optimiser avec la méthode **Grid search**. Pour cela, nous avons proposé un ensemble de valeurs qui sont considérées comme les entrées de la méthode, puis nous les testons pour voir l'étendue de leur impact et amélioration sur le performance du modèle grâce aux valeurs du métriques (RMSE,MAE,R^2) puis choisir le meilleur résultat qui donne les meilleures performances (ces expériences seront faites dans la section 5.4). les valeur proposé sont illustré dans le tableau suivant :

TABLEAU 4.2 – Les valeur proposés pour la taille du lot

la taille du lot (Batch size)
32
100
128
256
420
500
512
1024
1500
2048

4.3.5.3 Entraînement

pour lentraînement du modèle, il utilise :

— une couche dentrée qui a été fixée à (None x 1 x 1 x 12 x 5).

— trois couches cachées dans lencodeur .

— 128 filtres de longueur 4 pour chaque couche de convolution.

— une couche cachée dans le décodeur

— une unité cachée de 512

— une couche dense entièrement connectées de 100 unités.

La fonction dactivation "Relu" (unité linéaire rectifiée) a été utilisée pour activer toutes les couches de modèle et la fonction de perte de régression "le MSE" (lerreur quadratique moyenne) a été utilisée pour calculer la somme des carrés de la distance entre la valeur prédite et la valeur réelle. l'utilisation du "Adam" comme optimiseur, qui a démontré sa bonne généralité et sa capacité de convergence rapide dans les modèles

d'apprentissage profond avec un taux d'apprentissage (lr = 0.0001) pour compiler le modèle . l' utilisation du "Modelcheckpoint" pour sauvegarder le modèle ou ses poids dans un fichier .hdf5 sil y a une diminution de la valeur de loss sur l'ensemble de données de validation, sinon à travers la stratégie darrêt précoce (early stopping) l'entraînement est terminé après la valeur de patience (le paramètre de patience est de 15).

La figure ci-dessus présente le partie de code qui exprime l'entraînement.

```
es = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=15)
checkpointer = ModelCheckpoint(filepath="best_weightsconvBi.hdf5",
                               monitor='val_loss',
                               verbose=2,
                               save_best_only=True)
callbacks_list = [checkpointer, es]  # early
model.summary()
plot_model(model, to_file='model_plotconv.png', show_shapes=True, show_layer_names=True)
# entrainement le modèle
history = model.fit(train_x, train_y, epochs=epochs, callbacks=callbacks_list,
                    batch_size=batch_size,
                    verbose=verbose,
                    validation_split=0.2)
plot_result(history) #dessin le graphe de perte

model.load_weights('best_weightsconvBi_hyperparametré.hdf5')
model.save('modelconvBi_hyperparametré.h5')
```

FIGURE 4.15 – L'entraînement du modèle.

4.3.6 Test et évaluation du modèle :

4.3.6.1 Test du modèle :

A ce stade, le modèle est testé à travers les données de test. Ce test est réalisé en deux étapes :

— La première étape consiste à télécharger le modèle déjà entraîné(sous forme d'un fichier .h) et ses poids (sous forme d'un fichier .hdf5) ce téléchargement est réalisé par les instructions suivantes :

```
model = load_model (" modelconvBi-hyperparamétré .h5",custom_objects =None
                    , compile = True )#telechargé le modéle
model . load_weights (" best_weightsconvBi-hyperparamétré . hdf5 ")#telechargé les poids du modéle
model.save('modelconvBi-hyperparamétré.h5')#sauvgarder le modéle entrainé

#retourne le modéle .
 return model
```

FIGURE 4.16 – Téléchargement du modèle et ces poids

— La deuxième étape consiste à faire la prédiction de flux de trafic sur les données de test avec la fonction **forecast()** .(voir le partie du code suivant)

```
model = load_model (" modelconvBi-hyperparamétré .h5",custom_objects =None
                    , compile = True )#telechargé le modéle
model . load_weights (" best_weightsconvBi-hyperparamétré . hdf5 ")#telechargé les poids du modéle
model.save('modelconvBi-hyperparamétré.h5')#sauvgarder le modéle entrainé

#retourne le modéle .
 return model
```

FIGURE 4.17 – La fonction **forecast()**

4.3.6.2 L'évaluation du modèle :

Aprés l'étape du test du modèle on va évalué la performance du modèle avec l'approche la validation croisée k-fold (Walk-Forward validation)(qui mentionné dans la section 4.2.8).On implémente ce dernier avec la fonction "evaluate-model()"

```python
# évaluer le modèle
def evaluate_model(train, test, n_steps, n_length, n_input):
    # entrainement du modèle
    model = build_model(train, n_steps, n_length, n_input)
    # historique est liste de données d'heureur
    history = [x for x in train]
    # walk-forward validation sur chaque heure
    predictions = list()
    for i in range(len(test)):
        # prédire l'heure
        yhat_sequence = forecast(model, history, n_steps, n_length, n_input)
        # stocker les prédictions
        predictions.append(yhat_sequence)
        # obtenir une observation réelle et l'ajouter à l'historique pour prédire l'heure suivante.
        history.append(test[i, :])
    # évaluer les prédictions
    predictions = np.array(predictions)
    score, score1, score2, scores, scores1, scores2 = evaluate_forecasts(test[:, :, 0], predictions)

    return score, score1, score2, scores, scores1, scores2
```

FIGURE 4.18 – La fonction d'évaluation du modèle.

ce dernier nécessite à appeler trois fonctions :

— **build-model()** : pour construire le modèle des données dentraînement.

```python
model = build_model(train, n_steps, n_length, n_input)
```

FIGURE 4.19 – L'appel du La fonction **"build-model()"**

— **forecast()** :pour faire des prédictions à partir lhistorique pour chaque pas de temps de 5 min pour une heure.

```python
# prédire l'heure
yhat_sequence = forecast(model, history, n_steps, n_length, n_input)
```

FIGURE 4.20 – L'appel du La fonction**"forecast()"**

— **evaluate-forecast()** : pour faire l'évaluation de chaque pas est prédit à partir d'une entrée (les donnes réelles) et puis calculer le score total et partiel de RMSE, MAE et R2 de prédiction de chaque pas et les afficher comme une sortie de la fonction **evaluate-model()** pour évaluer la performance de modèle.

```python
# évaluer une ou plusieurs prédictions horaire par rapport aux valeurs prédites
def evaluate_forecasts(actual, predicted):
    scores = list()
    scores1 = list()
    scores2 = list()
    # calculer un score RMSE,MAE et R2 pour chaque échantillon
    for i in range(actual.shape[1]):
        # calcule MSE
        mse = mean_squared_error(actual[:, i], predicted[:, i])
        # calcule RMSE
        rmse = np.sqrt(mse)
        # calcule MAE
        mae = mean_absolute_error(actual[:, i], predicted[:, i])
        # calcule R2
        r2 = r2_score(actual[:, i], predicted[:, i])
        # sauvgarder les scores de chaque metrique
        scores.append(rmse)
        scores1.append(mae)
        scores2.append(r2)
    # calculer le RMSE,MAE et R2 totale
    print(sum(scores2))
    s = 0
    a = 0
    for row in range(actual.shape[0]):
        for col in range(actual.shape[1]):
            s += (actual[row, col] - predicted[row, col]) ** 2
            a += abs(actual[row, col] - predicted[row, col])
    score = np.sqrt(s / (actual.shape[0] * actual.shape[1]))#RMSE totale
    score1 = (a / (actual.shape[0] * actual.shape[1]))#MAE totale
    score2 = (sum(scores2) / 12)# R2 totale
    return score, score1, score2, scores, scores1, scores2
```

FIGURE 4.21 – La fonction d'évaluation de performance **"evaluate-forecast()"**

Une fois que nous avons l'évaluation d'un modèle, nous pouvons résumer les performances. La fonction ci-dessous nommée **resume-scores()** affichera les performances d'un modèle sur un seul ligne.

```
# affiche les resultats d'un modèle sur une seule ligne pour chaque metrique d'evaluation
def summarize_scores(name, names, nom, score, score1, score2, scores, scores1, scores2):
    s_scores = ', '.join(['%.3f' % s for s in scores])
    s_scores1 = ', '.join(['%.3f' % s for s in scores1])
    s_scores2 = ', '.join(['%.3f' % s for s in scores2])

    print('%s: [%.3f] %s ' % (name, score, s_scores))
    print('%s: [%.3f] %s ' % (names, score1, s_scores1))
    print('%s: [%.3f] %s ' % (nom, score2, s_scores2))
#############################################################################
```

FIGURE 4.22 – La fonction d'affichage les métriques "**resume-scores()**"

4.4 Expérimentations et discussion les résultats obtenus :

4.4.1 Expérimentations

Dans notre travail, nous effectuerons dix expériences de prédiction différentes sur les variables d'entrée. Où, dans chaque expérimentation, nous prendrons une valeur proposé du l'hyperparamètre " la taille du lot" (Batch size) et nous verrons les résultats des métriques RMSE, MAE, R^2 obtenus afin de les comparer et de choisi la meilleure valeur qui donne la meilleure performance de prédiction. Le tableau suivant présente les résultats obtenus pour chaque valeur proposée.

TABLEAU 4.3 – Les résultats obtenus

Le valeur du batch size	RMSE	MAE	R^2
32	0.398	0.291	0.873
100	0.399	0.290	0.873
128	0.393	0.287	0.877
256	0.398	0.290	0.874
420	0.394	0.286	0.876
500	0.397	0.288	0.874
512	0.399	0.290	0.873
1024	0.400	0.292	0.873
1500	0.396	0.288	0.875
2048	0.398	0.290	0.874

4.4.1.1 Expérimentation 1 :

Dans cette expérimentation on prend **le batch size=32** avec les autres hyperparametres qui nous avons déjà mentionné avec un nombre de filtre =128. Pour calculé le flux de trafic pour une durée de 5 à 60 minutes cest à dire les résultats est composer de 12 valeurs équivalentes à un prédiction dun heure suivante.

