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 RTIC01/M2/2021

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 routier par une méthode d'apprentissage profond

Par :
SAADI IBTISSAM

Soutenu le 04/07/2021 devant le jury composé de :

Kahlou Laid	Prof	Président
Bitam Salim	**Prof**	**Rapporteur**
Zernadji Tarek	MCB	Examinateur

Année universitaire 2020-2021

Résumé

La sécurité routière est un sujet très important dans le monde, elle est affectée par de différents problèmes qui surviennent sur la route tels que : les embouteillages et les émissions de CO2. La prédiction de flux de trafic routier est un élément clé dans les systèmes de transport intelligents (ITS) en raison de leurs caractéristiques non linéaires et stochastiques ce qui font rendre de la prédiction un problème difficile.

Pour faire face à ces problèmes et afin d'améliorer la précision de la prédiction pour améliorer la sécurité routière, le confort des usagers de la route et d'atténuer les embouteillages, nous proposons dans ce mémoire de master une méthode d'apprentissage profond issue de l'intelligence artificielle qui est basée sur un réseau de neurones récurrents à une mémoire convolutive à long-court terme (ConvLSTM2D) avec un LSTM bidirectionnel (Bi-LSTM), cette proposition est appelée Encodeur-Décodeur (ConvLSTM2D-Bi LSTM). Chaque module de cette méthode est adopté pour extraire les caractéristiques de flux de trafic, on suggère le module de ConvLSTM2D comme un encodeur afin d'extraire les informations spatio-temporelles sur le flux du trafic. Par ailleurs, le module LSTM bidirectionnel est initié comme un décodeur pour analyser les données historiques de flux de trafic pour obtenir la caractéristique de périodicité de flux de trafic. Pour valider cette proposition, un ensemble d'expérimentation a été mené sur la base de données de Transport for Grand Manchester (TfGM).

Les résultats expérimentaux ont montré que le modèle proposé permet d'obtenir une meilleure précision de prédiction par rapport aux modèles comparés qui sont : Attention based CNN-LSTM et ConvLSTM2D (notre modèle sans le module Bi-LSTM).

Mot clés : Système de transport intelligent, sécurité routière, prédiction de flux de trafic, apprentissage profond, LSTM (Mémoire à long-court terme).

Abstract

Road safety is a very important topic in the world, it is affected by different problems that occur on the road such as : traffic jams, CO2 emissions. The prediction of road traffic flows is the key element in Intelligent Transport Systems (ITS) due to their non-linear and stochastic characteristics which make the prediction a difficult problem.

To address these issues and to improve the accuracy of prediction to improve road safety, road user comfort and alleviate traffic congestion, In this master thesis, we propose a deep learning method from artificial intelligence which is based on a recurrent neural network with a long-short term convolutional memory (ConvLSTM2D) with a bidirectional LSTM (Bi-LSTM), this proposal is called Encoder-Decoder (ConvLSTM2D-Bi LSTM). Each module of this method is adopted to extract the traffic flow characteristics, the ConvLSTM2D module is suggested as an encoder to extract the spatio-temporal information about the traffic flow. On the other hand, the bidirectional LSTM module is initiated as a decoder to analyze the historical traffic flow data to obtain the periodicity feature of the traffic flow. To validate this proposal, a set of experiments was conducted on the Transport for Grand Manchester (TfGM) database.

The experimental results showed that the proposed model achieves better prediction accuracy than the compared models which are Attention based CNN-LSTM and ConvLSTM2D (our model without the Bi-LSTM module) .

Keywords : Intelligent transportation system (ITS), road safety, traffic flow prediction, deep learning, LSTM (Long short-term memory)

ملخص

تعتبر السلامة على الطرق موضوعًا مهمًا للغاية في العالم، فهي تتأثر بالمشكلات المختلفة التي تظهر على الطريق مثل: الاختناقات المرورية وانبعاثات ثاني أكسيد الكربون. يعد التنبؤ بتدفق حركة المرور على الطرق عنصرًا أساسيًا في أنظمة النقل الذكية (ITS)نظرًا لخصائصها غير الخطية والعشوائية التي تجعل التنبؤ مشكلة صعبة.

لمواجهة هذه المشاكل ومن أجل تحسين دقة التنبؤ لتحسين السلامة على الطرق وراحة مستخدمي الطريق وتقليل الاختناقات المرورية، نقترح في أطروحة الماجستير هذه طريقة التعلم العميق المستمدة من الذكاء الاصطناعي والتي تقوم على أساس طويل - ذاكرة تلافيفيه قصيرة المدى (ConvLSTM2D) شبكة عصبية متكررة معLSTM ثنائي الاتجاه(Bi-LSTM) ، وهذا الاقتراح يسمى Encoder-Décodeur (ConvLSTM2D-BiLSTM) يتم اعتماد كل وحدة من هذه الطريقة لاستخراج خصائص تدفق حركة المرور، نقترح وحدةConvLSTM2D كمشفّر لاستخراج المعلومات الزمانية المكانية حول تدفق حركة المرور. من ناحية أخرى، يتم بدء وحدة LSTM ثنائية الاتجاه كوحدة فك ترميز لتحليل بيانات تدفق حركة المرور التاريخية للحصول على خاصية الدورية لتدفق حركة المرور. للتحقق من صحة هذا الاقتراح، تم إجراء مجموعة من التجارب على قاعدة بيانات النقل لمانشستر الكبرى (TfGM)

أظهرت النتائج التجريبية أن النموذج المقترح يسمح بالحصول على دقة تنبؤ أفضل مقارنة بالنماذج Attention based CNN-LSTM و ConvLSTM2D (نموذجنا بدون وحدة Bi-LSTM)

الكلمات الرئيسية: نظام النقل الذكي، السلامة على الطرق، التنبؤ بتدفق حركة المرور، التعلم العميق،LSTM (ذاكرة طويلة المدى).

Remerciements

Tout d'abord, je tiens à remercier ***ALLAH*** *le tout puissant de m'avoir donner la volonté, la patience et la santé pour d'accomplir ce travail.*

Ensuite, Je tiens à exprimer ma gratitude à mon encadreur le professeur ***BITAM Salim****, pour ses efforts, ses conseils et critiques constructives, ses corrections, sa gentillesse et disponibilité.*

Mes vont également au Professeur ***Mohamed Chaouki BABAHENINI*** *pour son soutien et à toute l'équipe du laboratoire "Les Systèmes experts, l'Imagerie et leurs applications dans l'Ingénierie (LESIA)" de recherche en université de Mohamed khider, biskra , qui m' ont ouvert les portes afin de me permettre de travailler dans les meilleures conditions.*

*Un remerciement spécial à dr.****baya lina MENAI****, pour son soutien émotionnel, répondre à mes questions, son orientations et sa patience.*

Je tiens à remercier aussi, tous mes professeurs du département d'informatique. J'adresse mes sincères remerciements aux membres du jury, pour avoir accepter de juger ce travail.
Je remercie ma grande famille et mes amies de m'encourager durant cette année, et d'avoir toujours être disponible quand j'en avais besoin.

De peur d'avoir oublier, je souhaite remercier tous ceux qui ont contribué de près ou de loin à l'élaboration de ce mémoire ainsi qu'à la réussite de ce parcours universitaire.

Dédicace

ce travail est dédié :

***À l'âme de mon Grand-père**, décédé le 19/11/2020, qui m'a toujours poussé et motivé à avoir les bons résultats, et qui attendait avec impatience le jour de mon diplôme.*

***À mes toutes mères**, qui m'ont élevé à la personne que je suis aujourd'hui, aucune dédicace peut exprimer l'amour l'estime et le respect que j'ai toujours eue pour elles.*

***À mon très cher père**, Tu as toujours été pour moi un exemple du père respectueux, honnête, de la personne méticuleuse, je tiens à honorer l'homme que tu es. Grâce à toi papa j'ai appris le sens du travail et de la responsabilité. Je voudrais te remercier pour ton amour, ta générosité et compréhension.*

***À toute ma grande famille**, aucun langage ne peut exprimer mon respect et ma considération pour votre soutien et encouragements. Je vous dédie ce travail en reconnaissance de l'amour que vous m'offrez quotidiennement et votre bonté exceptionnelle.*

***A mes amies** Rym, Maissoun, Rania, Feriel... Je ne peux pas trouver les mots justes et sincères pour vous exprimer mon affection , vous êtes pour moi des sœurs et des amies sur qui je peux compter, je vous dédie ce travail et je vous souhaite une vie pleine de santé et de bonheur.*

Table des matières

Table des figures

Introduction générale

Chaque jour dans le monde, on recense un nombre important d'accidents et de pertes humaines sur les routes. De plus, les embouteillages deviennent de plus en plus un problème mondial, car ils perturbent le déroulement des activités humaines en provoquant d'énormes pertes de temps et d'argent, ainsi qu'ils contribuent à la pollution atmosphérique par l'émission du CO2. Avec l'application de nouvelles technologies de communication dans le domaine des transports, les systèmes de transport intelligents (ITS) ont été développés pour faire face aux problèmes routiers. Parmi ces applications on mentionne la prédiction de flux de trafic routier qui joue un rôle important dans le système de transport intelligent en raison de sa sensibilité de prédire avec précision des situations futures à partir de données passées et actuelles. On prédit par exemple le débit du trafic, la vitesse des véhicules dans la route afin d'aider les usagers de la route à prendre de meilleures décisions en matière de déplacement, d'améliorer l'efficacité de l'exploitation du trafic, de réduire les émissions de carbone et d'atténuer les embouteillages.

Dans cet égard, la prédiction de flux de trafic est considérée comme un problème des séries temporelles dans lequel la valeur future de flux de trafic est à estimer en fonction des données passées provenant de diverses sources, ce qui introduit l'idée de big data dans le domaine des transports. Dans ce contexte, les méthodes traditionnelles ont des difficultés à traiter de ce type de problème tels que elles ne peuvent pas explorer efficacement les caractéristiques non linéaire et stochastiques de flux de trafic ce qui influence négativement sur la précision de la prédiction.

Grâce aux récentes avancées dans le domaine de l'intelligence artificielle (IA) , les prédictions ont connu de grandes améliorations grâce aux algorithmes qui se sont avérés capables de capturer des relations non linéaires entre les données d'entrée et de sortie, ces algorithmes

sont généralement connus sous le nom de d'apprentissage profond et qui sont basés sur les réseaux de neurone (NN) et en particulier les réseaux de neurones récurrents (RNN).

Ce projet de master, vise la conception et le développement d'un système de prédiction de flux de trafic routier pour guider les conducteurs à circuler de manière optimale afin d'améliorer la sécurité routière, le confort des usagers de la route à l'aide de prédiction préciser. Pour cela, on opte pour une méthode d'apprentissage profond issue de l'intelligence artificielle qui est basée sur un réseau de neurones récurrents, et spécifiquement le RNN de type 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.

Notre mémoire est organisé en quatre chapitres principaux :

Le premier chapitre est consacré à la présentation des concepts des réseaux véhiculaires, l'Internet des véhicules et le système de transport intelligent et ses applications, les normes relatives aux réseaux véhiculaires, la prédiction des flux de trafic pour améliorer la sécurité routière, et l'apprentissage machine, ses différents types et leur phase d'apprentissage.

Le deuxième chapitre expose une introduction sur l'apprentissage profond pour la prédiction de flux de trafic et les réseaux de neurones récurrents (RNN) avec ses types. On présente aussi un état de l'art sur l'apprentissage profond pour la prédiction de flux de trafic et les réseaux de neurones récurrents (RNN) avec ses types LSTM et GRU, une comparaison et une discussion vont être effectuées.

Dans le troisième chapitre, nous présenterons la conception de notre système de prédiction à savoir la conception générale et détaillée, et les améliorations envisagées. Dans le chapitre quatre, nous illustrerons l'implémentation de notre système, en présentant les environnements matériels et logiciels pour le développement de notre système et les différents expérimentions et les résultats obtenus.

Ce travail sera terminé par une conclusion générale qui permettre d'examiner nos idées et nos résultats en donnant des améliorations possibles et quelques perspectives.

Chapitre 1

Prédiction des flux de trafic

1.1 Introduction

Les recherches dans le domaine des systèmes de transport intelligents ITS sont en plein développement. Le réseau véhiculaire VANET fait partie essentielle du système ITS due à sa contribution à éviter les problèmes routiers, ainsi que la prédiction de flux de trafic, c'est un élément important dans ce domaine, en raison de son rôle dans l'amélioration de sécurité routière.

Ce premier chapitre est consacré à la définition des concepts de base des réseaux véhiculaires, de l'Internet des véhicules et le système de transport intelligent. Nous décrirons également les normes relatives aux réseaux véhiculaires, la prédiction des flux de trafic pour améliorer la sécurité routière, et l'apprentissage machine, ses différents types et les notions les plus importantes dans la phase d'apprentissage seront également abordés.

1.2 Réseaux de véhicules, ITS et IoV

Le développement et la maturation des communications sans fil et des technologies de réseau au cours des dernières décennies ont fait des réseaux de véhicules un champ d'application possiblement très prometteur, pour améliorer la sécurité routières, optimiser l'efficacité du trafic routier et pour fournir un environnement plus sûr grâce aux diverses applications [20]. Un réseau véhiculaire considéré comme un composant important du développement du

système de transport intelligent (ITS) , est constitué des véhicules intelligents en mouvement et de stations de base routières. Trois entités principales forment le réseau véhiculaire qui sont : TA, RSU et OBU.

TA est l'autorité de confiance (appelée aussi CA autorité de certification) qui est une source d'authenticité de l'information. Elle garantit la gestion et l'enregistrement de toutes les entités sur le réseau. Les stations de base routières : (Road Side Unit - RSU-) sont des entités subordonnées des TA, elles sont installées sur le bord de la route. Certaines RSU peuvent servir de passerelle pour la connectivité à d'autres réseaux de communication, et les ordinateur de bord (On-Board Unit - OBU -). Ce 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 connecter directement à d'autres véhicules et/ou RSU qui sont dans sa portée de communication, voir le figure 1.1 et figure 1.2 .

FIGURE 1.1 – Les dispositifs RSU et OBU [9].

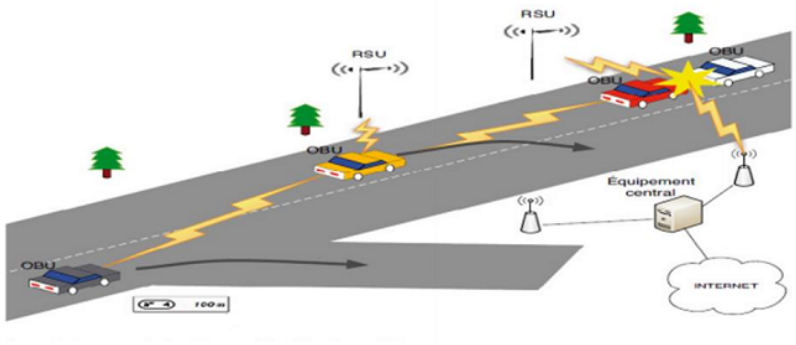

FIGURE 1.2 – Un réseaux véhiculaire [4].

Dans la littérature, ce type de réseau est connu sous le nom de Vehicular Ad-Hoc Network (VANET), qui fait partie de la famille des réseaux mobiles MANET [4][26] L'architecture de communication des VANETs peut être divisée en trois catégories : Véhicule à infrastructure (V2I), Véhicule à véhicule (V2V) et hybride [6] pour échanger des informations générées par des applications véhiculaires mobiles.

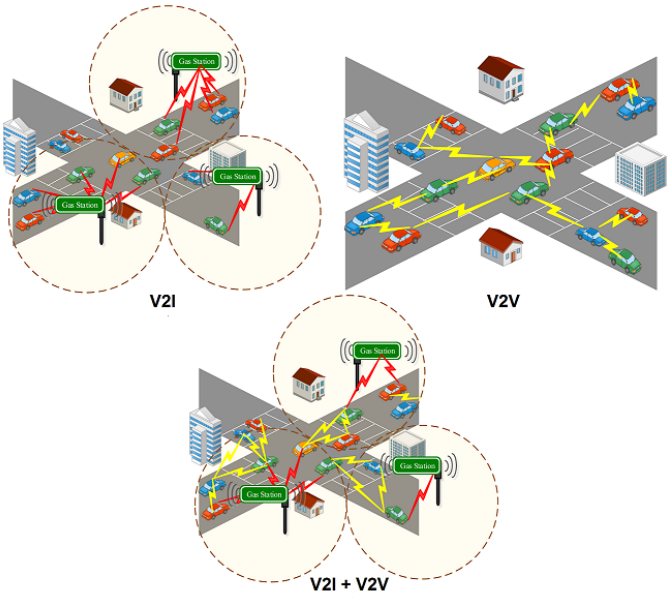

FIGURE 1.3 – Types de communications dans un VANET [6] .

L'application de nouvelles technologies dans le domaine des transports a conduit au développement de systèmes de transport intelligents (ITS), pour faire face aux problèmes routiers tels que la congestion, l'utilisation des infrastructures et la pollution environnementale. Avec

le développement des capteurs et des interfaces radio et des réseaux de nouvelle génération, une nouvelle architecture véhiculaire est apparue qui est l'Internet des véhicules (Internet of Vehicles –IoV-)[27].

IoV est considérée comme un partie de l'internet des objets (IoT) ou‘ les objets sont des véhicules intelligents connectés sur Internet. La communication dans l'IoV utilise des transmission connues sous le nom : V2X (véhicule a tous), on mentionne la communication V2V (véhicule à véhicule), V2R (véhicule a route), V2I (véhicule à infrastructure), V2P (véhicule a personne) et les communications V2S (véhicule à capteur). Les types de communication de l'IoV sont représentés dans la figure 1.4.

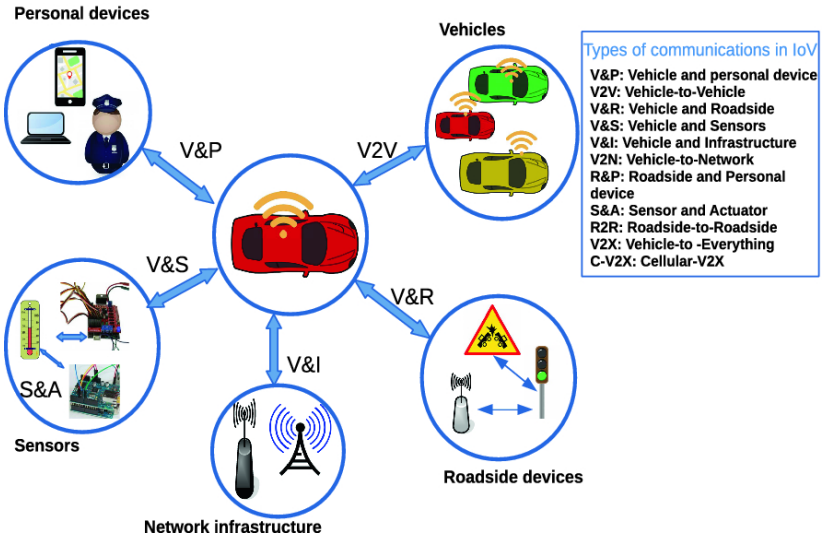

FIGURE 1.4 – Les types de communication de l'IoV [6].

Les applications des VANET, de l'IOV et des applications liées aux ITS peuvent être classées en deux catégories principales. La première est celle des applications de divertissement visant à fournir des services au conducteur, aux passagers ou à d' autres participants au système de transport. La deuxième catégorie d'applications se concentre sur la sécurité de la conduite et les applications liées au confort.

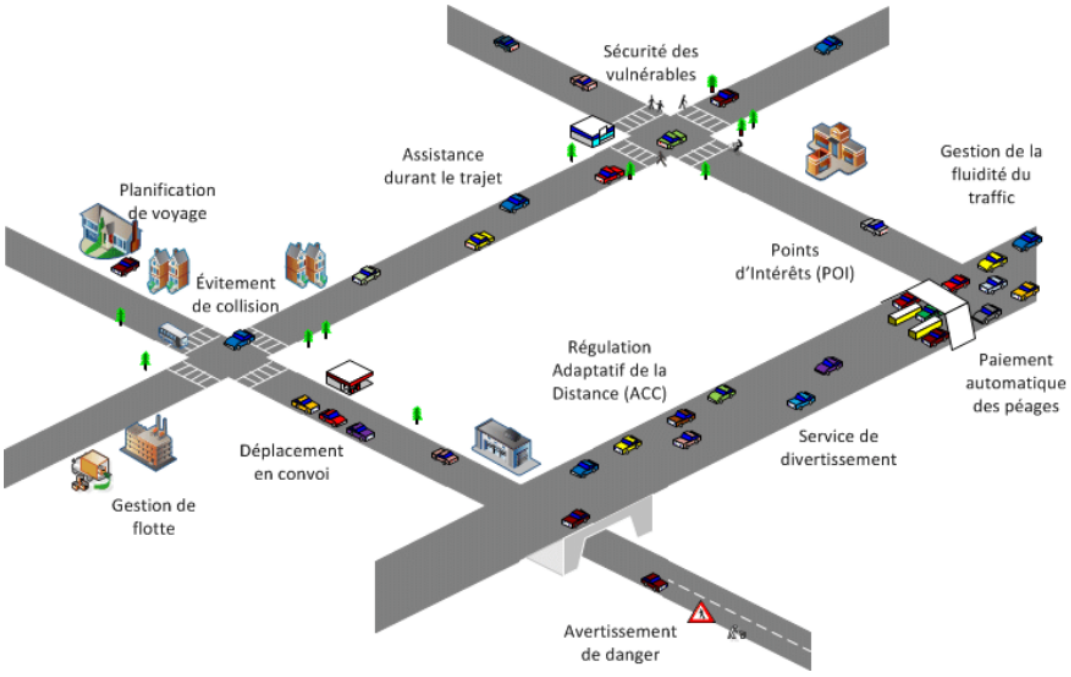

FIGURE 1.5 – Système de transport intelligent ITS [9].

1.2.1 Applications de sécurité

Ces applications se basent sur l'envoi des messages, périodique ou non, sur l'état de la route et des véhicules voisins. Ces messages peuvent prendre en charge les notifications de danger, la prévention des collisions et les avertissements. Ceci vise à réduire les probabilités d'accidents de la circulation et de congestion pour améliorer la sécurité des passagers sur les routes [4]. Ces applications sont classées dans les catégories suivantes [8] :

1.2.1.1 Gestion intelligente du trafic

Les applications de gestion intelligente du trafic visent à faciliter la circulation des véhicules en prévenant les encombrements ou même en évitant les zones encombrées et donc réduire les pertes humaines et la consommation d'énergie [7] . C'est l'un des objectifs de conception les plus importants des ITS. Ces applications sont basées sur deux méthodes.

La première est une méthode de contrôle de flux de trafic en temps réel (méthode réactive), par le biais d'un système de contrôle des signaux de trafic (TSC) pour assurer la fluidité du trafic. Cette méthode a un rôle essentiel dans la régulation de flux de trafic et dans le maintien

de la fluidité du système de transport. Mais lorsque le flux de trafic dépasse la capacité de la route, cette méthode est inefficace. En même temps, cette méthode ne peut pas réduire les embouteillages qui se sont produits.

Contrairement, le deuxième type de méthode de gestion proactive du trafic basée sur la prédiction du trafic peut aider la méthode réactive à prévenir les embouteillages dans une certaine mesure, tel que le système peut deviner une éventuelle congestion sur la route grâce à la prédiction de flux de trafic. Ceci peut servir pour planifier et gérer la circulation. Donc, la méthode réactive responsable de la micro-régulation et la méthode proactive responsable de la macro-régulation sont complémentaires et indispensables, dans un système de transport intelligent à grande échelle [8].

1.2.1.2 Conduite autonome

l'amélioration de la sécurité de la conduite des véhicules est importante pour améliorer la sécurité générale du système de transport, pour cela le développement de la technologie de conduite autonome joue un rôle crucial dans l'amélioration de la sécurité du trafic, car en tant que participant essentiel du système de transport, les technologies de conduite autonome peuvent être divisées en deux catégories.

La première catégorie est celle des technologies d'aide à la con- duite liées à l'état du mouvement du véhicule ou du conducteur, par exemple, le système d'alerte de maintien dans la voie/de sortie, le système de détection de la somnolence du conducteur (conduite autonome de niveau 1, il y a six niveaux d'autonomies d'un véhicule , dans la figure 1.6 : Selon la norme de la Society of Automotive Engineers (SAE)).

Le deuxième type est constitué des techniques liées à la détection des conditions de circulation autour des véhicules (la détection des animaux, la détection des véhicules ,etc.). Cette technologie utilise l'apprentissage profond pour résoudre le problème de la détection des objectifs. En raison de l'angle de vue limité de l´équipement de détection et des obstacles, la détection avec succès des piétons et autres cibles sera parfois impossible [8].

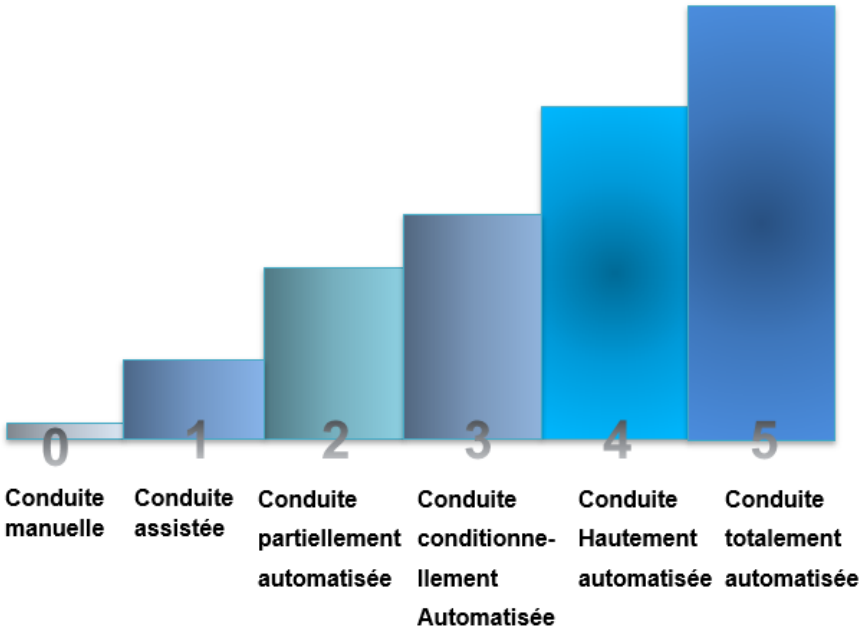

FIGURE 1.6 – Les niveaux d'autonomies d'un véhicule [6].

