Apprentissage de représentation pour la prédiction et la classification de séries temporelles

Ali Yazid Ziat

Université Pierre et Marie Curie
École Doctorale Informatique, Télécommunications et Électronique

Apprentissage de représentation pour la prédiction et la classification de séries temporelles

Ali Ziat
sous la direction de Ludovic Denoyer
et l'encadrement de Nicolas Baskiotis

THÈSE
pour obtenir le titre de
Docteur en Sciences mention Informatique

soutenue le XX octobre 2017
devant le jury composé de

Ahlame Douzal	Rapporteur
Latiffa Oukhellou	Rapporteur
Matthieu Cord	Examinateur
Jean Michel Loubes	Examinateur
Bertrand Leroy	Co-encadrant
Ludovic Denoyer	Directeur de thèse

RÉSUMÉ

Le traitement de données séquentielles représente une part importante des problèmes abordés en apprentissage automatique. Les méthodes classiques de classification ou de régression (SVM, KNN, Perceptron...) ne sont pas toujours adaptées aux caractéristiques de ces données temporelles (séquentialité, parcimonie, relations entre exemples d'apprentissage...). Nous nous intéressons aux données séquentielles où chaque exemple est une série temporelle, où il représente une quantité numérique qui évolue dans le temps à intervalle fixé. Le but de cette thèse est de développer des méthodes qui répondent aux difficultés posées par l'analyse des séries temporelles. Les contributions présentées dans ce manuscrit se focalisent sur deux tâches : la prédiction de séries temporelles et la classification de séries temporelles.
Le manuscrit s'ouvre sur une description des séries temporelles et de leurs caractéristiques dans le cadre de l'apprentissage automatique. Une revue des différentes tâches traitées dans la littérature est également donnée. Une des tâches motivant cette thèse, la prédiction de trafic automobile, est décrite, et nous expliquons en quoi elle s'inscrit dans le cadre du traitement de séries temporelles. Nous rappelons aussi quelques concepts clés de l'apprentissage de représentation qui est au cœur de nos contributions.
Notre première contribution présente une méthode de prédiction et de complétion de séries temporelles multivariées et relationnelles. Le but est d'être capable de prédire simultanément l'évolution d'un ensemble de séries temporelles reliées entre elles selon un graphe (pouvant représenter la proximité géographique entre plusieurs capteurs par exemple), ainsi que de compléter les valeurs manquantes dans ces séries (pouvant correspondre par exemple à une panne d'un capteur pendant un intervalle de temps donné). On se propose d'utiliser des techniques d'apprentissage de représentation pour prédire l'évolution des séries considérées tout en complétant les valeurs manquantes et prenant en compte les relations qu'il peut exister entre elles. Des expériences sur des problèmes de prédiction de trafic sont menées pour montrer l'intérêt de notre modèle.
Des extensions de ce modèle sont proposées et décrites : d'abord dans le cadre de la prédiction de séries temporelles hétérogènes puis dans le cas de la prédiction de séries temporelles avec une incertitude exprimée.
Un modèle de prédiction de séries spatio-temporelles est ensuite proposé, avec lequel les relations entre les différentes séries peuvent être exprimées de manière plus générale, et où ces dernières peuvent être apprises. Des expériences dans des domaines divers, allant de la prédiction d'évolution d'épidémie à la prédiction des vents, sont menées.
La dernière partie de ce manuscrit s'intéresse à la classification de séries temporelles. Un modèle d'apprentissage joint de métrique et de classification de séries est proposé. Une famille de problèmes de classification de séries temporelles pour laquelle les méthodes de l'état de l'art se révèlent peu satisfaisantes est décrite et nous mon-

trons en quoi le modèle que l'on propose est adéquat.

TABLE DES MATIÈRES

TABLE DES FIGURES

LISTE DES TABLEAUX

CHAPITRE 1

INTRODUCTION

1.1 Contexte : UPMC/VEDECOM

Cette thèse est le fruit d'une collaboration entre l'Institut VEDECOM au sein de l'équipe MOB04 (Développement des Espaces Numériques) et l'Université Pierre et Marie Curie au sein de l'équipe MLIA (Machine Learning for Information Access). l'Institut VEDECOM est un institut de recherche sur des problématiques liées au transport et dans ce cadre, l'explosion de la quantité de données de mobilité et leurs caractéristiques spécifiques posent des questions précises. La thèse réalisée et décrite dans ce manuscrit vise au développement des nouvelles méthodes de traitement de données de mobilité issues de différentes sources d'information et de leur mise en application pour des applications concrètes laissant présager des usages à venir dans les véhicules. L'objectif est principalement de répondre à la problématique de la prédiction du trafic et l'évaluation du taux de places disponibles de stationnement ainsi que la classification de séries temporelles (pouvant être appliquée par exemple à la classification d'évènements routiers). Nous nous proposons d'utiliser des techniques d'apprentissage automatique et d'apprentissage de représentation en particulier pour traiter ces problèmes.

1.2 Données de transport et données séquentielles

Les dernières années ont été marquées par l'explosion de la quantité de données temporelles dans différents domaines tels que la météorologie, la biologie, le trafic automobile et la finance entre autres. Les données sont produites sous la forme de séries temporelles qui sont le plus souvent multi-variées et qui exhibent des dépendances spatio-temporelles. Celà est spécialement vrai dans le domaine du transport où le développement des ITS (Intelligent Transportation System) qui nécessite la collecte d'informations sur le trafic et l'infrastructure routière en temps réel est en pleine expansion. Depuis plusieurs années, dans le but d'améliorer les conditions et la fluidité du trafic routier, les méthodes de collecte de données ont considérablement évolué et l'accès aux informations de trafic en temps réel est devenu courant. Les capteurs statiques (caméras, boucles magnétiques) et mobiles (GPS embarqués dans les véhicules ou les smartphones) capturent en temps réel l'évolution de l'activité urbaine.

L'usage des capteurs traditionnellement installés directement dans la chaussée (e.g. boucles magnétiques, pneumatiques) pour collecter des données est toujours nécessaire mais pas suffisant étant données les limitations inhérentes à ces méthodes : couverture partielle d'un réseau routier, forts coûts d'installation et de maintenance. Récemment, on a pu observer l'émergence de sources de données alternatives. C'est par exemple le cas pour des méthodes basées sur la position du véhicule ; les FCD

(Floatting Car Data ; véhicules équipés de GPS) en particulier sont considérés comme la solution la plus à même de pallier les limitations des capteurs fixes et à un moindre coût. La croissance importante de cette famille de collecte de données est liée non seulement au fait qu'une demande importante des utilisateurs aujourd'hui est d'avoir des services leur permettant d'avoir des informations pertinentes sur l'état des réseaux de transport, mais aussi aux perspectives atteignables grâce aux informations précises et en temps réel : la prédiction de l'évolution du trafic et des congestions, la détection automatique d'évènements routiers en temps réel, l'évaluation de la durée des futures congestions, la prédiction d'évènements routiers... Ces questions requièrent que les données soient au maximum précises, fiables et complètes. Celà soulève plusieurs difficultés : les capteurs peuvent couvrir une zone spatiale importante ce qui mène à une grande quantité de séries observées à prédire, et cela durant une période de temps qui peut être longue. Une autre caractéristique de ces données réside dans leur incomplétude : les capteurs mobiles en particulier fournissent de l'information sur une partie du réseau étudié qui varie dans le temps et les données produites sont par nature partielle. Une autre difficulté inhérente aux données spatio-temporelles est la prise en compte de leur particularité structurelle : les séries temporelles étudiées sont corrélées les unes aux autres selon des relations spatiales qui peuvent être complexes.

Les données collectées peuvent être utilisées pour prédire l'évolution future des phénomènes mesurés : la prédiction de trafic est un problème qui a focalisé l'attention de la communauté scientifique depuis longtemps. Depuis les premiers travaux de recherche sur le sujet [3], un grand nombre de techniques ont été proposées, essentiellement basées sur de la prédiction de séries temporelles univariées, puis plus récemment sur de la prédiction multi-variée [108] [63]. Cependant, les méthodes développées ont très peu pris en compte l'aspect géographique des données et les dépendances spatiales que les séries temporelles pouvaient exhiber. Celà est spécialement vrai dans le cas de la prédiction de trafic, mais également pour les données spatio-temporelles collectées plus généralement dans des domaines comme la météorologie ou la médecine. Tandis qu'historiquement ces données étaient le plus souvent étudiées à travers des modèles issus de la physique statistique [34] et basés sur des lois a priori, les développements récents en machine learning permettent de proposer des méthodes pour extraire des modèles prédictifs prenant en compte les caractéristiques spatiales directement depuis les données. Ces méthodes ont par exemple récemment été appliquées au problème de la prédiction de trafic [5].

Nous nous intéressons dans la première partie de cette thèse à l'étude et à la prédiction de séries temporelles multivariées dans les cas où il existe une interdépendance entre ces séries, et en prenant en compte le fait que ces séries peuvent être

incomplètes. Nous nous basons sur des techniques d'apprentissage de représentations où les données temporelles sont représentées par des points dans un espace vectoriel latent qui capture la dynamique du système et prend en compte les relations entre les sources des différentes séquences observées. Plusieurs modèles ont été proposés, non seulement pour la prédiction mais également pour la complétion de séries temporelles : en particulier, les modèles de l'état de l'art ne sont pas capables de traiter en même temps les deux tâches de prédiction et de complétion.
Nous proposons des modèles permettant de traiter ce problème. Nous nous intéressons également à la prédiction multi-vues de séries temporelles dans le cas où plusieurs séquences de différents types sont prédites, puis à la prédictions de phénomènes spatio-temporels (phénomènes environnementaux, propagation de maladies, etc.). Les contributions de ce travail sont les suivantes :

Un modèle de complétion et prédiction de séries temporelles (RAINSTORM) présenté dans le chapitre 3 : nous avons tout d'abord proposé un modèle qui s'appuie sur l'idée que chaque série temporelle à chaque pas de temps est représentée par un point dans un espace latent. Deux séries dont les sources sont proches sont également proches dans l'espace latent de représentation. Nous avons fait des expérimentations sur le domaine du transport et de la prédiction de trafic. Ce travail a donné lieu à deux publications :

— MUD2 (*Mining Urban Data)*, ICML (*International Conference on Machine Learning*) Workshop 2015 (publié), Ali Ziat, Gabriella Contardo, Nicolas Baskiotis, Ludovic Denoyer : Car-Traffic Forecasting : A Representation Learning Approach

— ESANN (*European Symposium on Artificial Neural Networks*) 2016 (publié), Ali Ziat, Gabriella Contardo, Nicolas Baskiotis, Ludovic Denoyer : Learning Embeddings for Completion and Prediction of Relational Multivariate Time-Series

Une extension du modèle pour la prédiction de séries temporelles hétérogènes présentée dans la première partie du chapitre 4 : elle repose sur l'idée que des séries de types différents (trafic et parking par exemple) puissent être projetées dans le même espace latent. Des expérimentations sont réalisées sur des données issues de la ville de Lyon. L'influence mutuelle des séries hétérogènes est étudiée. Ce travail a donné lieu à une publication :

— ITSC (*Intelligent Transportation Systems Conference*) 2016, Ali Ziat, Nicolas Baskiotis, Ludovic Denoyer, Bertrand Leroy : Joint Prediction of Road-Traffic and

Parking Occupancy Over a City With Representation Learning.

Une extension du modèle utilisant des représentations gaussiennes présentée dans la deuxième partie du chapitre 4 : une extension qui s'intéresse à l'incertitude dans la prédiction de séries temporelle. Nous nous basons toujours sur l'apprentissage de représentation mais dans un cadre probabiliste et non plus déterministe. Ce travail a donné lieu à une publication et à la soumission d'un article de journal :

— PRL (*Pattern Recognition Letter*) 2017, Ludovic Dos Santos, Ali Ziat, Benjamin Piwowarsky, Ludovic Denoyer, Patrick Gallinari : Modeling Relational Time Series using Gaussian Embeddings.

— NTS (*NIPS - Time Series Workshop*) 2016, Ludovic Dos Santos, Ali Ziat, Benjamin Piwowarsky, Ludovic Denoyer, Patrick Gallinari : Modeling Relational Time Series using Gaussian Embeddings.

Un modèle de prédiction spatio-temporelle présenté et décrit dans le chapitre 5 : Nous avons ensuite proposé un modèle spatio-temporel où les relations entre les séries peuvent être exprimées de manières plus générales et où le poids de ces dernières peut être appris. Nous avons fait des expérimentations dans différents domaines. Ce travail a donné lieu à la rédaction d'un article soumis :

— ICDM (*Conference on Neural Information Processing Systems*) 2017, Ali Ziat, Ludovic Denoyer, Patrick Gallinari : Spatio-Temporal Neural Networks for Space-Time Series Forecasting

Un modèle de classification de séries temporelles présenté dans le chapitre 6 : Dans la dernière partie de cette thèse, nous nous intéressons à la classification de séries temporelles en utilisant un apprentissage joint de métrique et d'une fonction de classification. Ce travail a donné lieu à la rédaction d'un article soumis :

— PRL (*Pattern Recognition Letter*) 2016, Ali Ziat, Ludovic Denoyer : Deep Joint Metric Learning and Time-Series Classification

La suite de ce rapport est organisée comme suit : Dans le chapitre 2, un état de l'art succin est donné. Le chapitre 3 qui suit décrit notre première contribution, puis deux extensions sont proposées dans le chapitre 4. Ensuite, nous décrivons un modèle général de prédiction spatio-temporelle dans le chapitre 5. Suit alors le chapitre

6 qui aborde la tâche de la classification de séries temporelles et décrit notre contribution. Enfin, le chapitre 7 conclut ce manuscrit et donne de futures directions de travail envisagées.

CHAPITRE 2

ÉTAT DE L'ART

2.1 Séries temporelles

Les séries temporelles constituent une part importante des données produites et disponibles sur Internet dans de très différents domaines. Par exemple, dans le seul domaine médical, l'information enregistrée par les électroencéphalogrammes ou par des électrocardiogrammes, les données qui représentent l'expression de gènes [2], les données sur la croissance d'un individu etc., sont des séries temporelles fréquemment traitées. Les séries temporelles se retrouvent de la même manière dans d'autres domaines tels que la finance, la météorologie, le son... Si depuis plus d'un siècle la communauté scientifique s'est penchée sur le traitement des séries temporelles [101], la disponibilité de grandes quantités de données est relativement récente et de nouveaux challenges s'offrent à elle. Les principaux axes d'études autour des séries temporelles qui ont été proposés dans la littérature sont les suivants :

- **La prédiction :** étant donnée une série temporelle $X = x_1, x_2, ..., x_T$ contenant T points, il s'agit de prédire la ou les valeurs suivantes, c'est-à-dire les valeurs $x_{t+1}, x_{t+2}, x_{t+3}$... [128, 23, 70, 112].
- **La classification :** étant donnée une série temporelles X, il s'agit de l'assigner à une des (deux ou plus) classes prédéfinies [61, 55, 129].
- **La complétion :** étant donnée une série temporelle $X = x_1, x_2, ..., x_T$ contenant T points et un masque m_i tel que $m_i = 1$ si la valeur de x_i est connue et $m_i = 0$ sinon, il s'agit d'inférer la ou les valeurs manquantes, c'est-à-dire les valeurs pour lesquelles $m_i = 0$ [118].
- **L'indexation :** étant donnée une série temporelle X ainsi qu'une mesure de similarité (ou dissimilarité) notée $D(X, X')$ telle que $D(X, X')$ est grand si les séries X et X' sont similaires et petit sinon, il s'agit de trouver la ou les séries temporelles les plus similaires dans une base de données donnée [65, 47].
- **La segmentation :** étant donnée une série temporelle $X = x_1, x_2, ..., x_T$ avec $\forall i, x_i \in \mathbb{R}$, il s'agit de trouver une approximation $\hat{X} = k_1, k2, ..., k_K$ avec $\forall i, k_i \in \mathbb{R}$ et $K << T$ et où $\hat{X}$ est une bonne approximation de X [52, 66].
- **Le partitionnement :** il s'agit de regrouper des séries temporelles d'une base de données donnée en plusieurs partitions différentes selon une mesure de similarité (ou dissimilarité) notée $D(A, B)$) telle que $D(X, X')$ est grand si les séries X et X' sont similaires et petit sinon [78, 125, 86]
- **La détection d'anomalies :** étant donnée une série temporelle X que l'on considère comme étant "normale", déterminer quelles séries au sein d'une base de données contiennent une "anomalie" [48, 119].