4.4.1.1.1 Résultats obtenus :

Tous d'abord, on an la courbe de la variation de lerreur (Loss) au cours de lentraînement et de validation du la modèle avec la 1er valeur du batch size. La figure suivante est présenté la courbe de la variation de l'erreur (Loss).

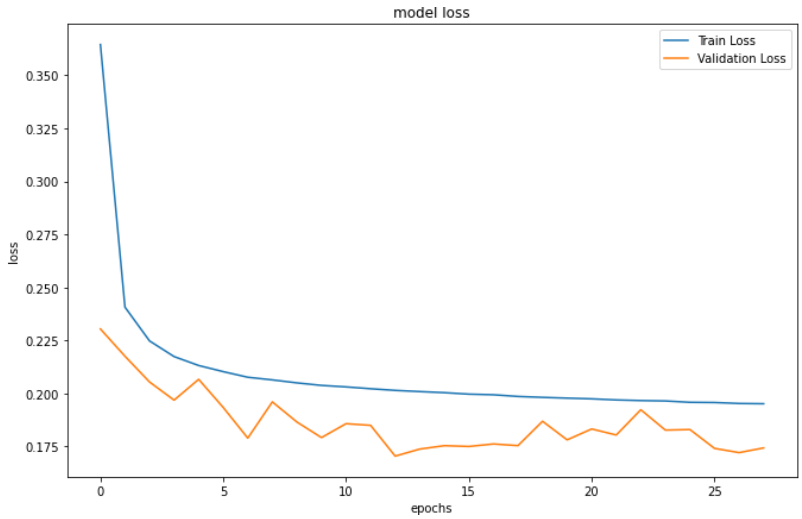

FIGURE 4.23 – la variation de lerreur (Loss) de l'expérimentation 1

Nous avons vu que la courbe de lentraînement ne cesse de décroître, mais la courbe validation continue de monter et de descendre jusqu'à l'époque numéro 28 .

Nous avons calculé les métrique d'evaluation RMSE ,MAE, R^2 pour chaque pas de temps de 5 à 60 min.Et le résultat d'exécution est illustré dans la figure suivante :

```
RMSE: [0.398] 0.322, 0.354, 0.356, 0.367, 0.383, 0.399, 0.404, 0.409, 0.427, 0.430, 0.447, 0.459
MAE: [0.291] 0.238, 0.260, 0.262, 0.268, 0.279, 0.293, 0.292, 0.298, 0.313, 0.313, 0.333, 0.338
R2: [0.873] 0.912, 0.897, 0.896, 0.891, 0.884, 0.875, 0.868, 0.867, 0.858, 0.854, 0.841, 0.838
[0.398484] [0.2905691] 0.8733108063032872
```

FIGURE 4.24 – le résultat du métriques dévaluation de l'expérimentation 1

Nous avons également représenté ces résultats dans les courbes suivantes :

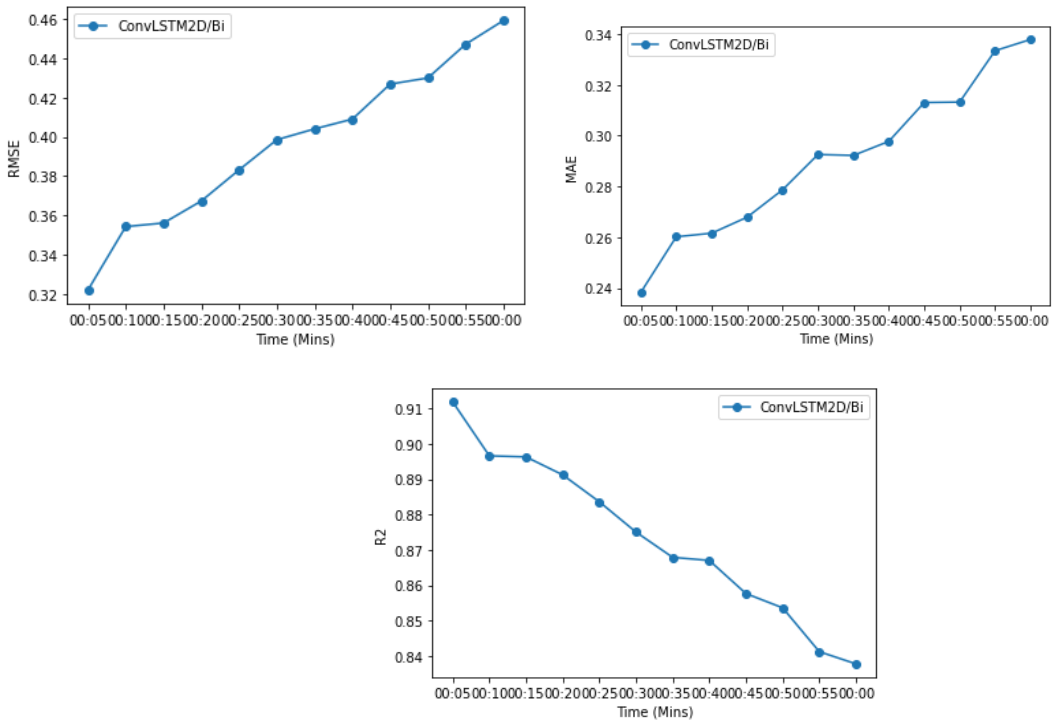

FIGURE 4.25 – les résultats du RMSE,MAE,R^2 de l'exprémentation 1

4.4.1.2 Expérimentation2 :

Dans cette expérimentation, on prend **le batch size=100**.

4.4.1.2.1 Résultats obtenus :

la courbe de la variation de lerreur (Loss) au cours de lentraînement et de validation du la modèle est représenté dans la figure 4.26

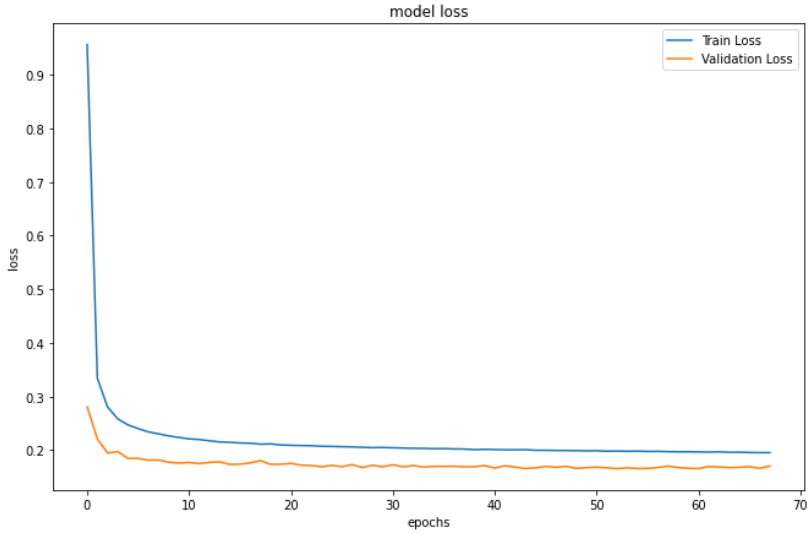

FIGURE 4.26 – la variation de lerreur de l'expérimentation 2

On remarque que dans cette expérience quand on prend la taille du lot =100 la courbe de lentraînement est diminuées et puis elle se stabilise depuis lépoque numéro 35 et la courbe de validation elle se diminué et connaît de très légères hausses .

Le résultats du métriques d'évaluation de cette expérimentation est représenté dans la figure 4.27

```
RMSE: [0.399] 0.316, 0.342, 0.347, 0.358, 0.374, 0.393, 0.410, 0.412, 0.430, 0.441, 0.458, 0.468
MAE: [0.290] 0.234, 0.253, 0.257, 0.263, 0.273, 0.289, 0.295, 0.298, 0.314, 0.320, 0.338, 0.342
R2: [0.873] 0.915, 0.903, 0.901, 0.897, 0.889, 0.878, 0.864, 0.865, 0.855, 0.846, 0.833, 0.832
[0.39853597] [0.28969562] 0.8733474147365975
```

FIGURE 4.27 – le résultat du métriques dévaluation de l'expérimentation 2

les courbes du RMSE,MAE etR^2 sont illustré ci-dessous

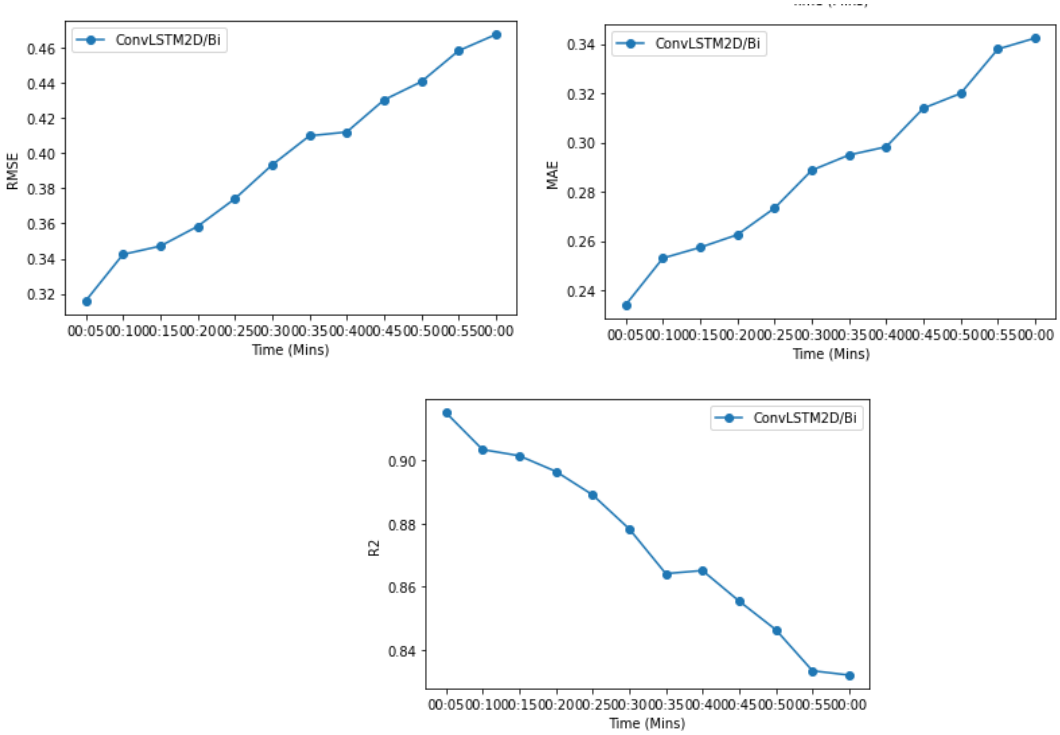

FIGURE 4.28 – les résultats du RMSE,MAE,R^2 de l'expérimentation 2

4.4.1.3 Expérimentation 3 :

On prend **le batch size=128**.