1.2.1.3 Informatique nuagique véhiculaire

l'objectif de la conception ces applications de Informatique nuagique véhiculaire (Vehicular cloud computing VCC en anglais) est l'amélioration de l'efficacité de l'utilisation de la puissance de calcul du véhicule , selon des différents états de fonctionnement des véhicules due leur différents équipement embarqué et savoir comment le véhicule puisse explorer et utiliser efficacement les ressources utilisées (la puissance de calcul et le stockage de véhicule en état de repos) d'autres participants au l'environnement IoV du système ITS, lorsque le véhicule se trouve dans de différents états de mouvement pour effectuer la tâche informatique du véhicule [8][7].

1.2.2 Applications de confort

Les applications de confort ou de divertissement contiennent toutes les applications qui contribuent au confort du conducteur et ne font pas partie de la gestion du trafic ou de la sécurité routière. Elles sont donc présentées comme des services fournis aux passagers du véhicule ou à d'autres participants du système de transport pour améliorer leur niveau de

confort. Parmi ces applications, on peut citer les panneaux d'affichage locaux et commerciaux tels que les restaurants, la présence de stations-service adjacentes, ou culturels tels que les informations touristiques sur la position du véhicule, l'estimation du temps de trajet, et les services d'accès à Internet, pour faciliter le voyage.

De plus, grâce à la prédiction de flux de trafic, les véhicules peuvent déterminer à l'avance les éventuels embouteillages et adapter leur itinéraire de fonctionnement pour les éviter, améliorant ainsi le confort de voyage [4][8].

1.3 Normes relatives aux réseaux véhiculaires

Les réseaux véhiculaires font une partie importante du système ITS. La normalisation des protocoles et la création des standards dans ce domaine est un act fondamental dans l'industrie des prochaines générations de véhicules. cet act à conduit une forte concurrence entre les Etats-Unis, l'Europe et le Japon pour créer des nouvelles normes ou modèles dans le système de réseau véhiculaire pour l'efficacité de leurs techniques et pour mieux répondre aux exigences de ce domaine de la grande vitesse de mouvement des véhicules, ainsi que le changement dynamique du type de communication[9]. Parmi ces normes, on cite les plus importants :

1.3.1 DSRC/WAVE

Les normes DSRC (Communications dédiées à courte portée) et WAVE (Environnement de véhicules sans fi) ont fait l'objet d'une attention particulière de la part de la communauté des chercheurs et des industriels, définies comme étant une technologie sans fil qui peut potentiellement répondre à l'exigence de latence extrêmement courte pour la messagerie et le contrôle de la securite routiere [21].

Le protocole DSRC a ete cree en 2002 par l'ASTM (Société Américaine pour les Tests et les Matériaux) spécialement pour les systèmes de transport intelligents ITS et représente un ensemble de protocoles et de normes définissant la communication à courte portée pour établir un réseau avance de véhicules connectés (communication V2X) notamment la communication V2V et V2I qui favorise l'efficacité et la sécurité du trafic.

Le protocole WAVE est la partie centrale du DSRC, il représente un groupe de normes et de protocoles d'accès sans fil possédant un ensemble de standards, de services et d'interfaces permettant de sécuriser les différents types de communications dans le système de réseau. Le terme DSRC est utilisé comme un terme plus général par rapport à WAVE [5][9].

La figure 1.7 représente l'architecture de DSRC/WAVE qui collecte la famille IEEE 1609 de WAVE, la norme IEEE802.11p qui peut être définie comme un système dédié aux communications à courte portée (DSRC) dans un environnement de véhicules sans fil (WAVE)) [9], et la norme J2735 de la Société des ingénieurs de l'automobile (SAE).

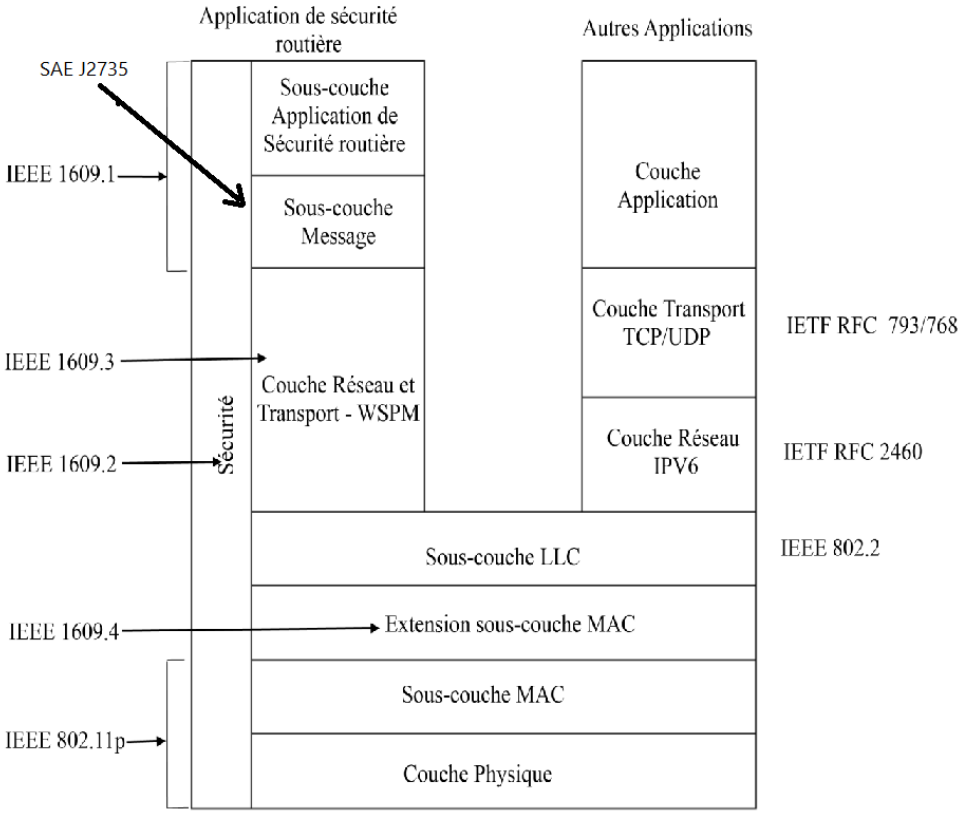

FIGURE 1.7 – L'architecture de DSRC/WAVE [9].

1.3.2 SAE J2735

La Société des ingénieurs de l'automobile (SAE) a développé la norme J2735 (SEA J2735) pour soutenir l'interopérabilité entre les applications des réseaux véhiculaires. Dans certaines situations, cette norme offre également des informations qui sont utiles dans la compréhension

de la façon d'appliquer l'ensemble des messages aux applications V2X.

La norme SAE J2735 définit environ 150 éléments de données standard et 70 trames de données standard et d'écrit 15 types d'ensembles de messages de données utilisés par les applications pour échanger des données sur le DSRC/WAVE et d'autres protocoles de communication. En outre, les catégories de messages liées à la norme SAE J2735 peuvent être réparties en messages de : transmission basique, sécurité, géolocalisation, informations aux voyageurs et paiement électronique [5] [9].

1.3.3 Messages de sécurité de base (BSM)

La technologie DSRC/WAVE et la norme SAE-J2735 offrent ensemble la possibilité d'échanger des informations de sécurité routière dans un réseau véhiculaire et de manière très efficace a l'aide de ce qui est présenté comme des messages de sécurité de base (Basic Safety Message) connu par le "heartbeat" message. Les BSM contiennent des informations essentielles sur l'état de véhicule (la vitesse, l'accélération, les feux, les freins, les essuie-glaces, l'horodatage, l'historique des trajets), la position et le mouvement du véhicule par rapport aux autres véhicules, ainsi qu'entre l'infrastructure. Ces informations n'existent que temporairement et ne sont pas stockées.

Le BSM est peut-être le message le plus important de la norme SAE J2735, car il permet d'éviter les collisions et offre la possibilité de transmettre des informations supplémentaires si nécessaire. Il comporte deux parties. La partie I comprend les informations d'état cruciales qui doivent être envoyées dans chaque BSM. La partie II est une zone facultative dans laquelle des éléments de données et des trames supplémentaires peuvent être inclus. La La figure 1.8 représente le format du BSM.

Ainsi, les BSM générés à partir de plusieurs véhicules connectés peuvent agir comme une source de données sur l'environnement de transports [5][7].

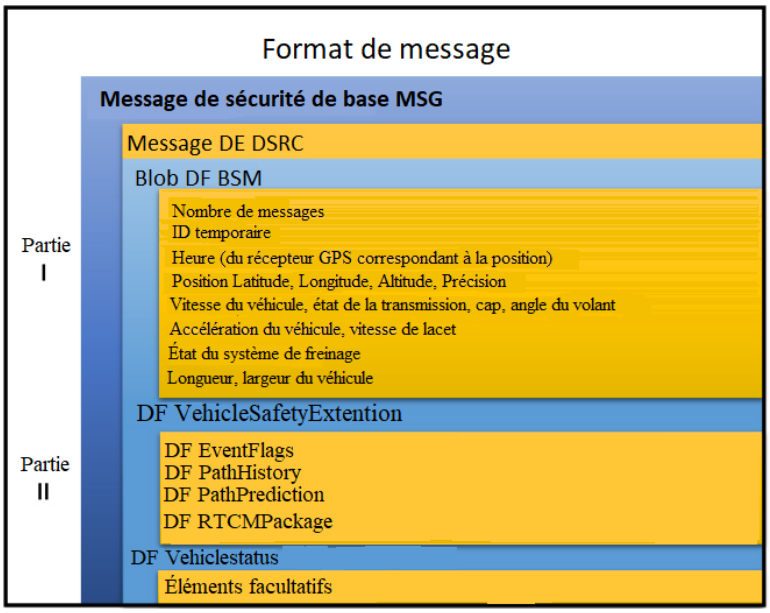

FIGURE 1.8 – Format du message de sécurité de base SAE J2735 [5].

1.4 prédiction de flux de trafic pour l'amélioration de la sécurité routière

La sécurité routière est une exigence nécessaire pour les conducteurs, les passagers, les piétons et tous les usagers de la route , elle doit être améliorée. Le système de transport intelligent (ITS) est un moyen d'aborder les problèmes liés au domaine du transport pour assurer la sécurité routière et son amélioration tels que la congestion du trafic, les émissions de carbone et les accidents de la circulation ... etc. Pour cela, la prédiction du flux de trafic est suggérée dans les applications liées à l'ITS.

La prédiction de flux de trafic est un élément fonctionnel crucial des ITS, car elle améliore l'efficacité de la transmission des données, et améliore la fiabilité et l'efficacité opérationnelle du transport. Ceci est atteint grâce à l'analyse de l'historique de l'information et en temps réel sur le flux de la trafic recueillie à partir de diverses sources (radars, caméras, dispositifs de véhicules, équipement d'infrastructure). Par conséquent, on enregistre une amélioration et une efficacité de l'établissement du routage des données et accroître la précision de calcul ou de stockage des données utilisée pour les véhicules dans un système (Vehicule Cloud

Computing). Par ailleurs, elle aide également à réduire les émissions de carbone, en évitant ou en atténuant les embouteillages.

La prédiction de flux de trafic est généralement considérée comme un moyen essentiel d'aide à la gestion de trafic et permettant aux conducteurs de choisir la voie appropriée en fonction des données qui leur sont fournies pour améliorer leur confort de déplacement et pour améliorer la sécurité routière.

En général, deux modèles sont utilisés pour la mise en œuvre de cette prédiction de flux du trafic, le premier est basé sur les statistiques et le seconde sur un apprentissage automatique[5].

1.5 Apprentissage automatique

L'apprentissage automatique (Machine Learning (ML) en anglais), en tant que branche importante de l'intelligence artificielle, est un axe de recherche qui a une nature multidisciplinaire en le qualifiant en une meilleure adaptabilité aux différents contextes et aux divers domaines, tels que la reconnaissance des formes, le diagnostic médical, l'exploration de données, la vision par ordinateur, la reconnaissance vocale, la traduction et la prédiction [35].

1.5.1 Types d'algorithmes d'apprentissage automatique

L'apprentissage automatique se divise en trois types : l'apprentissage supervisé, l'apprentissage non supervisé et l'apprentissage par renforcement, la compréhension de ces types est très importante pour choisir l'algorithme approprié au type de données spécifiées a dans le domaine de l'étude [35] (voir la figure 1.9).

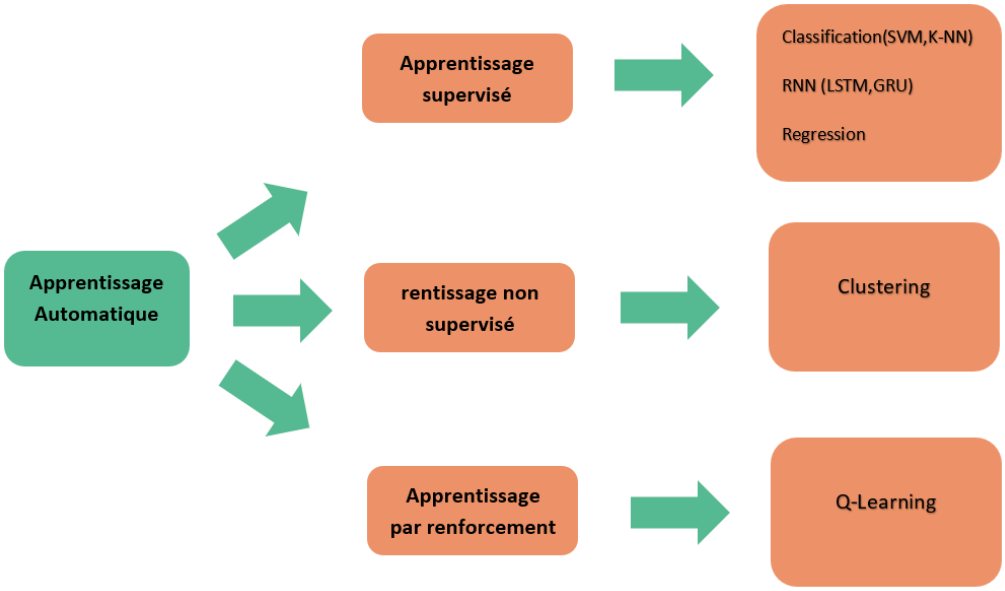

FIGURE 1.9 – taxonomie générale des méthodes de prédiction basées sur le ML[5].

1.5.1.1 Apprentissage supervisé

L'apprentissage supervisé est une technique d'apprentissage qui nécessite un entraînement avec des données étiquetées composées des variables d'entrée (x) et une variable de sortie souhaitée (Y) pour faire des prédictions sur des nouveaux donnes où seules les entrées sont fournies et les sorties sont prédites par le modèle, et ces prédites sont comparées aux variables cibles retenues et utilisées pour estimer la compétence du modèle. Il existe deux types principaux de problèmes d'apprentissage supervisé : la classification et régression.

Si la sortie est impliqué la prédiction d'une étiquette de classe (valeurs discrètes) il s'agit une classification, et si la sortie est impliquée la prédiction d'une étiquette numérique (valeurs réelles continue) il s'agit d'une régression. Les techniques les plus couramment utilisées dans cette catégorie sont les suivantes les réseaux neuronaux, les voisins les plus proches, les machines à vecteurs de support, les arbres de décision et l'algorithme de Bayes naïf [11].

1.5.1.2 Apprentissage non supervisé

L'apprentissage non supervisé ne fonctionne que sur les données d'entrée (X), sans sorties ni variables cibles correspondantes. En tant que tel, l'apprentissage non supervisé n'a pas

d'enseignant qui corrige le modèle et l'algorithme d'apprentissage automatique est censé trouver par lui-même d'éventuels modèles dans les données d'entraînement. Clustering (k-Means) et l'estimation de la densité sont deux types d'apprentissage non supervisé le plus fréquemment rencontrés, le clustering qui consiste à trouver des groupes dans les données et l'estimation qui consiste à résumer la distribution des données [11].

1.5.1.3 Apprentissage par renforcement

L'apprentissage par renforcement diffère fondamentalement des problèmes supervisés et non supervisés, il est permis d'apprendre à partir de 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'entraînement fixe mais plutôt un objectif ou un ensemble d'objectifs que l'agent doit atteindre. Il est fréquemment utilisé pour la robotique, les jeux et la navigation.

Par exemple, un agent (l'algorithme) interagit avec l'environnement pour trouver la solution optimale. L'agent essaie plusieurs solutions, observe la réaction de l'environnement et adapte son comportement (les variables) pour trouver la meilleure stratégie [11].

1.5.2 L'entraînement dans L'apprentissage automatique

Cette phase est très importante, il s'agit du cœur de la ML en général. lors de l'entraînement la machine cherche les paramètres de modèle qui minimisent la fonction de perte (coût), L'algorithme de rétro-propagation est la méthode couramment utilisée pour l'entraînement du modèle , où la fonction de perte et l'algorithme d'optimisation par descente de gradient jouent un rôle essentiel dans l'évaluation de performance du modèle pour obtenir un modèle à une performance équilibrée sur les données de l'entraînement et test (modèle sans overfitting) [34] .

1.5.2.1 Fonction de perte

La fonction de perte, également appelée fonction de coût, mesure la compatibilité entre les prédictions de sortie du réseau et la valeur réelle donnée étiquettes par le rétro- propagation en utilisant l'algorithme de descente de gradient, c'est à dire réduit tous les aspects du modèle

à un seul nombre de telle sorte que les améliorations de ce nombre soient le signe d'un meilleur modèle [14].

1.5.2.2 Hyperparamètres

Les paramètres et hyperparamètres sont des notions différentes et très importantes dans l'apprentissage automatique. Les paramètres changent au fur et à mesure que le modèle est entraîné (phase d'apprentissage) par exemple : le poids biais. Par contre certains paramètres utilisés n'appartiennent pas directement au modèle lui-même, et restent constants tout au long de la phase d'apprentissage. Ils sont appelés hyperparamètres et appartiennent plutôt à l'algorithme ML (qui entraîne le modèle ML) .Il n'y a pas de valeurs global à attribuer aux hyperparamètres, car elles sont fortement liées à la situation. Pour déterminer les valeurs optimales des hyperparamètres il faut utiliser une approche par essais et erreurs. Des exemples d'hyperparamètres importants associés aux réseaux de neurone sont : Nombre de couches cachées, Nombre de neurones dans chaque couche cachée, Nombre d'époques et le taux d'apprentissage [25].

1. Nombre d'époques :

   Une époque correspond à une itération à travers toutes les paires de mappings d'entrée/sortie. C'est rarement suffisant pour minimiser la fonction de coût. Le réseau peut donc avoir besoin de nombreuses époques d'apprentissage avant que toutes les connaissances n'aient été apprises. Cependant, un trop grand nombre d'époques peut entraîner un sur-ajustement (Overfitting) [25].

2. Taux d'apprentissage :

   Le taux d'apprentissage, (ou learning rate) peut être considéré comme l'hyperparamètre le plus important, c'est un hyperparamètre configurable utilisé dans l'entraînement de modèle qui a une petite valeur positive, comprise entre 0,0 et 1,0. Il contrôle également la vitesse à laquelle le modèle s'adapte au problème. Si le taux d'apprentissage trop élevé peut faire converger le modèle trop rapidement vers une solution sous-optimale et nécessitent moins d'époques de formation, tandis qu'un taux d'apprentissage trop faible peut bloquer le processus et nécessitent un plus grand nombre d'époques de formation [25].

1.5.2.3 La descente de gradient

La Descente de Gradient, (ou Gradient Descent en anglais) est un des algorithmes les plus importants de toute le Machine Learning et de tout le Deep Learning. Il est techniquement appelé algorithme d'optimisation de premier ordre car il utilise explicitement la dérivée de premier ordre de la fonction de coût (fonction de perte) , et qui met à jour de manière itérative les paramètres apprenables, c'est-à-dire les noyaux et les poids, qui permet également d'entraîner votre modèles pour minimiser la fonction de coût afin de trouver le meilleur modèle [13] [34].

1.5.2.4 Rétropropagation

L'objectif de l'algorithme de l'entraînement par rétro-propagation est de modifier les poids d'un réseau neuronal afin de minimiser l'erreur des sorties du réseau par rapport à une certaine sortie attendue en réponse aux entrées correspondantes. La rétro-propagation permet de calculer le gradient de l'erreur pour chaque neurone, de la dernière couche vers la première. Dans le cas des données séquentielles comme une série temporelle, l'entraînement effectué par la Backpropagation Through Time, (ou BPTT), tel que chaque pas de temps comporte un pas de temps d'entrée, une copie du réseau et une sortie. Les erreurs sont ensuite calculées et accumulées pour chaque pas de temps. Le réseau est déroulé à nouveau et les poids sont mis à jour. Lorsque le nombre de pas de temps augmente, la BPTT peut être coûteuse en termes de calcul [12].

1.5.2.5 Surapprentissage (Overfitting)

L'Overfitting est un problème incontournable dans le machine learning (réseau de neurone), il désigne un comportement indésirable d'un algorithme d'apprentissage automatique utilisé pour la modélisation prédicative. En d'autres termes, si la performance du modèle sur l'ensemble de données d'apprentissage (training set) est significativement meilleure que la performance sur l'ensemble de données de test (test set), alors le modèle peut avoir un overfitting sur l'ensemble de données d'apprentissage. Divers modèles d'apprentissage automatique utilisent différentes techniques de régularisation pour éviter le sur-apprentissage. Le

technique la plus simple mais efficace appelée "Dropping out (Dropout)". Dans le Dropout, un sous-ensemble d'activations choisies aléatoirement est mis à zéro dans une couche [2]. Le concept du dropout est illustré à la figure 1.10

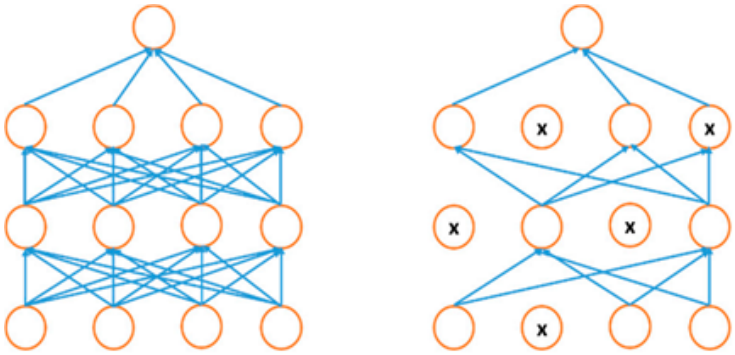

FIGURE 1.10 – Le concept du dropout[2]

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 d'améliorer la sécurité routière et d'offrir des services aux conducteurs pour leur confort.

On a mentionné dans ce chapitre à l'apprentissage automatique et ses différents types, qui ont été utilisés pour la prédiction de flux de trafic. Dans ce mémoire, nous concentrons sur l'apprentissage supervisé et notamment les modèles de prédiction basées sur les réseaux de neurones récurrents (RNN) car ils sont largement utilisés dans la prédiction de flux de trafic, due à leurs caractéristiques permettant une haute précision et une capacité robuste d'analyse de données volumineuses par rapport aux autres modèles. Dans le chapitre suivant, nous allons présenter l'apprentissage profond dans la prédiction de flux de trafic et quelques travaux sur la prédiction des flux de trafic réalisées par les différents types des RNN.

Chapitre 2

Les modèles RNN pour la prédiction de flux de Trafic : État de l'art

2.1 Introduction

La prédiction du 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 de deep learning, pour donner une précision de prédiction des flux de trafic très performante.

Dans ce chapitre, nous allons présenter dans la première partie une introduction sur l'apprentissage profond pour la prédiction de flux de trafic et les réseaux de neurones récurrents (RNN) avec ses types, et dans la deuxième partie un état de l'art sur la prédiction des flux de trafic par les différents types des RNN tels que LSTM et GRU.

2.2 Apprentissage profond pour la prédiction de flux de trafic

L'apprentissage profond (Deep learning (DL) en anglais), est un type de méthodes d'apprentissage automatique qui a suscité un grand intérêt académique et industriel, il a été appliqué avec succès dans de nombreuses applications. L'apprentissage profond est caractérisé par des architectures à couches multiples ou des architectures profondes inspirées des neurones du cerveau, illustrées dans la figure 2.1.

En raison de ses architectures, le (DL) a une forte capacité à extraire les caractéristiques inhérentes aux données du niveau le plus bas au niveau le plus élevé et à réaliser des calculs complexes, vu que la prédiction de flux de trafic est considérée comme processus très compliqué et contient de nombreuses caractéristiques dans les dimensions spatiales et temporelles. L'apprentissage profond peut représenter les caractéristiques du trafic sans connaissance préalable et donne de bonnes performances pour la prédiction de flux de trafic [33][24].

Ce processus a deux dépendances définies comme suit :

1. Dépendance temporelle : il s'agit une dépendance ou une relation lors de la prédiction de flux de trafic actuels ou futures avec le flux de trafic des instants passés [18].
2. Dépendance spatiale : 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 [18].