Nous nous intéressons dans la première partie de ce manuscrit aux tâches de prédiction et de complétion de séries temporelles dans le cas où les séries sont multivariées et relationnelles. Nous proposons ensuite une méthode de classification de séries temporelles.

2.2 Prédiction

Le sujet de la modélisation et de la prédiction de séries temporelles a donné lieu à une riche littérature depuis de nombreuses années en statistique et en apprentissage automatique. En statistiques, les approches linéaires classiques basées sur les modèles à moyenne mobile et autorégressifs ont été les plus utilisées. Ces modèles supposent que les séries temporelles sont stationnaires et qu'elles présentent des dépendances linéaires dans le temps [38]. En apprentissage automatique, des extensions non linéaires de ces modèles, basées essentiellement sur des réseaux de neuronnes, ont été proposées dès le début des années 1990, ouvrant la voie à de nombreuses extensions non linéaires comme les méthodes à noyaux [85].

Les modèles dynamiques à état comme les réseaux de neurones récurrents ont également été utilisés pour la prédiction séquentielle dans différents contextes [32]. Récemment, ces méthodes ont été à la base d'importants succès dans différents domaines en modélisation de séquence avec notamment de forts progrès en modélisation du langage [46], génération de langage naturel [111], traduction [28] et beaucoup d'autres [24]. Un modèle proche de ceux que l'on propose dans cette thèse est dénommé "dynamic factor graph" [83] conçu pour la modélisation de séries temporelles multivariées. Comme les nôtres, c'est un modèle génératif à vecteurs latents qui capture les dynamiques temporelles dans un espace de représentation et qui prédit le futur des séries à l'aide d'une fonction de décodage de l'espace latent. A la différence des modèles prédictifs que nous proposons, aucune dépendance spatiale n'est considérée dans cette approche.

Les statistiques spatio-temporelles ont également un long historique [34, 121]. Les méthodes traditionnelles se basent sur des approches descriptives qui utilisent les moments du premier et du second ordre pour modéliser les dépendances spatio-temporelles. Plus récemment, des modèles dynamiques à états où l'état courant est conditionné par les états précédents ont été explorés [120]. Pour ces modèles, le temps et l'espace peuvent être continus ou discrets, cependant la méthodologie usuelle est de considérer un temps discret ce qui mène à modéliser des processus spatiaux comme des séries temporelles. Quand l'espace est continu à l'inverse, les modèles sont généralement exprimés par des équations intégro-différentielles linéaires. Quand l'espace est discret, les formulations sont le plus souvent autorégressives.
Ces modèles font face à des difficultés de passage à l'échelle dans le cas où un grand nombre de sources produisent des séries incomplètes : pour nombre de processus complexes, les observations ne fournissent qu'une description incomplète des dynamiques des phénomènes observés. Différentes stratégies ont été adoptées pour traiter ces particularités comme représenter le processus par des espaces à petites dimensions, menant à des familles de modèles assez proches de celles utilisées en

machine learning pour modéliser des phénomènes dynamiques. Une propriété intéressante de ces approches est la possibilité d'incorporer de la connaissance a priori comme pour les processus spatio-temporels inspirés des phénomènes physiques. Cette stratégie consiste à s'inspirer de principes concrets comme par exemple les équations à dérivées partielles développées pour modéliser des phénomènes de diffusion en physique.
En climatologie, des modèles prenant en compte des composantes géographiques et temporelles ont aussi été développés comme les "Gaussian Markov Random Fields" [96]. En apprentissage automatique, la modélisaton spatio-temporelle a été assez peu considérée. Par exemple, [9] introduit un modèle de tenseur pour la prédiction et le kriging. Les auteurs de [69] utilisent des champs aléatoires conditionnels pour detecter de l'activité dans des vidéos ; le temps est discrétisé (image par image) et un des buts visés est la prédiction d'activité future. Le BCI (Brain Computer Interface) est un autre domaine pour l'analyse de données spatio-temporelles avec notamment des travaux s'intéressant à l'apprentissage de filtres spatio-temporels [39, 90]. En deep-learning, des approches ont également été proposées sans prendre en compte explicitement les liens entre les dimensions spatiales et temporelles [72].

2.2.1 Modèles de prédiction de séries temporelles univariées

2.2.1.1 Modèles linéaires

La plupart des applications réelles de modélisation de séries temporelles univariées utilisent des modèles linaires. Les modèles linéaires les plus populaires sont les modèles autorégressifs (AR). Un des avantages de ces modèles est qu'ils donnent une bonne approximation du premier ordre des dynamiques des processus sous-jacents aux données. Ces modèles peuvent théoriquement modéliser parfaitement des données qui sont décrites exhaustivement par le premier et le deuxième moment dans un monde de distributions gaussiennes. Ces méthodes sont aussi attractives de par leur simplicité et leur relative efficacité : même pour des problèmes connus comme présentant des dynamiques non linéaires, la non linéarité est soit pas assez significative soit pas assez constante dans le temps pour que les modèles autorégressifs présentent des performances acceptables.

Les modèles autorégressifs sont donc les modèles de séries temporelles les plus populaires ; leurs paramètres peuvent être appris entièrement en minimisant une erreur des moindres carrés [4]. Une série temporelle est modélisée par un modèle AR si à un instant t on a :

$$x_t = \sum_{i=1}^{p} \alpha_i x_{t-i} + \epsilon_t$$

Celà correspond au modèle autorégressif d'ordre (ou de lag) p. L'erreur ϵ est usuellement spécifiée comme un bruit blanc, c'est-à-dire non corrélé dans le temps, de variance constante et de moyenne zéro.

De nombreuses extensions de ces modèles [19] dont il n'est pas possible de faire un descriptif exhaustif ont été proposées.

2.2.1.2 Modèles non linéaires

Au début des années 90, les modèles non linéaires de prédiction de séries temporelles ont commencé à être popularisés [113]. Ces modèles nécessitant en général une quantité de données en apprentissage plus importante pour obtenir des gains de performances de prédictions, les applications visées ont été essentiellement la finance et le transport [50]. Là encore, un grand nombre de méthodes a été proposée, en statistique tout d'abord, avec le modèle ARCH [71] puis son extension GARCH, mais aussi en machine learning avec les modèles SVR [85] (Support Vector Regression) et surtout les réseaux de neurones qui sont devenus les modèles parmi les plus populaires en modélisation et prédiction de séquences.
Ces derniers sont utilisés classiquement comme fonction autorégressive prenant la forme usuelle :

$$x_t = f(x_{t-1}, x_{t-2}, ..., x_{t-p})$$

Ces modèles sont entraînés en utilisant une fenêtre glissante comme c'est le cas pour les modèles AR. Le mécanisme est illustré sur la Figure 2.1

2.2.2 Modèles de prédiction de séries temporelles multivariées

Les modèles de prédiction de séries temporelles ont tous été étendus au cas de la prédiction de séries temporelles multivariées, c'est-à-dire au cas où plusieurs valeurs évoluent simultanément - comme illustré en Figure 2.2. Parmi les modèles existants, le modèle VAR (vectorial autoregressiv) est l'un des plus répandus et les plus utilisés. Il s'agit d'une extension naturelle au modèle autorégressif univarié AR. Il part du principe que la valeur d'une série temporelle i dépend des valeurs précédentes de la série pendant une fenêtre temporelle donnée, mais également des valeurs des autres séries considérées pendant ce même intervalle de temps. Celà s'écrit :

$$x_t^i = \sum_{j=1}^{N} \alpha_{t-1}^j x_{t-1}^j + \alpha_{t-2}^j x_{t-2}^j + ... + \alpha_{t-p}^j x_{t-p}^j + \epsilon_t$$

Ce modèle du fait de sa simplicité a été extremement utilisé et il s'est montré particulièrement efficace dans certaines tâches prédictives [84, 57, 9]. Cependant les dépendances linéaires entre séries restent une hypothèse forte et limitant le pouvoir d'expression de ces modèles ; de plus, la forte complexité en nombre de paramètres

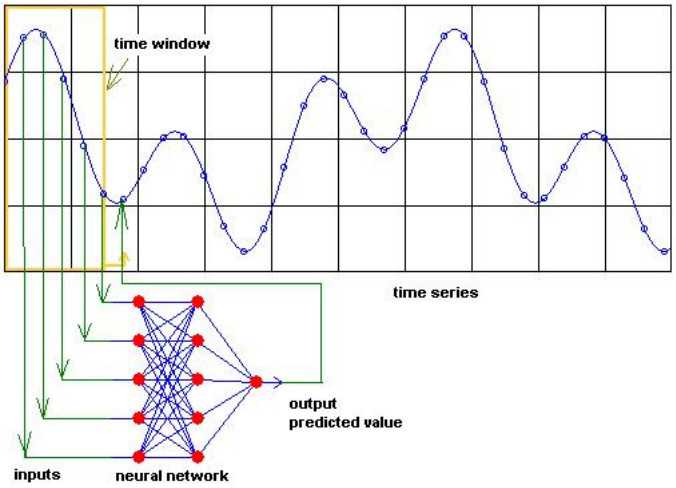

FIGURE 2.1 – Réseau de neurones entraîné pour prédire une série temporelle. Une fenêtre des valeurs passées est donnée en entrée du réseau qui prédit la valeur suivante.

à apprendre (quadratique en nombre de séries à prédire) rend cette approche difficilement utilisable dans le cas où le nombre de séries temporelles est grand.

2.2.3 Réseaux de neurones récurrents

Ces dernières années ont vu la résurgence de l'utilisation des réseaux de neurones récurrents pour le traitement de séquences, et en particulier dans des tâches prédictives. Les réseaux de neurones récurrents disposent d'une mémoire sur ce qui a été calculé par le passé et c'est ce qui les rend particulièrement adaptés au traitement de séquences. En théorie, les réseaux de neuronnes récurrents peuvent garder en mémoire l'information vue dans une séquence arbitrairement grande mais en pratique perdent de leur efficacité sur des dynamiques à très long terme. C'est dans ce sens que de récents travaux ont vu l'émergence d'architectures de réseaux de neurones récurrents disposant de mécanismes de "gate" permettant d'améliorer considérablement les capacités de mémorisation des modèles. Plus spécifiquement, certains types de réseaux récurrents, les LSTMs [53] et les GRUs [30] se sont avérés particulièrement efficaces pour modéliser des séquences dont la dynamique pouvait s'étaler loin dans le temps; ce n'est pas le cas de l'approche autorégressive qui de fait, au moment de l'inférence ne se base que sur une fenêtre temporelle de taille fixe dans le passé. Finalement, en apprentissage automatique, l'essentiel des progrès réalisés ces dernières années en modélisation de séquence a été avec des architectures

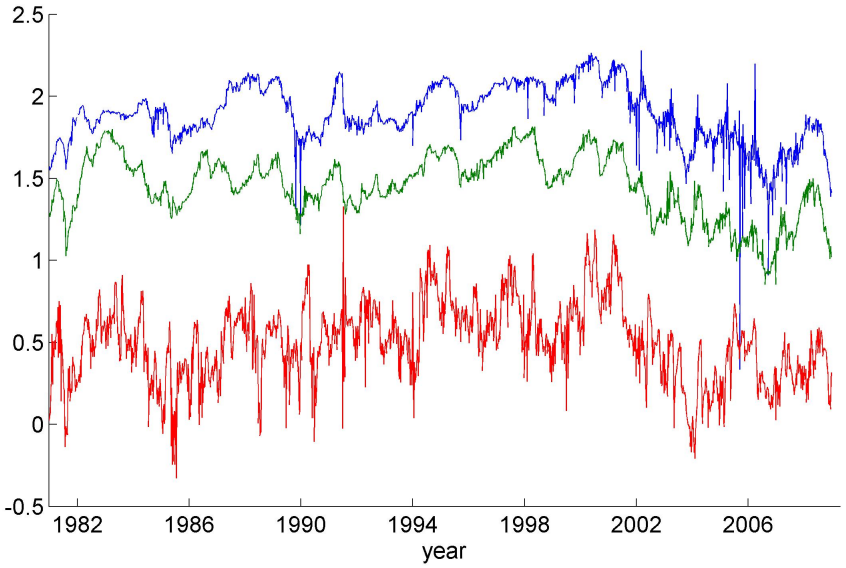

FIGURE 2.2 – Série temporelle multivariée : trois valeurs évoluent en parallèle et la tâche de la prédiction multivariée consiste à prédire la future évolution de chacune des séries considérées.

récurrentes à "gate". On peut par exemple citer [97, 15].

2.2.4 Autres modèles

De nombreux autres modèles ont été proposés pour la prédiction de séries temporelles et il ne serait pas possible de faire ici un état de l'art exhaustif de ceux-ci. Plus précisément, un grand nombre de modèles existe dans des cadres restreints : on peut par exemple citer le modèle de Croston particulièrement adapté dans le cas où les séries temporelles valent souvent zéro. La méthode de lissage exponentielle est également souvent utilisée en pratique mais elle diffère peu des méthodes autorégressives. Nous nous limitons dans ce travail aux modèles les plus familièrement utilisés dans la communauté de l'apprentissage automatique.

2.3 Données manquantes

L'immense majorité des méthodes d'analyse statistique requiert des datasets rectangulaires [1], sans valeur manquante. Celà est également vrai pour les séries tempo-

1. Nous parlons ici de jeux de données où tous les exemples ont les mêmes caractéristiques donc la même taille

relles pour lesquelles les modèles le plus souvent développés ne sont valides que si toutes les valeurs d'une série sont définies. Néanmoins, les données réelles dans les différentes applications industrielles et scientifiques présentent communément des données ou des plages de données manquantes ou incomplètes. La méthode usuelle pour faire face au problème de l'incapacité des modèles à traiter des données parcimonieuse est d'inférer les données manquantes dans un premier temps, comme étape de pré-traitement, puis d'utiliser des méthodes classiques de modélisation de séries sur des données complètes.