4.4.1.3.1 Résultats obtenus :

La figure 4.29 représente la courbe de la variation de lerreur

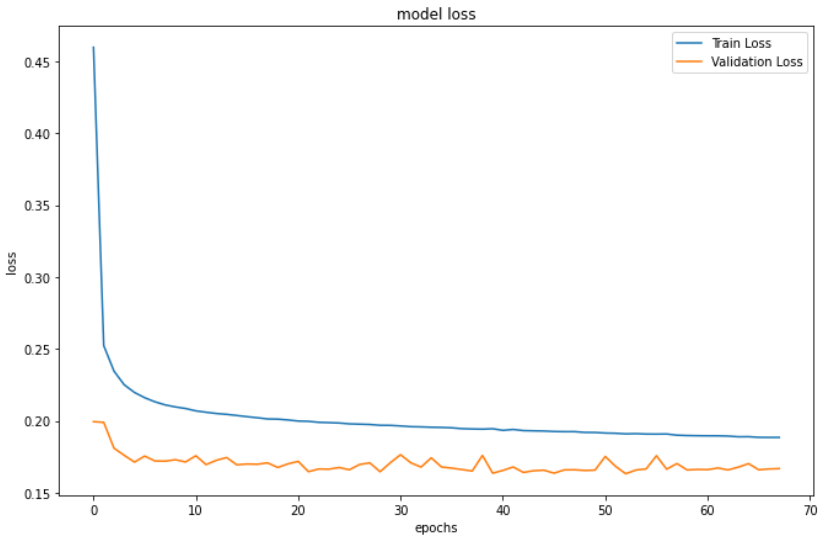

FIGURE 4.29 – la variation de lerreur de l'expérimentation 3

à travers la courbe la variation de lerreur de l'expérimentation 3, nous constatons que la courbe lentraînement diminue alors qu'elle continue de diminuer jusqu'à la fin. Quant à la courbe de validation, elle connaît de légères hausses.

La figure 4.30 représente le résultat du métriques pour cette expérimentation ,et 4.31 les courbes de ces métriques.

```
RMSE: [0.393] 0.325, 0.344, 0.345, 0.358, 0.373, 0.392, 0.403, 0.403, 0.423, 0.429, 0.444, 0.454
MAE: [0.287] 0.241, 0.256, 0.257, 0.262, 0.272, 0.288, 0.291, 0.294, 0.309, 0.313, 0.330, 0.334
R2: [0.877] 0.910, 0.902, 0.902, 0.897, 0.890, 0.879, 0.869, 0.871, 0.860, 0.854, 0.843, 0.841
[0.39321068] [0.28724185] 0.8766383564162491
```

FIGURE 4.30 – le résultat du métriques dévaluation de l'expérimentation 3

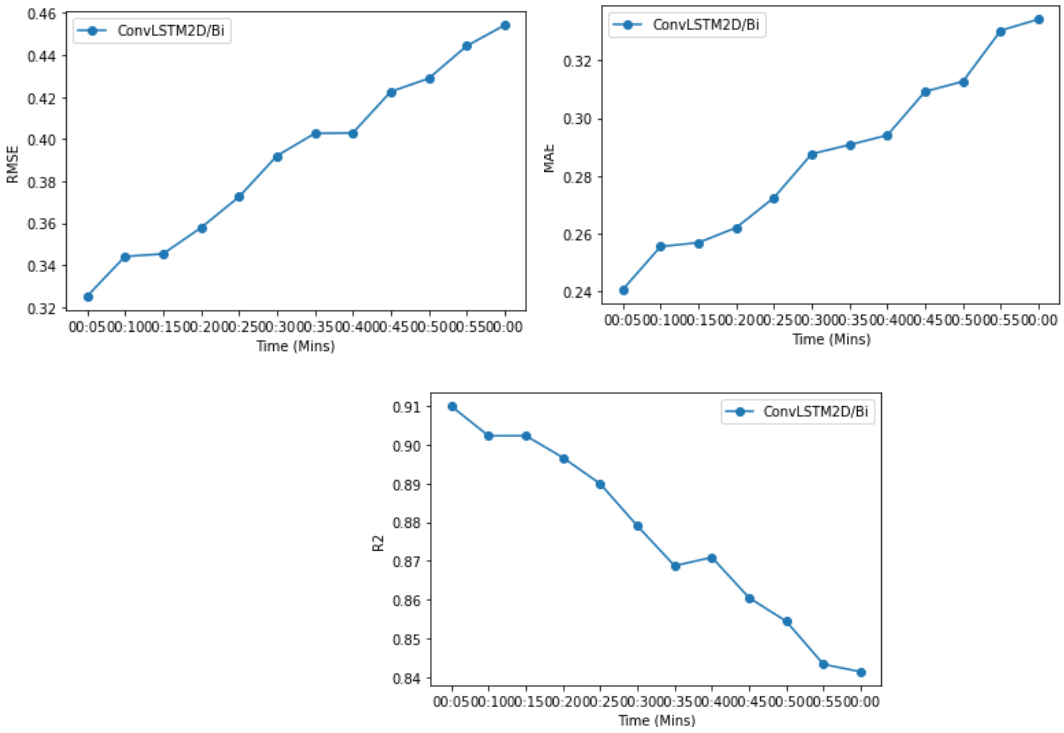

FIGURE 4.31 – les résultats du RMSE, MAE, R^2 de l'expérimentation 3

4.4.1.4 Expérimentation 4 :

la taille du lot= **256**.

4.4.1.4.1 Résultats obtenus :

Les mêmes figures sont affichées au niveau de chaque expérimentation, nous avons présenté les figures ci-dessous correspondantes à cette expérimentation.

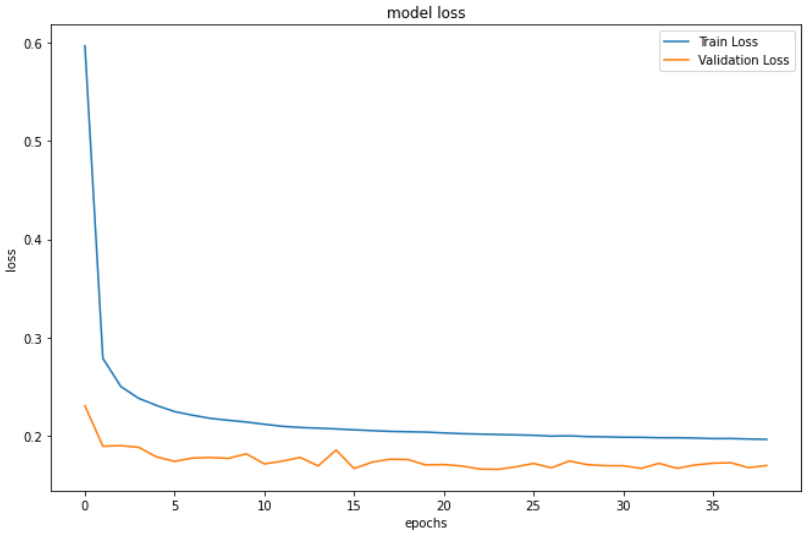

FIGURE 4.32 – la variation de lerreur de l'expérimentation 4

```
RMSE: [0.398] 0.325, 0.348, 0.349, 0.360, 0.375, 0.395, 0.408, 0.409, 0.427, 0.436, 0.451, 0.467
MAE: [0.290] 0.240, 0.258, 0.259, 0.264, 0.274, 0.290, 0.293, 0.296, 0.311, 0.316, 0.334, 0.342
R2: [0.874] 0.910, 0.900, 0.900, 0.896, 0.889, 0.877, 0.865, 0.867, 0.858, 0.850, 0.838, 0.832
[0.39810055] [0.28987378] 0.8735812405066786
```