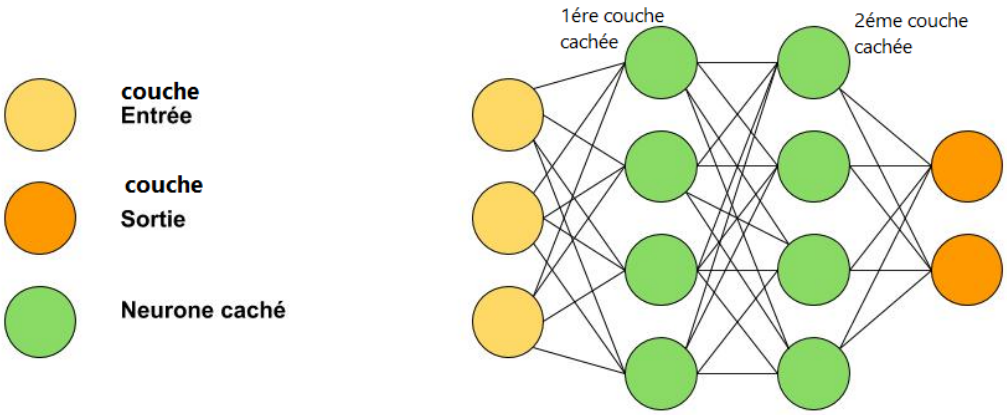

FIGURE 2.1 – Architecture des couches d'apprentissage profond [8].

Pour la prédiction de flux de trafic routier, l'apprentissage profond utilise des méthodes différentes 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[8].

2.2.1 Réseaux de neurones récurrents (RNN)

Les réseaux de neurones traditionnels ne sont pas appropriés au problème de la prédiction de flux de trafic, en raison de leur négligence de la dimension temporelle des données de séries. De plus, il n'y a pas de connexion entre les neurones d'une même couche, la connexion est seulement entre les neurones de deux couches adjacentes.

Contrairement, les RNN sont des réseaux de neurones supervisés qui préservent la dimension temporelle des données de séries en utilisant un état caché récurrent qui sont mis à jour par les informations séquentielles obtenues à partir des séries temporelles de données d'entrée. La sortie d'une séquence est dépendante de cet état caché c-à-d la sortie actuelle d'une séquence est liée à la sortie précédente (réseau neuronal récurrent) comme le montre la figure 2.2 . Ils se basent sur l'utilisation répétée de classificateurs ce qui permet d'éviter le besoin d'un grand nombre de classificateurs historiques.

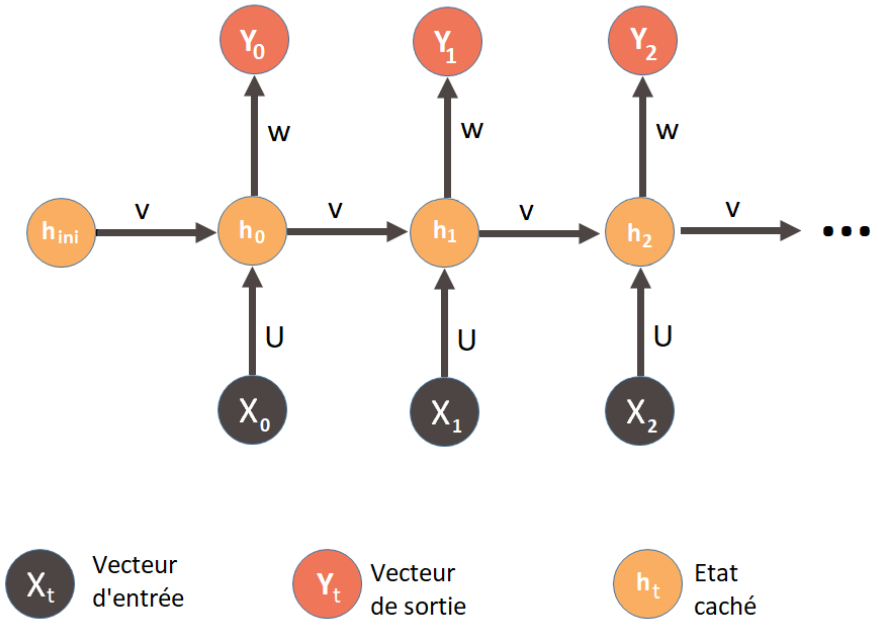

FIGURE 2.2 – Structure d'un réseaux de neurones récurrents (RNN)[(RNN)[10].

Par conséquent, le RNN est une approche plus performante pour les problèmes de prédiction de flux de trafic grâce aux couches cachées, car ils sont connectées, contrairement aux réseaux de neurones traditionnels, qui ne présente aucune connexion entre les couches cachées [8].

Les RNN ont deux modèles très connus pour résoudre le problème de prédiction des flux de trafic , ils sont le modèle de mémoire a long terme (LSTM) et le modèle d'unité récurrente à porte(GRU) [8].

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

Bien que le RNN est un modèle efficace pour la prédiction de flux de trafic, mais il peut être difficile d'apprendre la dépendance à long terme, principalement en raison des problèmes de disparition et d'explosion du gradient [30].

1. le problème de disparition est la diminution très rapide des valeurs des gradients pendant la rétro propagation entraînant l'annulation du gradient et l'arrêt de l'apprentissage [19].

2. le problème d'explosion est l'augmentation très rapide des valeurs des gradients pendant la rétro-propagation entraînant un dépassement de la capacité de la représentation interne des nombres et l'arrêt de l'apprentissage [19].

L'utilisation de la mémoire à long terme et à court terme (LSTM) est un moyen efficace de surmonter les problèmes posés en RNN, car elle utilise des neurones portés (gated) pour capturer toutes les séquences à court et à long terme dans le flux de trafic. En d'autres termes, la LSTM peut obtenir de meilleures performances dans des séquences plus longues que les RNN normaux [33][8].

La LSTM est composée d'une couche d'entrée, d'une couche cachée (considérée comme un bloc de mémoire) et d'une couche de sortie. La structure d'une LSTM est illustrée par la figure 2.3. Le bloc de mémoire contient un cellule de mémoire (état de la cellule) avec trois portes :

- Porte d'oubli : pour contrôler les caractéristiques de la série temporelle (qui contrôle les informations à rejeter de la mémoire).
- Porte d'entrée : pour contrôler l'entrée (qui contrôle les nouvelles informations ajoutées à l'état de la cellule à partir de l'entrée actuelle).
- Porte de sortie pour contrôler la sortie (décider de manière conditionnelle de ce qui doit être sortir de la mémoire) [30] [28].

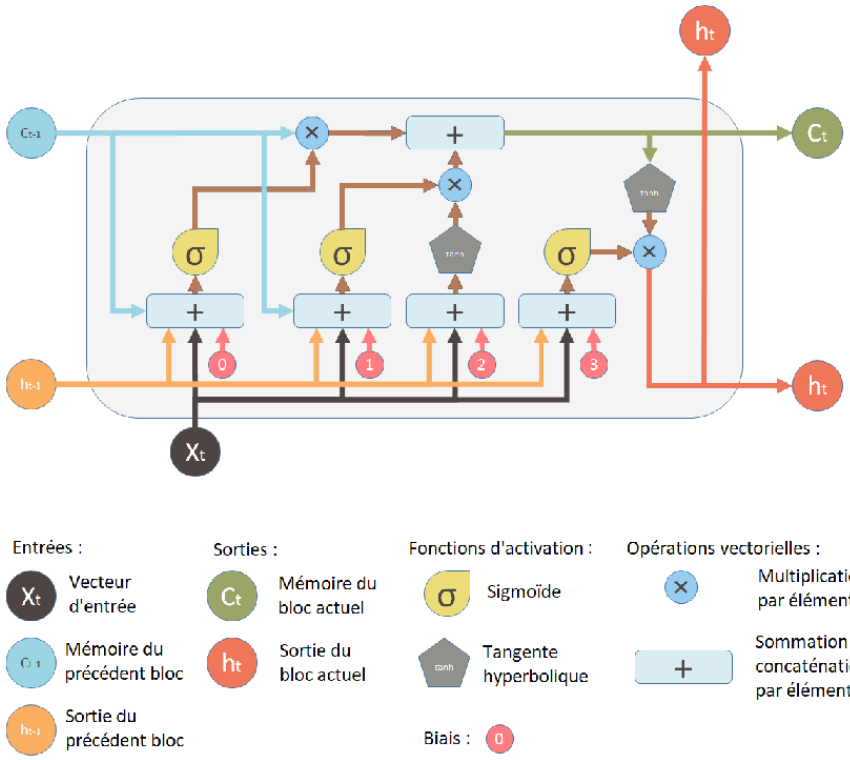

FIGURE 2.3 – La structure d'un LSTM [10].

Le modèle LSTM peut être représenté par les équations suivantes [28] :

$$f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \tag{2.1}$$

$$i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \tag{2.2}$$

$$o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \tag{2.3}$$

$$\tilde{c}_t = \sigma_c(W_c x_t + U_c h_{t-1} + b_c) \tag{2.4}$$

$$c_t = f_t \circ c_{t-1} + i_t \circ \tilde{c}_t \tag{2.5}$$

$$h_t = o_t \circ \sigma_h(c_t) \tag{2.6}$$

Les i_t, o_t, f_t et c_t représentent respectivement la sortie de la porte d'entrée, de la porte de sortie, de la porte d'oubli et l'état de la cellule de mémoire,les données d'entrée de la couche cachée à l'intervalle t représentent par x_t, où σ_g signifie la fonction d'activation sigmoïde, σ_c et σ_h sont les fonctions tangentes hyperboliques, W et U signifie les matrices de pondération et b signifie les vecteurs de biais, l'opérateur $\circ$ désigne le produit de Hadamard (produit par

éléments)[19].

le LSTM a une meilleure fonction de mémoire, ce qui permet d'éviter efficacement les problèmes de disparition et d'explosion du gradient, et il est donc plus adapté aux tâches de prédiction [8] .

2.2.3 Unité récurrente à porte (GRU)

La cellule Gated Recurrent Unit (GRU) a été développée par Cho et al[1], à une structure plus compacte, comme illustré par la figure 2.4. Contrairement à la structure à portes multiples de LSTM, GRU a moins de portes de contrôle et un état de la cellule, c'est pourquoi elle a un coût et une complexité de calcul plus faible que LSTM. Par ailleurs, elle a une influence à peu près similaire à celle de LSTM pour résoudre les problèmes des RNN (explosion et disparition du gradient)[8].

Les deux portes de contrôle dans GRU sont :

- La porte de reset r qui détermine si nous avons besoin de fusionner l'état actuel avec l' état précédent.
- La porte de mise à jour z qui détermine combien d'informations d'état ont été apportées au moment précédent.

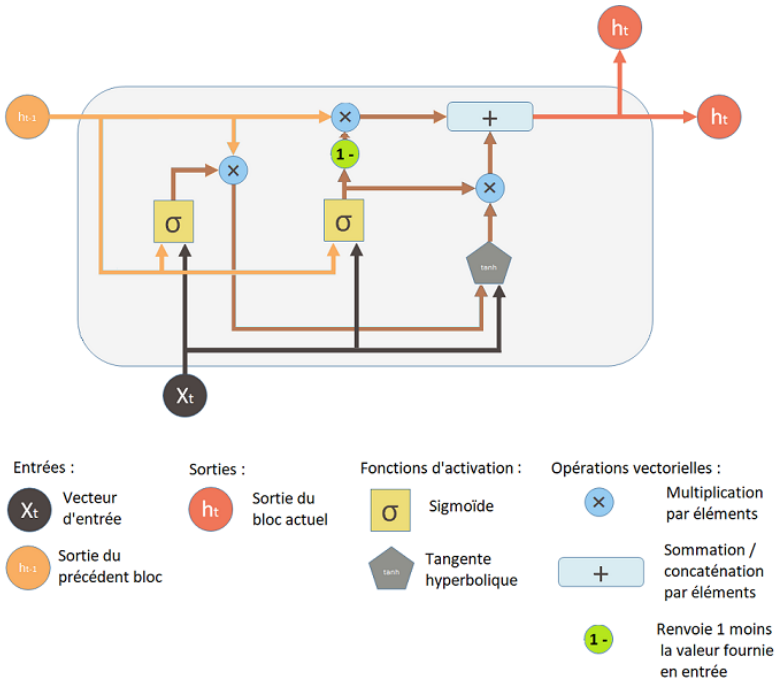

FIGURE 2.4 – La structure de GRU[10].

La cellule GRU est calculée comme suit [28] :

$$z_t = \sigma_g(W_z x_t + U_z h_{t-1} + b_z) \tag{2.7}$$

$$r_t = \sigma_g(W_r x_t + U_r h_{t-1} + b_r) \tag{2.8}$$

$$\hat{h}_t = \phi_h(W_h x_t + U_h(r_t \odot h_{t-1}) + b_h) \tag{2.9}$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \hat{h}_t \tag{2.10}$$

Une fonction Sigmoïde σ écrase les nombres entre 0 et 1, Le fonction tanhgate ϕ sont utilisées pour mettre la valeur entre -1 et 1, et L'opérateur $\odot$ désigne le produit de Hadamard

2.3 Le modèle LSTM pour la prédiction des flux de Trafic

Premièrement, nous allons exposer les deux méthodes qui basée sur LSTM :

2.3.1 La prédiction des flux de trafic basée sur un AutoEncodeur et un LSTM (AE-LSTM) [32]

1. **Principe :**

   Les auteurs de cet article [32] ont proposé une nouvelle méthode de prédiction de flux de trafic appelée méthode de prédiction AutoEncoder Long Short- Term Memory (AE-LSTM), Pour améliorer la précision de la prédiction , ce qui conduit a améliorer efficacement la qualité de la vie urbaine.

   Cette méthode combine l'AutoEncoder avec le LSTM, ou l'AutoEncoder est une sorte d'apprentissage non supervise , largement utilise dans le traitement de debruitage, et peut également être utilise comme extracteur de caractéristiques des données de flux de trafic en amont et en aval de la position actuelle. Ceci ne permet pas d'ajouter trop de calculs dans la prédiction ultérieure par LSTM ce qui aide à réduire la complexité de calcul. par ailleurs, le LSTM est une sorte d'apprentissage supervise, qui utilisée les caractéristiques acquises par L'AE et les données historiques pour prédire les données linéaires complexes du flux de trafic telles que la séquence de flux de trafic de sortie . donc le modèle AE-LSTM ne prend pas seule- ment en compte les caractéristiques temporelles mais il utilise également les données en amont et en aval pour capturer les caractéristiques spatiales de flux de trafic.

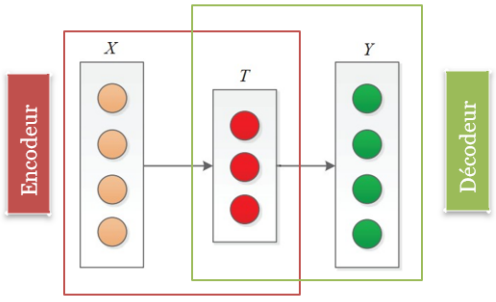

FIGURE 2.5 – La structure de l'AutoEncoder. Le codeur obtient la séquence caractéristique T basée sur la séquence originale X. Le décodeur obtient la séquence reconstruite Y en fonction de la séquence caractéristique T.

L'AutoEncoder est divisé en deux parties après son entraînement , à savoir l'enco-

deur et le décodeur. L'encodeur pour extraire de caractéristiques de données et Le décodeur est utilisé pour vérifier la validité des caractéristiques extraites, et est écarté après l'entrainement. comme représenter dans la (Figure 2.1). un modèle LSTM est attache après l'encodeur pour l'entraînement le modèle de prédiction AE-LSTM .voir la (Figure 2.6).

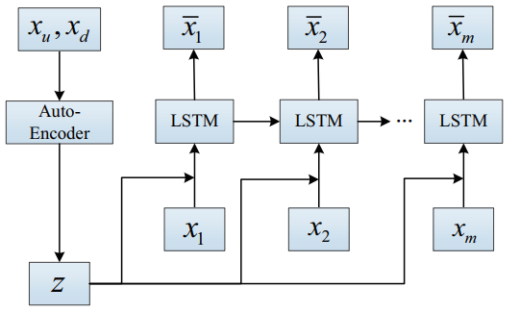

FIGURE 2.6 – La structure de l'AE-LSTM pour la prédiction de flux de trafic.

2. **Expérimentation de l' AE-LSTM :**

La performance de AE-LSTM est évalué par des expériences sur des ensembles de données réelles de flux de trafic qui ont été obtenues à partir du système de mesure de la performance de Caltrans (PeMS) [15] de la période allant d'avril à septembre 2018. Le flux de trafic est le nombre de véhicules passant par un détecteur de (PeMS) sur une période de temps, et l'intervalle de prédiction des données de flux de trafic était de 5 minutes.

Les ensembles de données ont été divises en deux sous-ensembles, à savoir l'ensemble de l'entraînement et l'ensemble de test, les données des 176 premiers jours des six mois utilises dans le processus de l'entraînement, donnant lieu au modèle. Pour le modèle LSTM , la sortie précédente est considérée comme une entrée dans l'itération actuelle lors de la prédiction . Le modèle généré est utilisée pour prédire les données des sept prochains jours, et le comparer aux données des sept derniers jours pendant les six mois. Par ailleurs, les données des cinq premiers mois sont utilisées comme données d'entraînement et prédits différentes étapes suivantes. En considérant les modèles temporels du flux de trafic, ils utilisent les données de flux de trafic de la période précédente pour prédire les données de la période suivante. En considérant le modèle

spatial de flux de trafic, on pris les données de flux de trafic en amont (upstream) et en aval (downstream) comme entrée du modèle de prédiction.

- Les données de flux de trafic en amont (upstream) : Si le flux de données va vers la source originale, ce flux est en amont.
- Les données de flux de trafic en aval (downstream) : Si le flux de données s'éloigne de la source originale, ce flux est en aval.

Pour la vérification de la performance du modèle de prédiction, les auteurs ont comparé AE-LSTM avec des modèles différents tels que l'algorithme proposé [32], le modèle SVM et CNN. Pour cette étude, trois indicateurs de mesure ont été visés l'erreur quadratique moyenne (RMSE), l'erreur relative moyenne (MRE) et l'erreur absolue moyenne (MAE).

3. **Résultats expérimentaux obtenus :**

   Lorsque la prédiction d'un petit pas d'avance (step ahead : 4, 6, 8) : 4 a été effectuée, l'erreur de l'algorithme de prédiction AE-LSTM était inférieure à celle des autres algorithmes comparés, ce qui indique que le modèle de prédiction a une bonne robustesse et stabilité. Lorsque le pas d'avance de la prédiction augmente, la précision de la prédiction diminue légèrement. Dans la comparaison de la valeur observée avec les résultats prédits des autres algorithmes de prédiction à savoir CNN, SVM, l'algorithme proposé dans [32], AE-LSTM a donné une meilleure performance de prédiction qui correspond bien aux valeurs observées. Pour les résultats de la distribution cumulative MAE de la prédiction de flux de trafic et pour chaque méthode de prédiction dans différentes pas d'avance, les erreurs du modèle de prédiction AE-LSTM sur différents ensembles de données étaient fondamentalement les mêmes, ce qui indique que le modèle de prédiction a une bonne robustesse.

4. **Critique :**
   - La méthode proposée dans cet article a une meilleure performance et précision avec des corrélations spatiaux simples, mais il est préférable qu'elle prend en compte des corrélations spatiales plus complexes telles que les corrélations spatiales de flux de trafic des positions adjacentes et distantes.
   - La LSTM a une structure un peu complexe, ce qui nécessite beaucoup de temps

lors de l'apprentissage. Ainsi Le modèle AE-LSTM sera moins rapide lors de l'entraînement .

2.3.2 La Prédiction des flux de trafic à court terme avec Conv-LSTM [23]

1. **Principe :**

   Cette étude présente une nouvelle méthode d'apprentissage profond (DL) de bout en bout sans prétraitement de données ni extraction manuelle des caractéristiques des données. C'est la combinaison de deux modules le réseau de neurones convolutif et le LSTM pour former un module appelé (Conv-LSTM). Par ailleurs, on a proposé un module LSTM bidirectionnel(Bi-LSTM) qui est considère comme un module auxiliaire, pour assurer une prédiction précise en atténuant les embouteillages. Le module Conv-LSTM peut extraire les informations spatio-temporelles des informations sur le flux de trafic. En outre, le module Bi-LSTM est adopte pour analyser les données historiques de flux de trafic du point de prédiction pour obtenir la caractéristique de périodicité de flux de trafic.

   Dans le module Conv-LSTM il existe deux opérations : la convolution et le mise en commun (Max Pooling)/fusion (Pooling), Ils ont utilisé 1D-Conv pour traiter chaque élément de la matrice d'entrée qui est représenté par un vecteur de série temporelle des données de flux de trafic. Un filtre à noyau de convolution unidimensionnel est utilisé pour acquérir les informations de convolution du domaine perceptuel local par filtre coulissant (sliding filter), afin de former les caractéristiques globales à partir les caractéristiques local spatial extrait entre les points proches. Le filtre de mise en commun est appliqué sur le vecteur de données, et n'effectue pas d'opérations complexes de convolution. Ce processus de mise en commun, certaines informations inutiles sont filtrées pour obtenir des données plus abstraites tels que les données de flux de trafic et la vitesse.

   Les résultats de sortie sont un vecteur de série temporelle. Comme chaque élément du vecteur est la corrélation spatiale de flux de trafic entre les points de la région, la

série temporelle est utilisée comme entrée pour la couche LSTM. Cette structure est représentée par la Figure 2.7.

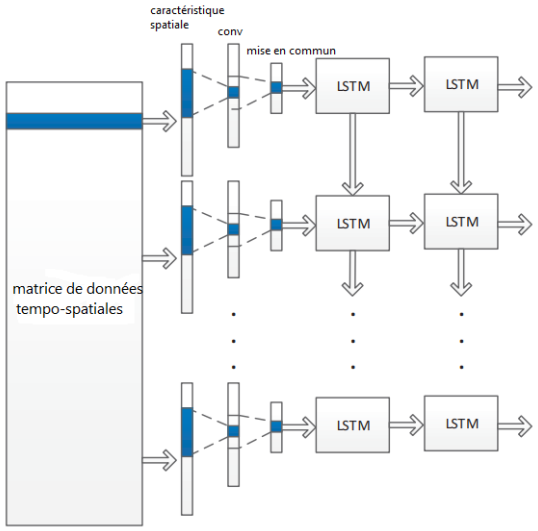

FIGURE 2.7 – La structure du Conv-LSTM.

Le module Bi-LSTM est compose de deux LSTM unidirectionnels empiles de haut en bas est utilise pour extraire la caractéristique périodique à partir du traitement des données historiques. L'entrée du Bi-LSTM contient les séries temporelles avant et après le moment de la prédiction, ces caractéristiques sont ajoutées comme informations supplémentaires pour prédire le flux de trafic à court terme. La structure du Bi-LSTM est présentée par la Figure 2.8.

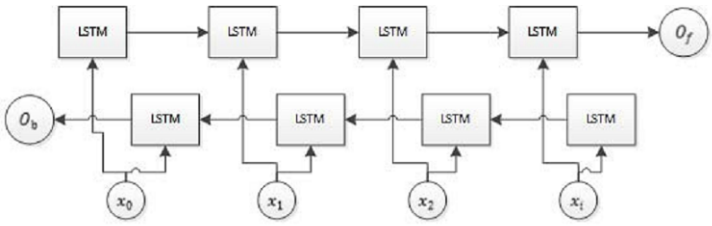

FIGURE 2.8 – La structure du Bi-LSTM.

Donc, La méthode proposée se compose de deux parties, la première étant le module Conv-LSTM et la seconde est le module Bi-LSTM, les caractéristiques extraites de chaque module sont concentrées or focalisés à la prédictions de flux de trafic, c-à-d,

les deux modèle utilise dans la prédiction de flux de trafic . La structure globale du réseau neuronal profond proposé est illustrée par la Figure 2.9.

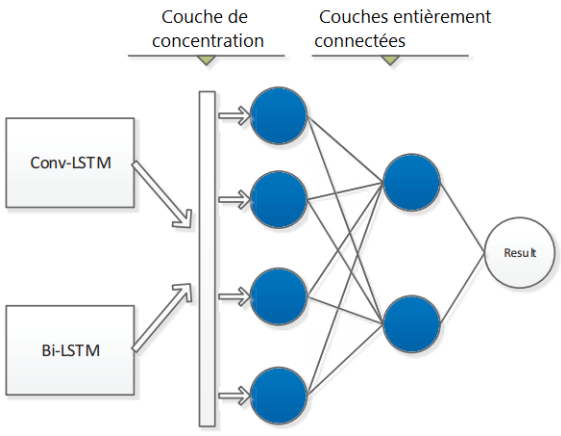

FIGURE 2.9 – Le modèle d'architecture profonde proposé pour la prédiction des flux de trafic à court terme

2. **Expérimentation :**

   Pour valider cette étude, on a effectué des simulations pour comparer la performance de prédiction de la méthode proposée avec les autres méthodes existantes, sur des données du monde réel utilisant l'ensemble de données du système de mesure de la performance de Caltrans (PeMS). Ces données sur les flux de trafic sont collectées toutes les 30 secondes à deux endroits différents, le premier étant les données de trafic sur autoroute de SR99-S District 10 et l'autre les données de trafic urbain sur la rue I980-E District 4 à Oakland City (au USA).

   La méthode proposée avec et sans Bi- LSTM a été comparée avec les méthodes ARIMA, SVM, SAE, LSTM, CNN-LSTM sur la base des trois mesures de : L'erreur absolue moyenne (MAE),l'erreur absolue moyenne en pourcentage](MAPE), l'erreur quadratique moyenne (RMSE).

3. **Résultats expérimentaux :**

   Les résultats expérimentaux montrent que le module Bi-LSTM peut améliorer la précision de prédiction de l'architecture proposée basée sur l'apprentissage profond, et montrent aussi que l'approche proposée permet d' atteindre une meilleure performance

par rapport aux autres approches.

4. **Critique :**
   - Il est préférable de traiter d'autres types de données de trafic afin d'améliorer la précision de la prédiction des flux de trafic à court terme tels que la densité , le nombre de véhicules, le nombre de piétons et de cyclistes...etc.
   - L'architecture de Conv-LSTM utilisée est limitée dans le cas d'un ensemble de données non varier et de taille élevée.