2.3.1 Heuristiques

Dans la pratique, les méthodes de complétion de données les plus utilisées sont souvent des méthodes à base d'heuristiques [12] ; celà se justifie par la simplicité de la mise en place de ces méthodes. Le problème de ces approches étant leur précision limitée et l'introduction occasionelle de biais importants dans les données [54]. Parmi les heuristiques utilisées, on peut citer les suivantes :

— Moyenne : Cette méthode consiste à remplacer les données manquantes d'une série temporelle par la moyenne des valeurs observées sur la séquence, par une moyenne sur les valeurs qui précédent la valeur manquante ou par une moyenne des valeurs observées dans une certaine fenêtre autour d'une valeur manquante.
— Dernière observation : L'idée est de remplacer une valeur manquante par la dernière valeur observée sur la séquence. Dans le cas où les valeurs manquantes sont peu nombreuses, cette méthode peut donner des résultats surprenamment bons [87].
— Knn-substitution : utilisée dans la complétion de données manquantes dans le cas de séries temporelles multivariées, cette méthode consiste à rechercher la série la plus similaire (au sens des plus proches voisins) à une série donnée et de remplacer les valeurs manquantes de cette dernière par celles du plus proche voisin.

2.3.2 Méthodes autorégressives

Plusieurs méthodes ont proposé d'utiliser les modèles autorégressifs pour inférer les valeurs non observées dans les séries temporelles. Par exemple, [106] propose de trouver pour une série contenant des valeurs manquantes les K séries les plus similaires (au sens de la distance euclidienne) avec K un hyper-paramètre du modèle, puis de calculer les coefficients autorégressifs sur ces séries similaires puis d'enfin utiliser ces coefficients pour estimer les valeurs manquantes de la série temporelle initiale.

2.3.3 Maximum de vraisemblance et E.M

Dans [11], une méthode de maximum de vraisemblance basée sur un algorithme EM est proposée. L'idée est que pendant l'étape E(spérance), l'espérance est évaluée selon les données observées et les paramètres du modèle tandis que lors de l'étape M(aximisation), cette espérance est maximisée. Selon les auteurs, cette méthode donne des résultats particulièrement intéressants lorsque les données manquantes occupent de larges portions continues. De façon générale, E.M a souvent été utilisé pour inférer des données manquantes.

2.3.4 Factorisation matricielle

Ces dernières années, plusieurs modèles de factorisation matricielle ont été proposés pour la complétion de données [91] et en particulier dans le domaine des données séquentielles et des séries temporelles [104, 105, 102]. Par exemple, un modèle pour compléter les données de trafic manquantes est formulé dans [102].

Si m séries temporelles (m routes) sont observées pendants t pas de temps, la matrice X est définie telle que :

$$X = \begin{array}{c} \begin{matrix} t_1 & t_2 & \dots & t_T \end{matrix} \\ \begin{pmatrix} x_1^1 & x_2^1 & \dots & x_t^1 \\ x_1^2 & x_2^2 & \ddots & x_t^2 \\ \vdots & \ddots & \ddots & \ddots \\ x_1^m & x_2^m & \dots & x_T^m \end{pmatrix} \end{array} \begin{matrix} r_1 \\ r_2 \\ \vdots \\ r_m \end{matrix}$$

Ensuite sont définies des matrices latentes R et T de dimension respectives mxK et KxT, avec K un hyper-paramètre, tel que le coût suivant est minimisé :

$$L(R,\ T) = \tfrac{1}{2}||X - RT||^2$$

L'apprentissage se fait par descente de gradient alternée ou par descente de gradient stochastique. Les facteurs latents appris correspondant aux routes (matrice R) sont utilisés pour obtenir des partitionnements des différents types de route. Cette méthode est étendue pour la prise en compte de l'information géographique et est en ce sens proche de celles que l'on développe dans cette thèse. Les auteurs considèrent une matrice E de contexte géographique correspondant par exemple à une connaissance expert sur différentes zones de la ville. Si L différentes zones sont considérées et que les experts s'attendent à observer une occupation de ces zones pendant T pas de temps, la matrice suivante est construite :

$$G = \begin{array}{c} \begin{matrix} t_1 & t_2 & \dots & t_T \end{matrix} \\ \begin{pmatrix} o_1^1 & o_2^1 & \dots & o_t^1 \\ o_1^2 & o_2^2 & \ddots & o_t^2 \\ \vdots & \ddots & \ddots & \ddots \\ o_1^l & o_2^l & \dots & o_T^L \end{pmatrix} \end{array} \begin{matrix} \\ g_1 \\ g_2 \\ \vdots \\ g_L \end{matrix}$$

Une nouvelle matrice latente G de dimension LxK est définie, telle que la nouvelle fonction de coût prenant en compte l'information géographique est formulée :

$L(R, T, G) = \frac{1}{2}||X - RT||^2 + \lambda\frac{1}{2}||E - GT||^2$

L'apprentissage est réalisé de manière jointe sur l'ensemble des facteurs latents et λ est un hyper paramètre équilibrant l'apport de la connaissance géographique.

2.4 Classification

La classification de séries temporelles consiste à assigner une série temporelle à l'une des classes présentes dans un ensemble prédéfini. Elle trouve une application dans de nombreux domaines : la reconnaissance de la parole[17], les sciences environnementales[124], les neurosciences[117], la finance[64] et beaucoup d'autres. Une large variété de problèmes de classification de séries existe : deux séries peuvent appartenir à la même classe pour plusieurs raisons, notamment la présence d'une forme particulière dans la série, la similarité dans le temps des deux séries temporelles (c'est-à-dire si les deux séries temporelles ont une allure semblable dans le temps) mais aussi pour plusieurs autres raisons, par exemple une variation subite à un moment donné. Un exemple illustratif est donné sur la Figure 2.4.

De nombreuses techniques ont été proposées pour traiter ces problèmes. Elles peuvent êtres catégorisées en deux familles principales : les méthodes de classification de séries temporelles basées sur des mesures de similarités et les méthodes de classification de séries temporelles basées sur l'extraction de caractéristiques descriptives.

2.4.1 Extraction de caractéristiques descriptives

Les modèles de classification de séries temporelles basés sur des caractéristiques descriptives supposent d'extraire un ensemble de caractériques qu'on espère être représentatif de la forme générale d'une série temporelle. Le plus communément, ces caractériques sont quantifiées pour former des "sacs de mots" (BoW pour "Bag

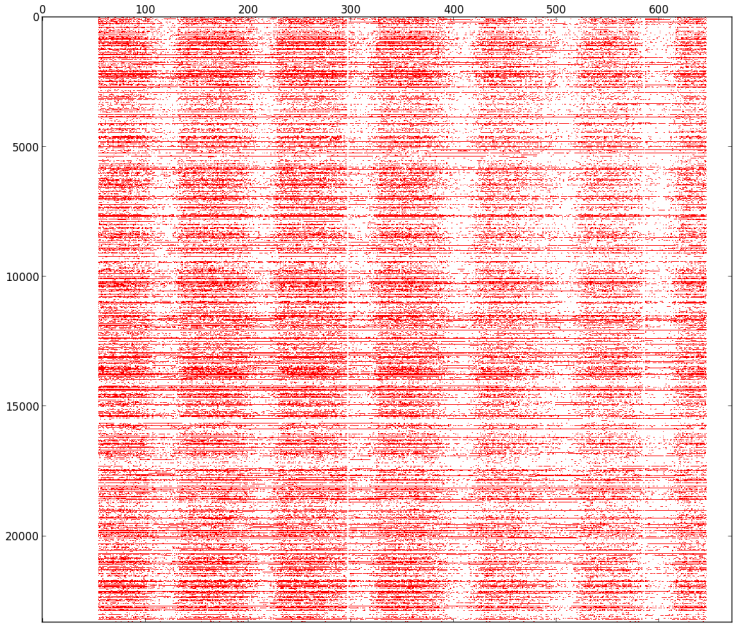

FIGURE 2.3 – Visualisation d'une semaine de données enregistrées sur le trafic dans la ville de Pékin. Le trafic est illustré pour environ 25000 routes (ce qui correspond à 25000 lignes) et quart d'heure par quart d'heure (ce qui correspond à 672 colonnes). Les cases blanches correspondent à des données manquantes alors que les cases rouges représentent les données observées. Les bandes verticales blanches correspondent aux nuits pendant lesquelles la quantité de données observées est beaucoup moins importante.

of Words"), qui sont ensuite donnés en entrée d'un classifieur. Ces approches diffèrent essentiellement dans les caractéristiques extraites. Pour nommer quelques approches récemment proposées, le modèle TSBF (Time-Series Bag of Features) [13] extrait des intervalles à différents niveaux d'échantillonnage, et chaque série temporelle est alors transformée en sac de mot, à la suite de quoi un classifieur supervisé (une forêt aléatoire) est construit pour classifier la série. Une autre approche récente est notée BOSS (Bag-of-SFA-Symbols) [99] qui propose une distance basée sur des histogrammes d'approximations de séries de Fourier. Cette méthode a été étendue par le modèle BOSSVS [100] qui combine le modèle original avec un modèle de représentation vectorielle pour réduire la complexité temporelle du modèle. Différents

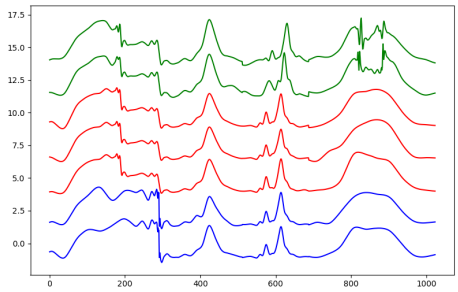

FIGURE 2.4 – Illustration d'une instance d'un problème de classification de séries temporelles à trois différentes classes : une classe marquée par des séries rouges (3 exemples), une classe marquée par des séries vertes (deux exemples) et une troisième classe marquée par des séries bleues (deux exemples).

modèles correspondant à différentes tailles de fenêtres glissantes sont agrégés. La classification est finalement effectuée dans l'espace de représentation en se basant sur le plus proche voisin.

2.4.2 Mesures de similarités

Les méthodes basées sur des mesures de similarité utilisent des comparaisons "point à point" de séries temporelles entières. Les plus communes étant les approches de plus proche voisin couplées à une distance euclidienne ou une distance de déformation dynamique temporelle, notée DTW (Dynamic Time Warping). La déformation dynamique temporelle (DTW) [115] ne souffre pas de ces maux et est sans doute la méthode la plus utilisée en tant que méthode de comparaison [7]. Elle permet non seulement de comparer des séries temporelles de tailles différentes mais permet de reconnaitre des formes même si elles ont subi des transformations (décalage, changement d'échelle). C'est la technique la plus populaire en classification de séries temporelles et obtient des résultats très compétitifs [25][123]. Elle souffre cependant également de certains handicaps : elle a une complexité très importante ($O(n^2)$ pour une série de longueur n) et lors de la phase d'inférence, le temps d'exécution dépend du nombre d'exemples dans la base d'apprentissage (comme c'est le cas pour toutes les méthodes basées sur des mesures de similarité) et peut être extrêmement long. Des techniques existent pour réduire cette complexité comme l'utilisation d'un seuil de restriction de taille variable (illustré en Figure 2.6) permettant de ne mapper qu'un point d'une série T à un sous-ensemble de points d'une série S plutot que tous. On peut citer également le "early-pruning" des séries candidates [89].

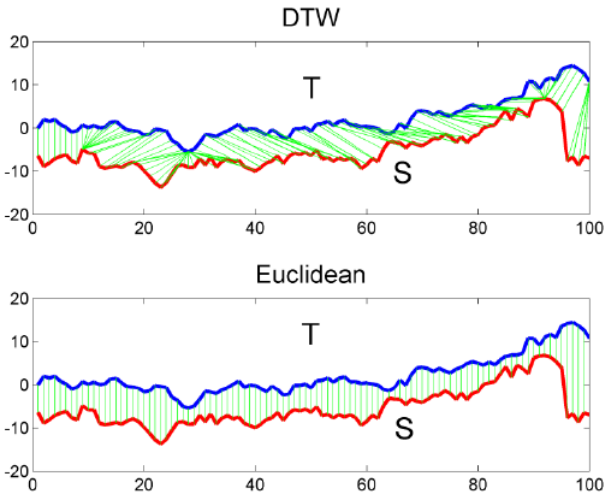

FIGURE 2.5 – Illustration de deux distances : DTW (en haut) et euclidienne (en bas)

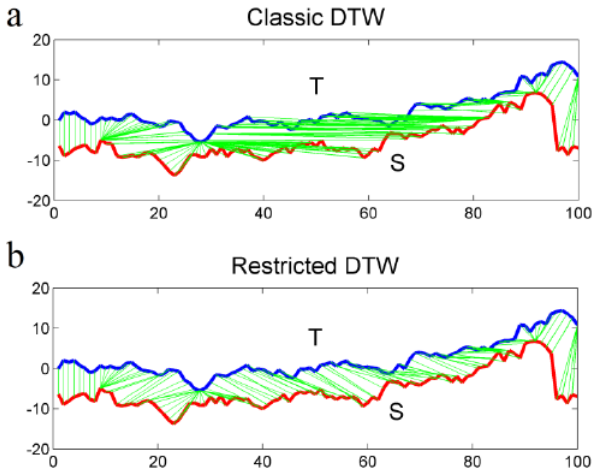

FIGURE 2.6 – Illustration de deux distances : DTW et DTW restreinte

Des méthodes d'ensemble qui combinent les résultats de plusieurs classifieurs -basés sur des mesures et similarité et basés sur l'extraction de caractéristiques discriminantes- existent et sont connues pour avoir de bons taux de bonne classification (voir [8] par exemple), mais ont souvent des temps d'exécution très longs en apprentissage et en inférence.

2.4.3 Apprentissage de métrique

L'apprentissage de métrique vise en général à apprendre une métrique satisfaisante, ce qui veut dire que la plupart du temps, la distance entre exemples ayant les mêmes labels est réduite tandis que la distance entre exemples ayant des labels différents est maximisée. Ce sujet d'étude est relativement ancien et peut être catégorisé en deux sous-familles de méthodes : les méthodes basées sur des distances de Mahalanobis et les méthodes basées sur les réseaux de neurones siamois. Les deux paradigmes sont décrits ci-dessous.

2.4.3.1 Distance de Mahalanobis

La majorité des travaux précédents en apprentissage de métrique apprennent une métrique dans l'espace d'entrée comme par exemple avec des métriques de Mahalanobis qui le plus souvent apprennent une transformation linéaire des exemples dans un nouvel espace de caractéristiques. La forme générale de ces métriques est paramétrisée par une matrice M et définie sur un ensemble d'exemples étiquetés $\mathscr{X} = (\mathbf{x}, \mathbf{y})$ tel que si $x_i, x_j \in \mathbf{x}$, la distance entre x_i et x_j est donnée par :

$$D(x_i, x_j) = \sqrt{(x_i - x_j)^T M (x_i - x_j)} \tag{2.4.1}$$

Plusieurs exemples de ces travaux peuvent être cités ; ils diffèrent essentiellement par la façon dont la transformation linéaire est optimisée. Par exemple, voire [73, 95, 36, 74, 59, 75].

2.4.3.2 Réseaux de neurones siamois

Plus en adéquation avec notre approche, plusieurs travaux ont exploré l'apprentissage de métriques non linéaires en rétro-propageant l'erreur dans des réseaux de neurones sur des paires d'exemples. La manière usuelle de procéder est d'utiliser des réseaux de neurones siamois qui sont deux réseaux de neurones aux topologies identiques et partageant les mêmes paramètres. Des paires d'exemples sont présentées aux siamois durant l'apprentissage et ils calculent en sortie un score de similarité indiquant si les deux exemples sont similaires ou pas. Différentes mesures de similarité sont utilisées, habituellement basées sur des mesures de distance comme la distance euclidienne ou la similarité cosinus. [82, 29, 68, 56, 20, 49, 98].