FIGURE 4.33 – le résultat du métriques dévaluation de l'expérimentation 4

les courbes du RMSE, MAE et R^2 sont illustré ci-dessous

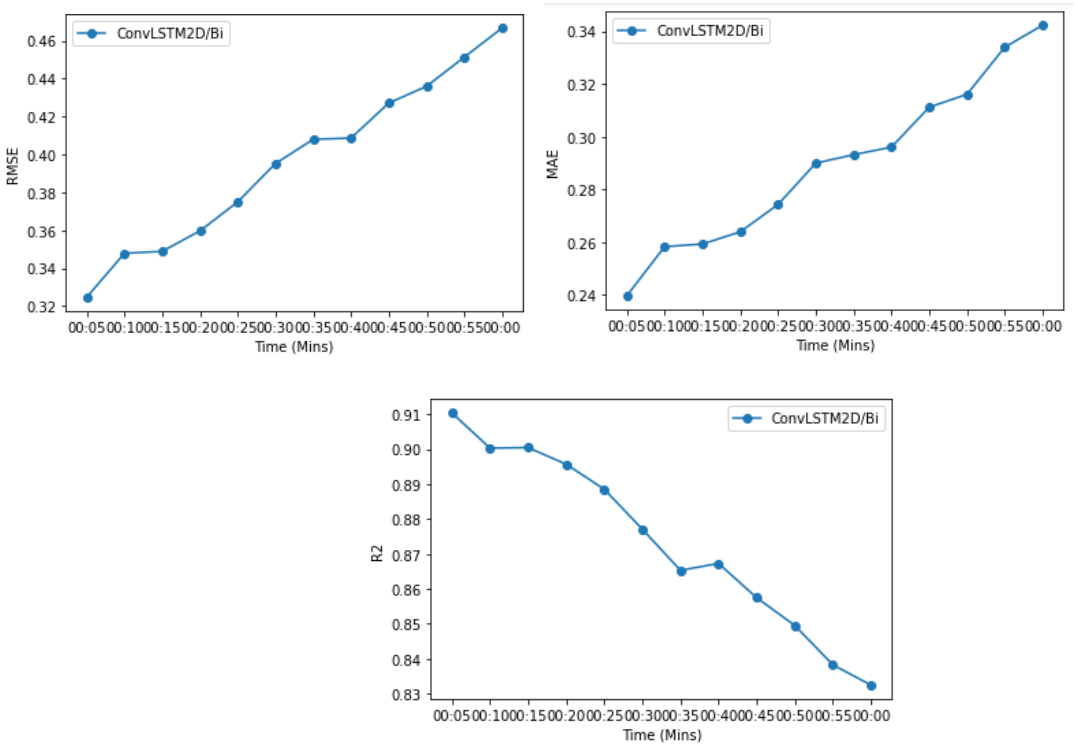

FIGURE 4.34 – les résultats du RMSE, MAE, R^2 de l'expérimentation 4

4.4.1.5 Expérimentation 5 :

On prend **le batch size=420**.

4.4.1.5.1 Résultats obtenus :

La figure 4.35 représente la courbe de la variation de lerreur

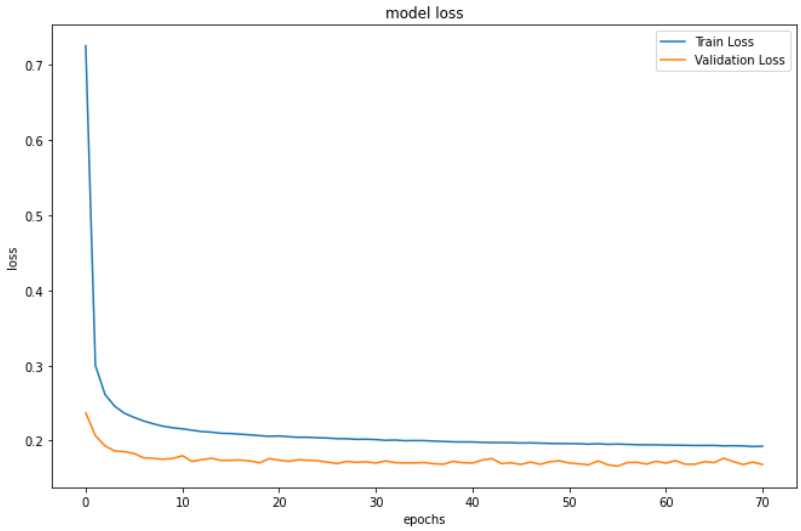

FIGURE 4.35 – la variation de lerreur de l'expérimentation 5

La figure 4.36 représente le résultats du métriques pour cette expérimentation ,et 4.37 les courbes de ces métriques.

```
RMSE: [0.394] 0.322, 0.345, 0.347, 0.359, 0.373, 0.391, 0.402, 0.403, 0.423, 0.428, 0.448, 0.457
MAE: [0.286] 0.238, 0.253, 0.255, 0.262, 0.272, 0.287, 0.290, 0.294, 0.309, 0.311, 0.331, 0.334
R2: [0.876] 0.912, 0.902, 0.901, 0.896, 0.890, 0.880, 0.869, 0.871, 0.860, 0.855, 0.841, 0.839
[0.39370537] [0.28633758] 0.8763415982311823
```

FIGURE 4.36 – le résultat du métriques dévaluation de l'expérimentation 5

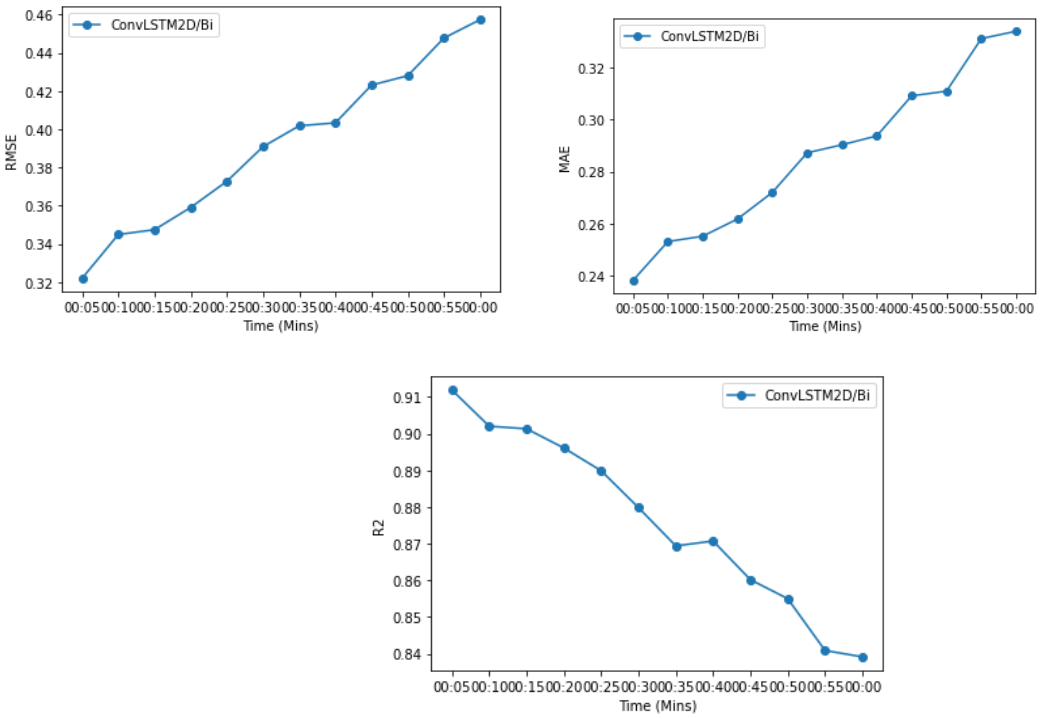

FIGURE 4.37 – les résultats du RMSE,MAE,R^2de l'expérimentation 5

4.4.1.6 Expérimentation 6 :

Dans cette expérimentation on prend **le batch size=500** avec les autres hyperparametres qui nous avons déjà mentionnés.

4.4.1.6.1 Résultats obtenus :

Les figures 4.38,4.39 et 4.40 représentent respectivement le courbe de la variation de l'erreur (model loss), le résultat d'exécution des métriques d'évaluation et les courbes de ces métriques de cette expérimentation.

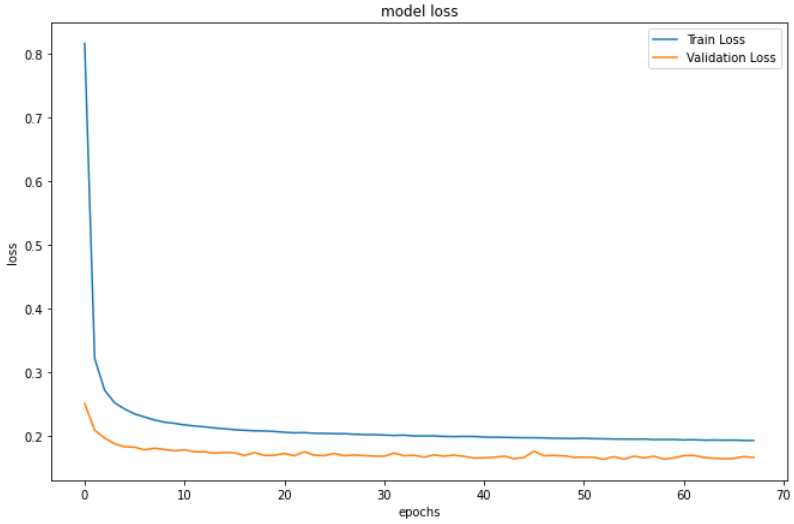

FIGURE 4.38 – la variation de lerreur(model loss) de l'expérimentation 6

```
RMSE: [0.397] 0.318, 0.342, 0.347, 0.358, 0.374, 0.393, 0.405, 0.408, 0.428, 0.436, 0.454, 0.465
MAE: [0.288] 0.234, 0.252, 0.257, 0.262, 0.273, 0.288, 0.291, 0.296, 0.311, 0.316, 0.334, 0.341
R2: [0.874] 0.914, 0.904, 0.902, 0.897, 0.889, 0.878, 0.867, 0.868, 0.857, 0.849, 0.836, 0.834
[0.396723] [0.2878621] 0.8744855622829587
```