2.4 Le modèle GRU pour la prédiction de flux de Trafic

Dans cette partie, nous allons exposer deux méthodes de prédiction de flux de trafic qui sont basées sur le modèle GRU :

2.4.1 La prédiction des flux de trafic basée sur un AutoEncodeur et un GRU (AE-GRU) [16]

1. **Principe :**

   Dans [11], les auteurs ont proposé une méthode de prédiction du flux de trafic à court terme basée sur un apprentissage profond qui combine un AutoEncoder et un GRU, elle est appelée (AE-GRU). Cette méthode convient au changements de flux de trafic et prend en compte la relation spatiale et temporelle de données de flux de trafic, et peut améliorer la précision de prédiction et résoudre le problème de la faible robustesse. Cette méthode prend en compte l'impact de flux de trafic des liens en amont (upstream) et en aval (downstream) du lien actuel et la relation spatiale topologique du réseau routier par l'utilisation d'un algorithme qui sélectionne les liens à haute pertinence qui est basée sur le coefficient de corrélation spatiale moyen hebdomadaire. Ce coefficient à son tour est bas´e sur le coefficient de corrélation de Pearson qui est un algorithme commun de discrimination de similitude visant certain nombre de liens en amont et en aval qui ont le plus de corrélation avec le lien actuel.

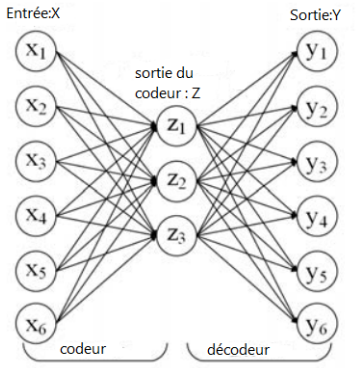

FIGURE 2.10 – La structure du Auto-Encodeur.

De plus, l'AutoEncoder est un algorithme d'apprentissage non supervisé qui peut être divise en deux parties : l'encodeur et le décodeur. La partie encodeur est responsable de la réduction des données à haute dimension en données à basse dimension (compression), et la partie décodeur reconstruit ensuite les données à basse dimension en données à haute dimension (décompression), la structure du AutoEncoder est présentée par la Figure Figure 2.10. L'AutoEncoder doit être pré-entraîné avec des données historiques avant d'être utilisé pour l'extraction de caractéristiques des données des liens sélectionnés , puis les introduire dans le GRU avec les données des liens actuels ce qui permet de réduire efficacement la dimension d'entrée des données spatiales et la complexité du calcul. L'architecture globale du modèle est présentée à la Figure 2.11.

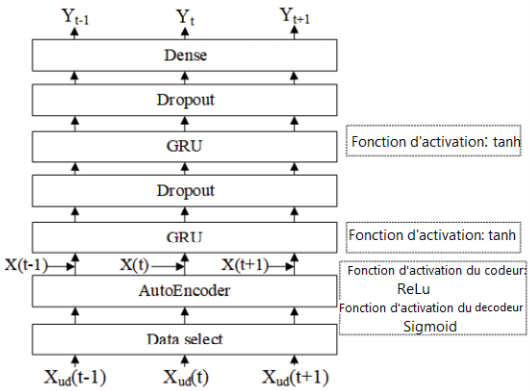

FIGURE 2.11 – La structure du Conv-LSTM.

Donc, cette méthode utilisée dans cet article peut être principalement divisée en trois

étape : sélection des liens par corrélation spatiale, extraction des caractéristiques de l'AutoEncoder et prédiction du réseau GRU.

2. **Expérimentation :**

   Afin de tester la performance prédicative du modèle propose et de prouver sa faisabilité, les auteurs ont utilisé des données réelles qui concerne la ville de Wuhan, en Chine, on a collecté des données de deux mois de allant du 15 avril au 15 juin 2019 sur Le lien (Link/lane ou segment de route) numéro 15929 a été utilise comme lien actuel, et chaque lien contient 288 données en une journée, Il est noter que les données du 15 au 21 avril sont utilisées pour le calcul des coefficients de corrélation spatiale moyenne hebdomadaire et choisir les liens dont le coefficient de corrélation spatiale moyen hebdomadaire supérieur ou égal à 0,8 comme lien de similarité, et les données du 22 avril au 1er mai 2019 sont utilisées pour la pré-entraîner de l'AutoEncoder en fonction des 8 liens sélectionne. de plus, les données du 2 mai au 2 juin 2019 sont utilisées pour l'algorithme AE-GRU, et les données du 3 au 8 juin 2019 ont été utilisées comme ensemble de test.

   Pour mesurer les performances du modèle de prédiction propos´e et les autres modèles existants GRU, LSTM, SVR, GRU-dims et AE- LSTM les auteurs ont choisi les trois indicateurs RMSE, MAPE, MAE comme dans [32] et [23].

3. **Résultats expérimentaux :**

   Les résultats des tests effectuant sur les données de test de la semaine du 3 mai au 2 juin 2019 qui concernent les MAPE, MAE et RMSE ont montré que lorsque la longueur du paramètre de décalage augment ou diminue, la précision de prédiction de tous les modèles s'amélioré. Plus précisément, les MAPE, MAE et RMSE de AE-GRU sont meilleurs que ceux de GRU, LSTM et SVR.

   Pour le GRU-dim qui n'a pas été sélectionné par le coefficient de corrélation spatiale moyen hebdomadaire , ainsi que l'effet de prédiction n'a pas été amélioré.

   Pour AE-LSTM, les auteurs ont remplace les unités de GRU dans l'AE-GRU par des unités LSTM, et ont enregistré le temps de l'entraınement et les erreurs de prédiction des deux modèles AE-GR eu AE-LSTM], les résultats obtenus ont montré que le temps de l'entra^ınement de AE-GRU est plus court et la précision de prédiction de

l'AE-GRU est proche de celle de l'AE-LSTM.

4. Critique :

   La méthode AE-GRU est basée sur gru dans la prédiction, à cause de la structure simple de son unité de mémoire, certaines informations importantes peuvent être ignorées dans le processus de prédiction telles que des informations liée au volume de trafic ce qui conduit vers une réduction de précision.

2.4.2 un réseau convolutif à graphes temporels pour la prédiction du trafic (T-GCN)[36]

1. **Principe :**

   Dans l'article [36], les auteurs ont présenté un nouveau modèle de prédiction du trafic basée sur un réseau de neurones pour capturer les dépendances spatiales et temporelles simultanément, le modèle de réseau convolutif de graphes temporels appelé (T-GCN), combine le réseau convolutif de graphes (GCN) et l'unité récurrente à porte (GRU).

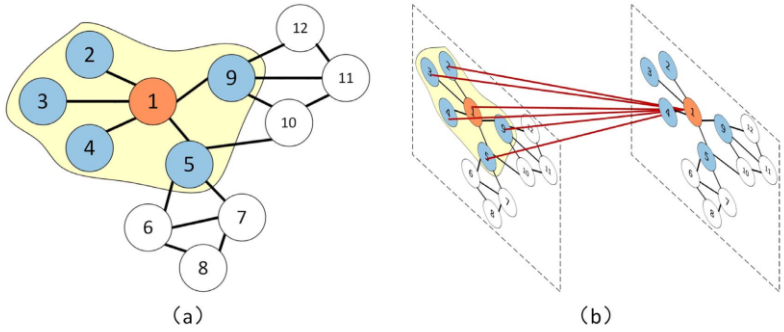

FIGURE 2.12 – En supposant que le nœud 1 est une route centrale. (a) Les nœuds bleus indiquent les routes connectées à la route centrale. (b) Nous obtenons les caractéristiques spatiales en obtenant la relation topologique entre la route 1 et ses routes environnantes.

Le GCN est utilisé pour apprendre des structures topologiques complexes du réseau routier à partir des données de trafic afin de capturer la dépendance spatiale comme le montre la Figure 2.12. et GRU en raison de sa structure simple et sa rapidité d'apprentissage. Ceci a été choisi pour apprendre les changements dynamiques des données de trafic afin de capturer la dépendance temporelle.

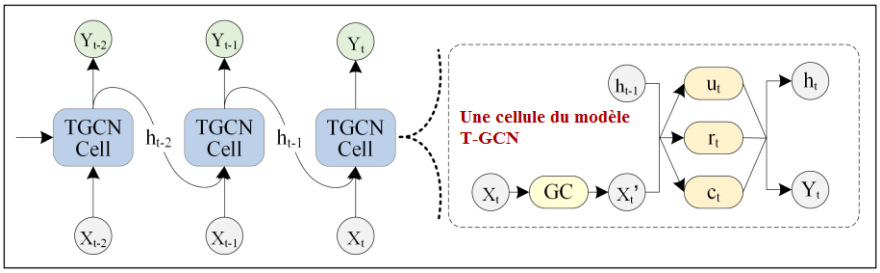

FIGURE 2.13 – Le processus global de la prédiction spatio-temporelle. La partie droite représente l'architecture spécifique d'une unité T-GCN, et GC représente la convolution du graphe.

Ensuite, le modèle T-GCN (voir laFigure 2.13), est utilisé pour la prédiction du trafic basée sur le réseau routier urbain. la Figure 2.14 illustre comment le modèle T-GCN est utilise pour réaliser la tache de prédiction de trafic basée sur les routes urbaines.

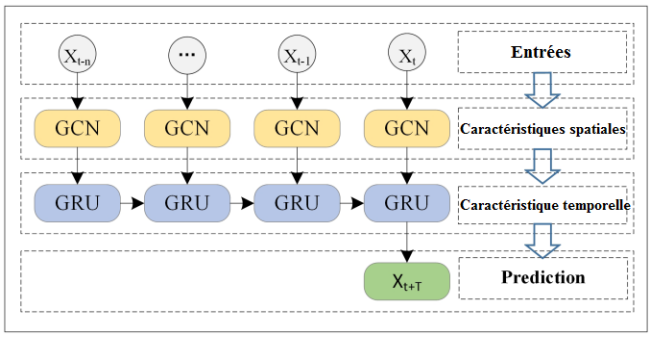

FIGURE 2.14 – Vue d'ensemble. Nous prenons les informations historiques sur le trafic en entrée et obtenons le résultat final de la prédiction grâce au réseau de convolution graphique et au modèle d'unité récurrente à porte.

2. **Expérimentation :**

   Dans cette expérience, les performances de prédiction du modèle T-GCN sont effectuées sur deux ensembles de données du monde réel sur la vitesse du trafic,l'ensemble de données SZ-taxi et Los-loop.

   - SZ-taxi : Cet ensemble de données est constitué de la trajectoire des taxis de Shenzhen du 1er au 31 janvier 2015 . Ils sélectionné comme zone d'étude contenant 156 routes principales du district de Luohu , la vitesse du trafic sur chaque route

est agrégée toutes les 15 minutes.

- Los-loop : Cet ensemble de données a été collecté sur l'autoroute du comté de Los Angeles en temps réel par des détecteurs de boucle. cet ensemble de donnes est liée à la vitesse du trafic, donc ils ont sélectionné un total de 207 capteurs avec leur trafic vitesse du 1er mars au 7 mars 2012. La vitesse du trafic sur chaque route est agrégée toutes les 5 minutes. Ils ont utilisé une méthode d'interpolation linéaire dans cet ensemble de données pour combler les valeurs manquantes.

Le but de cette expérience est de prédire la vitesse du trafic pour les 15 minutes, 30 minutes, 45 minutes et 60 minutes à venir, avec les méthodes de baseHA[22], ARIMA[1], SVR[29], GCN, GRU. et analyser le changement de la précision des prédictions avec différentes unités cachées. 80/100 des données sont utilisées comme ensemble d'apprentissage et les 20/100 restants sont utilisés comme ensemble de test, le modèle T-GCN est entraîné en utilisant l'optimiseur Adam (Adaptive Moment) est un algorithme d'optimisation du taux d'apprentissage adaptatif conçu spécifiquement pour l'entraînement de réseaux de neurones profonds. Parmi les mesures utilisées pour évaluer les performances de prédiction du modèle T-GCN, on mentionne RMSE et MAE, Ainsi l'exactitude (accuracy) est utilisée pour détecter la précision de la prédiction (R2) et var calculent le coefficient de corrélation, qui mesure la capacité du résultat prédit à représenter les données réelles.

Il y a d'autre expérience dans ce travail, pour tester la robustesse du modèle T-GCN par l'analyse de perturbation sur les données collectées. A partir d'ajouter deux types de bruit aux données, le bruit aléatoire obéit à la distribution Gaussienn et à la distribution de Poisson.

3. **Résultats expérimentaux :**

Les résultats obtenus montrent que le modèle T-GCN obtient la meilleure performance de prédiction pour presque toutes les mesures d'évaluation par rapport aux autres méthodes de base telles que le modèle HA, ARIMA, SVR et GCN qui prend en compte que les caractéristiques spatiales et GRU prennent en compte seulement les caractéristiques temporelles. Ceci prouve l'efficacité du modèle T-GCN pour les taches de prédiction spatio-temporelle du trafic. En outre, les résultats de prédiction

montrent un état stable sous différents horizons de prédiction, ce qui indique que le modèle T-GCN peut non seulement réaliser des prédictions à court terme mais aussi être utilise pour des taches de prédiction de trafic à long terme.

Pour les résultats d'analyse de la perturbation (bruit dans les donnes) montre que il y a un changement faible dans les métriques d'évaluation de performance de RMSE, MAE, R2...ect pendant l'analyse , quelle que soit la distribution du bruit (soit la distribution Gaussienn ou Poisson) . Ainsi, le modèle T-GCN est robuste et capable de géré les problèmes de bruit élevé.

4. **Critique :**

   Le modèle T-GCN utilise uniquement la convolution de diffusion de 1er ordre comme fonction de fusion du modèle de dépendance spatiale pour captures les caractéristique spatial et le réseau GRU pour modéliser la d´dépendance temporelle . dans ce cas , la fonction de fusion du réseau convolutif est fixe ce qui peut limiter la flexibilité du modèle pour capturer la dépendance spatiale. Il est donc préférable de rendre la fonction de fusion du réseau convolutionnel graphique entraînable, pour aider le modèle a obtenir plus de diversité et d'informations utiles et pour plus de flexible pour capturer la dépendance spatiale.

2.5 Conclusion

Dans ce chapitre, nous avons présenté l'apprentissage 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 l'art sur les méthodes utilisées pour la prédiction de flux de trafic routier. Nous avons constaté qu'il y a des méthodes ne prend qu'en considération les corrélations spatiales ou temporelles ou les corrélations spatio-temporelles mais il travaille avec un ensemble de données simple (C'est-à-dire donnée univariée), ce qui affectera sur la performance de modèle de prédiction.

Par conséquent, dans le chapitre suivant nous allons proposer un nouveau modèle de prédiction de flux de trafic contribuer à l'amélioration de la précision de la prédiction en présentant la conception générale et détaillés de ce système proposer.

Chapitre 3

Conception du système de prédiction proposé

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 nous ont permis de proposer un modèle de prédiction de flux de trafic à utiliser dans la conception et la réalisation de notre système .

Dans ce chapitre, nous allons aborder la tâche la plus importante dans ce travail, à savoir la tâche de conception. En effet, nous présentons, en premier lieu, l'architecture générale de notre système de prédiction de flux de trafic afin d'en extraire les différentes étapes et les modules qui la composent. Puis, nous détaillons chacune de ces étapes en précisant leurs fonctionnements.

3.2 La conception générale du système

La prédiction de flux de trafic routier joue un rôle important dans le système de transport intelligent, c'est idéal pour la sécurité et le confort sur les routes. Notre modèle proposé est basé sur l'amélioration de la précision de la prédiction de flux de

trafic pour améliorer la sécurité routière, pour cela on prend lors de l'apprentissage différents types de caractéristiques de trafic en utilisant des modules convenables.

Nous illustrons l'architecture de notre système de prédiction de flux de trafic comme suit :

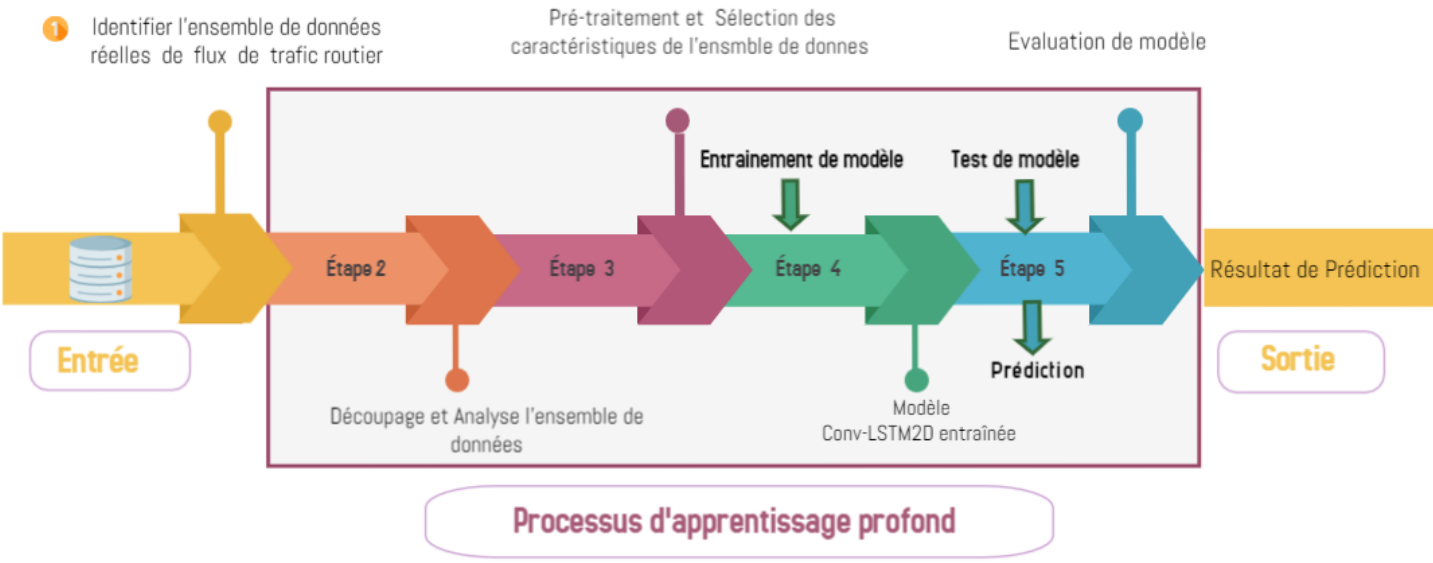

FIGURE 3.1 – L'architecture générale du système.

Comme illustré dans l'architecture précédente, nous pouvons diviser notre système en six étapes :

3.2.1 Identifier l'ensemble de données réelles (Entrée)

L'ensemble de données, ou dataset, c'est une série temporelle qui contient une suite d'observations chiffrées ordonnées dans le temps, et considérer comme une collection d'éléments variés.

Le choix de dataset joue un rôle important dans la réussite de la mise en place un modèle d'apprentissage profond car il représente le principal en jeu d'un modèle performant. Dans notre système on a identifié un dataset réel de flux de trafic diversifié (multivarié) et volumineux pour nous aidons à construire un modèle d'apprentissage plus précis.

Dans la section suivante on va présenter le processus d'apprentissage profond de notre système.

3.2.2 Découpage et Analyse de dataset

3.2.2.1 Découpage de dataset

Dans cette étape, nous découpons notre dataset de flux de trafic en deux parties (au minimum) : entraînement et test, Chacune a un échantillonnage spécifié.

- Données d'entraînement : sont utilisées pour l'apprentissage d'un modèle.
- Données de test : sont utilisées pour l'évaluation du modèle c-à-d vérifier la performance du modèle final et les résultats.

3.2.2.2 Analyse de dataset

C'est une étape importante qui nous permettre de faire :

- inventaire de notre donnée, à partir de déterminer quel est le type de donnée, le nombre d'observations(lignes), Nombre de features/variables (colonnes), Quelles sont les variables connues, les variables à prédire.
- Détecter les valeurs manquantes .
- Détecter si nous avons des valeurs corrompues ou non valides.

Grâce à cela, nous pouvons déterminer les étapes de prétraitement et de nettoyage appropriées.

3.2.3 Pré-traitement et Sélection des caractéristiques de dataset

Certaines techniques peuvent être appliquées afin de préparer les données et de construire un modèle d'apprentissage profond.

3.2.3.1 Pré-traitement de dataset

Le prétraitement des données comprend diverses opérations. Chaque opération vise à élaborer de meilleurs modèles prédicatifs, dans notre travail l'étape de prétraitement consiste en deux opérations :

- la standardisation : afin de normaliser la gamme de caractéristiques de l'ensemble de données d'entrée.
- Convertir le problème de prédiction de séries temporelles en problème d'apprentissage supervisé, où les données d'entrée ont été converties du format de série temporelles multivariée au format d'apprentissage supervisé à l'aide de la méthode de la fenêtre glissante « sliding window » pour les rendre exploitables par le modèle d'apprentissage profond.

3.2.3.2 Sélection des caractéristiques

Dans cette étape, nous allons choisir les colonnes que nous utiliserons comme caractéristiques d'entrée pour l'entraînement du modèle et qui contribuent le plus à notre variable de prédiction ou la sortie, tout en ignorant celles non pertinentes.

3.2.4 Entraînement du modèle

3.2.4.1 Modélisation

Lors de cette étape , on a choisir une méthode d'apprentissage profond issue de l'intelligence artificielle qui est basée sur un réseau de neurones récurrents (RNN) , à une mémoire convolutive à long-court terme (ConvLSTM2D) avec une LSTM bidirectionnels (Bi-LSTM). Notre modèle est appelé Encodeur-Décodeur ConvLSTM2D (ConvLSTM2D-Bi) comme une modèle de prédiction de flux de trafic.

3.2.4.2 Entraînement

Au cours de cette étape, nous allons entraîner notre modèle à apprendre à partir de nos données d'entraînement, afin d'améliorer progressivement leur capacité de prédiction,

jusqu' à obtenir un meilleur modèle entraîné, ce dernier est stocké pour être utilisé ultérieurement au moment du test.

3.2.5 Test et évaluation du modèle

Dans cette étape, nous obtenons une prédiction de flux de trafic à partir du modèle appris en faisant des données de test, et ainsi on peut évaluer la performance de notre modèle avec des métriques spécifiées aux problèmes de régression .

3.2.6 Résultat de prédiction

On va prédire le flux de trafic de 12 pas de temps à l'avance (c'est-à-dire 12 pas de temps d'horizons de prédiction de 5 minutes, correspondant au trafic de l'heure suivante).

3.3 La conception détaillée du système

Dans cette partie nous allons entamer la description des détails conceptuels relatifs à notre système de prédiction de flux de trafic. Ainsi, nous commencerons par détailler la première étape.

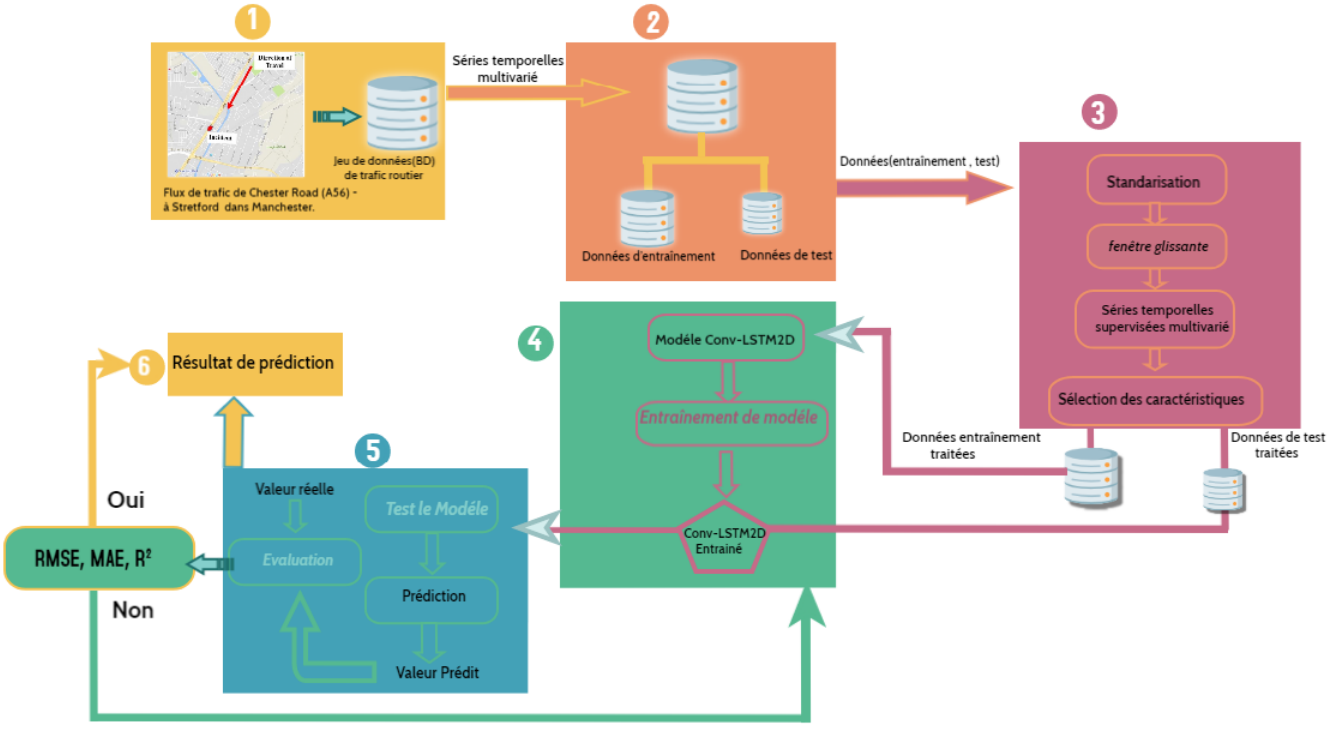

FIGURE 3.2 – L'architecture détaillés du système.