2.5 Apprentissage de représentation dans les graphes

Nous nous intéressons dans nos premières contributions à la prédiction de séries temporelles dans le cas où les séries sont inter-dépendantes. Notre première contri-

bution tire son origine de travaux effectués dans l'équipe sur l'apprentissage de représentation pour la classification dans les graphes hétérogènes. L'idée de l'apprentissage de représentation est de projeter les données depuis un espace d'entrée dans un espace vectoriel latent dans lequel on espère résoudre notre problème (régression, classification...) plus facilement (voir Figure 2.7).

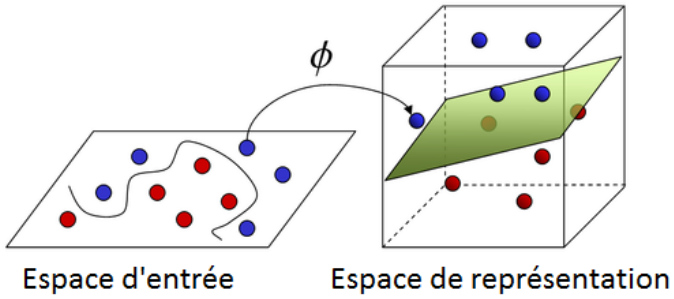

FIGURE 2.7 – Illustration du mécanisme de projection dans un espace de représentation.

Dans [58], l'auteur s'intéresse au cas où les données dans l'espace d'entrée sont reliées entres elles selon des relations connues, comme dans un réseau social par exemple.

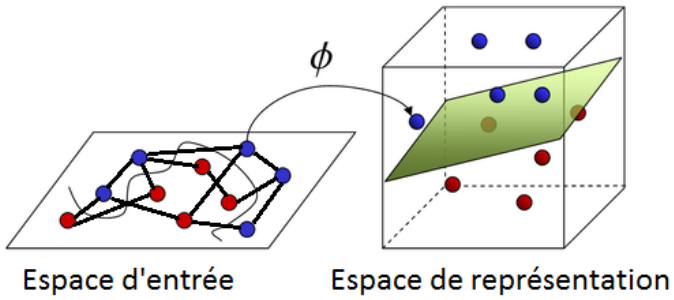

FIGURE 2.8 – Illustration du mécanisme de projection dans un espace de représentation dans le cas où les données d'entrée sont reliées par un graphe.

Une hypothèse de régularité est faite sur l'espace latent : deux nœuds reliés dans un graphe doivent avoir une représentation latente proche dans l'espace de représentation. La fonction de coût suivante est formulée :

$$\mathscr{L}(\theta, \mathbf{z}) = \sum_{i=1}^{n} \Delta(d_\theta(z_i, x_i) + \lambda \sum_{i=1}^{n} \sum_{j=1}^{n} W_{i,j} ||z_i - z_j||^2 \tag{2.5.1}$$

Où z_i est la projection du point x_i dans l'espace de représentation, $W_{i,j}$ représente la force de la relation entre les points i et j et où $\Delta(d_\theta(z_i), x_i)$ représente une erreur de classification ou une erreur de prédiction. Le terme λ est là pour équilibrer la force du terme de régularisation. C'est cette même idée que nous formulons dans le modèle RAINSTORM qui peut être vu comme une extension de ce modèle dans le cas où les données sont dynamiques.

Ce modèle est basé sur l'idée de projection des séries temporelles dans un espace latent de dimension N. Ainsi, à chaque pas de temps, chaque série est représentée par un vecteur Z_t^i, l'ensemble des représentations étant ainsi une matrice $Z_t \in \mathbb{R}^{N \times T}$ composée de tous les vecteurs latents des différentes séries. Comme pour le modèle de classification dans les graphes hétérogènes, une contrainte de régularisation structurelle est imposée : deux séries connectées doivent être proches dans l'espace latent.

Pour la modélisation des séries temporelles, nos approches sont basées sur deux idées clefs :

— d'une part la représentation latente de chaque série permet de retrouver la ou les valeurs numériques de cette série au temps t. Cette idée est assez triviale et contraint notre système à capturer les observations X_t fournies, et à être en mesure de prédire des observations futures

— d'autre part la représentation latente contient l'information permettant de modéliser la dynamique future de chaque série. Ainsi, chaque série correspond à une trajectoire "prédictible" dans l'espace latent.

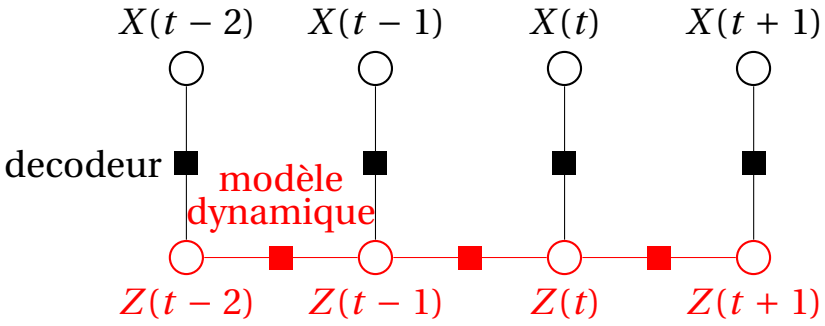

FIGURE 2.9 – Illustration du modèle proposé avec le décodeur en noir et le modèle dynamique en rouge.Les observations sont obtenues à partir du décodage des états latents.

CHAPITRE 3

RAINSTORM : APPRENTISSAGE DE REPRÉSENTATION POUR LA PRÉDICTION ET LA COMPLÉTION DE SÉRIES TEMPORELLES RELATIONNELLES

Résumé

Dans ce chapitre, nous proposons un modèle permettant d'apprendre sur plusieurs séries temporelles dans le cas où ces dernières sont reliées entre elles par une structure de graphe. Le modèle est à la fois à même de prédire les valeurs manquantes dans les série temporelles mais aussi de prédire les futures valeurs. Cette approche est basée sur l'apprentissage de représentation où chaque valeur de chaque série est représentée par un point dans un espace vectoriel latent qui capture la dynamique du processus ainsi que les relations entre les différentes séries. La complétion des valeurs manquantes et la prédiction sont opérées directement à partir de ces représentations latentes. Le modèle permet ainsi de faire face aux deux tâches (prédiction et complétion) simultanément alors qu'elles sont habituellement traitées séparément dans la littérature. Le modèle est testé et comparé à des méthodes de l'état de l'art sur des jeux de données en prédiction de trafic.

3.1 Introduction

Dans les chapitres précédents, nous avons vu que les séries temporelles pouvaient correspondre à une large variété de données et donner naissance à différentes tâches. Cette première contribution se penche sur celles de la prédiction et de la complétion. Certaines caractéristiques particulières des séries temporelles qui peuvent être produites sont traitées : les données contiennent souvent beaucoup de valeurs manquantes, peuvent êtres hétérogènes, c'est-à-dire qu'une source d'information peut produire différents types d'information et si plusieurs sources de données produisent de l'information simultanément, il peut être intéressant de considérer les relations éventuelles entre ces différentes sources.

La prédiction de trafic routier est une bonne illustration des phénomènes décrits : l'émergence des capteurs mobiles (i.e. GPS) dans les véhicules aboutit à la production de données temporelles compliquées. Les capteurs sont mobiles, mesurant le trafic sur différentes routes à différents moments. Pendant une période donnée, certaines routes sont monitorées tandis que d'autres ne le sont pas, ce qui résulte en la production de séries avec beaucoup de valeurs manquantes - voir Figure 3.1.

D'une part, si d'autres modèles ont été proposés pour la prédiction de séries temporelles (univariées ou multivariées), dont les plus couramment utilisés en pratiques sont les réseaux de neurones [22] (voir état de l'art pour plus de détails), ces modèles ne sont pas adaptés à la présence de données manquantes. D'autre part, si des modèles de complétions ont été proposés [54], et en particulier dans le domaine de données de trafic [103] [6], les tâches de complétion et de prédiction sont usuellement traitées de manière séparée. De plus, les données produites sont souvent de

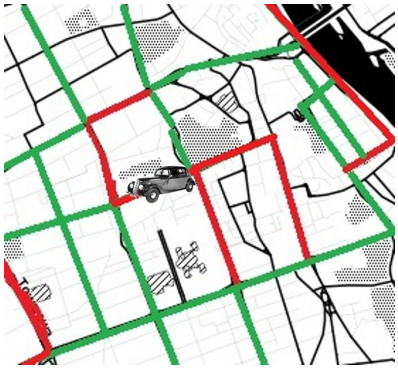

FIGURE 3.1 – Une partie d'un réseau routier où les routes sont surlignées en rouge si l'information sur le trafic est disponible pendant un pas de temps et en vert dans le cas contraire. Les capteurs (véhicules équipés de GPS) bougent en permanence ce qui résulte en une collecte de données très parcimonieuses.

natures diverses. Il n'existe pas à notre connaissance de modèle capable de faire face à ces caractéristiques conjointement.

Nous proposons une nouvelle méthode visant à prendre en compte tous les aspects et la complexité de ces données temporelles en un seul modèle. Cette approche se base sur de l'apprentissage de représentation où les observations sont projetées dans un espace latent continu, chaque séquence étant représentée par un point à chaque pas de temps dans cet espace. Cette méthode présente plusieurs avantages par rapport aux techniques existantes : (i) elle est capable de compléter les valeurs manquantes et d'apprendre à prédire le futur des séquences simultanément (ii) elle traite naturellement les séquences qui sont produites selon une structure de graphe (iii) elle est capable de traiter de l'information hétérogène, c'est-à-dire quand les séries sont constituées de plusieurs types d'informations. De plus, le modèle est basé sur un paradigme d'optimisation continue permettant un apprentissage efficient et passant facilement à l'échelle.

Les contributions présentées dans ce chapitre peuvent se résumer en :

1. Nous proposons un modèle d'apprentissage de représentation pour des données relationnelles et temporelles.
2. Nous montrons comment ce modèle peut résoudre conjointement les tâches de complétion et de prédiction.
3. Nous proposons une extension de ce modèle où chaque série temporelle est composée de différents types d'informations (valeurs réelles et labels discrets).

4. Nous menons des expériences en prédiction de trafic et notre méthode est comparée à plusieurs autres méthodes de l'état de l'art en complétion et en prédiction.

Ce chapitre est organisé comme suit : La section 3.2 décrit notre modèle d'apprentissage de représentation et détaille les procédures d'apprentissage et d'inférence. La section 3.4.5 décrit le protocole expérimental et donne les résultats expérimentaux. La section 3.5 introduit une extension possible à notre modèle. Enfin, la section 3.6 conclut cette première contribution.

3.2 Modèle

3.2.1 Notations et tâches

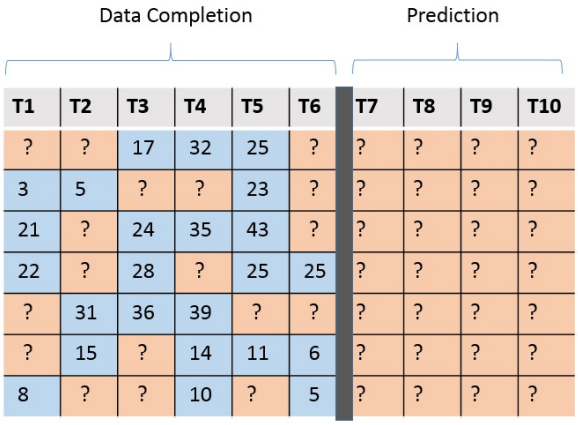

FIGURE 3.2 – Les deux tâches que nous voulons résoudre conjointement consistent en la complétion de T1 à T6 et à la prédiction de T7 à T10. Les carrés bleus sont des valeurs disponibles à l'apprentissage telles que $m_i^{(t)} = 1$

Nous considérons un ensemble de n séries temporelles notées $\mathbf{x_1}$, ..$\mathbf{x_n}$ telles que $x_i^{(t)} \in \mathcal{X}$ est la valeur de la i-ème série [1] au temps t définie par $\mathbf{x_i} = (x_i^{(1)}, .., x_i^{(T)})$. Dans le cas où $\mathcal{X}$ est $\mathbb{R}^m$, le contexte est celui de séries temporelles multivariées, mais notre approche peut également être utilisée pour des séquences de labels par exemples.

1. Nous introduisons d'abord une version "homogène" du modèle : la valeur de chaque séquence à chaque pas de temps est une valeur unique dans $\mathcal{X}$. Des expériences avec une version hétérogène dans la Section 3.5 sont menées, où $x_i^{(t)}$ est composé de plusieurs types d'informations.

Les séquences contiennent des valeurs manquantes. Est alors défini un masque $m_i^{(t)}$ tel que $m_i^{(t)} = 1$ si la valeur $x_i^{(t)}$ est observée - donc disponible pour entraîner le système - et $m_i^{(t)} = 0$ si $x_i^{(t)}$ est manquante - donc doit être prédite par le modèle. De plus, on considère l'existence a priori d'un ensemble de relations entre les séquences correspondant à de l'information extérieure, telle que la proximité spatiale. Les séquences sont alors organisées selon un graphe $\mathcal{G} = \{e_{i,j}\}$ tel que $e_{i,j} = 1$ signifie que $\mathbf{x_i}$ et $\mathbf{x_j}$ sont connectées, et $e_{i,j} = 0$ dans le cas contraire. Pour simplifier, nous considérons que le graphe de relations est statique et ne change pas dans le temps.

Les deux tâches que l'on veut résoudre conjointement - voir Figure 3.2 - sont les suivantes :

(i) le problème de la prédiction qui consiste à prédire *ce qui va arriver* à partir des données observées.

(ii) le problème de la complétion des valeurs manquantes dans les séquences en se basant sur les valeurs présentes. Notons $y_i^{(t)}$ la valeur de la séquence i au temps t prédite par le modèle, les deux tâches définies précédemment peuvent être explicitées de la façon suivante :

Prédiction des futures valeurs Cette tâche vise à prédire les valeurs au temps $T + t'$ où T est le temps des séquences observées durant l'apprentissage et t' l'horizon de prédiction. La qualité de prédiction peut être mesurée en prenant une erreur moyenne entre les valeurs prédites et les valeurs réellement observées :

$$\mathscr{P}^{pred}(t') = \frac{1}{n} \sum_{i=1}^{n} \Delta(y_i^{(T+t')}, x_i^{(T+t')})$$

où Δ est une mesure de l'erreur de prédiction définie telle que : $\Delta : \mathscr{X} \times \mathscr{X} \rightarrow \mathbb{R}^+$. L'erreur des moindres carrés peut par exemple être utilisée.

Complétion des valeurs manquantes La complétion vise à remplir les valeurs manquantes (où $m_i^{(t)} = 0$) à partir des valeurs observées. La performance peut être évaluée comme suit :

$$\mathscr{P}^{comp} = \frac{1}{n.T} \sum_{t=1}^{t=T} \sum_{i=1}^{n} (1 - m_i^{(t)}) \Delta(y_i^{(t)}, x_i^{(t)})$$

Une erreur des moindres carrés est également un choix commun pour la complétion. Il est à noter que la complétion est usuellement une tâche de prétraitement des données basée le plus souvent sur des heuristiques et dont les performances ne sont que rarement évaluées quantitativement.