FIGURE 4.39 – le résultat du métriques dévaluation de l'expérimentation 6

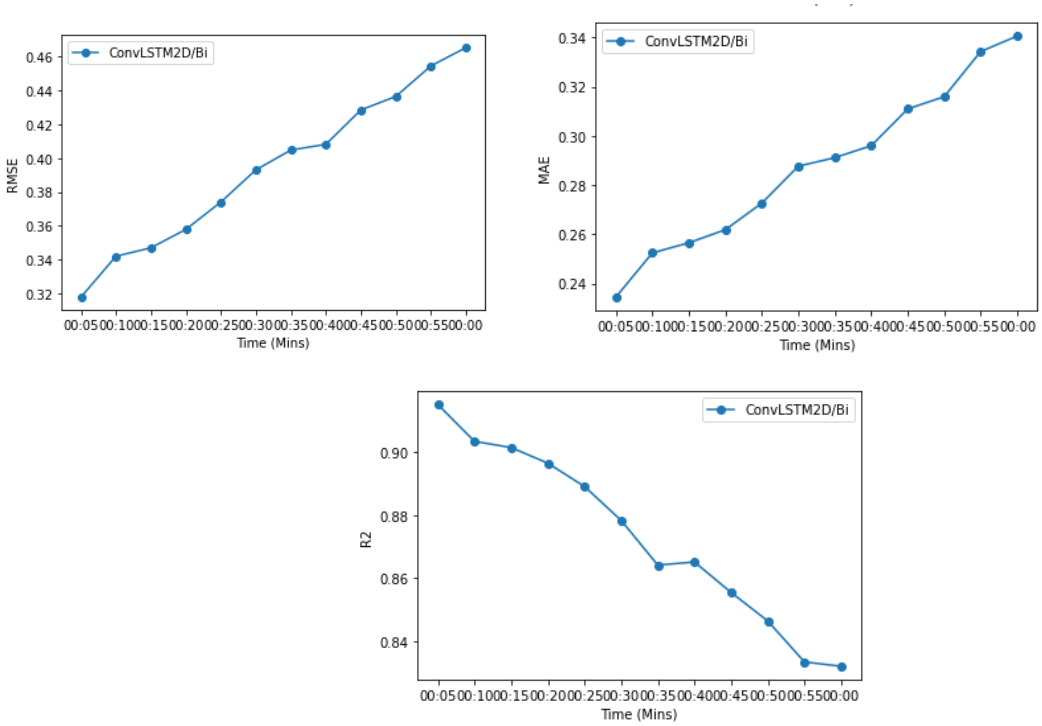

FIGURE 4.40 – les résultats du RMSE,MAE,R^2 de l'expérimentation 6

4.4.1.7 Expérimentation 7 :

la taille du lot= **512**.

4.4.1.7.1 Résultats obtenus :

Les mêmes figures sont affichées au niveau de chaque expérimentation, nous avons présenté les figures ci-dessous correspondantes à cette expérimentation.

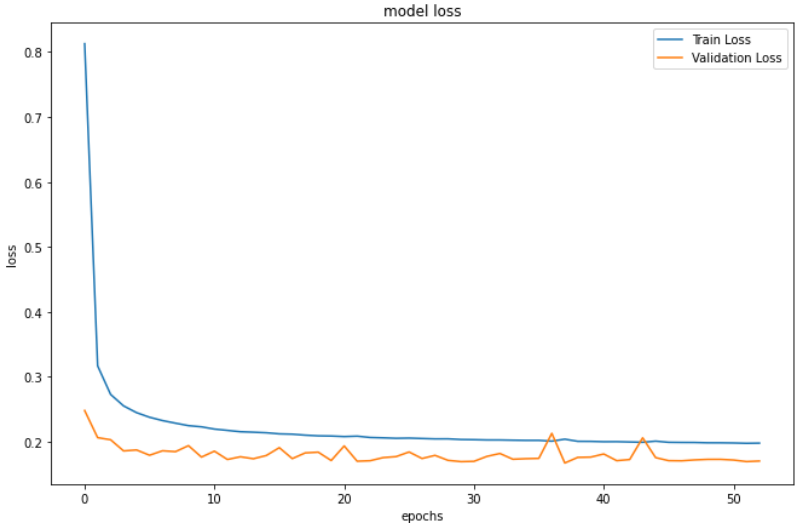

FIGURE 4.41 – la variation de lerreur de l'expérimentation 7

Nous remarquons dans cette expérience une diminution et une stabilité de la courbe d'entraînement comme dans les expériences précédentes, mais nous remarquons une augmentation de la perte dans la courbe de validation , surtout dans l'époque numéro 38.

```
RMSE: [0.399] 0.319, 0.345, 0.349, 0.359, 0.375, 0.395, 0.409, 0.412, 0.430, 0.440, 0.458, 0.469
MAE: [0.290] 0.235, 0.255, 0.258, 0.262, 0.275, 0.290, 0.294, 0.297, 0.314, 0.319, 0.336, 0.341
R2: [0.873] 0.914, 0.902, 0.901, 0.896, 0.888, 0.877, 0.864, 0.865, 0.855, 0.847, 0.834, 0.831
[0.39922246] [0.28964978] 0.872901472795793
```

FIGURE 4.42 – le résultat du métriques dévaluation de l'expérimentation 7

les courbes du RMSE,MAE etR^2 sont illustré ci-dessous

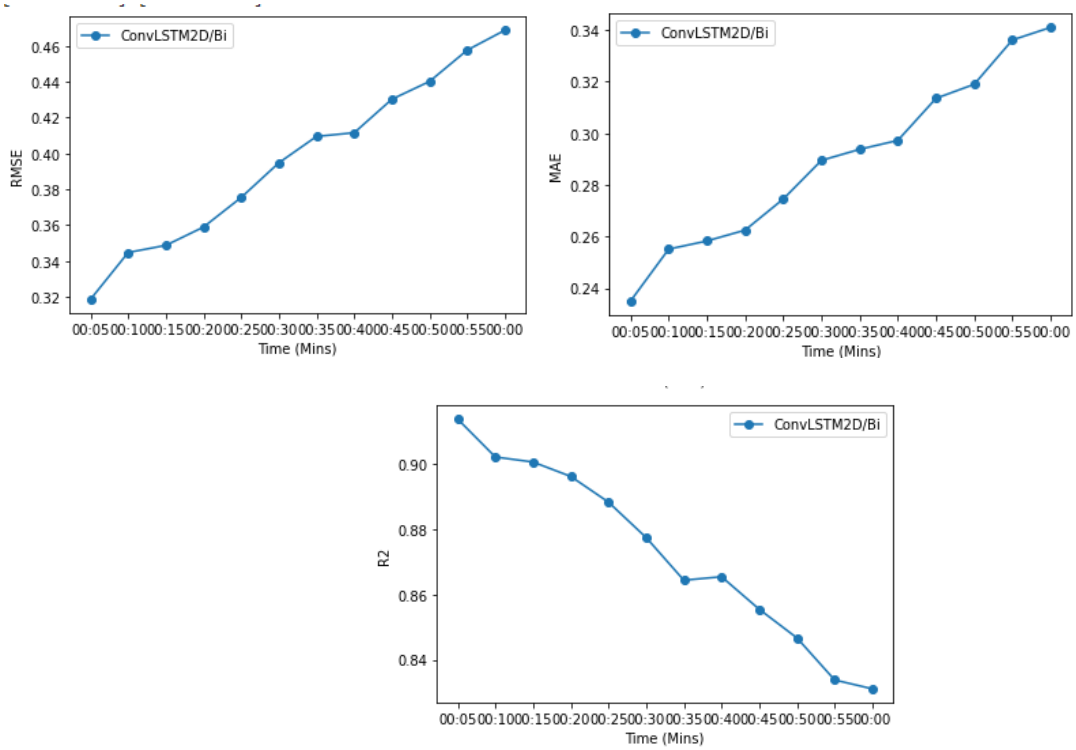

FIGURE 4.43 – les résultats du RMSE, MAE, R^2 de l'expérimentation 7

4.4.1.8 Expérimentation 8 :

dans la 8ème expérimentation on utilise **le batch size=1024**.

4.4.1.8.1 Résultats obtenus :

La figure suivante représente la courbe de model loss de cette expérimentation.

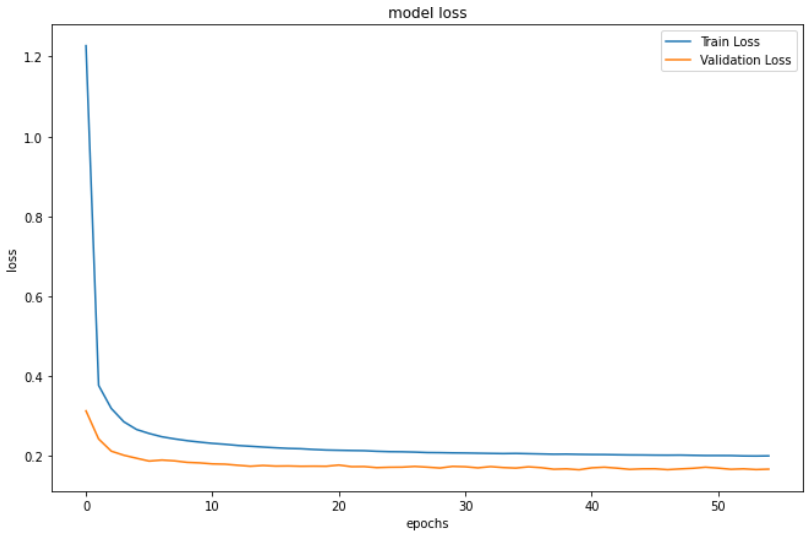

FIGURE 4.44 – la variation de lerreur de l'expérimentation 8

On observe une décroissance puis une constante à la fois pour la courbe d'entrainement et pour la courbe de validation .

La figure suivante représente le résultats du métriques pour cette exprémentation .

```
RMSE: [0.400] 0.319, 0.345, 0.351, 0.360, 0.377, 0.396, 0.408, 0.412, 0.430, 0.439, 0.455, 0.471
MAE: [0.292] 0.237, 0.257, 0.262, 0.265, 0.276, 0.291, 0.296, 0.300, 0.314, 0.320, 0.338, 0.346
R2: [0.873] 0.913, 0.902, 0.900, 0.895, 0.887, 0.877, 0.865, 0.865, 0.856, 0.847, 0.836, 0.830
[0.39952478] [0.2917861] 0.8727094271529042
```

FIGURE 4.45 – le résultat du métriques dévaluation de l'expérimentation 8

Et la figure suivante représente les courbes de les métriques d'erreur RMSE,MAE et R^2.