3.3.1 Identifier le dataset réelles (Entrée)

Le dataset que nous avons choisi concerne une zone d'étude choisie dans le Grand Manchester, au Royaume-Uni (UK). Notre scénario de ce dataset reproduit un accident routier historique le long d'un segment de la zone d'étude qui a entraîné la fermeture d'une voie et une forte congestion. L'accident s'est produit le 2 mai 2014, à 13 h 04. Les données relatives à l'accident ont été obtenues à partir d'une base de données en ligne [17], qui rapport les détails de l'accident, notamment la date et l'heure, le lieu et de l'accident, les coordonnées géographiques de la route concernée, la cause de l'incident, voie(s) fermée(s), etc. Nous avons choisi ce dataset[3] en raison de l'impact de les accidents de la route sur les réseaux de circulation urbaine en particulier aux heures de pointe. La Figure 3.3 montre un extrait de l'emplacement spécifique de l'accident le long du segment de route.

FIGURE 3.3 – Zone d'étude montrant le lieu de l'incident [18].

En outre, le dataset contenait 245 376 observations des cinq (5) variables - vitesse, débit, densité, précipitations et température, soit 852 jours de données d'entrée [18]. c'est un autre cause pour laquelle nous avons choisie ce dataset (volumineuse et multivariée c'est-à-dire elle contient des données sur le flux de trafic et la météo) afin d'avoir un apprentissage plus précis.

Les données de trafic utilisées dans cette étude ont été fournies par le Transport for Greater Manchester (TfGM), Les observations par minute des caractéristiques de flux de trafic (vitesse, débit et densité), recueillies à l'aide de capteurs à boucle inductive. La zone d'étude comprenait 10 capteurs de mesure du trafic, chacun d'entre eux étant espacé de 0,3 miles sur la route artérielle. La zone d'étude est une artère urbaine (Chester Road - A56) à Stretford, dans le Grand Manchester, au Royaume-Uni, dont les coordonnées de longitude et de latitude sont comprises entre (53.46281, -2.28398) et (53.43822, -2.31394). Les points de repère aux alentours sont les suivants stade de football de Manchester United - Old Trafford - en plus d'autres points de loisirs tels

que des centres commerciaux (Stretford Mall), des clubs, des restaurants, etc. Bien qu'il s'agisse d'une route "A", ce qui signifie qu'elle devrait être une autoroute ou une voie rapide, le tronçon considéré est limité à 30 mph car il s'agit d'un segment très fréquenté, avec de nombreux passages pour piétons, des commerces et des magasins, c'est un autre raison pour choisir ce dataset. Les données météorologiques obtenues pendant la période d'étude auprès du Centre for Atmospheric Studies (CAS) de l'Université de Manchester comprenaient des observations horaires de la température (Celsius), et des précipitations (mesurées en millimètres) [18].

En outre, notre dataset est sous forme de Fichier.csv de série temporelle de flux de trafic, utilisé comme entrée dans le processus d'apprentissage profond.

3.3.2 Découpage et Analyse de dataset

La division de dataset en ensembles d'entraînement, de validation et de test est la stratégie de division de l'ensemble de données la plus courante en apprentissage profond, le rapport optimal d'échantillons distribués dans chaque ensemble varie en fonction du problème. Dans notre étude on sépare le dataset en ensemble d'entraînement et de test tels que la plupart des données sont utilisées pour l'entraînement (90%), et une plus petite partie des données est utilisée pour le test (10%), et également (20%) de donnés d'entraînement utiliser pour la validation afin d'évaluer le modèle) (voir la Figure 3.4)

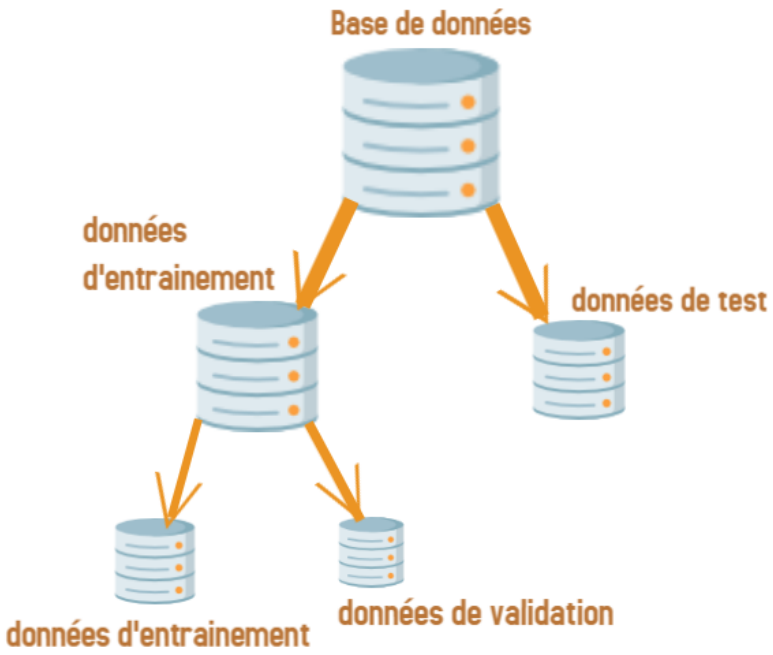

FIGURE 3.4 – La division de dataset .

Cette division permet des entraînements multiples en utilisant le même ensemble d'entraînement pour rechercher les hyperparamètres optimaux afin de maximiser la performance sur l'ensemble de validation. Lorsque la meilleure performance est obtenue sur l'ensemble de validation, le modèle est finalement utilisé une fois sur l'ensemble de test pour mesurer et confirmer la performance finale.

Après les analyse nous remarquons ce dataset ne contient pas des valeurs manquantes ou nulle, ainsi nous allons faire un prétraitement afin de préparer notre données pour faire l'entraînement et l'évaluation de modèle .

3.3.3 Pré-traitement et Sélection des caractéristiques de dataset

3.3.3.1 Pré-traitement

- Standardisation :

  Comme nous l'avons mentionné dans l'étape 1 (3.2.1), notre dataset est volumineux et multivariée qui contient 5 variables d'entrée (vitesse, débit, densité, précipita-

tions et température), avec des unités différentes (mph, veh/h, veh/mi, mm, °c) qui, à leur tour, peuvent signifier que les variables ont des échelles différentes. En raison de sa variété nous allons faire une standardisation des donnes d'entrée afin réduire la difficulté du problème modélisé, causés par la mise à jour du gradient et ainsi donner un modèle stable pendant l'apprentissage avec une bonne performance

.

La standardisation (Z-Score normalisation) d'un ensemble de données consiste à Re-dimensionner la distribution des valeurs observées de manière à ce que la moyenne μ soit égale à 0 et l'écart type σ à 1.

la formule de standardisation est la suivante :

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

Où x est les données d'entrée (features) originale.

- Conversion des données de séries temporelles multivariées aux séries temporelles supervisées :

  Notre dataset est à propos d'un série temporelle multivariée, ces dernier est souvent plus difficiles à traiter et plus difficiles à modéliser et de nombreuses méthodes classiques d'analyse de série temporelle ne donnent pas de bons résultats, en raison de ça nous allons convertir cette série temporelles multivariées à une série temporelle supervisée par la méthode de fenêtre glissante « Sliding window ».

  la fenêtre glissante est la base de la façon dont nous pouvons transformer n'importe quel ensemble de données de séries temporelles en un problème d'apprentissage supervisé, à partir d'utilisation le pas de temps précédent comme variables d'entrée (X) et le pas de temps suivant comme variables de sortie (y).

3.3.3.2 Sélection des caractéristiques

Après le prétraitement de données nous obtenons un dataset (fichier.csv) qui est prêt pour l'entraînement et le test , nous allons choisir pour chaque expérimentation, différents nombre de colonnes comme des caractéristiques (débit, température, précipitations, densité, vitesse) dans les variables d'entrée pour faire l'entraînement de notre

modèle.

3.3.4 Entraînement de modèle

L'entraînement du réseau de prédiction est l'un des travaux les plus importants, qui est directement lié à la performance finale de la prédiction.

Tous d'abord, nous allons commencer par la définition de notre modèle proposé ConvLSTM2D-Bi avant faire l'entraînement. Notre modèle est composé de deux modules : ConvLSTM2D comme un encodeur et Bi-LSTM comme un décodeur.

Le ConvLSTM2D 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 (deux LSTM unidirectionnels) est utilisé pour agréger les informations d'entrée dans le passé et le futur d'un 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 (décrite à la Figure 3.5).

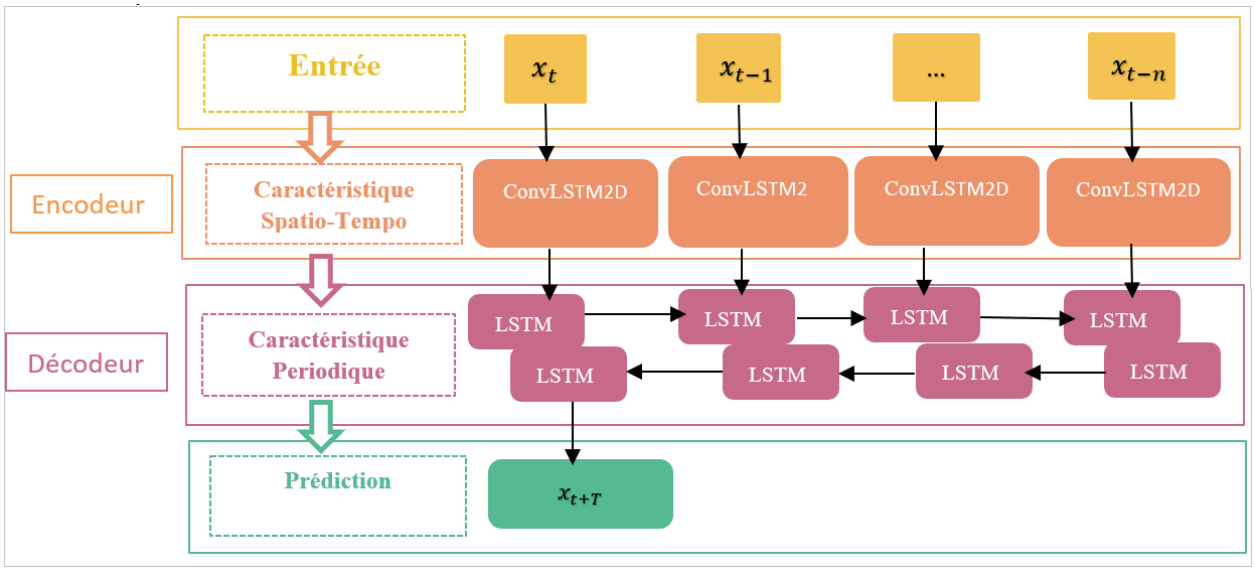

FIGURE 3.5 – Aperçu, Nous prenons l'information historique du trafic comme entrée et obtenons le résultat final de la prédiction par une mémoire convolutive à long-court terme et le modèle LSTM Bidirectionnel.

Le modèle comprend cinq éléments principaux : l'encodeur, le vecteur de répétition, l'aplatissement (Flatten), le décodeur et les couches entièrement connectées (Fully Connected FC), respectivement.

La première couche est la couche d'entrée, qui reçoit le vecteur d'entrée, comprenant les données relatives au trafic, à la météo (débit, température, précipitations, densité, vitesse) sous la forme d'un vecteur bidimensionnel m × n, où m représente le nombre d'échantillons dans l'ensemble de données d'apprentissage, et n représente le nombre de caractéristiques (les colonnes). Le deuxième ensemble des couches comprennent 3 couches de ConvLSTM2D qui forment l'ensemble de la couche encodeur, qui lit la séquence de vecteurs en entrée et extrait les caractéristiques spatio-temporelles par le filtre, Après la lecture de la dernière séquence, une Couche d'aplatissement est effectuée pour le convertir à vecteur unidimensionnel. Une couche de répétition de vecteurs est appliquée, qui - comme son nom l'indique - répète la séquence de vecteurs à reproduire par les couches encodeurs. La couche décodeur comprend 1 couche LSTM bidirectionnel qui reprend ensuite la séquence de la couche de répétition de vecteurs et produit une prédiction sous la forme d'une séquence de vecteurs à une seule ligne, Et ensuite une séquence de vecteurs de décodeur transmis à une couche FC, où la séquence cible est prédite. Entre chaque couche, une couche d'exclusion (dropout) est insérée afin d'éviter le sur-apprentissage (overfitting) et d'accélérer le processus d'apprentissage (voir la Figure 3.6).

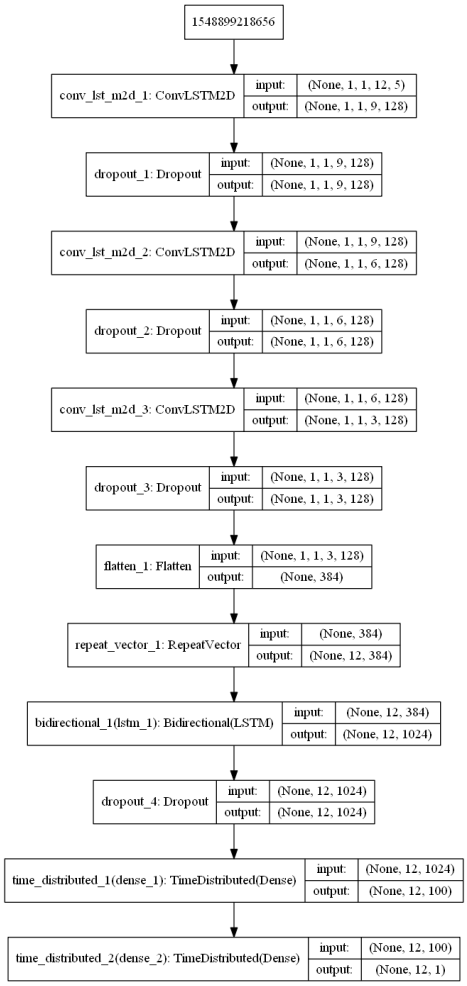

FIGURE 3.6 – Un aperçu de l'architecture du modèle proposé.

L'initialisation des paramètres et l'optimisation des hyper-paramètres du modèle a un impact important sur la performance globale de la prédiction. Nous avons utilisé l'algorithme de rétro-propagation par le temps (BPTT) pour entraîner les paramètres de notre modèle et Les détails des hyperparamètres utilisés dans cette étude sont présentés dans le tableau suivant :

Couches	*Paramètres*	*Hyperparamètres*
couche convolutive	Noyaux	Taille des kernels, nombre de kernels, stride, padding, fonction d'activation.
couche Bi-LSTM	poids , biais	Nombre de poids, fonction d'activation.
couche entièrement connectée	Poids	Nombre de poids, fonction d'activation.
autres	/	Taux d'apprentissage, fonction de perte, taille du époques, initialisation des poids, division l'ensemble de données.

FIGURE 3.7 – Les paramètres et hyperparamètres utilisés dans notre modèle

3.3.5 Test et évaluation de modèle

Idéalement, un ensemble de test n'est utilisé qu'une seule fois, à la toute fin du projet, afin d'évaluer les performances du modèle final qui est affiné et sélectionné au cours du l'entraînement avec des donnes d'entraînement. Comme on a déjà mentionné dans la section (3.2.4) nous allons utiliser la méthode et les métriques qui sont expliquées ci-dessous :

'Walk-Forward validation' est la meilleure méthode d'évaluation des modèles. C'est la validation croisée k-fold des séries temporelles, qui donnerait au modèle la meilleure chance de faire de bonne prédiction à chaque pas de temps. Le principe de cette approche (voir la Figure 3.8), en commençant au début de la série temporelle, un nombre minimum d'échantillons est utilisé pour entraîner un modèle et Le modèle fait une prédiction pour le pas de temps suivant, le résultat de prédiction est stocké ou évalué par rapport à la valeur connue et ensuite la taille échantillonnons est étendue pour inclure la valeur connue et le processus est répété pour la prédiction de chaque pas de temps suivante.

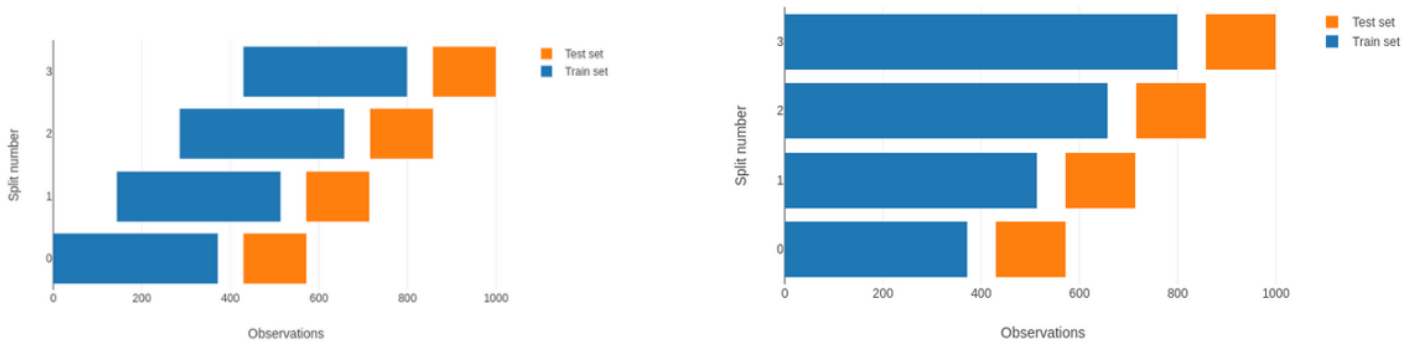

FIGURE 3.8 – L'approche de Walk-Forward validation.

Pour évaluer la précision de prédiction de notre modèle proposée, la racine de l'erreur quadratique moyenne (RMSE), Erreur absolue moyenne (MAE) ont été calculées et le carré du coefficient de corrélation de l'échantillon ou Coefficient de détermination (c'est-à-dire le carré r (R^2)) sont appliqués. Il s'agit de trois fonctions de coût couramment utilisées dans le domaine de l'apprentissage automatique, qui sont permettent d'évaluer les performances des modèles.

- RMSE est la racine carrée de la moyenne du carré de toutes les erreurs, elle est souvent utilisée comme norme pour mesurer les résultats de prédiction des modèles d'apprentissage automatique.
- MAE est représenté la moyenne de la différence absolue entre les valeurs réelles et prédites dans l'ensemble de données, elle mesure la moyenne des résidus dans l'ensemble de données. Une petite valeur pour ces critères (RMSE et MAE) signifie que le modèle estimé est proche de la valeur réelle.
- R2 est une mesure statistique d'ajustement pour tester l'ajustement du modèle, qui indique la quantité de variation d'une variable dépendante expliquée par la ou les variables indépendantes dans un modèle de régression. La valeur de 0 à 1 est interprétée comme des pourcentages. Plus la valeur est élevée, plus le modèle est bon.

Les formulations détaillées de ces métriques d'évaluation sont présentées ci-dessous :

$$RMSE = \sqrt{\frac{1}{n}\sum_{1}^{n}(y_i - \hat{y})^2} \tag{3.2}$$

$$MAE = \frac{1}{n}\sum_{1}^{n}|y_i - \hat{y}| \tag{3.3}$$

$$R^2 = 1 - \sum_{1}^{n}\frac{(\hat{y} - y_i)^2}{(y_i - \bar{y})^2} \tag{3.4}$$

Où , y_i , $\hat{y}$ sont respectivement la valeur prédite , la valeur réelle de base et $\bar{y}$ la moyenne des mesures, et n est le nombre de toutes les valeurs prédites.

3.3.6 Résultat de prédiction

Le résultat de prédiction est un vecteur de sortie de 12 pas de temps à l'avance correspondant au trafic (La vitesse dans notre étude) de l'heure suivante. Après cette prédiction on va comparer le résultat d'erreur (Loss) de notre modèle avec d'autres modèles existants dans littérature afin de valider la performance de notre modèle proposé.

3.4 Conclusion

A travers ce chapitre, nous avons présenté la conception globale de l'organisation de notre système. Ensuite, nous avons expliqué étape par étape le processus de notre travail à savoir l'identification de dataset, le découpage et analyse de ce dataset, le prétraitement, la sélection des caractéristiques et l'apprentissage jusqu'au résultat de prédiction.

Dans le chapitre suivant, nous allons présenter l'étude expérimentale afin de valider cette étude, on illustre l'implémentassions du modèle proposé, on va aussi décrire l'environnement de développement sur lequel nous avons développé le système que ce soit matériel ou logiciel. A la fin, nous allons discuter les différents résultats obtenus en les comparant avec d'autres modèles.

Chapitre 4

Étude expérimentale et résultats

4.1 Introduction

Dans le chapitre précédent, nous avons présenté notre modèle de prédiction de flux de trafic proposé afin d'améliorer la précision de la prédiction. Par ailleurs, nous avons détaillé les différentes étapes de conception de notre système pour atteindre cette prédiction.

Afin de réaliser notre système, nous avons eu recours à plusieurs outils de développement que ce soit logiciel ou matériel. Ce chapitre est consacré à la présentation de l'environnement de travail, le langage de programmation et les outils que nous avons utilisés pour construire ce système. Par la suite, nous allons expliquer toutes les expérimentations conduites sur la méthode proposée ainsi que la discussion des résultats obtenus.

4.2 Outils et langages de développement

4.2.1 Outils matériel

Nous avons réalisé notre système en utilisant une station graphique avec Windows de 64-bits , un processeur *Intel(R) Core (TM) i7-9700F CPU @ 3.00GHz*, 16 GB RAM et *NVIDIA GeForce RTX 2060M* avec la version 456.71 du pilote.

4.2.2 Outils logiciel

Pour le développement de notre système, nous avons utilisé la langage Python avec quelques bibliothèques bien connues travaillant sur l'apprentissage profond. Les différents environnements et outils utilisé sont présentés dans la Figure 4.1.

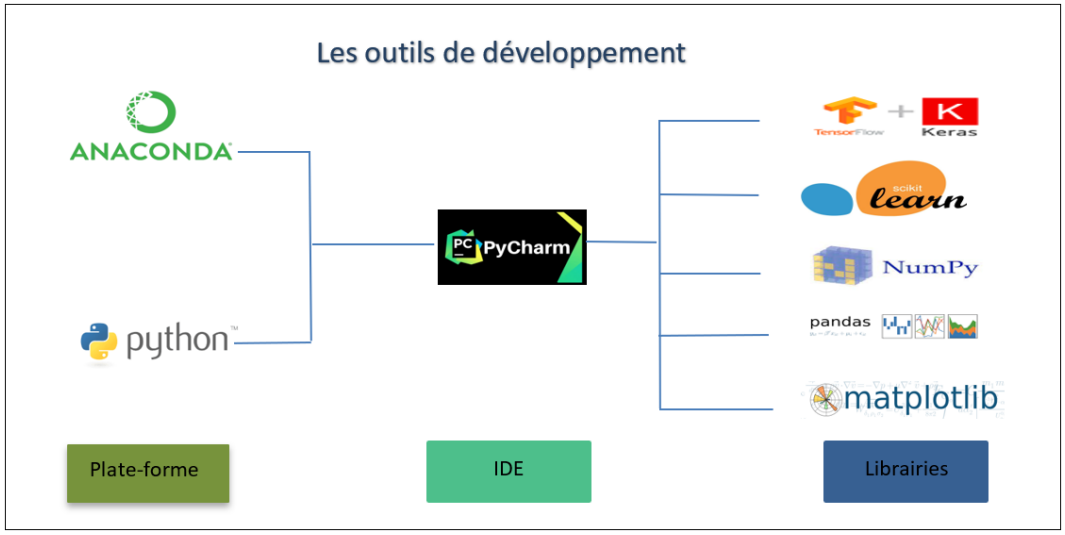

FIGURE 4.1 – Les outils de développement de notre système .

4.2.2.1 Plate-forme

- Anaconda

  Anaconda est une distribution gratuite et open-source des langages de programmation, livrée avec l'interpréteur Python et divers paquets liés à l'apprentissage automatique et à la science des données afin de permettre aux personnes intéressées par ces domaines d'installer facilement tous (ou la plupart) des paquets nécessaires en une seule installation et ainsi simplifier la gestion et le déploiement des paquets.

- Python

  Python est un langage de programmation 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. Parce que sa syntaxe est élégante, que son typage est dynamique et qu'il est interprété, Python est un langage idéal pour l'écriture de scripts et le développement rapide d'applications dans de nombreux

domaines et sur la plupart des plateformes. L'interpréteur Python et sa vaste bibliothèque standard sont disponibles librement, et peut être facilement étendu par de nouvelles fonctions et types de données implémentés en C ou C++ (ou tout autre langage appelable depuis le C).

4.2.2.2 IDE (environnement de développement intégré)

- PyCharm

  PyCharm est un environnement de développement intégré (IDE) Python dédié qui fournit une large gamme d'outils essentiels pour les développeurs Python, étroitement intégrés pour créer un environnement pratique pour le développement productif de Python, du web et des sciences des données.

4.2.2.3 Librairies

Les librairies ci-dessus sont appelés au début de programme pour utiliser des fonctions prédéfinies.

- Tensorflow

  TensorFlow est une bibliothèque logicielle open source pour le calcul numérique haute performance. Son architecture flexible permet de déployer facilement le calcul sur une variété de plateformes (CPU, GPU, TPU), et des ordinateurs de bureau aux clusters de serveurs en passant par les appareils mobiles et périphériques. Développé par des chercheurs et des ingénieurs de l'équipe Google Brain au sein de l'organisation d'IA de Google, il offre une prise en charge solide de l'apprentissage automatique et de l'apprentissage profond, et son noyau de calcul numérique flexible est utilisé dans de nombreux autres domaines scientifiques.

- Keras

  Keras est une bibliothèque Python open source puissante et facile à utiliser pour développer et évaluer des modèles d'apprentissage profond. Elle englobe les bibliothèques de calcul numérique efficaces Theano et TensorFlow et vous permet de définir et d'entraîner des modèles de réseaux neuronaux en quelques lignes de code seulement.

- Sckit-learn (Sklearn)

  Sklearn est la bibliothèque la plus utile et la plus robuste pour l'apprentissage automatique en Python. Elle fournit une sélection d'outils efficaces pour l'apprentissage automatique et la modélisation statistique, notamment la classification, la régression, le regroupement et la réduction de la dimensionnalité, via une interface cohérente en Python. Cette bibliothèque, qui est en grande partie écrite en Python, s'appuie sur NumPy, SciPy et Matplotlib.