Dans la partie expérimentale, ces mesures seront évaluées sur un ensemble de test pour comparer l'approche proposée avec d'autres modèles de l'état de l'art - voir Section 3.4.5

3.2.2 Idée principale

Le but de notre approche est de prendre en compte les différents types d'informations disponibles dans les données et qui sont les suivants :

— Les valeurs observées
— Les relations entre les différentes séries temporelles
— La dynamique des séries observées

On se propose de capturer cette information dans un espace latent de grande dimension noté $\mathcal{Z}$, dans lequel chaque observation correspondra à un point particulier (ou une représentation) à chaque pas de temps, dénoté $z_i^{(t)} \in \mathcal{Z}$. Traitant n sources d'informations pendant une durée T, le modèle apprendra alors $T \times n$ représentations dans $\mathcal{Z}$, chaque source d'information $\mathbf{x_i}$ étant décrite par une trajectoire apprise ($z_i^{(1)}$, ..., $z_i^{(T)}$) dans l'espace latent.

Ce genre d'approche a déjà été proposé pour capturer des informations à la structure complexe, comme dans [18] pour la diffusion d'information, [58] pour les graphes relationnels ou [33] pour les processus de décision markovien, mais jamais encore pour les données temporelles et relationnelles.

Pour capturer toute l'information disponible, les représentations latentes qui sont construites respecteront les contraintes suivantes :

Pas de perte d'information : Premièrement, chaque représentation de chaque série temporelle à chaque pas de temps $z_i^{(t)}$ doit permettre de prédire la valeur de la série $x_i^{(t)}$. Cette propriété correspond au fait qu'à partir de la représentation apprise, il doit être possible de retrouver l'observation correspondante, et donc de compléter les valeurs manquantes ou de prédire les futurs valeurs.

Dynamicité : Deuxièmement, nous voulons également capturer la dynamique de la source $\mathbf{x_i}$. Cette caractéristique sera utilisée pour prédire le comportement futur d'une série. Pour cela, chaque représentation $z_i^{(t)}$ est enrichie avec de l'information dynamique.

Relations entre les différentes sources : Enfin, pour intégrer l'information relationnelle entre les séquences, nous considérerons que deux sources d'information reliées tendront à se comporter de manière similaire, et donc les trajectoires (dans l'espace de représentation) de deux séquences reliées doivent être proches l'une de l'autre.

Suivant ces contraintes, nous décrivons comment les représentations peuvent être apprises depuis les observations dans la section suivante. Notons que, à l'opposé des méthodes inductives classiques (réseaux de neurones par exemple) où des représentations sont calculées depuis les observations, notre modèle est un modèle transductif où des représentations $z_i^{(t)}$ sont des paramètres à partir desquels les observations peuvent être calculées. Le modèle proposé agit plus comme un modèle génératif que comme un modèle discriminant, ce qui est utile puisque cela permet de prédire les valeurs manquantes aussi bien que de générer la future évolution des séries temporelles.

3.3 RepresentAtIoN-baSed TempORal relational Model

Nous présentons maintenant comment les contraintes décrites dans la section précédente peuvent être intégrées dans un seul modèle. Le modèle RAINSTORM (*RepresentAtIoN-baSed TempOral Relational Model*) est un modèle basé sur un coût à minimiser qui est décrit par une fonction continue et dérivable et dont l'optimisation se fera en utilisant des techniques classiques. L'approche proposée est proche de celles qui existent dans la littérature des réseaux de neurones profonds mais diffère dans la mesure où une représentation explicite $z_i^{(t)}$ est apprise pour chaque pas de temps et pour chaque source (comme fait dans [33]). L'intérêt principal de procéder ainsi est d'être capable de traiter naturellement et facilement les valeurs manquantes tandis que les approches classiques ne sont pas ou peu adaptées pour faire face à ce problème.

On définit $\mathscr{L}(\theta, \gamma, \mathbf{z})$ la fonction de coût à minimiser où $\mathbf{z}$ est l'ensemble de tous les vecteurs $z_i^{(t)}$ pour $i \in [1..n]$ et $t \in [1..T]$, T étant la taille de la fenêtre temporelle observée i.e. l'historique des séries temporelles. On définit $\mathscr{L}$ ainsi :

$$\begin{aligned}\mathscr{L}(\theta, \gamma, \mathbf{z}) = \frac{1}{O} \sum_{i=1}^{n} \sum_{t=1}^{T} m_i^{(t)} \Delta(d_\theta(z_i^{(t)}), x_i^{(t)}) \text{ (terme 1)} \\ + \lambda_{dyn} \sum_{i=1}^{n} \sum_{t=1}^{T-1} ||z_i^{(t+1)} - h_\gamma(z_i^{(t)})||^2 \text{ (terme 2)} \\ + \lambda_{struct} \sum_{i,j \in [1..N]^2} \sum_{t=1}^{T} e_{i,j} ||z_i^{(t)} - z_j^{(t)}||^2 \text{ (terme 3)}\end{aligned} \tag{3.3.1}$$

Où O est le nombre de valeur observées, i.e. des valeurs telles que $m_i^{(t)} = 1$.

Cette fonction de coût contient trois termes, chacun d'entre eux étant associé à l'une des contraintes présentées précédemment :

— Le terme 1 vise à apprendre simultanément $\mathbf{z}$ et les paramètres d'une fonction d_θ - appelée **fonction de décodage** - tel que, depuis $z_i^{(t)}$, d_θ peut être uti-

lisée pour inférer la valeur $x_i^{(t)}$. La fonction $d_\theta(z_i^{(t)})$ est définie comme suit : $d_\theta : \mathbb{R}^N \to \mathscr{X}$. Δ est utilisé pour mesurer l'erreur entre la prédiction $d_\theta(z_i^{(t)})$ et la valeur réellement observée $x_i^{(t)}$, $m_i^{(t)}$ jouant le rôle d'un masque restreignant le calcul de cette fonction aux seules valeurs observées, ignorant les valeurs manquantes qui sont inconnues et ne peuvent donc pas êtres utilisées en apprentissage.

— Le terme 2 vise à trouver les valeurs $z_i^{(.)}$ et un modèle dynamique h_γ tel que, lorsqu'il est appliqué à $z_i^{(t)}$, h_γ permet de prédire la représentation du prochain état de la série temporelle i i.e. $z_i^{(t+1)}$. h_γ est la **fonction dynamique** qui modélise la dynamicité de chaque série directement dans l'espace latent : $h_\gamma : \mathbb{R}^N \to \mathbb{R}^N$. Les paramètres γ seront appris pour minimiser la distance euclidienne (erreur des moindres carrés) entre la prédiction $h_\gamma(z_i^{(t)})$ et $z_i^{(t+1)}$, assurant le fait que h_γ es un bon modèle dynamique de la série dans $\mathbb{R}^N$.

— Enfin, le terme 3 correspond à un terme de **régularisation structurelle** sur la structure du graphe qui force le modèle à apprendre des représentations proches pour des séries temporelles qui sont reliées. Les représentations ainsi apprises refléteront la structure du graphe considéré.

λ_{dyn} et λ_{struct} sont des coefficients dont la valeur est fixée manuellement et qui représentent le poids de l'importance des différents termes dans la fonction à minimiser. Pour d_θ et h_γ, différentes architectures peuvent être choisies (elles se doivent d'être continues et dérivables) et nous en proposons plusieurs dans la partie expérimentale. Les valeurs de ces hyper-paramètres sont choisies en utilisant des techniques de validation classiquement utilisées et décrites dans la section expérimentale.

3.3.1 Apprentissage

Le problème d'apprentissage vise à minimiser la fonction de coût $\mathscr{L}(\theta, \gamma, \mathbf{z})$ simultanément sur θ, γ et $\mathbf{z}$. En restreignant d_θ et h_γ à des fonctions continues dérivables, nous nous proposons d'utiliser des approches de descente de gradient. La fonction de coût n'est clairement pas convexe mais les algorithmes de descente de gradient permettent de trouver des minimums locaux pour ces problèmes. L'optimisation de coûts complexes et non convexes a récemment connu d'importants succès dans plusieurs différentes applications (i.e. réseaux de neurones profonds). Nous vérifions que ce type de modèle est capable de prendre en compte des contraintes complexes, et que même dans des minimums locaux, est capable d'avoir des résultats intéressants - voir Section 3.4.5.

Différentes techniques de descente de gradient peuvent être utilisées : descente de gradient batch, stochastique, descente de gradient alternée,... on se propose d'utili-

ser une descente de gradient stochastique décrite dans l'algorithme 3. L'algorithme prend en entrée les séquences observées, le masque défini sur les séquences, le graphe de relations entre séquences, des fonctions dynamiques et de décodages initialisées aléatoirement et une représentation pour chaque pas de temps de chaque séquence (ligne 2-6). Puis, itérativement, il modifie les paramètres de la fonction de décodage (ligne 12) et les représentations correspondantes (ligne 11) dans la direction de leur gradient. Cela correspond au terme 1 dans l'équation (1). Les paramètres de la fonction dynamique et les représentations concernées sont modifiées de la même manière (ligne 15-17). Le modèle met également à jour ses paramètres qui correspondent à la régularisation structurelle (ligne 19-20). L'algorithme à convergence aura appris un ensemble de points dans l'espace latent $\mathbf{z}$ mais également les paramètres des fonctions de décodage et dynamiques, respectivement d_θ et h_γ. Toute cette information peut alors être utilisée pour la prédiction et la complétion. L'expressivité du modèle est essentiellement contrôlée par la taille de l'espace latent N qui sera choisie par validation.

3.3.2 Inférence

Maintenant, considérons qu'ayant à notre disposition un ensemble de séquences observées, $\mathbf{z}$, d_θ et h_γ ont été appris. Nous expliquons ci-dessous comment ils peuvent être utilisés en pratique.

3.3.2.1 Complétion de valeurs manquantes

Pour toutes les valeurs $x_i^{(t)}$ telles que $m_i^{(t)} = 0$, l'algorithme d'apprentissage proposé a appris une représentation $z_i^{(t)}$ dans l'espace latent. Cette représentation a été apprise en se basant sur les termes 2 et 3 de la fonction de coût (Équation 1) tandis que la fonction de décodage n'a pas été influencée par $z_i^{(t)}$ comme $x_i^{(t)}$ est inconnue. Pour inférer cette valeur manquante, notre modèle calcule simplement la valeur $d_\theta(z_i^{(t)})$ qui produit une valeur plausible en sortie. Notons que, quand la fonction de décodage est linéaire, la valeur prédite est $\theta^T z_i^{(t)}$ qui est très proche de la complétion obtenue classiquement avec des techniques de factorisation matricielle ; cependant θ et $\mathbf{z}$ ont été appris en se basant sur des informations multiples (en particulier le graphe de relations entre séries temporelles).

3.3.2.2 Prédiction des futures valeurs

Maintenant, nous expliquons comment le modèle peut être utilisé pour prédire les futures valeurs $x_i^{(t)}$. Pour tout $t > T$, le modèle ne calcule pas de représentation

Algorithm 1 Stochastic Gradient Descent Algorithm

input :
  $\forall i,\ t,\ x_i^{(t)} \in \mathcal{X}$
  $\forall i,\ t,\ m_i^{(t)}$
  $\epsilon \leftarrow$ learning rate
procedure LEARNING(θ, γ, $\mathbf{z}$)
    $\forall i,\ t,\ z_i^{(t)} \leftarrow random, \gamma \leftarrow random, \theta \leftarrow random$
  for number of iteration **do**
    Sample $i,\ t,\ j$
    %Gradient descent for term (1)
    if $m_i^{(t)} == 1$ **then**
      $z_i^{(t)} \leftarrow z_i^{(t)} - \epsilon \nabla_{z_i^{(t)}} \Delta(d_\theta(z_i^{(t)}),\ x_i^{(t)})$
      $\theta \leftarrow \theta - \epsilon \nabla_\theta \Delta(d_\theta(z_i^{(t)}),\ x_i^{(t)})$
    end
    %Gradient descent for term (2)
    $z_i^{(t)} \leftarrow z_i^{(t)} - \epsilon \lambda_{dyn} \nabla_{z_i^{(t)}} \Delta(h_\gamma(z_i^{(t)}),\ z_i^{(t+1)})$
    $z_i^{(t+1)} \leftarrow z_i^{(t+1)} - \epsilon \lambda_{dyn} \nabla_{z_i^{(t+1)}} \Delta(h_\gamma(z_i^{(t)}),\ z_i^{(t+1)})$
    $\gamma \leftarrow \gamma - \epsilon \lambda_{dyn} \nabla_\gamma \Delta(h_\gamma(z_i^{(t)}),\ z_i^{(t+1)})$
    %Gradient descent for term (3)
    $z_i^{(t)} \leftarrow z_i^{(t)} - \epsilon \lambda_{struct} \nabla_{z_i^{(t)}} ||z_i^{(t)} - z_j^{(t)}||^2 e_{i,j}$
    $z_j^{(t)} \leftarrow z_j^{(t)} - \epsilon \lambda_{struct} \nabla_{z_j^{(t)}} ||z_i^{(t)} - z_j^{(t)}||^2 e_{i,j}$
    end
end procedure

et les $z_i^{(t)}$ correspondants sont inconnus. Mais le modèle apprend une fonction dynamique h_γ dont le but est de permettre la prédiction de $z_i^{(t+1)}$ connaissant $z_i^{(t)}$. Ainsi, pour tout i, $z_i^{(T+1)}$ peut être calculé par $h_\gamma(z_i^{(T)})$, $z_i^{(T+2)}$ peut être calculé par $h_\gamma(h_\gamma(z_i^{(T)}))$ et ainsi de suite. h_γ agit comme une fonction générative de la séquence dans l'espace latent. La future valeur $x_i^{(T+s)}$ peut alors être prédite en calculant simplement $d_\theta(h_\gamma(h_\gamma(h_\gamma(.....h_\gamma(z_i^{(T)})..)))$ où h_γ est appliqué s fois et le vecteur ainsi obtenu étant transformé en prédiction en utilisant d_θ.

3.4 Prédiction de trafic et expériences

3.4.1 Trafic routier

On considère qu'un réseau de routes est un graphe où chaque nœud correspond à une route et où chaque arête correspond à une connexion entre routes : deux routes sont connectées si elles partagent une intersection. On considère que, à chaque pas de temps, un ensemble de capteurs donne une mesure du trafic sur un sous-ensemble de routes (i.e. les routes telles que $m_i^{(t)} = 1$)). Cette mesure du trafic peut typiquement être la vitesse moyenne des voitures qui empruntent la route considérée pendant une période de temps, ou un volume de véhicules [2]. On note que le sous-ensemble de routes pour lesquelles on obtient des mesures au temps t est susceptible de changer au temps $t+1$ car les capteurs sont mobiles et se déplacent dans la ville (des véhicules équipés de GPS par exemple). Notre objectif dans cette tâche de prédiction de trafic est de compléter les valeurs manquantes, c'est-à-dire les valeurs qui correspondent aux routes dont le trafic n'a pas été mesuré au temps t, et également de prédire ce qui arrivera sur l'ensemble des routes pour les prochains pas de temps. Dans la littérature en prédiction de trafic à court terme, le pas de temps correspond typiquement à un intervalle de quelques minutes (e.g. 15 minutes) et la prédiction est faite pour les prochaines heures.