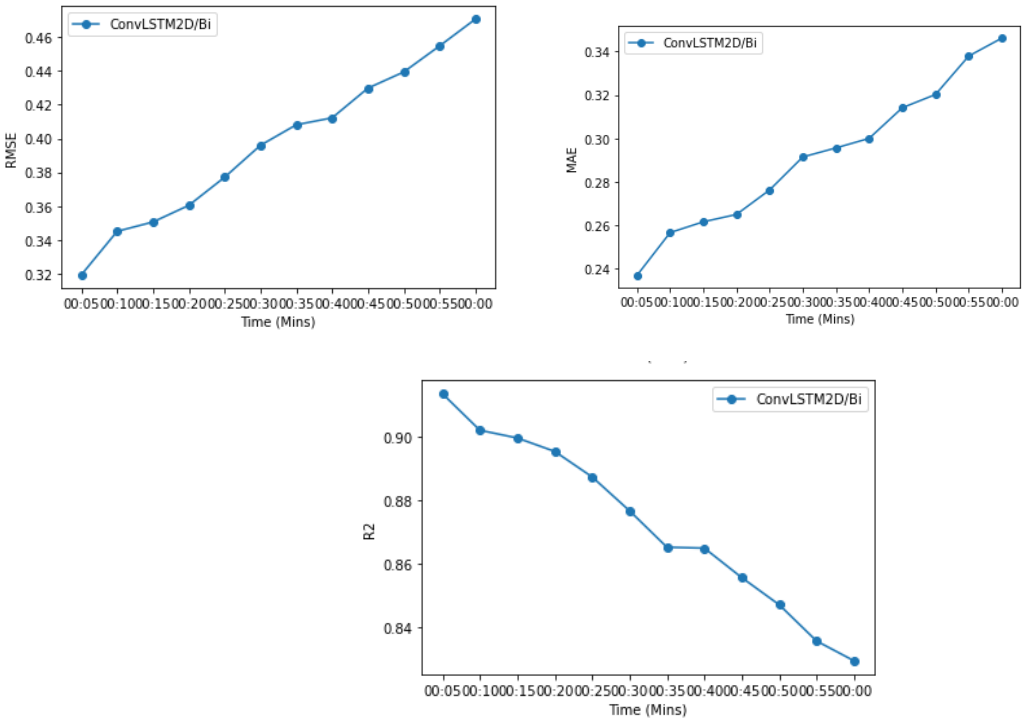

FIGURE 4.46 – les résultats du RMSE,MAE,R^2 de l'expérimentation 8

4.4.1.9 Expérimentation 9 :

dans le 9éme expérimentation on utilise **le batch size=1500**.

4.4.1.9.1 Résultats obtenus :

La figure suivante représente la courbe de model loss de cette expérimentation.

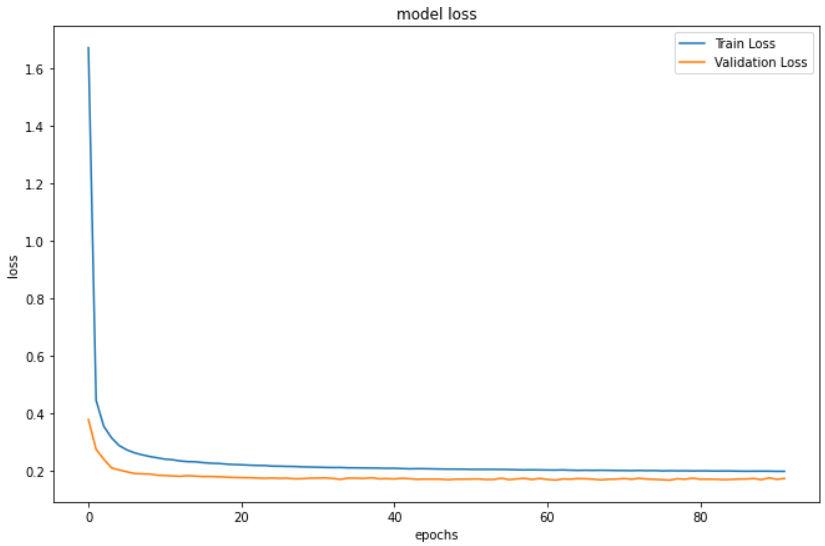

FIGURE 4.47 – la variation de lerreur de l'expérimentation 9

On observe une décroissance puis une constante pour la courbe d'entraînement et la courbe de validation à partir de l'époque numéro 20.
La figure suivante représente le résultat du métriques pour cette expérimentation .

```
RMSE: [0.396] 0.320, 0.345, 0.346, 0.358, 0.373, 0.392, 0.406, 0.408, 0.426, 0.434, 0.451, 0.464
MAE: [0.288] 0.236, 0.254, 0.255, 0.261, 0.272, 0.288, 0.292, 0.296, 0.311, 0.315, 0.334, 0.342
R2: [0.875] 0.913, 0.902, 0.902, 0.897, 0.890, 0.879, 0.867, 0.868, 0.859, 0.851, 0.839, 0.835
[0.39590907] [0.28810894] 0.8749788261084239
```

FIGURE 4.48 – le résultat du métriques dévaluation de l'expérimentation 9

Et la figure suivante représente les courbes de les métriques d'erreur RMSE, MAE et R^2.

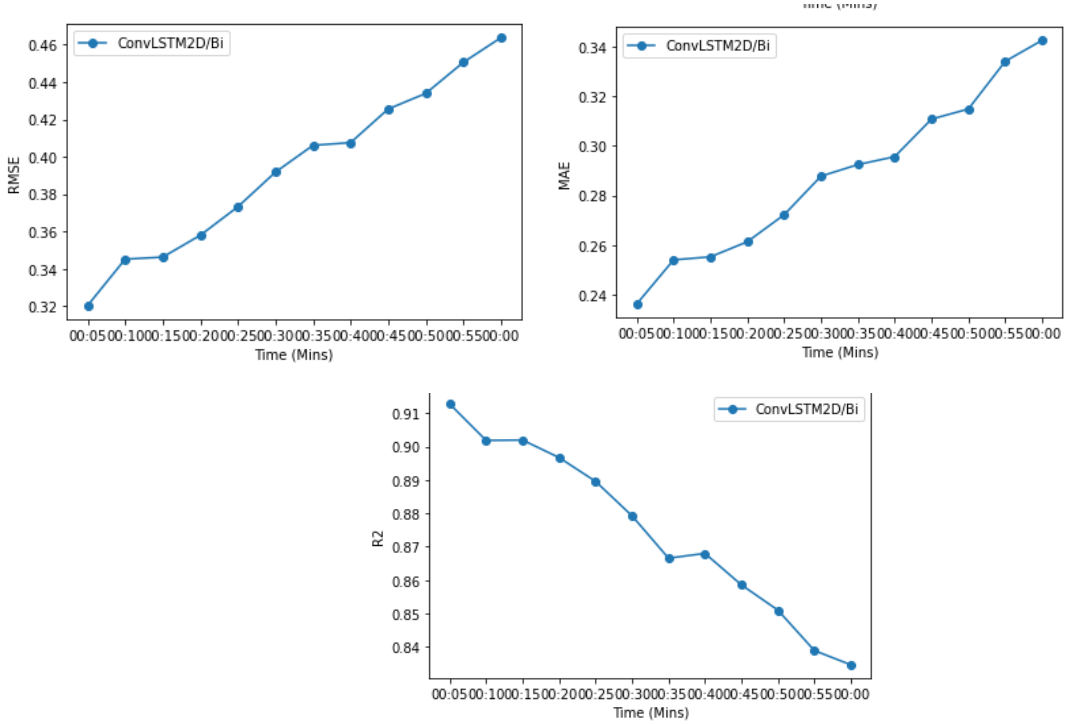

FIGURE 4.49 – les résultats du RMSE,MAE,R^2 de l'expérimentation 9

4.4.1.10 Expérimentation 10 :

Dans cette dernière expérience, nous travaillerons avec **une taille du lot =2048** et en gardant les mêmes autres hyperparamètres. les résultats obtenu dans cette expérimentation sont illustrés ci-dessous.

4.4.1.10.1 Résultats obtenus :

La figure suivante représente la courbe de la variation d'erreur (model loss) d'entraînement et de validation de cette expérimentation.

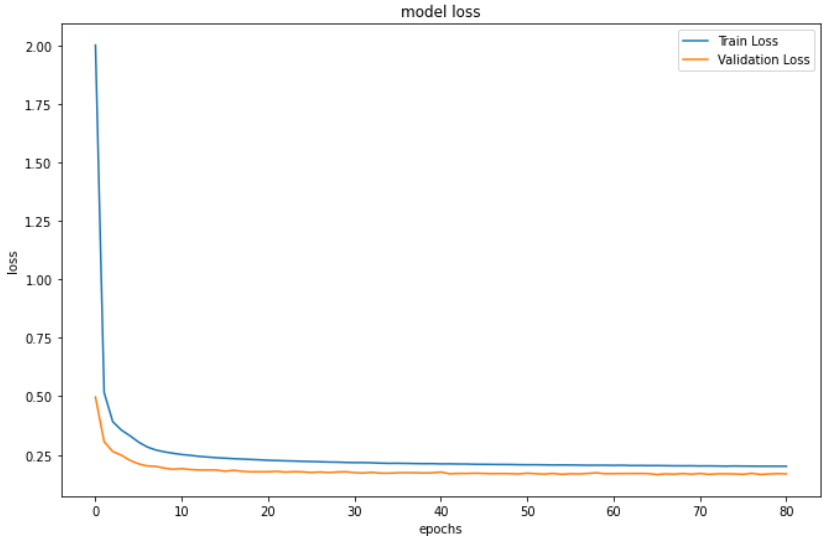

FIGURE 4.50 – la variation de lerreur de l'expérimentation 10

En ce qui concerne la courbe de perte, nous remarquons les mêmes changements qui se sont produits dans l'expérience précédente

La figure suivante représente le résultat du métriques d'évaluation .

```
RMSE: [0.398] 0.321, 0.346, 0.351, 0.361, 0.378, 0.396, 0.409, 0.410, 0.428, 0.435, 0.451, 0.463
MAE: [0.290] 0.237, 0.256, 0.260, 0.264, 0.275, 0.290, 0.295, 0.297, 0.312, 0.316, 0.334, 0.339
R2: [0.874] 0.912, 0.901, 0.899, 0.895, 0.887, 0.876, 0.865, 0.866, 0.857, 0.850, 0.839, 0.835
[0.39812532] [0.28950745] 0.8735699353999992
```

FIGURE 4.51 – le résultat du métriques dévaluation de l'expérimentation 10

Et la figure suivante représente les courbes de les métriques d'évaluation RMSE, MAE et R^2.