- Numpy

  NumPy, qui signifie Numerical Python, est une bibliothèque composée d'objets de tableaux multidimensionnels et d'un ensemble de routines permettant de traiter ces tableaux. Numpy permet d'effectuer des opérations mathématiques et logiques sur des tableaux.

- Pandas

  Pandas est un paquetage Python qui fournit des structures de données rapides, flexibles et expressives conçues pour rendre le travail avec des données structurées (tabulaires, multidimensionnelles, potentiellement hétérogènes) et des séries temporelles à la fois facile et intuitif.

- Matplotlip

  Matplotlib est une bibliothèque complète pour créer des visualisations statiques, animées et interactives en Python. Elle produit des figures de qualité publication dans une variété de formats papier et d'environnements interactifs sur toutes les plateformes. Matplotlib peut être utilisée dans des scripts Python, dans le shell Python et IPython, dans des serveurs d'applications Web et dans divers kits d'outils d'interface utilisateur graphique.

4.3 Implémentation

Les étapes suivantes présentent l'implémentions de notre modèle d'apprentissage profond ConvLSTM2D-Bi qui sont mentionnés dans le chapitre précédent, avec Python et Keras essentiellement.

4.3.1 Identifier le dataset de flux de trafic routier (Entrée)

Le dataset utilisé pour réaliser notre système est expliqué au chapitre précédent (la section 3.2.1), il est sous la forme d'un fichier (.csv) avec le nom *'Trafficdataset.csv'* du taille (245376 lignes, 6 colonnes), sa forme originale est affichée ci-dessous :

	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
...	...	...	...	...	...	...
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

[245376 rows x 6 columns]

FIGURE 4.2 – Les données de flux de trafic .

La description des champs de données est présentée dans le tableau 4.1 :

Nom de champs	Description
Vitesse (mph) ‘ Miles par heure ‘	La vitesse du trafic, ou vélocité, v est définie comme la vitesse de déplacement d’un véhicule sur une distance donnée par unité de temps.
Densité (veh/mi) ‘Véhicules par mile ‘	Le nombre de véhicules par unité de longueur d’une voie ou d’un segment de route par instant de moment donné. Elle est exprimée en unités de trafic par unité de distance.
Débit (veh/h) ‘ Véhicule par heure ‘	Il est le taux de véhicules passant par un point ou un segment de route particulier. Il est exprimé en unités de trafic par unité de temps.
Température (°c) ‘Celsius ‘	Température des routes à utiliser pour prévoir si la neige ou le verglas vont tenir au sol et mesurent également les températures dans le sol.
Précipitations (mm) ‘Millimètres’	Apports d’eau parvenant au sol sous forme liquide (pluie ou rosée) ou solide (neige ou grêle) en provenance directe ou indirecte de la condensation de la vapeur d’eau atmosphérique.

Tableau 4.1 – la description des champs de dataset

4.3.2 Découpage de dataset

On divise notre dataset avec la fonction *split_ dataset()* en deux ensembles d’entraînement et de test, pour l’entraînement on prend 90% équivalant aux données du 01 janvier 2014 [00 :00 :00] au 06 février 2016 [18 :55 :00] (220836 Observations) et prend pour le test 10% équivalant aux données du 06 février 2016 [19 :00 :00] au 01 mai 2016 [00 :00 :00] (24540 observation), la portion de l’entraînement diviser aussi à 20% pour la validation du modèle. La fonction ci-dessous exprimer la division de dataset :

```
#Diviser l'ensemble de donnees (dataset) en ensembles d'entrainement
    et de test
def split_dataset(data):
     # divise en ensembles d'entrainement / test
     train, test = data[:-24540, :], data[-24540:, :]
     # restructurer les donnees en d'heures standard (12 valeur de 5
    min %)
     train = np.array(np.split(train, len(train) / 12))
```

```
7       test = np.array(np.split(test, len(test) / 12))
8     return train, test
```

Listing 4.1 – La fonction de division de dataset

Cette fonction est prise comme une entrée des données de notre dataset et les organise en heure standard à l'aide de la fonction *split ()* en la renvoyant dans la sortie.

4.3.3 Pré-traitement et Sélection des caractéristiques de dataset

4.3.3.1 Pré-traitement

- **Standardisation**

  À travers la standardisation, on change la forme de notre dataset à données des petites valeurs pour éviter les problèmes qui face au réseau neuronal durant l'entraînement, tels que le ralentissement d'apprentissage. Certaines données d'entraînement sont traitées à cette étape sont présentées au Figure 4.3

```
Donnes d'entrainement standardiser
 [[[ 0.0246675  -0.50368861 -0.55153371 -0.42245846]
  [-0.33546617 -0.50440888 -0.55154584 -0.3864451 ]
  [-0.09537706 -0.50648965 -0.55154638 -0.41197457]
  ...
  [-0.28744835 -0.49728624 -0.55154638 -0.50857043]
  [-0.35947509 -0.4915241  -0.55151303 -0.45951222]
  [-0.35947509 -0.49080383 -0.5498811  -0.40885341]]

 [[-0.23943053 -0.49144407 -0.54567153 -0.33466588]
  [-0.35947509 -0.49256449 -0.55071407 -0.33682668]
  [-0.383484   -0.48192053 -0.55154558 -0.30409453]
  ...
  [-0.40749291 -0.46663486 -0.55154638 -0.38468444]
  [-0.50352856 -0.46847554 -0.55154638 -0.45959225]
  [-0.35947509 -0.47487792 -0.53336629 -0.49336478]]
```

FIGURE 4.3 – Partie de données d'entraînement sont standardiser .

Nous avons faire la standardisation par les instructions suivantes :

```
def split_dataset(data):
# standardiser (train/test)
mu = np.mean(train) #la Moyenne de donnes de train
sig = np.std(train) #l'ecart type de train
dataTrainStandardized = (train - mu) / sig #datatrain standardiser
```

Listing 4.2 – Standardisation de données

- **Conversion des données de séries temporelles multivariées aux séries temporelles supervisée**

  Comme nous l'avons vu dans les chapitres précédents, un dataset de séries temporelles est une série de nombres triés par index temporel tandis que le dataset d'apprentissage supervisé se compose du mode d'entrée X et du mode de sortie Y. Par conséquent, le problème de prédiction des séries temporelles doit être reconstruit en un problème d'apprentissage supervisé d'une simple séquence à une paire d'entrées et de sorties de séquence, avant d'utiliser Deep Learning.

  Pour ce faire, nous pouvons suivre les index de début et de fin des entrées et des sorties et configurer une fenêtre temporelle qui se chevauchent, se déplace en fonction de sa longueur (un pas de temps de notre étude) de chaque itération de données et ainsi générer des ensembles de données sur une période de temps à chaque fois, chaque ensemble de données, sauf que la dernière ligne, est combiné pour former une entrée, et la dernière colonne (ou d'autres colonnes) de données de la dernière ligne forme la sortie correspondante (prédit les 12 valeurs de pas de temps suivants).

  Le format de données traitées est transformé aux formes X = [220813, 12, 5] et y= [220813, 12] à l'aide de la fonction *to_supervised()* est présentée ci-dessous

```
# convertir les édonnes historiques  en éentres et sorties par '
   Sliding window'
def to_supervised(train, n_input, n_out=12):
    train = np.array(train)
    # aplatir les édonnes (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 èaprs l'autre
    for _ in range(len(data)):
        #  édfinir la fin de la ésquence d'éentre
        in_end = in_start + n_input
        out_end = in_end + n_out #la fin de la ésquence de sortie
        # determiner les édonnes d'éentre et de sortie en cas (out_end <= len(data))
        if out_end <= len(data):
            X.append(data[in_start:in_end, :]) #prend la sequence avec tout les features comme en éentre
            y.append(data[in_end:out_end, 0]) #la sortie specifie a le flux de trafic

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

Listing 4.3 – La fonction de conversion de série temporelle au problème supervisé

Exemple :

```
la taille de trainX (220813, 12, 5)
la taille de trainY (220813, 12)
TrainX [[[ 2.46675028e-02 -5.03688615e-01 -5.51533710e-01 -4.22458464e-01
    1.99000000e+01]
  [-3.35466173e-01 -5.04408882e-01 -5.51545843e-01 -3.86445096e-01
    2.08000000e+01]
  [-9.53770559e-02 -5.06489655e-01 -5.51546379e-01 -4.11974572e-01
    2.62000000e+01]
  ...
  [-2.87448350e-01 -4.97286238e-01 -5.51546379e-01 -5.08570427e-01
    2.46000000e+01]
  [-3.59475085e-01 -4.91524100e-01 -5.51513031e-01 -4.59512217e-01
    1.87000000e+01]
  [-3.59475085e-01 -4.90803832e-01 -5.49881097e-01 -4.08853414e-01
    1.84000000e+01]]
```

```
TrainY [[-0.23943053 -0.35947509 -0.383484   ... -0.40749291 -0.50352856
  -0.35947509]
 [-0.35947509 -0.383484   -0.383484   ... -0.50352856 -0.35947509
  -0.43150182]
 [-0.383484   -0.383484   -0.28744835 ... -0.35947509 -0.43150182
  -0.35947509]
 ...
 [ 1.22511309  1.63326459  1.27313091 ...  1.92137153  0.86497941
   0.86497941]
 [ 1.63326459  1.27313091  1.53722894 ...  0.86497941  0.86497941
   1.22511309]
 [ 1.27313091  1.53722894  1.75330915 ...  0.86497941  1.22511309
   1.39317547]]
```

FIGURE 4.4 – Données d'entraînement supervisé.

4.3.3.2 Sélection des caractéristiques

Dans notre étude, nous allons faire des expérimentations avec différentes nombre de caractéristique pour faire la prédiction afin d'évaluer de notre modèle proposé, donc le choix de caractéristique est fait manuellement.

4.3.4 Entraînement du modèle

Tout d'abord, pour entraîner notre modèle nous commençons par leur construction et puis nous déterminerons l'hyperparamètre qui a joué un rôle majeur dans la capacité d'apprentissage. Ainsi, le choix des hyperparamètres appropriés peut nous aider à obtenir de meilleurs résultats.

4.3.4.1 Construire le modèle ConvLSTM2d-Bi

L'implémentation de notre modèle qui est détaillé dans la section 3.4.2, est implémenté par la fonction *build_ model ()* :

```
1 model = Sequential()
2     #couche 1
3     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))
 #couche 3
 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))
 #couche 4
 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=optimizers.Adam(lr=0.0001)
)
```

Listing 4.4 – l'implémentation de notre modèle proposer

4.3.4.2 Hyperparamètres

Les hyperparamètres (expérimentaux) impliqués dans notre modèle sont ajustés manuellement par différentes expérimentations jusqu'au atteindre un bon résultat de RMSE.

Nous avons commencé par l'utilisation d'une seule couche cachée de ConvLSTM2D, deux couches de ConvLSTM2D avec des changements différents en la Taille du lot (128, 256), nombre d'époque (100, 150) , le nombre de filtre (32, 64, 128) , taille de noyaux dans le convolution avec un taux d'apprentissage (1e-5), mais la valeur de

loss n'est pas diminuée, le même résultat dans le cas en ajoutons troisième couche de convLSTM2D, ensuite nous avons ajouté un deuxième module est Bi-LSTM on fixons le nombre de couche convolutif de LSTM a 3 ConvLSTm2D et nous travaillons avec une couche bidirectionnelle LSTM avec 128, 256 et 512 unité mais lorsque on prend 512 unité le modèle est donne un résultat satisfait avec une taille du lot 512 , si on ajoute une deuxième couche de Bi-LSTM le résultat n'est pas bon. Les différents résultats qui nous avons obtenu dans chaque expérimentation varier de 0.403 à 0.427, et lorsque nous changeons le taux d'apprentissage à 0.0001 (1e-4) le résultat de RMSE est diminué à 0.389 (en notre cas toutes les caractéristiques de dataset sont prises en compte). Le tableau suivant d'écrit différente hyperparamètres qui ont été utilisé dans notre modèle :

Hyperparamètres	**Valeur**
Taux d'apprentissage (lr)	0.0001
Taille du lot (Batch size)	512
Époque (Epochs)	200
Nombre de couches cachées ()	6
Nombre d'unités cachées (bi-LSTM, Dense, Dense)	512, 100, 1
Pas de temp (Timestamps)	12

Tableau 4.2 – L'Hyperparamètres de notre modèle ConvLSTM2D-Bi proposé.

4.3.4.3 Entraînement

La partie présentée dans la figure ci-dessous exprime l'entraînement de notre modèle :

```
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 èmodle
    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.hdf5')
    model.save('modelconvBi.h5')
```

Listing 4.5 – L'entrainement de notre modèle.

Dans l'entraînement de notre modèle, on a une couche d'entrée qui a été fixée à (None $\times$ 1 $\times$ 1$\times$ 12$\times$ 5) (dans le cas en travaille avec 5 features), Le nombre de couches cachées dans l'encodeur était de 3, avec 128 filtres de longueur 4 pour chaque couche de convolution, et Le nombre de couches cachées dans le décodeur était de 1 avec une unité cachée de 512, et une couche dense entièrement connectées de 100 unités.

Relu (unité linéaire rectifiée) a été choisie comme fonction d'activation de toutes les couches de modèle et le MSE (l'erreur quadratique moyenne) est la fonction de perte de régression la plus couramment utilisée qui calcule la somme des carrés de la distance entre la valeur prédite et la valeur réelle. nous utilisons 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 , et nous utilisons aussi le Modelcheckpoint pour sauvegarder le modèle ou ses poids dans un fichier HDF5 s'il y a une diminution de la valeur de loss sur l'ensemble de données de validation, sinon à travers la stratégie d'arrêt précoce (early stopping) (et le paramètre de patience est de 15)

l'entraînement est terminé après la valeur de patience.

4.3.5 Test et évaluation du modèle

4.3.5.1 Test du modèle

Dans l'étape de test on a téléchargé le modèle déjà entraîné et ses poids sous forme des fichier *(.h) et (.hdf5)* respectivement par les instructions ci-dessous, pour faire la prédiction sur les données de test avec la fonction *forecast ()* :

```
model = load_model("../Mult/modelconvBi.h5",
                        custom_objects=None,
                        compile=True)
model.load_weights("../Mult/best_weightsconvBi.hdf5")
return model
```

Listing 4.6 – Téléchergement le models et ses poids.

```
def forecast(model, history, n_steps, n_length, n_input):
    #flatten data pour d'obtenir 12 ésries temporelles èparallles.
    data = np.array(history)
    data = data.reshape((data.shape[0] * data.shape[1], data.shape
    [2]))
    #éércuprer les èdernires observations pour les édonnes d'éentre
    input_x = data[-n_input:, :] #les èdernires 12 valeur de flux de
    trafic
    # reshape l'éentre à (5) dim [1, n_input, 1]
    input_x = input_x.reshape((1, 1, 1, input_x.shape[0], input_x.
    shape[1]))
    #faire la éprdiction
    yhat = model.predict(input_x, verbose=0)
    # pour faire la éprdiction en vectorielle
    yhat = yhat[0]
    # print("yhat",yhat)
    return yhat
```

Listing 4.7 – la fonction de prédiction

Cette fonction prend comme Entrée le modèle entraîné sur l'ensemble de données d'entraînement, l'historique des données observées qui sont représenté un échantillon de 12 valeurs de 5 min (une heure précédente) des caractéristiques de flux de trafic et le nombre de pas de temps d'entrée attendu par le modèle afin de prédire dans la sortie l'heure suivante de flux de trafic.

4.3.5.2 Évaluation de modèle

Afin d'évaluer la performance notre modèle, nous utilisons la fonction *evaluate_ model()* qui implémente l'approche de 'Walk-Forward validation' :

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

Listing 4.8 – la fonction d'évaluation le modèle.

Cette fonction prend en entrée les ensembles de données d'entraînement et de test, le nombre de pas de temps, le nombre de séquence que nous va prédire et le nombre

d'observations antérieures. lors de cette fonction on a fait une appel à deux méthodes qu'on a déjà mentionnée : *build_ model()* pour construire un modèle à partir des données d'entraînement et *forecast()* pour faire des prédictions en utilisant l'historique pour chaque pas de temps de 5 min pour une heure et ensuite on va faire évaluation de chaque pas prédites avec la fonction *evaluate_ forecasts()* (voir la figure 4.11) qui permet de calculer le score totale et partiel de RMSE , MAE et R2 de prédiction de chaque 5min pour une heure et afficher comme une sortie de la fonction *evaluate_ modele()*.

La fonction *evaluate_ forecasts()* ci-dessous prend comme une entrée les donnes réelles et prédites et puis calculer les différente métriques (RMSE, MAE et R2) et retourne comme une sortie les résultats de calcul afin d'évaluer la performance de modèle.

```
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
```

Listing 4.9 – La fonction de perfeormance.

4.3.6 Expérimentations et discussion les résultats obtenus

Dans notre travail, nous allons faire trois expérimentations de prédiction différentes sur les variables d'entrée, ces expérimentations sont menées et évaluée séparément. Dans chaque expérimentation nous allons comparons notre modèle proposé convLSTM2D-Bi avec le modèle existante Attention based CNN-LSTM (Att-ba CNN-LSTM) en implémentant leur modèle d'écrit en [31] et avec notre modèle sans le module bidirectionnelle LSTM (Bi-LSTM), la comparaison est effectuée en termes d'efficacité dans les mêmes conditions.

L'implémentation de modèle Att-ba CNN-LSTM qui nous utilisons dans les trois expérimentations comme une modèle comparer est montrer dans la figure ci-dessous :

```
model = Sequential()
    model.add(Dense(100, activation='sigmoid', input_shape=(n_timesteps, n_features)))
    model.add(Conv1D(filters=64, kernel_size=4, input_shape=(n_timesteps, n_features)))
    model.add(BatchNormalization())
    model.add(LeakyReLU())
    model.add(Conv1D(filters=64, kernel_size=4))
    model.add(BatchNormalization())
    model.add(LeakyReLU())
    model.add(Conv1D(filters=32, kernel_size=4))
    model.add(BatchNormalization())
    model.add(LeakyReLU())
```

```
12      model.add(LSTM(128, activation='relu', return_sequences=True))
13      model.add(BatchNormalization())
14      model.add(LSTM(64, activation='relu', return_sequences=True))
15      model.add(BatchNormalization())
16      model.add(LSTM(64, activation='relu'))
17      model.add(BatchNormalization())
18      model.add(Dropout(0.2))
19      model.add(Dense(n_outputs))
```

Listing 4.10 – Le modèle attention based CNN-LSTM.

4.3.6.1 Expérimentation (1)

Dans cette première expérimentation, on prend ainsi que le flux de trafic (la vitesse) des intervalles précédents, les donnes de débit de notre dataset en tant que caractéristique de flux de trafic, pour faire la prédiction.

- **Résultat d'expérimentation (1) obtenues et Discussion**

  Dans notre modèle, nous avons calculé le flux de trafic pour une durée de 5 à 60 minutes c'est à dire les résultats est composer de 12 valeurs équivalentes à un prédiction d'1 heure' suivante. Les données réelles de flux de trafic (vitesse) et les résultats prédits et la variation de l'erreur (Loss) du notre modèle au cours de l'entraînement sont présentés dans les Figure 4.5,Figure 4.6 respectivement.

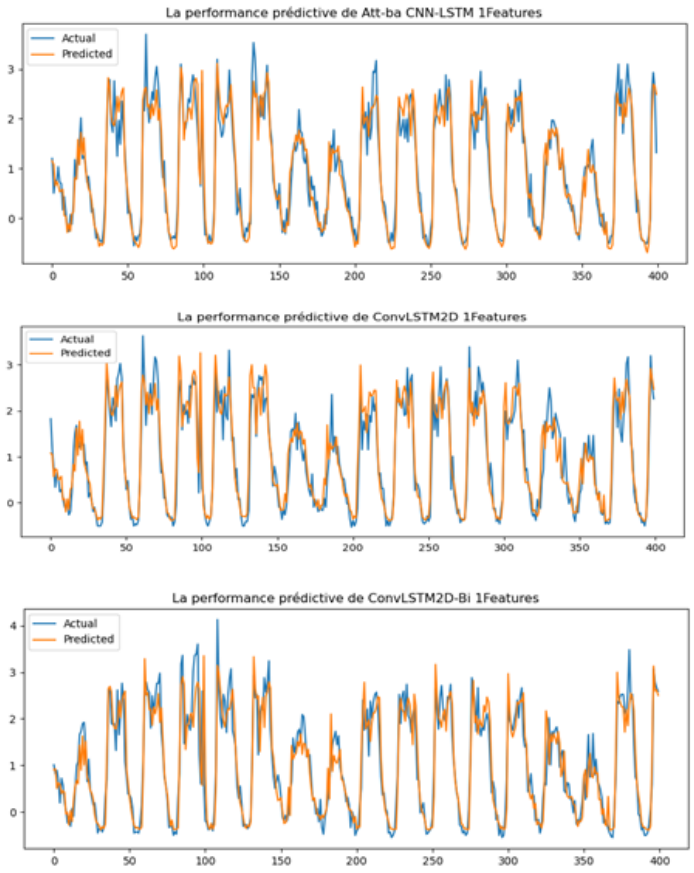

FIGURE 4.5 – Les différentes résultat de prédictions de chaque modèle.

Nous remarquons que les résultats de prédiction est un peu similaire au les donnes réelles de flux de trafic, et surtout avec notre modèle.

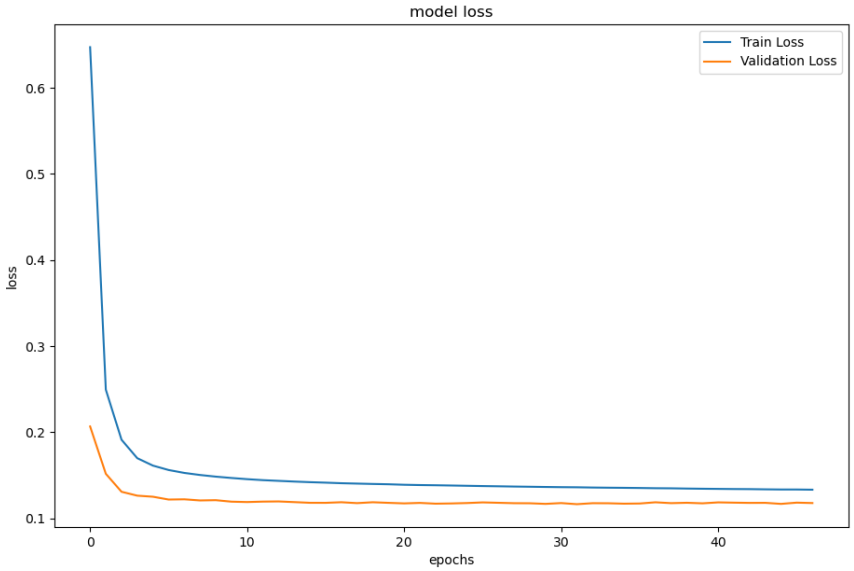

FIGURE 4.6 – la variation de l'erreur (Loss) du notre modèle au cours de l'entraînement.

Nous avons vu que les deux courbes de l'entraînement et validation sont diminuées et puis elles se stabilisent depuis l'époque numéro 11. L'erreur de notre modèle a pratiquement disparu après 46 Itérations d'entraînement.

Nous avons calculé aussi les erreurs et le coefficient de détermination de 3 modèles de prédiction : Att-ba CNN-LSTM, convLSTM2d et ConvLSTM2D-Bi. Les résultats de RMSE, MAE et R2 de chaque pas de temps de 5 à 60 min pour chaque modèle sont présentés dans la Figure A.3.

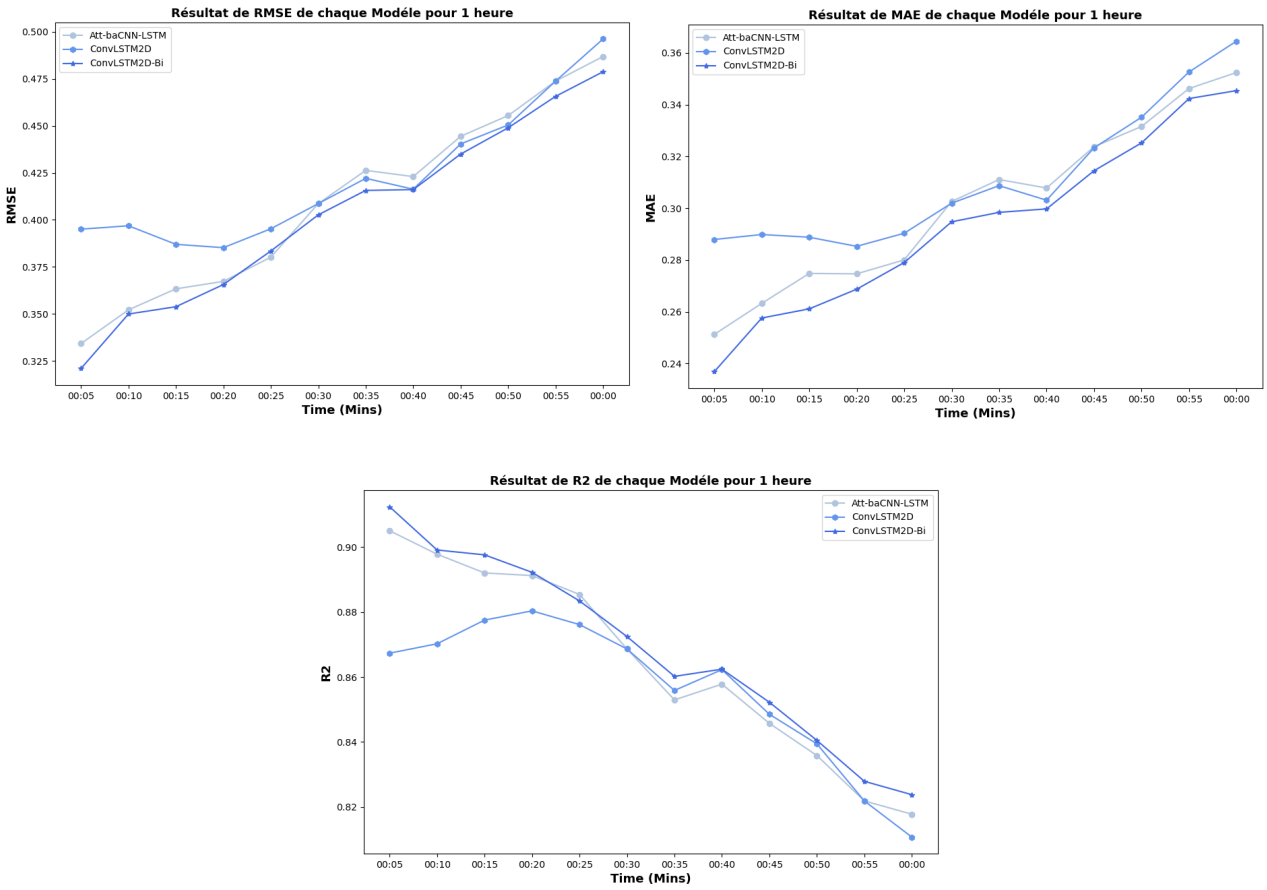

FIGURE 4.7 – Les résultats de RMSE, MAE et R2 pour différentes modèles (Expérimentation 1).

Nous constatons que pour chaque métrique d'erreur (RMSE et MAE) qui nous avons choisi pour faire la comparaison entre différents modèles, que notre modèle pour chaque pas de temps à prédire a une valeur minimal par rapport àles autres modèles ce que signifie que la précision de notre modèle est bien que les autres modèles, et les modèles Att-ba CNN-LSTM, ConvLSTM2D a chaque fois l'une est gain à l'autre.

La Figure 4.9 et le tableau 4.3 ci-dessous expriment les différents résultats totaux de chaque métrique en fonction des 3 modèles et La Figure 4.5 ci-dessous présente le résultat d'exécution :