3.4.2 Données réelles

Nous menons des expériences pour valider l'intérêt du modèle proposé sur deux datasets. Les données collectées consistent en des trajectoires GPS d'un grand nombre de véhicules dans les villes de Pékin et de Varsovie qui sont converties en séries temporelles relationnelles par le traitement qui suit :

— Les trajectoires sont map-matchées [3] (de manière analogue à ce qui est fait dans [60]).
— En se basant sur le matching obtenu, les vitesses et volumes de véhicules sont calculés pour chaque route à chaque pas de temps.
— Le graphe relationnel entre séries est construit en considérant que deux routes qui s'intersectent sont reliées. Des statistiques concernant les deux datasets

2. La quantité numérique représentant le trafic varie dans la littérature ; il en existe plusieurs et le choix de la plus adaptée peut faire débat. Cette thèse n'a pas vocation à répondre à cette question et les modèles proposés sont versatiles et peuvent prendre en entrée n'importe quelle quantité numérique mesurant le trafic

3. Il s'agit du procédé qui consiste à projeter un point GPS appartenant à la trajectoire d'un véhivule et de le projeter sur une carte pour identifier la route ou le segment de route emprunté.

TABLE 3.1 – Statistiques sur les deux datasets : Le volume est dérivé à partir de la présence d'un véhicule sur une route durant une fenêtre temporelle donnée. L'information sur la vitesse n'est présente que sur le dataset polonais. La "sparsité" est le pourcentage de valeurs manquantes, i.e. le pourcentage de routes qui à certains moments ne sont traversées par aucun véhicule.

Datasets	Pekin	Varsovie
Volume	Oui	Oui
Speed	Non	Oui
Nb de routes/séquences	24 000	20 000
Taille des time-steps t	15 min	10 min
Nb de time-steps	672	100
Sparsité	69%	81%

sont données dans le Tableau 3.1.

Dataset Pékin : ce dataset est rendu disponible par [130] et consiste en des trajectoires d'environ 10500 taxis pendant une semaine, pour un total de 17 millions de points. Le volume du trafic est alors agrégé en fenêtres de 15 minutes.

Dataset Varsovie : Ce jeu de données a été publié en 2010 dans le contexte du "ICDM data mining challenge" [1]. Le but du challenge était la prédiction de trafic et les données fournies l'ont été pour la ville de Varsovie en Pologne. Les données provenaient de 500 simulations de 10 heures chacune avec une très forte fréquence (un point toutes les 10 secondes), pour un total d'environ 130 millions de points. Nous agrégeons les données sur ~20 000 routes sur des fenêtres temporelles de 10 minutes et la vitesse moyenne et le volume de véhicules sont calculés pour chaque routes durant la période considérée.

3.4.3 Protocole expérimental

Dans le but d'évaluer l'efficacité de notre méthode, nous utilisons le protocole expérimental suivant : les données sont séparées en deux groupes, l'un d'entraînement et l'autre de test. Les valeurs de test peuvent être de deux types : une partie d'entre elles tirée aléatoirement (loi uniforme) dans l'ensemble des valeurs observées pour $t \in [1..T]$, T étant la taille de la plage temporelle étudiée. Ces valeurs de test seront utilisées pour évaluer les performances en complétion de notre modèle. Toutes les valeurs pour $t > T$ seront traitées comme des valeurs de tests pour évaluer les performances prédictives du modèle. De plus, un sous-ensemble des va-

T1	T2	T3	T4	T5	T6	T7	T8	T9	T10
12	21	17	32	25	28	30	34	30	24
3	5	12	11	23	23	27	19	11	14
21	21	24	35	43	44	51	67	38	45
22	26	28	27	25	25	21	29	18	22
35	31	36	39	51	44	60	57	57	59
24	15	13	14	11	6	12	8	6	7
8	5	3	10	10	5	9	13	14	17

FIGURE 3.3 – Les cases bleues correspondent aux valeurs de train, les vertes aux données de validation et les rouges appartiennent à l'ensemble de test.

leurs de test sera utilisé comme données de validation pour optimiser les hyperparamètres du modèle et les meilleures architectures pour d_θ et h_γ. Le protocole apprentissage/validation/test est illustré dans la Figure 3.3. On note que, pour chaque dataset, on extrait plusieurs problèmes de tailles T en utilisant des fenêtres glissantes sur les données collectées. La taille des fenêtres est de 96 time-steps pour Pékin et 40 time-steps pour Varsovie.

3.4.4 Méthodes de l'état de l'art

Nous proposons de comparer notre approche notée "RAINSTORM" aux méthodes de l'état de l'art qui suivent ; certaines étant des méthodes de complétion de données et d'autres de prédiction.

3.4.4.1 Complétion

— **RecentObs** : Une heuristique qui complète une valeur manquante dans une séquence par la dernière valeur observée dans la séquence tel que $x_i^{(t)} \leftarrow x_i^{(t-1)}$ si $m_i^{t-1} = 1$, ou $x_i^{(t)} \leftarrow x_i^{(t-2)}$ si $m_i^{t-1} = 0$ et ainsi de suite.
— **MF** : Méthode de factorisation matricielle classique, décrite par exemple dans le cadre de la complétion de données de trafic dans [102].
— **MF-avec contexte géographique** : Cette méthode est décrite sous le nom de TSE (traffic speed estimation) dans [102]. Elle consiste en la minimisation d'un coût de reconstruction sur une matrice contenant l'information de trafic et pour laquelle de l'information annexe comme la position géographique dans une ville est incorporée.

3.4.4.2 Prédiction

— **Last-Point** : Cette heuristique prédit que le trafic au temps $t+1$ sera le même que le trafic au temps t tel que $x_i^{(t)} \leftarrow x_i^{(t-1)}$.
— **NeuralNetwork** : Une baseline classique en prédiction de trafic utilisant une architecture de réseaux de neurones, décrite par exemple dans [41].
— **SAE** : Méthode décrite dans [79] ; elle consiste en une architecture profonde d'auto-encodeurs empilés entraînés sur l'historique de trafic.

Nous comparons aussi RAINSTORM avec un modèle basé sur une heuristique capable de traiter les deux tâches, c'est-à-dire de complétion et de prédiction que nous notons **RoadMean** et qui peut être décrite de la façon suivante : ce modèle prédit et complète les valeurs manquantes avec la moyenne des valeurs observées sur la séquence.
Le modèle RAINSTORM a été testé avec les spécificités suivantes : (i) d_θ est une fonction linéaire (ii) trois différentes architectures pour h_γ ont été testées :

— **RAINSTORM-lin** qui utilise une fonction h_γ linéaire
— **RAINSTORM-trans** qui utilise une fonction h_γ telle que $h_\gamma(z) = z + \gamma$
— **RAINSTORM-mlp** qui utilise un perceptron multicouche avec une couche cachée pour h_γ, avec 200 neurones sur la couche cachée et la fonction tangente hyperbolique comme fonction d'activation. Celà permet l'apprentissage d'une dynamique plus complexe dans l'espace latent.

3.4.5 Expériences et résultats

Complétion des données manquantes
Dans le premier ensemble d'expériences, nous nous attaquons à la tâche de complétion des valeurs manquantes sur les deux datasets. Nous utilisons 50% des observations en entraînement, 40% en test et 10% en validation. Chaque performance reportée dans le tableau a été calculée en moyennant les résultats obtenus sur 20 différentes expériences. Ici, la fonction Δ est la fonction RMSE (root mean squared error). Le Tableau 3.2 illustre les résultats obtenus par les baselines et par notre modèle pour différentes tailles de l'espace latent N. Une première observation est que les trois modèles RAINSTORM obtiennent de meilleures performances que les méthodes concurrentes pour presque toutes les dimensions de N testées. En particulier, sur le dataset de la ville de Pékin, le modèle RAINSTORM_mlp avec $N = 50$ obtient un RMSE de 2.97 tandis que la meilleure baseline (SAE) n'obtient que 3.24. Celà correspond à un gain de performance de $\sim 10\%$. Celà s'explique par la capacité du modèle à, à la fois capturer les dynamiques des séquences mais également à tirer parti de l'information relationnelle. Le Tableau 3.4 montre les performances de ce modèle lorsque l'on considère uniquement les termes de décodeur et de dynamique

N	Modèle/Données	Beijing	Warsaw	
		Volume	Volume	Speed
	RoadMean	5.55	5.00	11.10
	RecentObs	5.31	5.11	11.38
	MF	3.58	3.16	6.80
	MF-Geo	3.24	2.99	6.49
RAINSTORM				
5	Linear	3.37	3.13	6.41
	Translation	3.19	3.12	6.32
	MLP	2.99	3.12	6.49
10	Linear	3.01	3.09	6.40
	Translation	3.05	3.17	6.51
	MLP	3.03	3.00	6.24
20	Linear	3.52	2.97	6.47
	Translation	3.21	**2.92**	6.45
	MLP	3.22	2.94	**6.23**
50	Linear	3.53	2.99	6.32
	Translation	3.08	2.95	6.35
	MLP	**2.97**	2.93	6.70

TABLE 3.2 – Complétion pour 50% de données manquantes; comparaison entre les modèles de l'état de l'art et le modèle RAINSTORM pour différentes tailles d'espace latent N avec la racine d'une erreur des moindres carrés (root mean square error (RMSE))

($\lambda_{struc} = 0$) ainsi que lorsque l'on considère uniquement les termes de décodage et structurels ($\lambda_{dyn} = 0$). Il montre que le gain par rapport aux baselines vient de ces deux informations et qu'enlever l'un de ces termes fait chuter les performances du modèle.

Prédiction

Le deuxième ensemble d'expériences s'intéresse au problème de la prédiction. Ici, les valeurs aux temps $t > T$ sont ôtées de l'ensemble d'apprentissage; 20% d'entre elles sont utilisées comme valeurs de validation et les 80% restants sont utilisées comme valeur de test. Le Tableau 3.4.5 montre les performances des différents modèles en utilisant la fonction RMSE comme métrique d'évaluation au temps $t = T +$

N	Modèle/Données	Beijing	Warsaw	
		Volume	Volume	Speed
	RoadMean	5.51	5.09	11.02
	Last-Point	5.28	4.73	12.01
	NeuralNetwork	4.77	4.27	8.05
	SAE	4.75	4.27	7.85
RAINSTORM				
5	Linear	5.07	4.28	7.48
	Translation	4.89	4.32	7.72
	MLP	4.82	4.28	7.74
10	Linear	4.91	4.25	7.40
	Translation	4.85	4.29	8.00
	MLP	4.78	**4.20**	7.21
20	Linear	4.72	4.24	7.33
	Translation	4.67	4.22	7.54
	MLP	**4.54**	4.21	**7.19**
50	Linear	4.73	4.27	7.22
	Translation	4.68	4.39	7.45
	MLP	4.66	**4.20**	7.60

TABLE 3.3 – Prédiction à $T+1$, comparaison entre les modèles de l'état de l'art et le modèle RAINSTORM pour différentes tailles d'espace latent N avec la racine d'une erreur des moindres carrés (root mean square error (RMSE))

Space size	5	10	20	50
$\lambda_{dyn} = 0$	8.01	7.71	7.82	7.67
$\lambda_{struct} = 0$	7.59	7.54	7.55	7.48
$\lambda_{dyn}, \lambda_{struct} \neq 0$	6.49	6.24	6.23	6.70

TABLE 3.4 – Performance en complétion (RMSE) lorsque l'on enlève le terme structurel, lorsque l'on enlève le terme dynamique et lorsque les deux informations sont gardées. Les résultats sont donnés pour le dataset de la ville de Varsovie avec 50% de valeurs manquantes et avec un perceptron multicouche comme fonction dynamique.

1 sur le volume de véhicules ou sur la vitesse moyenne sur les routes considérées. Il montre également que, utilisé comme modèle prédictif, RAINSTORM obtient de meilleures performances que les baselines pour les deux villes (Pékin et Varsovie). Lorsque l'on considère la qualité de prédiction à plus long terme (Figure 3.4), notre

modèle obtient toujours de bonnes performances prédictives de $T+1$ à $T+7$ mais est moins performant pour la prédiction du trafic à long terme, ce qui s'explique par le fait que la fonction h_γ ne modélise jamais parfaitement la dynamique des phénomène sous-jacents.

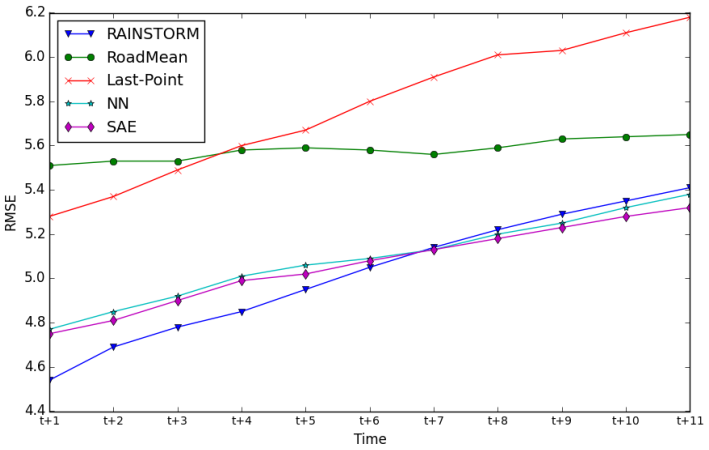

FIGURE 3.4 – Évolution de l'erreur en prédisant les futurs time-steps, de $T+1$ à $T+11$ sur le dataset de la ville de Pékin

Prédiction et Complétion Simultanées

Le troisième ensemble d'expériences se focalise sur la résolution simultanée des problèmes de prédiction et de complétion. Ici, l'ensemble d'apprentissage est construit en ignorant des observations pour $t \leq T$ mais également toutes les valeurs aux temps $t > T$ qui seront utilisées pour évaluer la qualité de prédiction. Nous comparons les performances en complétion et prédiction aux modèles de l'état de l'art. On note que, pour la tâche de prédiction, les baselines sont apprises sans considérer de valeurs manquantes sur l'ensemble d'apprentissage pour $t \leq T$ car ces modèles ne sont pas capables d'être appris dans cette configuration. Le Tableau 3.5 présente les performances obtenues. On y voit que si la qualité de prédiction de notre modèle décroît en comparaison aux résultats obtenus en prédiction pure (Tableau 3.4.5), et les performances de prédiction et les performances de complétion restent meilleures que celles des modèles de l'état de l'art. La diminution des performances en prédiction s'explique facilement par le fait que le modèle est entraîné avec des valeurs manquantes pour $t \leq T$.

	Complétion		Prédiction	
Pourcentage de valeurs manquantes	RM	RS	RM	RS
5%	4.88	**2.89**	5.51	**4.63**
10%	4.89	**2.89**	5.55	**4.67**
20%	5.04	**2.91**	5.55	**4.79**
30%	5.11	**2.96**	5.64	**4.91**
50%	5.55	**3.22**	5.65	**5.06**
75%	5.90	**3.78**	5.81	**5.26**

TABLE 3.5 – Performance en Complétion et Prédiction pour différents niveaux de valeurs manquantes sur le dataset issu de la ville de Pékin pour le modèle RoadMean (RM) et le modèle RAINSTORM-mlp (RS) avec un espace latent de taille 20

3.5 Information hétérogène

Nous considérons également un autre problème expérimental où, à chaque pas de temps, l'observation $x_i^{(t)}$ est composée de labels dénotés $x_{i,lab}^{(t)}$ et de valeurs réelles dénotées $x_{i,real}^{(t)}$. Notons que nous présentons un modèle hétérogène dans ce cas, mais des données plus complexes peuvent être intégrées et le modèle n'est pas limité au traitement des données réelles ou des labels. $x_{i,real}^{(t)}$ correspond aux valeurs utilisées précédemment, et $x_{i,lab}^{(t)}$ est un label (dans un ensemble de trois labels possibles) qui peuvent être *forte congestion*, *congestion moyenne* et *faible congestion*. Nous étendons le modèle précédent en considérant deux fonctions de décodages différentes :

— $d_{\theta,real}$ dont le but est de prédire la valeur $x_{i,real}^{(t)}$.
— $d_{\theta,lab}$ dont le but est de prédire le bon label parmi les trois possibles.