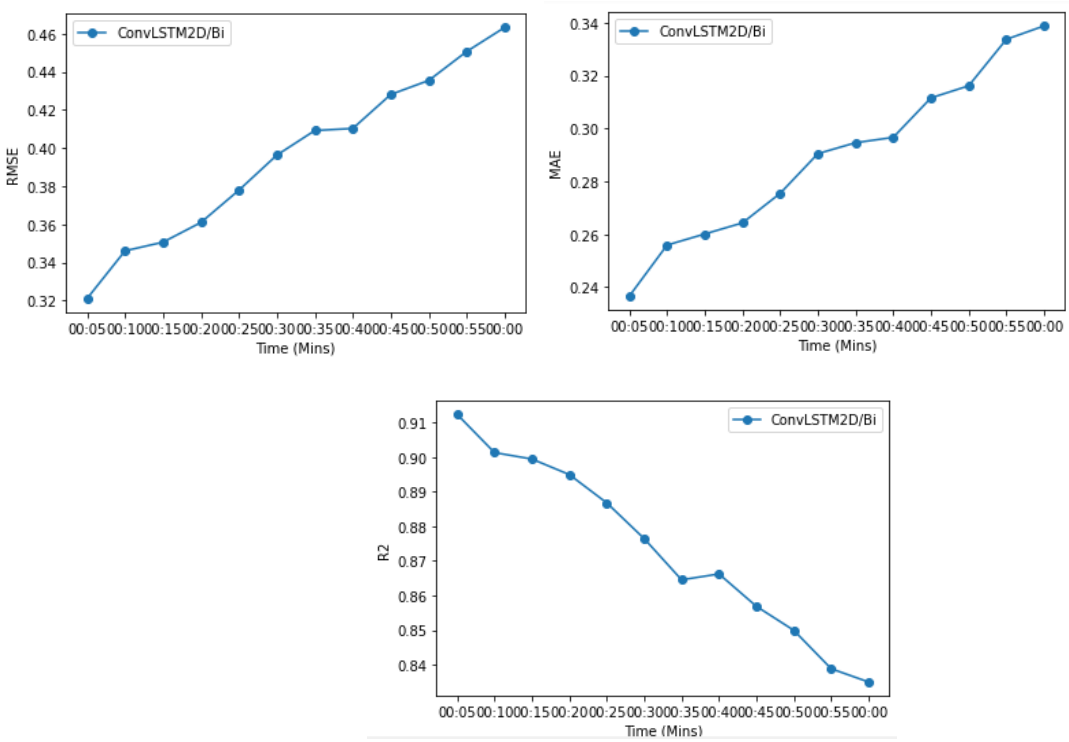

FIGURE 4.52 – les résultats du RMSE,MAE,R^2 de l'expérimentation 10

4.4.2 Discussion des résultats obtenus :

Suite à nos expérimentations qui stipulent de changer à chaque fois la valeur du l'hyper-paramètre "la taille du lot (Batch size)" et de calculer les valeurs du métriques d'évaluation RMSE et MAE et R^2 de chaque expérimentation .

Nous avons obtenu les valeurs métriques présentées par ordre de meilleures performances dans le tableau ci-dessous .

TABLEAU 4.4 – L'ordre des résultats obtenus

Les valeurs du batch size	RMSE	MAE	R^2
128	0.393	0.287	0.877
420	0.394	0.286	0.876
1500	0.396	0.288	0.875
500	0.397	0.288	0.874
256	0.398	0.290	0.874
2048	0.398	0.290	0.874
32	0.398	0.291	0.873
100	0.399	0.290	0.873
512	0.399	0.290	0.873
1024	0.400	0.292	0.873

La figure suivante représente les résultats obtenus de la métrique RMSE pour chaque valeur du batch size .

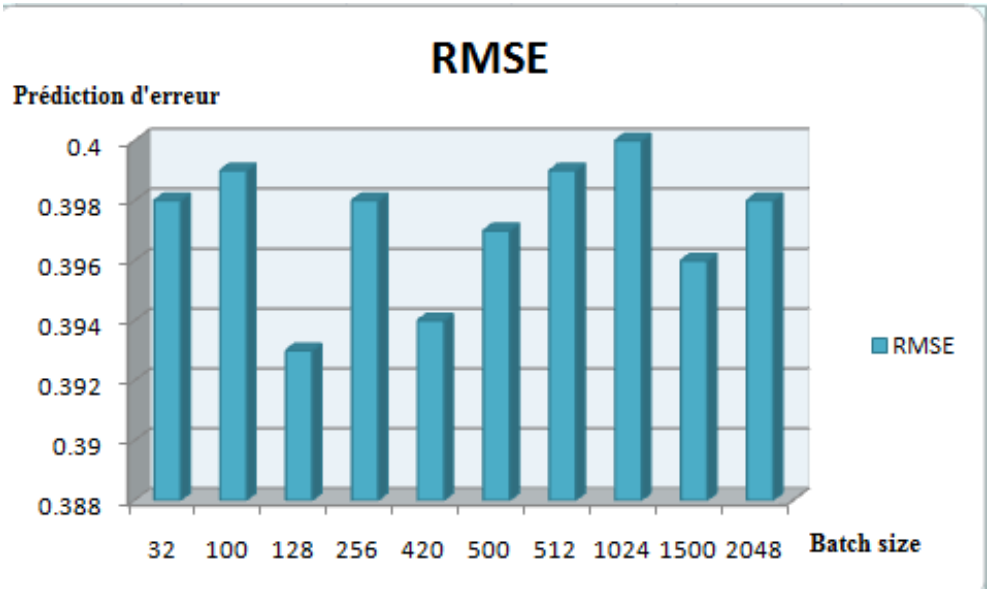

FIGURE 4.53 – Les résultats obtenus du métrique RMSE

On note la grande convergence des valeurs obtenues dans les expériences, mais la plus petite valeur de la métrique a été produite à 128 où elle est égale à 0,393.

En ce qui concerne la métrique MAE, nous présentons la figure suivante qui montre les différences entre les valeurs proposés du batch size .

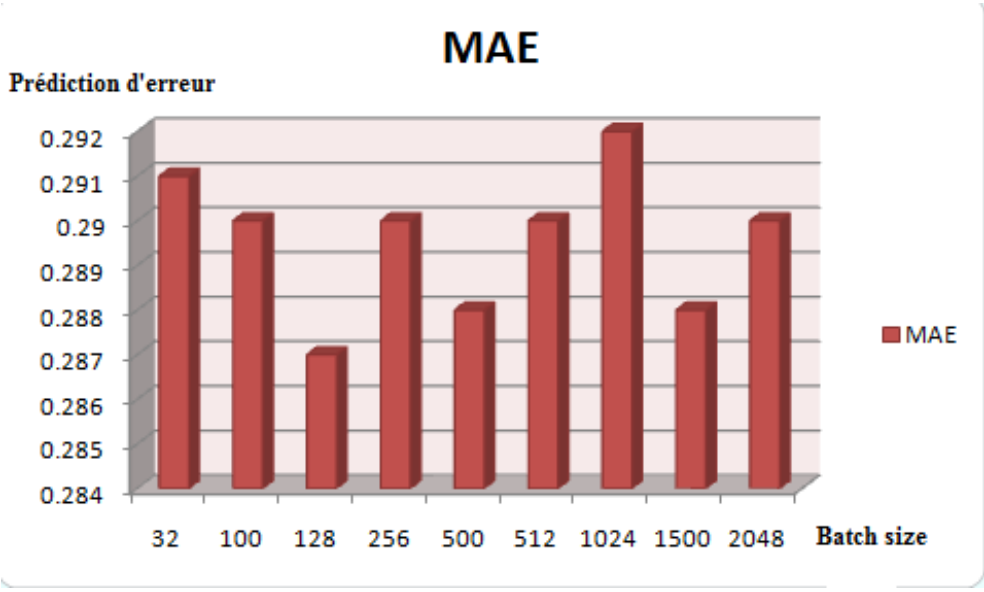

FIGURE 4.54 – Les résultats obtenus du métrique MAE

La plus petite valeur de cette métrique est produite par la valeur de batch de 128. Cela indique que la différence absolue moyenne entre la sortie prédit et la sortie réelle est très petite, ce qui se traduit par une bonne prédiction.

La figure suivante montre la variance des valeurs de la métrique R^2

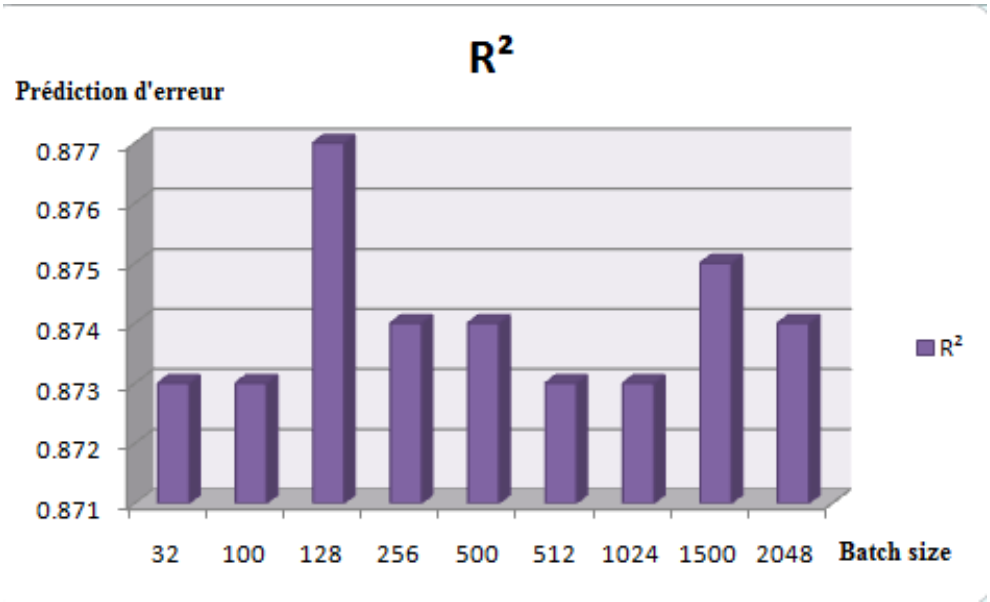

FIGURE 4.55 – Les résultats obtenus du métrique R^2

Nous remarquons clairement que la valeur de la métrique R^2 a augmenté à la valeur 128 (0.877 proche de 1), ce qui indique une augmentation de la possibilité d'ajuster la prédiction du modèle aux valeurs réelles.