```
Score de Att-ba CNN-LSTM pour 1 features
RMSE: [0.412] 0.334, 0.352, 0.363, 0.367, 0.380, 0.409, 0.426, 0.423, 0.444, 0.455, 0.474, 0.487
MAE: [0.302] 0.251, 0.263, 0.275, 0.275, 0.280, 0.303, 0.311, 0.308, 0.324, 0.332, 0.346, 0.352
R2: [0.864] 0.905, 0.898, 0.892, 0.891, 0.885, 0.869, 0.853, 0.858, 0.846, 0.836, 0.822, 0.818
Score de ConvLSTM2D pour 1 features
RMSE: [0.424] 0.395, 0.397, 0.387, 0.385, 0.395, 0.409, 0.422, 0.416, 0.440, 0.450, 0.474, 0.496
MAE: [0.311] 0.288, 0.290, 0.289, 0.285, 0.290, 0.302, 0.309, 0.303, 0.323, 0.335, 0.353, 0.364
R2: [0.857] 0.867, 0.870, 0.878, 0.880, 0.876, 0.869, 0.856, 0.862, 0.849, 0.839, 0.822, 0.811
Score de ConvLSTM-2D pour 1 features
RMSE: [0.406] 0.321, 0.350, 0.354, 0.366, 0.383, 0.403, 0.416, 0.416, 0.435, 0.449, 0.466, 0.479
MAE: [0.294] 0.237, 0.258, 0.261, 0.269, 0.279, 0.295, 0.298, 0.300, 0.314, 0.325, 0.342, 0.345
R2: [0.869] 0.912, 0.899, 0.898, 0.892, 0.883, 0.872, 0.860, 0.862, 0.852, 0.840, 0.828, 0.824

Process finished with exit code 0
```

FIGURE 4.8 – Le résultat total et individuel pour chaque pas de temps de différents modèles.

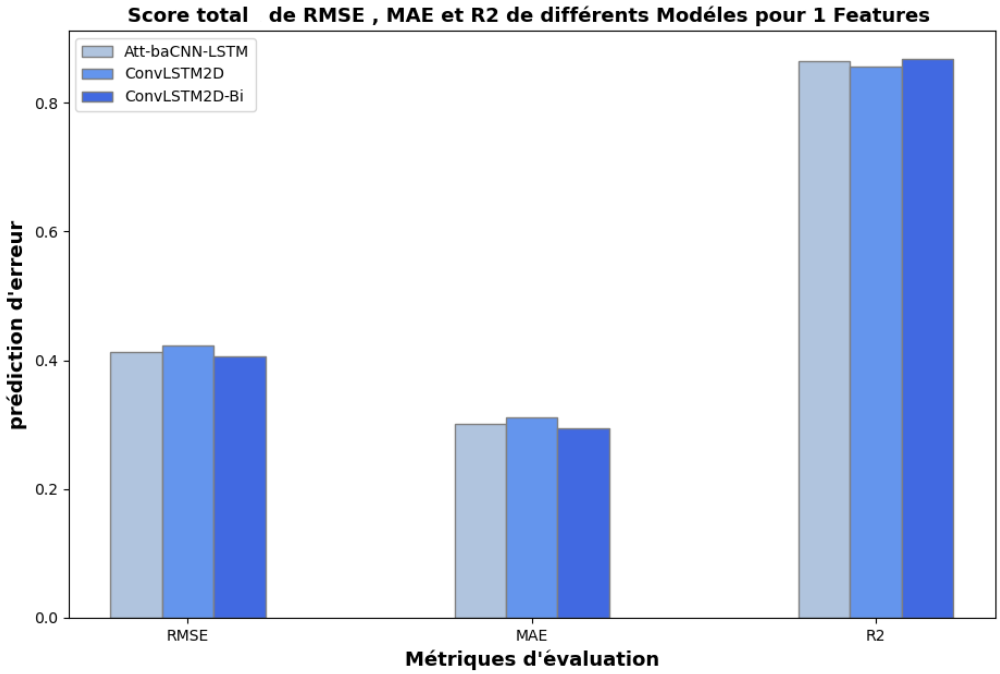

FIGURE 4.9 – Le résultat total de chaque métriques d'évaluation pour différentes modèle .

Modèle prédicatifs	Métrique d'evaluation		
	RMSE	MAE	R2
Att-ba CNN-LSTM [2021]	0.412	0.302	0.864
ConvLSTM2D	0.426	0.311	0.857
ConvLSTM2D-Bi LSTM	0.406	0.294	0.869

Tableau 4.3 – Comparaison des résultats de différentes modèle.

Notre modèle a une petite valeur de RMSE et MAE : 0.406 et 0.294 inferieur que les deux autres valeurs, ce qui montre que la précision de prédiction de notre modèle est meilleure que celles des deux autres modèles.

4.3.6.2 Expérimentation (2) : Les données de trafic

Dans cette expérimentation, on inclut dans l'entraînement des 3 modèles, la troisième caractéristique de flux de trafic (la densité) avec les données entraînées lors de la première expérimentation.

- **Résultat d'expérimentation (2) obtenues et Discussion**

  La Figure 4.10 qui représentés les données réelles de flux de trafic et les résultats prédits

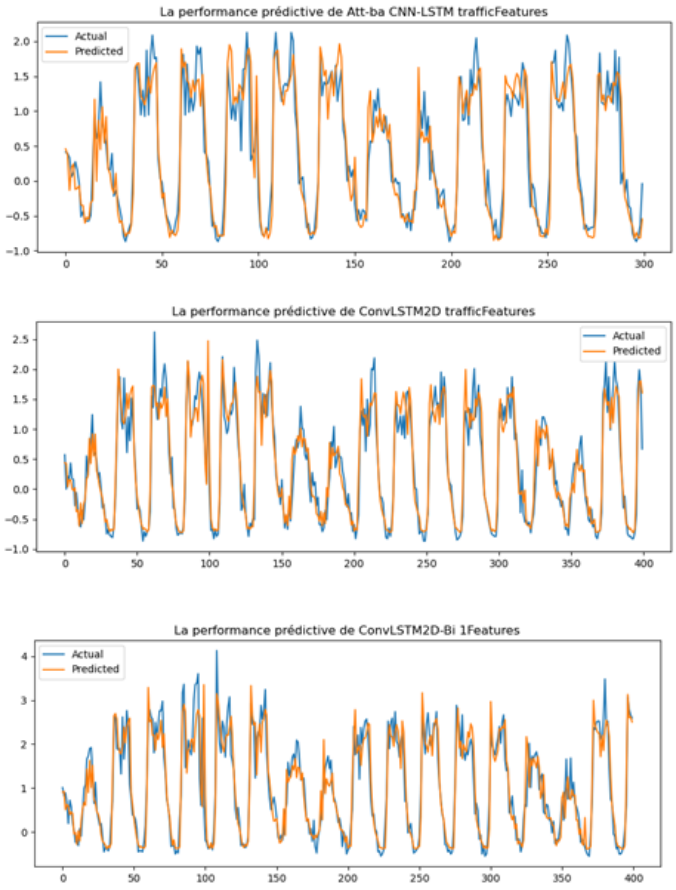

FIGURE 4.10 – Les différentes résultat de prédictions de chaque modèle .

Nous remarquons qu'il y a une similarité élever entre les donnes prédites et les données réelles par rapport à ce que nous avons dans la première expérimentation. La variation de l'erreur (Loss) du notre modèle au cours de l'entraînement est présentés dans la Figure 4.11.

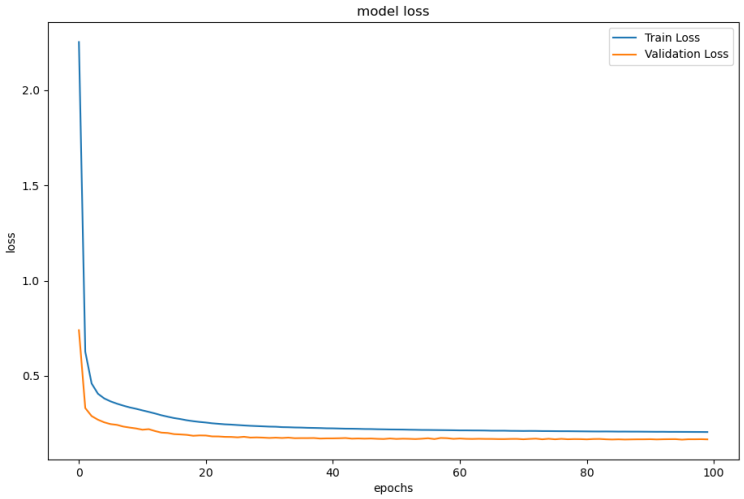

FIGURE 4.11 – Les différentes résultat de prédictions de chaque modèle .

A travers les différentes graphes précédentes associe à cette expérimentation (2), et le résultat de RMSE , MAE et R2 pour chaque modèle, que ce soit les résultat ci-dessous de chaque pas de temps ou le score totale et le tableau d'évaluation , qui sont représenter dans les Figure 4.12, Figure 4.14 et le tableau 4.4 respectivement, Nous pouvons notez que notre modèle proposé a une valeur d'erreur inférieur à celui des autres (Att-ba CNN-LSTM et ConvLSTM2D) et ainsi notre modèle est performant dans la prédiction des différentes pas de temps.

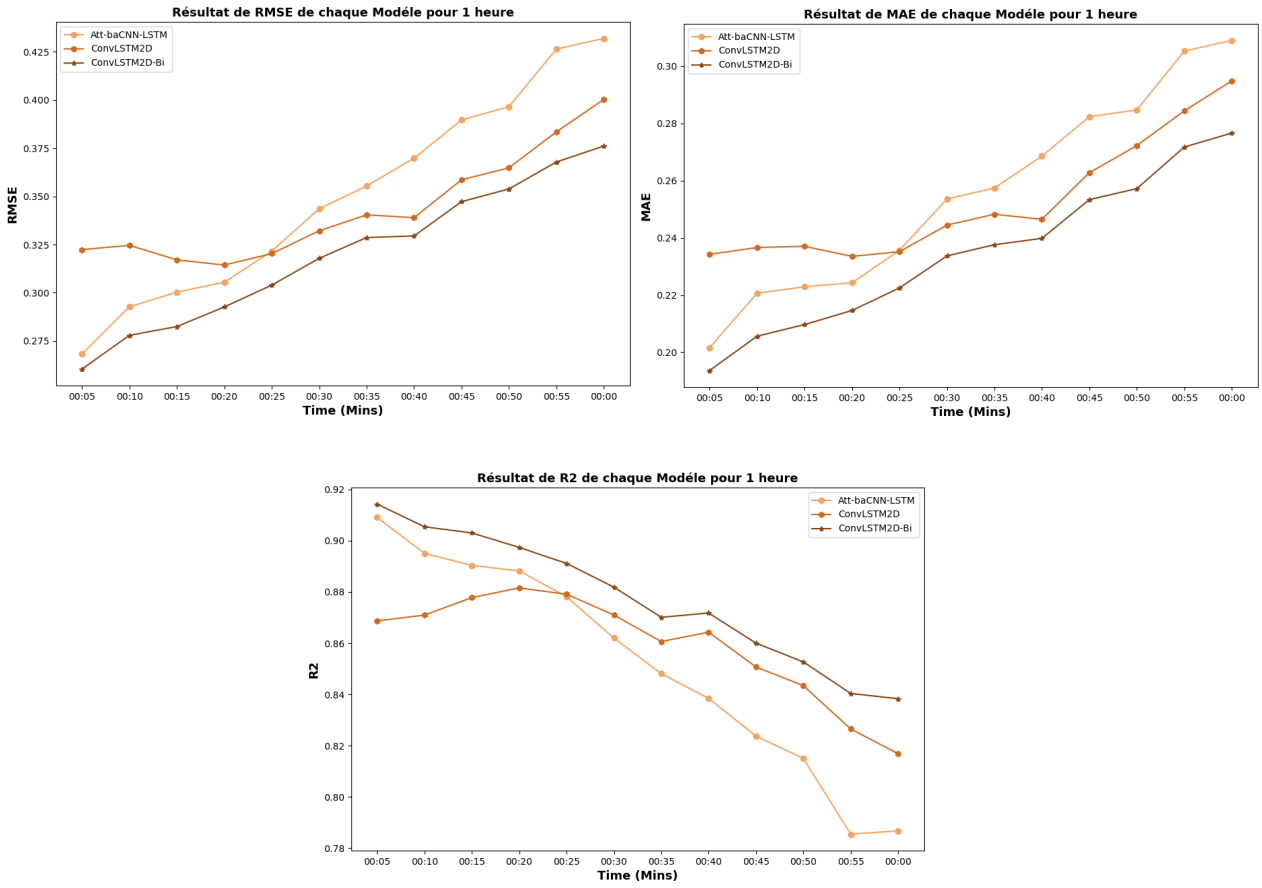

FIGURE 4.12 – Les résultats de RMSE, MAE et R2 pour différentes modèles (Expérimentation 2).

La Figure 4.13 ci-dessous présente le résultat d'exécution dans cette expérimentation(2) :

```
Score de Att-ba CNN-LSTM pour trafficfeatures
RMSE: [0.354] 0.268, 0.293, 0.300, 0.305, 0.322, 0.344, 0.355, 0.370, 0.390, 0.397, 0.426, 0.432
MAE: [0.255] 0.202, 0.221, 0.223, 0.224, 0.236, 0.254, 0.257, 0.269, 0.282, 0.285, 0.305, 0.309
R2: [0.852] 0.909, 0.895, 0.890, 0.888, 0.878, 0.862, 0.848, 0.838, 0.824, 0.815, 0.786, 0.787
Score de ConvLSTM2D pour trafficfeatures
RMSE: [0.344] 0.322, 0.325, 0.317, 0.314, 0.320, 0.332, 0.340, 0.339, 0.359, 0.365, 0.383, 0.400
MAE: [0.252] 0.234, 0.237, 0.237, 0.234, 0.235, 0.244, 0.248, 0.246, 0.263, 0.272, 0.284, 0.295
R2: [0.859] 0.869, 0.871, 0.878, 0.882, 0.879, 0.871, 0.861, 0.864, 0.851, 0.843, 0.827, 0.817
Score de ConvLSTM2D-Bi pour trafficfeatures
RMSE: [0.322] 0.260, 0.278, 0.282, 0.293, 0.304, 0.318, 0.329, 0.329, 0.347, 0.354, 0.368, 0.376
MAE: [0.235] 0.194, 0.206, 0.210, 0.215, 0.222, 0.234, 0.238, 0.240, 0.253, 0.257, 0.272, 0.277
R2: [0.877] 0.914, 0.905, 0.903, 0.897, 0.891, 0.882, 0.870, 0.872, 0.860, 0.853, 0.840, 0.838
[0.32183683]

Process finished with exit code 0
```

FIGURE 4.13 – Le résultat total et individuel pour chaque pas de temps de différents modèles

.

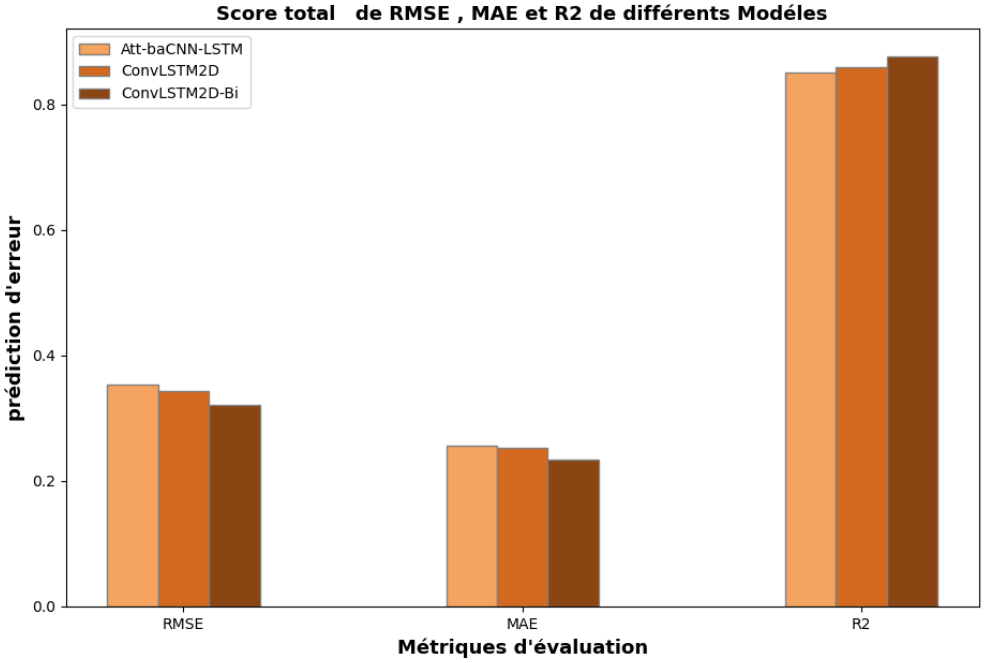

FIGURE 4.14 – Le résultat total de chaque métriques d'évaluation pour différentes modèle .

Modèle prédicatifs	Métrique d'évaluation		
	RMSE	MAE	R2
Att-ba CNN-LSTM [2021]	0.354	0.255	0.852
ConvLSTM2D	0.344	0.252	0.859
ConvLSTM2D-Bi LSTM	0.322	0.235	0.877

Tableau 4.4 – Comparaison des résultats de différentes modèle.

Nous remarquons aussi lors de cette expérimentation, il y a clairement une amélioration significative dans la valeur d'erreur dans notre modèle tel qu'est réduite ($0.\ 322 < 0.406$), par conséquent le modèle entraîné en utilisant tous les donnes de caractéristique de flux de trafic (vitesse, débit et densité) est plus performante que le modèle entraîné en utilisant uniquement (l'historique de vitesse et les donnes de débit).

4.3.6.3 Expérimentation (3) : Les données de trafic et la météo

Dans cette expérimentation, on ajoute aux données de trafic, les donnes de météo (température et précipitation) durant l'étape de l'entraînements afin d'évaluer la précision de notre modèle dans le cas on a d'entrée multivariée.

- **Résultat d'expérimentation (3) obtenues et Discussion**

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

  La Figure 4.15 représente les données réelles de flux de trafic et les résultats prédits pour chaque modèle.

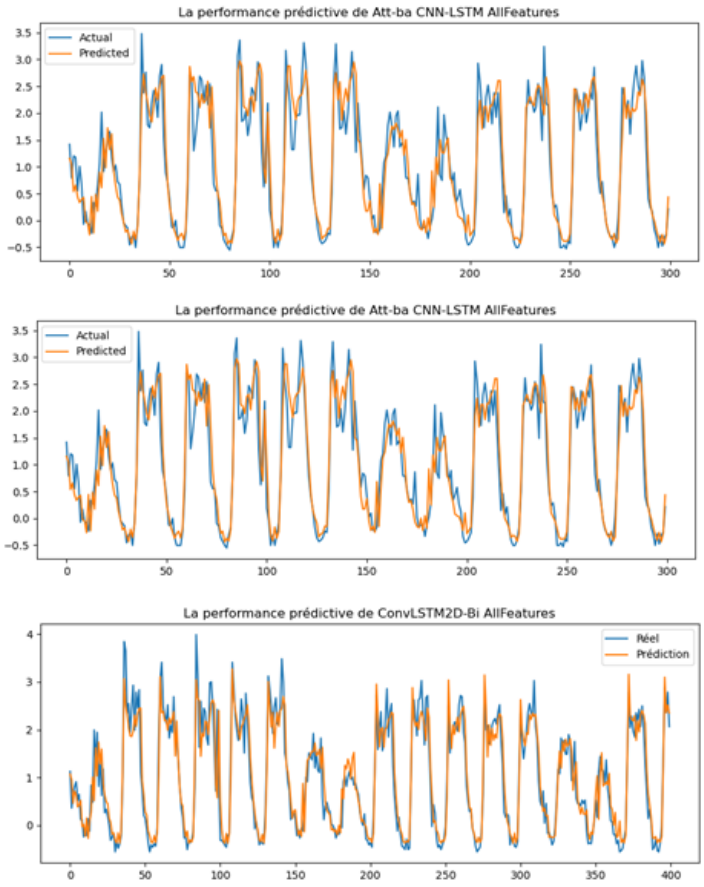

FIGURE 4.15 – Les différentes résultat de prédictions de chaque modèle.

La variation d'erreur (Loss) du notre modèle au cours de l'entraînement sont présentés dans la Figure 4.16 .

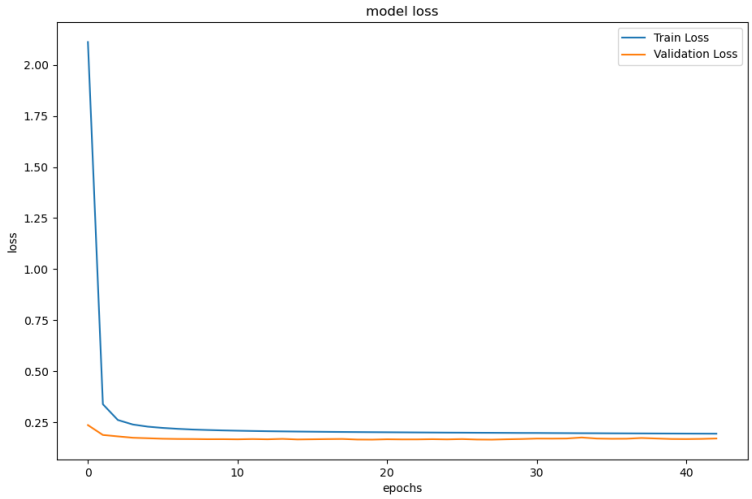

FIGURE 4.16 – la variation de l'erreur (Loss) du notre modèle au cours de l'entraînement.

Le résultat de RMSE, MAE et R2 pour chaque pas de temps de 5 à 60 min de différentes modèle est présenter dans la Figure 4.17 :

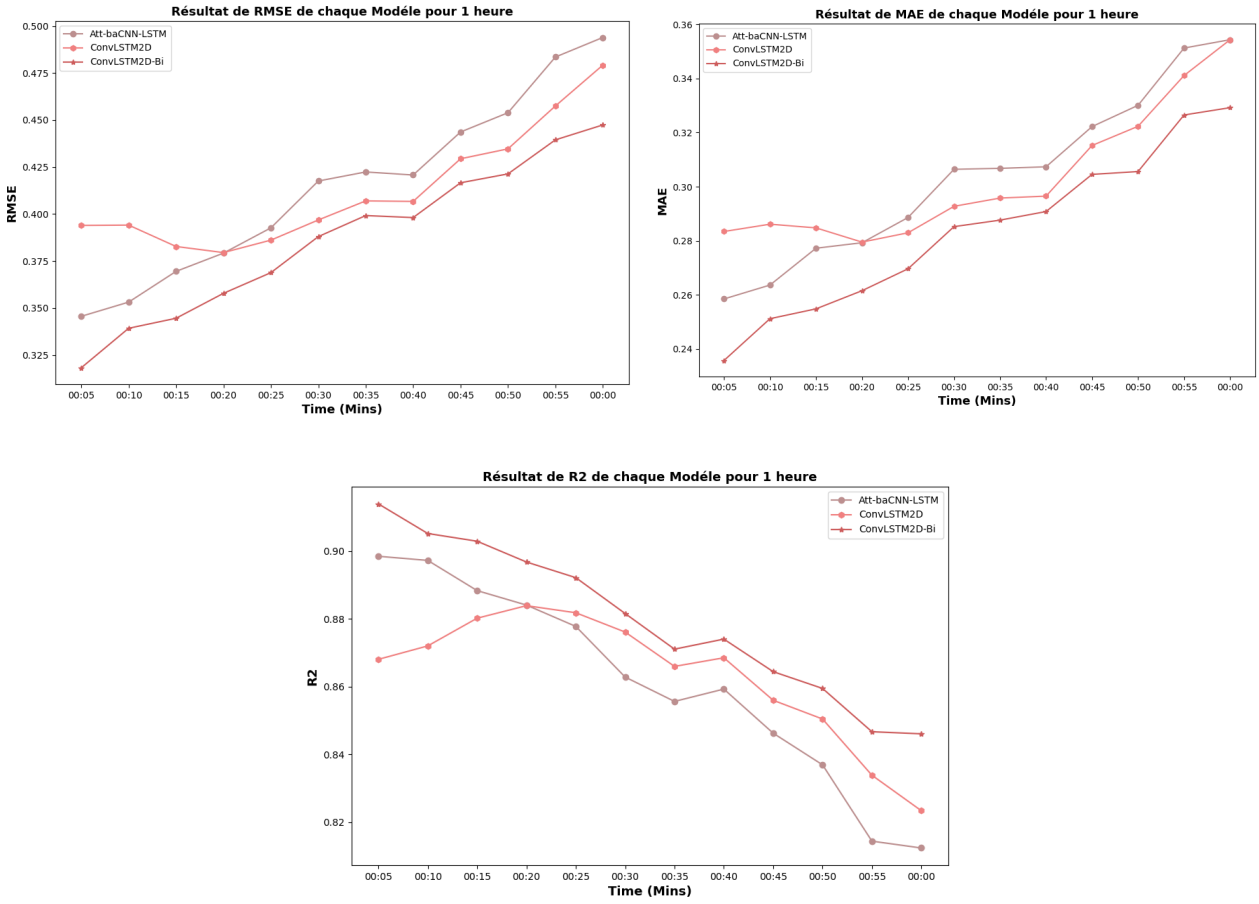

FIGURE 4.17 – Les résultats de RMSE, MAE et R2 pour différentes modèles (Expérimentation 3).

Nous allons voir le résultat d'exécution totale et le résultat pour chaque pas de temps (voir la Figure 4.18)