Ces deux fonctions de décodage sont intégrées naturellement dans la fonction de coût de notre modèle en utilisant l'écriture suivante :

$$\begin{aligned}\mathscr{L}(\theta, \gamma, \mathbf{z}) = \frac{1}{O} \sum_{i=1}^{n} \sum_{t=1}^{T} & m_{i,real}^{(t)} (d_{\theta,real}(z_i^{(t)}) - x_{i,real}^{(t)})^2 \\ & + m_{i,lab}^{(t)} \Delta_{hinge}(d_{\theta,lab}(z_i^{(t)}), x_{i,lab}^{(t)}) \\ & + \lambda_{dyn} \sum_{i=1}^{n} \sum_{t=1}^{T-1} ||z_i^{(t+1)} - h_\gamma(z_i^{(t)})||^2 \\ & + \lambda_{struct} \sum_{i,j \in [1..N]^2} \sum_{t=1}^{T} e_{i,j} ||z_i^{(t)} - z_j^{(t)}||^2\end{aligned} \tag{3.5.1}$$

Dans nos expériences, les valeurs notées "real" et celles notées "lab" ne sont pas

forcément manquantes aux mêmes moments et aux mêmes endroits, d'où la nécessité de définir deux masques différents $m^{(t)}_{i,real}$ et $m^{(t)}_{i,lab}$. Δ_{hinge} est un hinge-loss usuel de classification.

Le Tableau 3.5 montre les résultats de notre approche en considérant uniquement les valeurs réelles quantitatives observées, uniquement les labels présents et également en considérant les deux types de données simultanément. On peut remarquer qu'en considérant les deux types d'information simultanément, notre modèle est capable d'obtenir de meilleures performance en termes de RMSE pour les valeurs "réelles" ainsi qu'en termes de précision pour les labels, montrant la capacité de RAINSTORM à intégrer l'information hétérogène dans l'espace latent et également de tirer profit de cette information complémentaire.

TABLE 3.6 – Complétion pour 50% de données manquantes avec la version hétérogène du modèle RAINSTORM pour le dataset issu de la ville de Pekin. Les résultats sont donnés pour RAINSTORM-mlp : trois configurations différentes sont considérées : l'une avec des valeurs réelles seulement, une autre avec des labels uniquement et une troisième avec les deux types de données disponibles.

	x real only	**x** label only	both **x**
	Accuracy	Rmse	Accuracy/Rmse
N=5	0.71	2.99	0.74/3.11
N=10	0.77	3.03	0.77/3.15
N=20	0.79	3.22	0.84/3.24
N=50	0.76	2.97	0.82/3.05
N=60	0.78	2.99	**0.81/2.96**
N=80	0.77	3.05	0.82/2.98

3.6 Conclusion

Dans la première partie de cette thèse, nous avons proposé une nouvelle manière d'apprendre les dynamiques de multiples séquences temporelles et relationnelles et ayant la particularité de posséder un grand nombre de valeurs manquantes. Notre approche appelée RAINSTORM est basée sur des techniques d'apprentissage de représentation et vise à intégrer dans un espace latent l'information observée, la dynamique sous-jacente aux séquences observées et les relations entre ces dernières. De plus, une simple modification du modèle permet de traiter des sources de données hétérogènes. En comparaison avec des modèles de l'état de l'art qui ont été développés pour la prédiction seulement ou pour la complétion seulement, notre approche montre d'intéressantes performances et est capable de résoudre les deux

tâches décrites de manière satisfaisante. L'intérêt de considérer l'information relationnelle comme a priori est montré par nos expériences : en effet, les autres modèles de l'état de l'art testés qui ne prennent pas en compte cette information ont de moins bonnes performances prédictives. L'approche que nous avons proposée est donc particulièrement adaptée à des problèmes de prédiction de séquences relationnelles ; sa capacité à également traiter les valeurs manquantes la rend spécialement adaptée à la tâche de prédiction de trafic et aux données de transport en général.

CHAPITRE 4

AUTRES CONTRIBUTIONS

Résumé

Nous proposons deux variantes du modèle décrit dans le chapitre précédent. Dans la première, nous donnons une formulation du modèle permettant de prédire différents types de séries temporelles. Nous menons des expériences sur le trafic automobile et la disponibilité des places de stationnement dans la ville de Lyon : nous montrons comment notre approche peut s'attaquer à la prédiction jointe du trafic et de l'occupation des parkings. Dans la seconde, une extension du modèle RAINSTORM est également donnée où nous montrons que l'utilisation de représentations gaussiennes au lieu des représentations déterministes permet d'une part d'avoir des performances de prédiction améliorées mais aussi d'avoir une interprétation quantitative de l'incertitude de prédiction. Les composantes relationnelles et temporelles des séries sont ici modélisées par des distributions gaussiennes dans l'espace latent.

4.1 Prédiction de Parkings et de Trafic

Les systèmes adaptatifs de contrôle de la circulation au sein des villes utilisent l'information en temps réel sur l'importance du trafic automobile pour minimiser la congestion. La durée des feux de circulation est également un facteur permettant d'optimiser la fluidité du trafic. Si l'efficacité de ces méthodes dépend surtout de la fiabilité de l'évaluation du trafic en temps réel, l'information prévisionnelle est cruciale pour anticiper les congestions. Dans cette optique et sachant qu'une part substantielle est due à des véhicules recherchant une place de stationnement, prédire l'occupation des parkings est également très important : récemment, à San Francisco, il a été évalué que 30% de la congestion était due à des véhicules à la recherche de stationnement [92]. En effet, en cherchant une place, les automobilistes se dirigent vers un parking qu'ils espèrent libre et s'ils se trompent, peuvent avoir à parcourir de grandes distances pour trouver un nouvel emplacement ce qui résulte en une perte de temps doublée d'une aggravation de la congestion. Une solution intéressante serait d'avoir des modèles capables, en plus de prédire le trafic, de prévoir à quels endroits le stationnement sera le plus susceptible d'être disponible dans les prochaines heures. Si la prédiction de trafic et de stationnement ont déjà été étudiées dans la littérature [37, 10, 21, 88] , ces deux phénomènes ont été modélisés séparément. Nous proposons une extension du modèle RAINSTORM pouvant modéliser ces deux phénomènes conjointement. Chaque route et chaque parking sont représentés par un point dans l'espace latent. Le modèle sera alors capable de capturer les relations mutuelles entre stationnement et trafic et mènera à des performances de prédiction accrues. A notre connaissance, il s'agit du seul modèle pouvant prédire les deux phénomènes sur une même région géographique.

4.1.1 Contexte et Tâches

4.1.1.1 Notations

Nous utilisons les mêmes notations que pour le modèle RAINSTORRM à la différence qu'ici, nous ne considérons pas de valeurs manquantes dans les séries temporelles. La tâche de complétion n'est donc pas abordée. Nous n'utilisons donc pas de masque sur les valeurs manquantes. Nous introduisons une nouvelle notation qui correspond au type d'une série temporelle $m \in \{1, .., M\}$, M étant le nombre de types de séries temporelles existantes. Pour nos expériences, nous avons $M = 2$ car nous considérons la prédiction du trafic et de la disponibilité du stationnement. Nous rappelons que $x_i^{(t)} \in \mathbb{R}$ correspond à la valeur d'une série i au temps t.
Comme précédemment, additionnellement aux valeurs des séries, la proximité spatiale des capteurs qui produisent les séries temporelles est prise en compte. Nous notons ici $\mathbf{W} = \{w_{i,j}\}_{i,j \in [1;n]^2}$ la structure du graphe, telle que $w_{i,j} \in \mathbb{R}^+$ représente la proximité spatiale entre la série temporelle i et la série temporelle j.
Notre tâche est alors la prédiction pour les différents types de séries temporelles considérées. Le problème de la prédiction de séries hétérogènes est illustré dans la Figure 4.1 dans le cas particulier de la prédiction de trafic et de parking. Nous nous focaliserons dans la partie expérimentale sur de la prédiction à court terme [1].

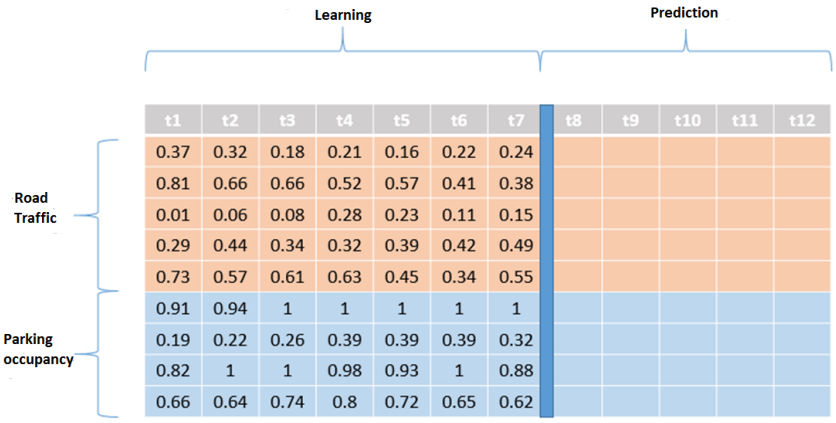

FIGURE 4.1 – Les tâches qui nous intéressent sont la prédiction des séries de trafic (cases roses) et celles qui correspondent à l'occupation des places de stationnement (cases bleues). Ces séries sont observées entre $t = 1$ et $t = T$, et doivent être prédites pour $t = T + 1$ à $t = T + \tau$.

1. Il s'agit ici de prédiction n'excédant pas quelques heures.

4.1.2 Modèle

Nous proposons une réécriture du modèle RAINSTORM avec une formulation hétérogène. L'idée est de projeter les séries temporelles des différents types dans le même espace latent dans lequel elles pourront s'influencer mutuellement. La différence par rapport au modèle précédent se fera dans la présence d'un décodeur spécifique pour chaque type de série temporelle. Le nouveau coût à optimiser s'écrit :

$$\begin{aligned}\mathscr{L}(\theta, \gamma, \mathbf{z}) = &\sum_{i=1}^{n}\sum_{m=1}^{M}\sum_{t=1}^{T}\Delta_m(d_\theta^m(z_i^{(t)}), x_i^{(t)}) \\ &+ \lambda_D \sum_{i=1}^{n}\sum_{t=1}^{T-1}||z_i^{(t+1)} - h_\gamma(z_i^{(t)})||^2 \\ &+ \lambda_{STC}\sum_{i,j}^{n}\sum_{t=1}^{T} w_{i,j}||z_i^{(t)} - z_j^{(t)}||^2\end{aligned} \tag{4.1.1}$$

Les notations λ_D et λ_{STC} correspondent respectivement à la valeur des hyperparamètres de la fonction dynamique et du terme de régularisation structurelle. Le premier terme change en revanche par rapport à l'écriture du modèle RAINSTORM. Ce terme apprend simultanément $\mathbf{z}$ et une fonction d_θ^m **pour chaque type de série**, tel que, à partir de $z_i^{(t)}$, d_θ^m puisse être utilisée pour prédire la valeur $x_i^{(t)}$. La fonction $d_\theta^m(z_i^{(t)})$ est définie pour que $d_\theta^m : \mathbb{R}^N \to \mathbb{R}$. Ceci permet de traiter les séries hétérogènes en décodant les représentations différemment selon leur type.

4.1.2.1 Apprentissage

L'apprentissage consiste en la minimisation de la fonction de coût $\mathscr{L}(\theta, \gamma, \mathbf{z})$ simultanément sur les paramètres θ, γ and $\mathbf{z}$. Nous illustrons la procédure d'apprentissage dans l'algorithme 3.
Les entrées de l'algorithme sont : les séquences observées, un pas d'apprentissage, les paramètres des fonctions dynamiques et de décodage initialisés aléatoirement et une représentation pour chaque séquence à chaque pas de temps (ligne 2-3). Puis, itérativement, les paramètres de la fonction de décodage sont modifiés dans la direction de leur gradient (ligne 10) tout comme les représentations correspondantes (ligne 9). Pour la fonction dynamique, les représentations et les paramètres de la fonction sont modifiés de la même manière (ligne 13-14). Enfin, les paramètres correspondants à la régularité structurelle sont mis à jour (ligne 17-18).

Algorithm 2 Stochastic Gradient Descent Algorithm

input :
  $\forall i,\ t,\ x_i^{(t)} \in \mathcal{X}$
  $\epsilon \leftarrow$ learning rate
procedure LEARNING($\theta,\ \gamma,\ \mathbf{z}$)
    $\forall i,\ t,\ z_i^{(t)} \leftarrow random, \gamma \leftarrow random, \theta \leftarrow random$
  for number of iteration **do**
    Sample $i,\ t,\ j$
    %Descente de gradient pour le terme (1)
    $z_i^{(t)} \leftarrow z_i^{(t)} - \epsilon \nabla_{z_i^{(t)}} \Delta_m(d_\theta^m(z_i^{(t)}),\ x_i^{(t)})$
    $\theta \leftarrow \theta - \epsilon \nabla_\theta \Delta_m(d_\theta^m(z_i^{(t)}),\ x_i^{(t)})$
    end
    %Descente de gradient pour le terme (2)
    $z_i^{(t)} \leftarrow z_i^{(t)} - \epsilon \lambda_{dyn} \nabla_{z_i^{(t)}} \Delta(h_\gamma(z_i^{(t)}),\ z_i^{(t+1)})$
    $z_i^{(t+1)} \leftarrow z_i^{(t+1)} - \epsilon \lambda_D \nabla_{z_i^{(t+1)}} \Delta(h_\gamma(z_i^{(t)}),\ z_i^{(t+1)})$
    $\gamma \leftarrow \gamma - \epsilon \lambda_D \nabla_\gamma \Delta(h_\gamma(z_i^{(t)}),\ z_i^{(t+1)})$
    %Descente de gradient pour le terme (3)
    $z_i^{(t)} \leftarrow z_i^{(t)} - \epsilon \lambda_{STC} \nabla_{z_i^{(t)}} ||z_i^{(t)} - z_j^{(t)}||^2 w_{i,j}$
    $z_j^{(t)} \leftarrow z_j^{(t)} - \epsilon \lambda_{STC} \nabla_{z_j^{(t)}} ||z_i^{(t)} - z_j^{(t)}||^2 w_{i,j}$
    end
end procedure

4.1.3 Expériences

4.1.3.1 Données

Nos expériences sont menées sur des données informant sur l'occupation des parkings et les conditions de trafic dans la ville de Lyon (France).

— Les données de trafic (voir Figure 4.2) sont fournies par des boucles magnétiques (capteurs statiques) placées dans la chaussée de certaines routes dans la ville. Nous gardons les 50 routes les plus empruntées dans nos expériences, ce qui résulte en 50 séries temporelles.

— Les données sur les parkings informent de leur occupation. Les 30 parkings les plus fréquentés sont gardés pour les expériences, ce qui mène à 30 nouvelles séries temporelles additionnelles.

Les données proviennent de `http://data.grandlyon.com/` et ont été collectées

pendant 15 jours et agrégées [2] en fenêtres de 20 minutes. Un total de 80 séries temporelles est utilisé dans ces expériences.