Et en fin , après ce que nous avons atteint, nous déterminons que la meilleure valeur du batch size qui donne les meilleures performances pour la prédiction, est de **128 (expérimentation 3)** en raison des bons résultats qu'il a donnés.

4.5 Conclusion

Ce dernier chapitre consiste à présenter le côté d'implémentation du notre projet.
Premièrement ,nous avons présenté dans ce chapitre les outils materièl et logiciels de la langage de développement utilisé pour implémenter notre système. Et puis nous avons exposé limplémentation du système avec ces différentes étapes :l'identificationde l'ensemble de données, découpage du dataset , pré-traitement et sélection des caractéristiques , la méthode de la recherche par grille (Grid Search) qui suit par l'entraînement du modèle pour être prêt à tester et évaluer.
Et puis faire des expérimentations et nous avons présenté les résultats obtenus, et à la fin nous avons discuté de ces résultats avec une explication de la meilleure performance obtenue.

Conclusion générale

La sécurité routière est l'un des sujets qui attirent le plus l'attention des chercheurs qui cherchent à réduire les accidents , la circulation et la congestion du trafic.Par conséquent, le système de transport intelligente (ITS) a été développé, ce qui rend les routes plus intelligentes, plus efficaces et mieux gérées. L'un des facteurs de ce système qui assure la sécurité routière est la prédiction de flux de trafic sur les routes, car il joue un rôle important dans ce système en anticipant la variabilité des flux de trafic dans un futur proche.
Pour une prédiction précise de flux de trafic, nous avons travaillé dans ce mémoire sur le réglage ou l'optimisation d'un hyperparametre en changeant les valeurs de taille du lot (Batch size) pour réaliser une meilleure performance du prédiction à l'aide du modèle ConvLSTM2D-Bi proposé précédemment . Ce modèle utilise un réseau de neurones récurrents à une mémoire convolutive à long-court terme (ConvLSTM2D)comme un encodeur afin dextraire les informations spatio-temporelles sur le flux du trafic et un LSTM bidirectionnel (Bi-LSTM) comme un décodeur pour analyser les données historiques de flux de trafic pour obtenir la caractéristique de périodicité de flux de trafic.
Où dans ce travail, nous avons faire dix expérimentations pour tester l'effet de ce hyperparamètre sur les performances de prédiction. Dans chaque expérimentation, nous avons utilisé une valeur proposée pour le batch size , puis on calcule les métriques d'évaluation "RMSE, MAE et R^2".Et après avoir obtenu les résultats, nous avons fait une comparaison sur ces résultats, et nous avons trouvé que la meilleure valeur pour la taille du lot est 128, car elle a donné les meilleurs résultats dans chacun des RMSE ,MAE et R^2 ce qui signifie qu'elle donne les meilleures performances de prédiction par rapport à d'autres valeurs.

Cette recherche peut être améliorée en :

— L'optimisation d'un ensemble des hyperparamètres avec d'autres méthodes telles que les méthodes bio-inspirées comme l'algorithme génétique.

— Une mise en marche et une installation realiste est aussi à envisager.

Ces pointes sont des perspectives pour notre travail.

Bibliographie

[1] crashmap crashmap. https ://www.crashmap.co.uk/Search/. Accessed : 07-05-2022.

[2] Machine learning mastery. https ://machinelearningmastery.com/types-of-learning-in-machine-learning/. Accessed : 19-03-2022.

[3] Walaa Alajali, Wanlei Zhou, and Sheng Wen. Traffic flow prediction for road intersection safety. In *2018 IEEE SmartWorld, Ubiquitous Intelligence Computing, Advanced Trusted Computing, Scalable Computing Communications, Cloud Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, pages 812–820, 2018.

[4] Fabio Arena, Giovanni Pau, and Alessandro Severino. A review on ieee 802.11 p for intelligent transportation systems. *Journal of Sensor and Actuator Networks*, 9(2) :22, 2020.

[5] Mariette Awad and Rahul Khanna. *Efficient learning machines : theories, concepts, and applications for engineers and system designers*. Springer nature, 2015.

[6] Khireddine Benaissa, Salim Bitam, and Abdelhamid Mellouk. Bsm-data reuse model based on in-vehicular computing. *Applied Sciences*, 10(16) :5452, 2020.

[7] Qifang Bi, Katherine E Goodman, Joshua Kaminsky, and Justin Lessler. What is machine learning ? a primer for the epidemiologist. *American journal of epidemiology*, 188(12) :2222–2239, 2019.

[8] Azzedine Boukerche, Yanjie Tao, and Peng Sun. Artificial intelligence-based vehicular traffic flow prediction methods for supporting intelligent transportation systems. *Computer Networks*, 182 :107484, 2020.

[9] Walid Bouksani. *Gestion de la protection de la vie privée dans les réseaux véhiculaires (VANET)*. PhD thesis, Université du Québec à Trois-Rivières, 2017.

[10] Aniekan Essien. *TAG-F : A Traffic Predictive Analytics Guidance Framework*. The University of Manchester (United Kingdom), 2019.

[11] Aniekan Essien, Ilias Petrounias, Pedro Sampaio, and Sandra Sampaio. A deep-learning model for urban traffic flow prediction with traffic events mined from twitter. *World Wide Web*, 24(4) :1345–1368, 2021.

[12] Shuai Gao, Yuefei Huang, Shuo Zhang, Jingcheng Han, Guangqian Wang, Meixin Zhang, and Qingsheng Lin. Short-term runoff prediction with gru and lstm networks without requiring time step optimization during sample generation. *Journal of Hydrology*, 589 :125188, 2020.

[13] Basharat Hussain, Muhammad Khalil Afzal, Shafiq Ahmad, and Almetwally M Mostafa. Intelligent traffic flow prediction using optimized gru model. *IEEE Access*, 9 :100736–100746, 2021.

[14] Banar Fareed Ibrahim, Mehmet Toycan, and Hiwa Abdulkarim Mawlood. A comprehensive survey on vanet broadcast protocols. In *2020 International Conference on Computation, Automation and Knowledge Management (ICCAKM)*, pages 298–302, 2020.

[15] SAADI IBTISSAM. La prédiction de flux de trafic routier par une méthode dapprentissage profond. Université du Mohamed Khider Biskra 2020,2021.

[16] Youssef Lahrouni. *Détection d'intrusions dans les réseaux véhiculaires sans fil*. PhD thesis, Université du Québec à Trois-Rivières, 2017.

[17] Paul-Antoine Leboeuf. Introduction à lapprentissage automatique en pharmacométrie : concepts et applications. Université de Montréal 2021.

[18] Fan Li and Yu Wang. Routing in vehicular ad hoc networks : A survey. *IEEE Vehicular Technology Magazine*, 2(2) :12–22, 2007.

[19] Yisheng Lv, Yanjie Duan, Wenwen Kang, Zhengxi Li, and Fei-Yue Wang. Traffic flow prediction with big data : a deep learning approach. *IEEE Transactions on Intelligent Transportation Systems*, 16(2) :865–873, 2014.

[20] Sumit Roy. Dsrc/wave enabled connected vehicles infrastructure. Technical report, 2020.

[21] Hassan Hadi Saleh and Saad Talib Hasson. A survey of routing algorithms in vehicular networks. In *2019 International Conference on Advanced Science and Engineering (ICOASE)*, pages 159–164. IEEE, 2019.

[22] Zhou Su, Yilong Hui, Tom H Luan, Qiaorong Liu, and Rui Xing. *The Next Generation Vehicular Networks, Modeling, Algorithm and Applications*. Springer, 2021.

[23] Greg Van Houdt, Carlos Mosquera, and Gonzalo Nápoles. A review on the long short-term memory model. *Artificial Intelligence Review*, 53(8) :5929–5955, 2020.

[24] Hao Ye, Le Liang, Geoffrey Ye Li, JoonBeom Kim, Lu Lu, and May Wu. Machine learning for vehicular networks. *arXiv preprint arXiv :1712.07143*, 2017.

[25] Hongsuk Yi and Khac-Hoai Nam Bui. An automated hyperparameter search-based deep learning model for highway traffic prediction. *IEEE Transactions on Intelligent Transportation Systems*, 22(9) :5486–5495, 2020.

[26] Gülsüm Yiğit and Mehmet Fatih Amasyali. Simple but effective gru variants. In *2021 International Conference on INnovations in Intelligent SysTems and Applications (INISTA)*, pages 1–6. IEEE, 2021.

[27] Hongyuan Zhan, Gabriel Gomes, Xiaoye S Li, Kamesh Madduri, and Kesheng Wu. Efficient online hyperparameter learning for traffic flow prediction. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 164–169. IEEE, 2018.

[28] Yaying Zhang and Guan Huang. Traffic flow prediction model based on deep belief network and genetic algorithm. *IET Intelligent Transport Systems*, 12(6) :533–541, 2018.