```
Score de Att-ba CNN-LSTM pour Allfeatures
RMSE: [0.417] 0.346, 0.353, 0.370, 0.379, 0.393, 0.418, 0.422, 0.421, 0.444, 0.454, 0.484, 0.494
MAE: [0.304] 0.258, 0.264, 0.277, 0.279, 0.289, 0.306, 0.307, 0.307, 0.322, 0.330, 0.351, 0.354
R2: [0.861] 0.898, 0.897, 0.888, 0.884, 0.878, 0.863, 0.856, 0.859, 0.846, 0.837, 0.814, 0.812
Score de ConvLSTM2D pour Allfeatures
RMSE: [0.413] 0.394, 0.394, 0.383, 0.380, 0.386, 0.397, 0.407, 0.407, 0.429, 0.435, 0.457, 0.479
MAE: [0.303] 0.283, 0.286, 0.285, 0.280, 0.283, 0.293, 0.296, 0.297, 0.315, 0.322, 0.341, 0.354
R2: [0.863] 0.868, 0.872, 0.880, 0.884, 0.882, 0.876, 0.866, 0.869, 0.856, 0.850, 0.834, 0.823
 Score de ConvLSTM2D-Bi pour Allfeatures
 RMSE: [0.389] 0.318, 0.339, 0.345, 0.358, 0.369, 0.388, 0.399, 0.398, 0.417, 0.421, 0.439, 0.447
 MAE: [0.284] 0.236, 0.251, 0.255, 0.262, 0.270, 0.285, 0.288, 0.291, 0.305, 0.306, 0.327, 0.329
 R2: [0.880] 0.914, 0.905, 0.903, 0.897, 0.892, 0.882, 0.871, 0.874, 0.864, 0.859, 0.847, 0.846

 Process finished with exit code 0
```

FIGURE 4.18 – Le résultat total et individuel pour chaque pas de temps de différents modèles.

La Figure 4.19 et le tableau 4.5 ci-dessous expriment les différents résultats totaux de chaque métrique en fonction des 3 modèles prédicatif.

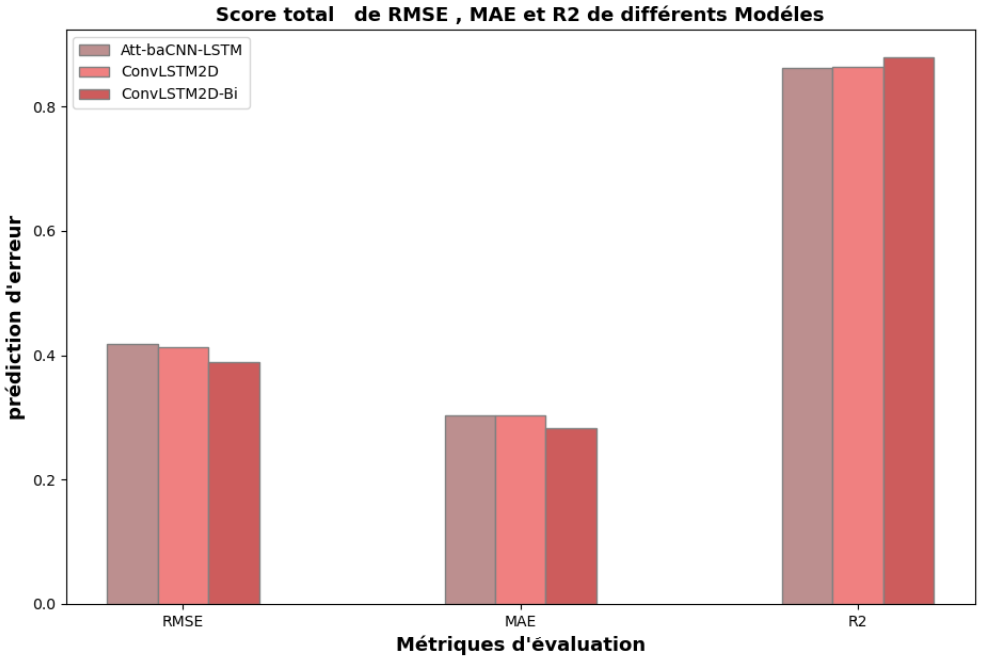

FIGURE 4.19 – Le résultat total de chaque métriques d'évaluation pour différentes modèle .

Modèle prédicatifs	Métrique d'evaluation		
	RMSE	MAE	R2
Att-ba CNN-LSTM [2021]	0.417	0.304	0.861
ConvLSTM2D	0.413	0.303	0.863
ConvLSTM2D-Bi LSTM	0.389	0.284	0.880

Tableau 4.5 – Comparaison des résultats de différentes modèle.

A partie de cela, l'erreur de notre modèle de prédiction ConvLSTM-Bi était plus faible que les autres modèles comparés dans chaque expérimentation, par conséquences notre modèle est performant par rapport les deux autres modèles.

4.3.7 Discussion

Pour chaque expérimentation, notre modèle reste le modèle le plus performante que les autres modèles comparés mais on remarque que le résultat de prédiction des premiers pas de temp (5min, 10 min ...etc.) sont moins que faible le résultat de pas de temps suivant. En outre, plus le pas de temps de la prédiction est petit, plus la précision de la prédiction est élevée et meilleure sinon la précision de la prédiction diminue légèrement, car dans la prédiction à plusieurs étapes doit utiliser les résultats de la prédiction précédente, et le résultat de cette dernière n'est pas exacte il doit y avoir une erreur entre la valeur prédite et la valeur réelles.

La valeur de RMSE et MAE qui nous utilisons pour la comparaison avec d'autre modèle sont variée pour chaque expérimentation, ce que signifie que les variables d'entrée dans l'entraînement sont jouée un rôle très important dans la précision de modèle prédicative finale, tel que dans l'expérimentation (1) ou uniquement les donnes de débit sont entraînées avec l'historique de donnes de flux de trafic qui donne une précision de prédictions moins que l'expérimentation (2) et (3), et lorsque nous entraînons notre modèle avec tous les donnes de caractéristique de flux de trafic nous obtenons une précision meilleure que ce que nous obtenons dans l'expérimentation (1) et (3) .En

raison, on peut réfère ça à la corrélation entre les variables indépendantes d'entrée et le variable qui nous allons prédire (sortie) . En d'autres termes il y a des variables d'entrée régalement a un effet sur la variable de sortie mais avec un petit pourcentage (le débit et précipitation), ça influence négativement sur la précision de prédictions finale de notre modèle. La matrice de corrélation ci-dessous qui montre ce que nous avons dite :

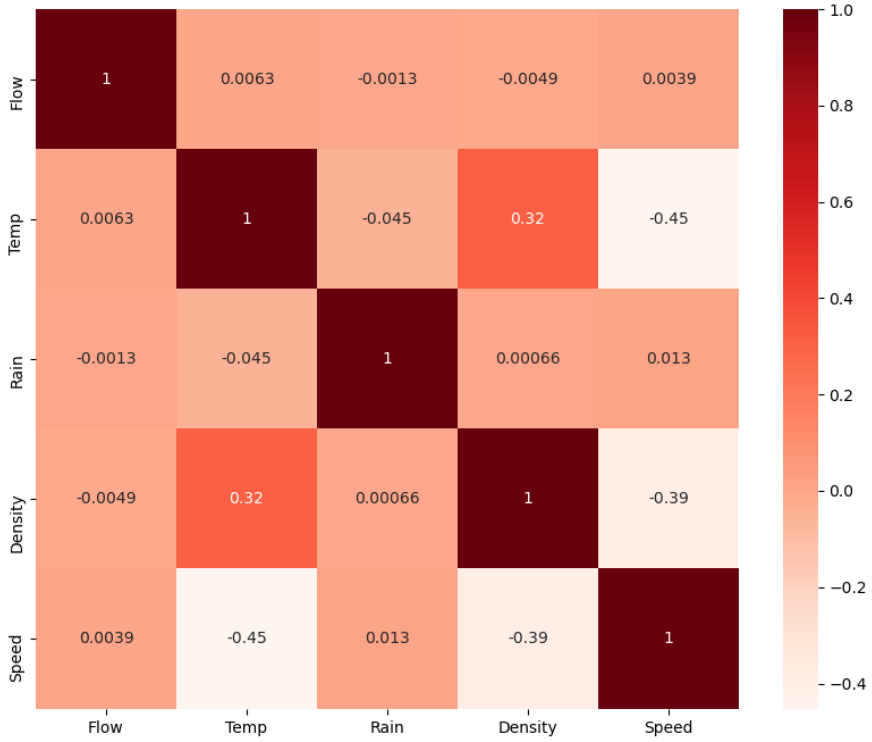

FIGURE 4.20 – Matrice de corrélation de Pearson (Heatmap).

Cette matrice a des valeurs comprises entre -1 et 1.

— Une valeur proche de 0 implique une corrélation plus faible (la valeur exacte 0 implique une absence de corrélation).

— Une valeur proche de 1 implique une corrélation positive plus forte.

— Une valeur proche de -1 implique une corrélation négative plus forte.

Nous remarquons, la température et la densité a une des corrélation négative plus forte

avec la vitesse, avant entraîner les modèles avec ces variables d'entrées nous notez qu'il y a une corrélation entre ces deux variables, donc il faut supprimer une entre eux, à la fin nous obtenons une variable d'entrée qui est : la température. Nous entraînons les 3 modèles, on se trouve les résultats ci-dessous dans les figures 4.29 , 4.30 et 4.31.

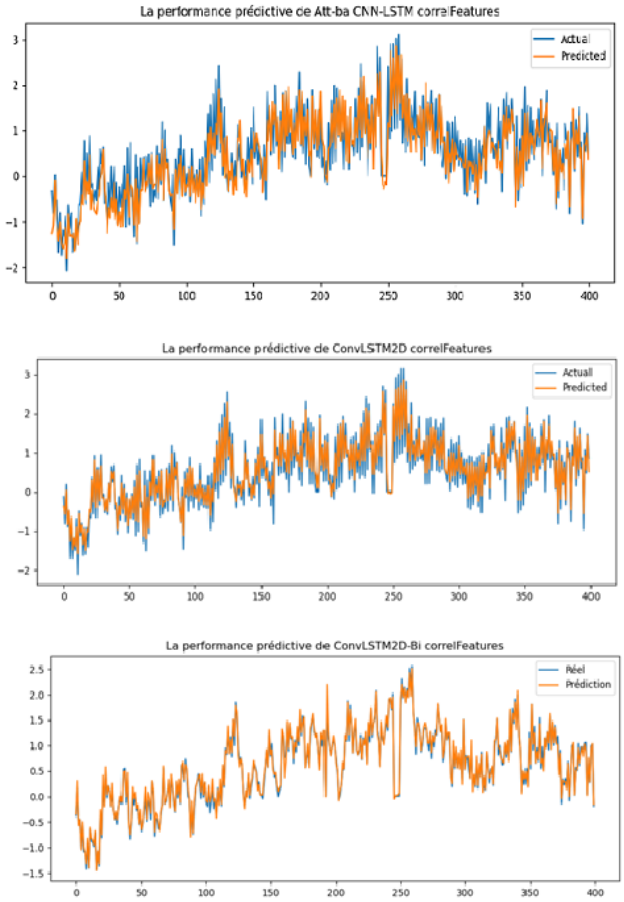

FIGURE 4.21 – Les différentes résultat de prédictions de chaque modèle.

Nous constatons que notre modèle a une bonne similarité à les donnes réelles.

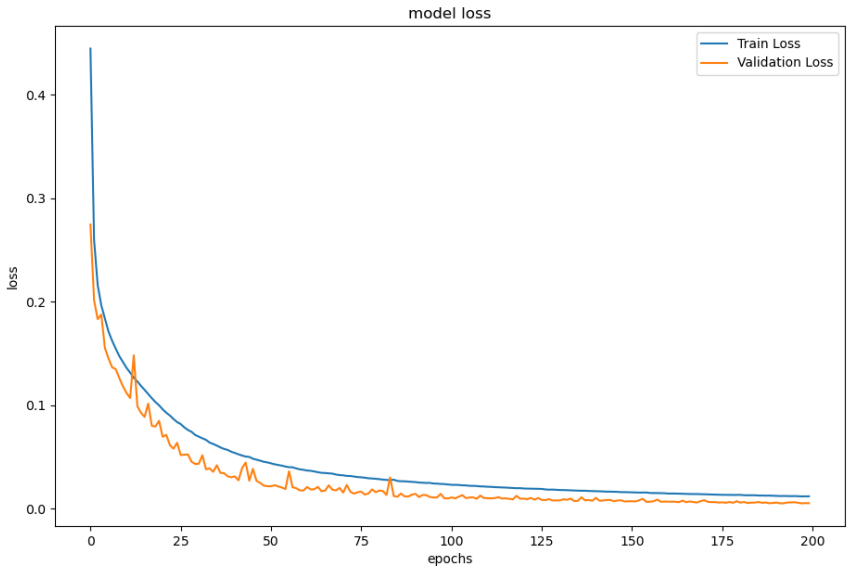

FIGURE 4.22 – la variation de l'erreur (Loss) du notre modèle au cours de l'entraînement.

Les résultats d'exécution de chaque modèle sont affichés ci-dessous :

Att-ba CNN-LSTM

```
RMSE: [0.280] 0.146, 0.179, 0.209, 0.235, 0.259, 0.281, 0.305, 0.322, 0.328, 0.329, 0.331, 0.344
MAE: [0.209] 0.104, 0.132, 0.160, 0.178, 0.198, 0.215, 0.237, 0.249, 0.255, 0.257, 0.257, 0.268
R2: [0.919] 0.977, 0.967, 0.957, 0.947, 0.936, 0.926, 0.911, 0.899, 0.888, 0.881, 0.876, 0.867
```

ConvLSTM2D

```
RMSE: [0.255] 0.246, 0.222, 0.219, 0.218, 0.217, 0.212, 0.202, 0.185, 0.163, 0.217, 0.337, 0.468
MAE: [0.189] 0.188, 0.163, 0.162, 0.167, 0.172, 0.166, 0.157, 0.145, 0.127, 0.166, 0.272, 0.381
R2: [0.931] 0.935, 0.950, 0.953, 0.954, 0.955, 0.958, 0.961, 0.966, 0.972, 0.949, 0.870, 0.753
```

ConvLSTM2D-Bi LSTM

```
RMSE: [0.072] 0.066, 0.060, 0.065, 0.071, 0.080, 0.074, 0.075, 0.075, 0.070, 0.072, 0.072, 0.080
MAE: [0.054] 0.052, 0.046, 0.050, 0.051, 0.058, 0.056, 0.057, 0.059, 0.056, 0.055, 0.053, 0.061
R2: [0.995] 0.995, 0.996, 0.996, 0.995, 0.994, 0.995, 0.995, 0.994, 0.995, 0.994, 0.994, 0.993

Process finished with exit code 0
```

FIGURE 4.23 – Le résultat total et individuel pour chaque pas de temps de différents modèles

.

Les valeurs de RMSE et MAE de notre modèle sont diminuées à 0.072 et 0.054 par rapport ce que nous avons obtenu dans les expérimentations précédentes, ce que signifie que lorsque nous allons faire de l'entraînement avec les variables approprie, notre modèle va donner une précision de prédiction plus élevée.

4.4 Conclusion

Dans ce dernier chapitre, nous avons présenté les différentes environnements matériels ou outils logiciels qui nous avons utilisé pour l'implémentation de notre projet. Ensuite nous avons exposé l'implémentation de notre système à partir de l'étape d'identifier le dataset, division, analyse et prétraitement de dataset et sélection différentes caractéristique qui sont inclus à l'entraînement du modèle et puis le test et validation de notre modèle par différentes expérimentation mise en œuvre.

À la fin, nous avons discuté les différents résultats obtenus de chaque expérimentation qui montre la performance de notre modèle en comparons avec d'autres modèles prédicative.

Conclusion générale

Les routes sont confrontées à plusieurs problèmes de la gestion de trafic, réduire la pollution et contribuer à améliorer la sécurité routière. Le système de transport intelligent (ITS) aide au résoudre ces problèmes en rendre les routes plus intelligentes, plus efficaces et mieux gérées. La prédiction de flux de trafic routier joue un rôle crucial du système de transport intelligent a travers de prédire la variation de flux de trafic dans un proche avenir. La prédiction permet d'éviter la congestion de la circulation dans le temps, de réaliser un rappel intelligent de la congestion du trafic.

Pour une prédiction précise de flux de trafic, nous avons proposé dans ce travail une méthode d'apprentissage profond encodeur-décodeur avec le module ConvLSTM2D et le module Bi-LSTM. Ces modules prédisent l'information sur le trafic les caractéristiques spatiales et temporelles de l'information sur la circulation et en la complétant par des caractéristiques périodiques respectivement.

Afin de valider notre travail, nous avons réalisé trois expérimentations, dans chacune nous avons comparé notre modèle avec le modèle att-ba CNN-LSTM et ConvLSTM2D. Dans la première, deuxième et troisième expérimentation on a fait un entraînement avec un caractéristique de flux (débit) de trafic, toutes les caractéristiques de flux de trafic (débit et densité) et la météo (température et précipitation) et à la fin on a fait un entraînement avec les donnes de température seulement en raison de sa corrélation élever avec la caractéristique (vitesse) qui nous avons à prédire.

Lorsque on a comparé notre modèle avec chacune des expérimentations on a trouvé

que notre modèle donne le meilleur résultat en termes de métriques d'évaluations suivantes : RMSE , MAE et R^2 .

Cette recherche peut être améliorer sur certains points que nous considérons comme des perspectives pour notre travail :

- utiliser plus d'ensembles de données sur le trafic tels que l'information prévient des réseaux sociaux (tweeter) afin d'améliorer l'exactitude des prédictions de flux de trafic.
- effectuer des expérimentations différentes par de temps (l'horizon de prédiction est long).
- tuning l'hyperparamètres de l'entraînement par l'utilisation des méthodes d'optimisation tels que le Grid search pour déterminer l'hyperparamètres optimale afin de trouver un meilleur modèle d'apprentissage profond.
- faire l'apprentissage avec des poids statique pas aléatoire.

Bibliographie

[1] Mohammed S Ahmed et Allen R Cook. *Analysis of freeway traffic time-series data by using Box-Jenkins techniques*. 722. 1979.

[2] Md Zahangir Alom et al. "A state-of-the-art survey on deep learning theory and architectures". In : *Electronics* 8.3 (2019), p. 292.

[3] Essien Aniekan. `https://github.com/nakessien/tagf_evaluation/`. Accessed : 2021-04-12. 2021.

[4] Amar ASSAM et al. "Une approche multi-agents pour la simulation de la mobilité urbaine et de la communication V2X". Thèse de doct. université akli mohande-oulhadj bouira, 2019.

[5] Khireddine Benaissa, Salim Bitam et Abdelhamid Mellouk. "BSM-Data Reuse Model Based on In-Vehicular Computing". In : *Applied Sciences* 10.16 (2020), p. 5452.

[6] Rima Benelmir, Salim Bitam et Abdelhamid Mellouk. "An efficient autonomous vehicle navigation scheme based on LiDAR sensor in vehicular network". In : *2020 IEEE 45th Conference on Local Computer Networks (LCN)*. IEEE. 2020, p. 349-352.

[7] Azzedine Boukerche et E Robson. "Vehicular cloud computing : Architectures, applications, and mobility". In : *Computer networks* 135 (2018), p. 171-189.

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

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

[10] Loïck BOURDOIS. *LES RNN, LES LSTM, LES GRU ET ELMO*. https://lbourdois.github.io/blog/nlp/RNN-LSTM-GRU-ELMO/. Accessed : 2021-03-22. 2019.

[11] Jason BROWNLEE. *14 Different Types of Learning in Machine Learning*. https://machinelearningmastery.com/types-of-learning-in-machine-learning/. Accessed : 2021-03-24. 2019.

[12] Jason BROWNLEE. *A Gentle Introduction to Backpropagation Through Time*. https://machinelearningmastery.com/gentle-introduction-backpropagation-time/. Accessed : 2021-03-24. 2017.

[13] Jason BROWNLEE. *Gradient Descent With Momentum from Scratch*. https://machinelearningmastcom/gradient-descent-with-momentum-from-scratch/. Accessed : 2021-03-24. 2021.

[14] Jason BROWNLEE. *Loss and Loss Functions for Training Deep Learning Neural Networks*. https://machinelearningmastery.com/loss-and-loss-functions-for-training-deep-learning-neural-networks/. Accessed : 2021-03-24. 2019.

[15] CALTRANS. *Performance Measurement System (PeMS)*. https://pems.dot.ca.gov/. Accessed : 2021-04-02. 2018.

[16] Dejun CHEN, Hao WANG et Ming ZHONG. "A Short-term Traffic Flow Prediction Model Based on AutoEncoder and GRU". In : *2020 12th International Conference on Advanced Computational Intelligence (ICACI)*. IEEE. 2020, p. 550-557.

[17] *crashmap*. https://www.crashmap.co.uk/Search/. Accessed : 2021-03-24. 2021.

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

[19] Aniekan ESSIEN et al. "A deep-learning model for urban traffic flow prediction with traffic events mined from twitter". In : *World Wide Web* (2020), p. 1-24.

[20] Shuangshuang HAN et al. "Parallel vehicular networks : A CPSS-based approach via multimodal big data in IoV". In : *IEEE Internet of Things Journal* 6.1 (2018), p. 1079-1089.

[21] Yunxin Jeff Li. "An overview of the DSRC/WAVE technology". In : *International Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness*. Springer. 2010, p. 544-558.

[22] Jing Liu et Wei Guan. "A summary of traffic flow forecasting methods [J]". In : *Journal of Highway and Transportation Research and Development* 3 (2004), p. 82-85.

[23] Yipeng Liu et al. "Short-term traffic flow prediction with Conv-LSTM". In : *2017 9th International Conference on Wireless Communications and Signal Processing (WCSP)*. IEEE. 2017, p. 1-6.

[24] Yisheng Lv et al. "Traffic flow prediction with big data : a deep learning approach". In : *IEEE Transactions on Intelligent Transportation Systems* 16.2 (2014), p. 865-873.

[25] Bernhard Mehlig. "Artificial neural networks". In : *arXiv preprint arXiv :1901.05639* (2019).

[26] Hassan Aboubakr Omar et al. "Performance evaluation of VeMAC supporting safety applications in vehicular networks". In : *IEEE Transactions on Emerging Topics in Computing* 1.1 (2013), p. 69-83.

[27] Jose A Onieva et al. "Edge-assisted vehicular networks security". In : *IEEE Internet of Things Journal* 6.5 (2019), p. 8038-8045.

[28] Saurabh Rathor. "Simple RNN vs GRU vs LSTM :-Difference lies in More Flexible control". In : *Web page : https ://medium. com/@ saurabh. rathor092/simple-rnn-vs-gru-vs-lstm-differencelies-in-more-flexible-control-5f33e07b1e57, Retrieval Date* 12 (2018), p. 2019.

[29] Alex J Smola et Bernhard Schölkopf. "A tutorial on support vector regression". In : *Statistics and computing* 14.3 (2004), p. 199-222.

[30] Yan Tian et al. "LSTM-based traffic flow prediction with missing data". In : *Neurocomputing* 318 (2018), p. 297-305.

[31] Balachandran Vijayalakshmi et al. "An attention-based deep learning model for traffic flow prediction using spatiotemporal features towards sustainable smart city". In : *International Journal of Communication Systems* 34.3 (2021), e4609.

[32] Wangyang Wei, Honghai Wu et Huadong Ma. “An autoencoder and LSTM-based traffic flow prediction method”. In : *Sensors* 19.13 (2019), p. 2946.

[33] Yuankai Wu et al. “A hybrid deep learning based traffic flow prediction method and its understanding”. In : *Transportation Research Part C : Emerging Technologies* 90 (2018), p. 166-180.

[34] Rikiya Yamashita et al. “Convolutional neural networks : an overview and application in radiology”. In : *Insights into imaging* 9.4 (2018), p. 611-629.

[35] Hao Ye et al. “Machine learning for vehicular networks : Recent advances and application examples”. In : *IEEE Vehicular Technology Magazine* 13.2 (2018), p. 94-101.

[36] Ling Zhao et al. “T-gcn : A temporal graph convolutional network for traffic prediction”. In : *IEEE Transactions on Intelligent Transportation Systems* 21.9 (2019), p. 3848-3858.