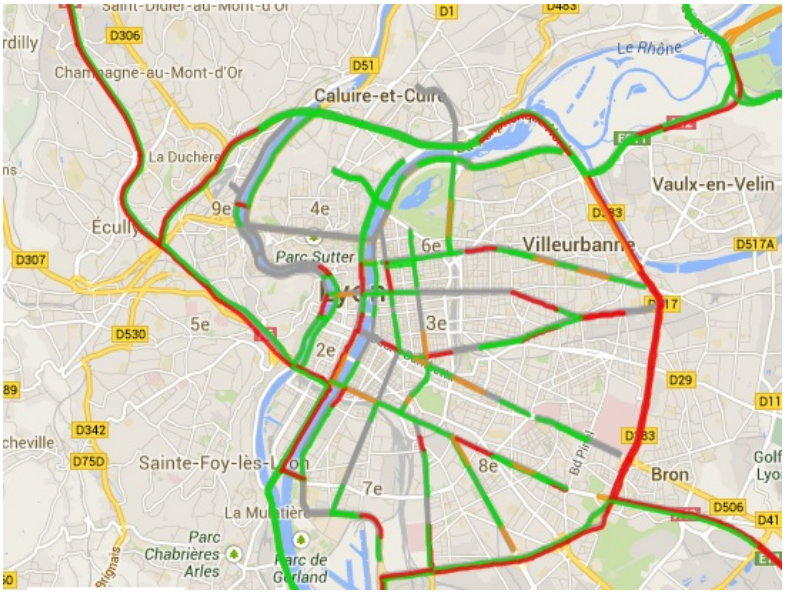

FIGURE 4.2 – Illustration de l'information de trafic à un instant donné pour une partie donnée de la ville. Les couleurs -rouge, orange et vert- sont utilisées pour illustrer les conditions de trafic, respectivement congestionné, dense et fluide. Le gris signifie que l'information était manquante à l'instant donné.

4.1.4 Méthodes Concurrentes

Nous comparons la version du modèle que nous décrivons dans ce chapitre et que nous notons HTSR (Heterogeneous Time-Series Representation) avec les modèles concurrents suivants :

— **MEAN** : Une heuristique qui prédit les futures valeurs comme la moyenne des valeurs observées sur la fenêtre d'entraînement de taille T.

— **VAR** : Un modèle "vectorial autoregressiv" où la valeur prédite d'une série au temps $t+1$ dépend des valeurs passées pour toutes les séries pendant un laps de temps donné et contrôlé par un hyper-paramètre dont la valeur est décidée

2. Le volume de trafic et le nombre de places libres de stationnement sont sommés par tranche de 20 minutes

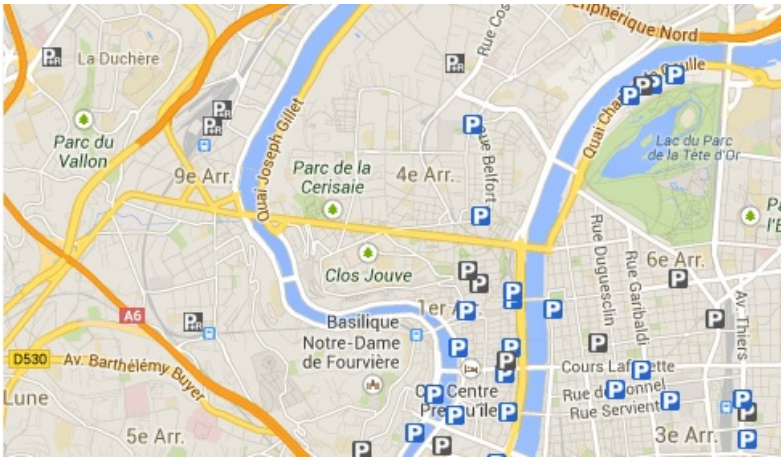

FIGURE 4.3 – Localisation des parkings utilisés pour une partie de la ville à un instant donné. Une couleur grise signifie que l'information est manquante pour la fenêtre temporelle illustrée.

par validation croisée. Ce modèle prédictif est linéaire.

— **NN** : Méthode basée sur une architecture de réseau de neurones classique, décrit par exemple dans [41].

— **ARIMA** : (Autoregressive Integrated Moving Average). L'équation de prédiction pour une série temporelle consiste en une combinaison linéaire des précédentes valeurs observées et des précédentes erreurs de prédiction. Le modèle ARIMA est largement utilisé en prédiction de trafic [122].

4.1.5 Résultats

Dans cette section, nous utilisons comme pour le modèle RAINSTORM des décodeurs linéaires d^m et un perceptron multi-couches avec 150 neurones sur la couche cachée pour h. Des évaluations quantitatives pour une série de tests sur les données décrites sont fournies avec différentes configurations d'expériences.
Nous comparons différents modèles dans une configuration homogène (voir Figure 4.4 et 4.6) , c'est-à-dire en prédiction de trafic ou de disponibilité de place de stationnement avec exclusivement des séries du même type en entraînement, tout comme une configuration hétérogène où le trafic et le stationnement sont prédits conjointement.
Une première observation est que le modèle HTSR qui prend en compte l'informa-

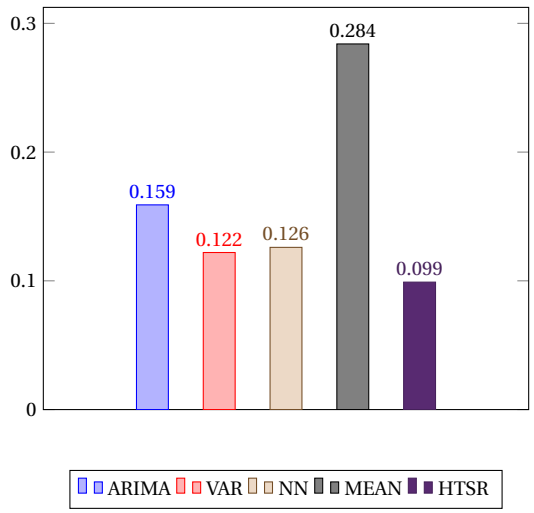

FIGURE 4.4 – RMSE pour la prédiction de trafic routier pour les différents modèles à $T+1$, en utilisant seulement l'information de trafic en entraînement (séries homogènes)

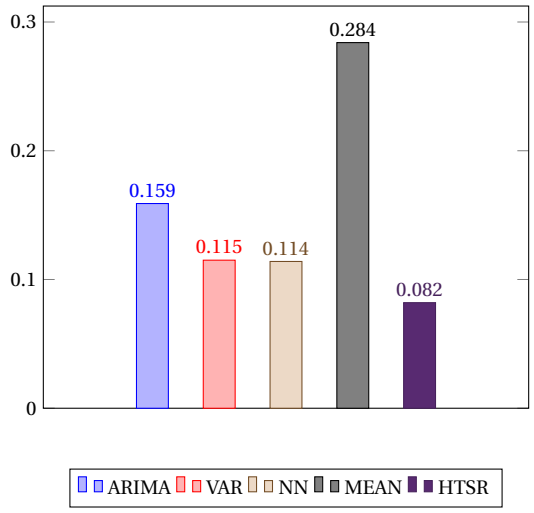

FIGURE 4.5 – RMSE pour la prédiction de trafic routier pour les différents modèles à $T+1$, en utilisant l'information hétérogène (trafic + parking)en entraînement

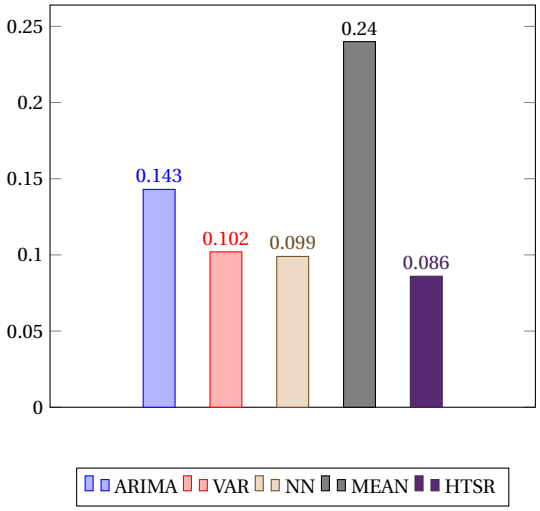

FIGURE 4.6 – RMSE pour la prédiction d'occupation de parking pour les différents modèles à $T + 1$, en utilisant seulement l'information de stationnement en entraînement (séries homogènes)

tion relationnelle surpasse significativement les autres modèles dans toutes les configurations expérimentales. Une autre observation importante est que les modèles de prédiction de séries temporelles multivariées bénéficient tous de l'information hétérogène; en effet, considérer la prédiction de trafic et de disponibilité de stationnement simultanément améliore les performances pour les deux tâches. Ceci s'explique par la forte intrication des deux phénomènes.
Nous donnons aussi les performances du modèle HTSR à différents horizons de prédiction $T + 1$, $T + 2$, ...$T + 15$ – Figure 5.3 pour les configurations hétérogènes et homogènes. Les résultats montrent clairement que considérer les séries hétérogènes est plus intéressant que l'approche homogène classique à tous les horizons de prédiction et permet donc de mieux capturer les dépendances long-terme.

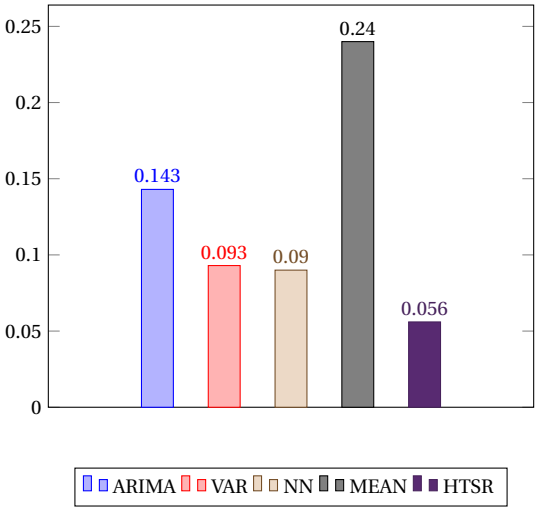

FIGURE 4.7 – RMSE pour la prédiction d'occupation de parking pour les différents modèles à $T+1$, en utilisant l'information hétérogène (trafic + parking)en entraînement

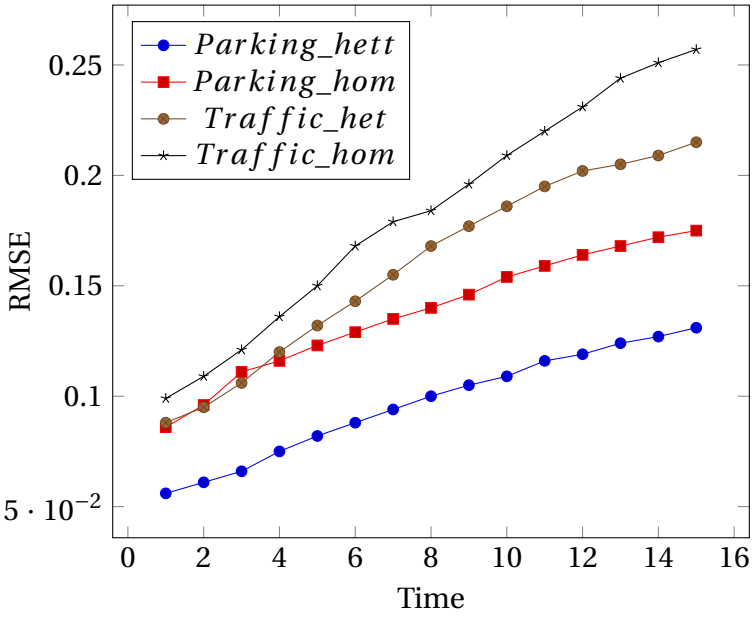

FIGURE 4.8 – Évolution de l'erreur en termes de RMSE pour les deux tâches avec le modèle HTSR en prédisant les 15 prochains pas de temps pour la variante homogène du problème (seulement le trafic ou seulement le parking en entraînement) et la version hétérogène (trafic et parking disponible à l'entraînement).

4.1.6 Conclusion

Nous avons dans ce chapitre proposé une nouvelle façon d'apprendre à prédire des séries temporelles multivariées de types différents. Pour cela, nous avons étendu le modèle RAINSTORM avec plusieurs décodeurs, un pour chaque type de séries. Dans nos expériences, les séries sont de deux types : trafic routier et occupation de parking. Notre approche obtient de bonnes performances de prédiction et plus généralement, les expériences illustrent l'intérêt de considérer ces deux phénomènes liés simultanément plutôt qu'indépendamment. Une perspective que ce travail ouvre serait par exemple de traiter d'autres phénomènes temporels liés au transport conjointement aux deux présentés ici.

4.2 Utilisation de Représentations Gaussiennes

4.2.1 Principes

Nous explorons une écriture différente du modèle RAINSTORM où la nature des représentations varie. Les composantes relationnelles et temporelles des séries sont ici modélisées par des distributions gaussiennes dans l'espace latent et non plus par des représentations déterministes. L'utilisation de ces représentations qui sont stochastiques nous permet de modéliser naturellement l'incertitude et le bruit inhérent aux observations et de prédire les futures valeurs des séries temporelles avec un taux de confiance dans la prédiction.
Ce modèle s'écrira toujours comme la minimisation d'une fonction de coût à trois termes mais le paradigme change en passant de représentations vectorielles à des représentations probabilistes.

4.2.2 Notations et Tâches

Nous considérons qu'à chaque série correspond une trajectoire dans l'espace latent. Nous gardons les mêmes notations que précédemment, à la différence que les représentations ne sont plus déterministes :
Chaque série $x_i^{(1)}$, ..., $x_i^{(T)}$ est alors associée à une série de variables aléatoires dans $\mathbb{R}^N$ dénotées $Z_i^{(1)}$, ..., $Z_i^{(T)}$ (avec N la taille de l'espace). Nous modélisons désormais chaque $Z_i^{(t)}$ comme une loi normale multidimensionnelle $\mathcal{N}(\mu_i^{(t)}, \Sigma_i^{(t)})$. Les observations peuvent être calculées à partir de ces distributions en utilisant une fonction de décodage d qui associe $Z_i^{(t)}$ à $X_i^{(t)} = d(Z_i^{(t)})$. Comme précédemment, nous supposons que la dynamique des séries est capturée par une fonction h telle que $Z_i^{(t+1)} = h(Z_i^{(t)})$. Ces différents mécanismes sont illustrés sur la Figure 4.9.

Maintenant, nous expliquons comment les distributions correspondant aux variables aléatoires $Z_i^{(t)}$ sont apprises conjointement aux fonctions d et h.

4.2.3 Définition du modèle

Nous supposons que les variables aléatoires $Z_i^{(t)}$ suivent une distribution $Z_i^{(t)} \sim \mathcal{N}(\mu_i^{(t)}, \Sigma_i^{(t)})$ où $\mu_i^{(t)}$ et $\Sigma_i^{(t)}$ doivent être estimés à partir des observations connues. Par simplicité et pour limiter la complexité spatiale du modèle, nous nous restreignons dans la suite au cas où $\Sigma_i^{(t)}$ est une matrice diagonale, avec $\sigma_{i,j}^{(t)}$ qui dénote la $j$ème valeurs de la diagonale de $\Sigma_i^{(t)}$. Cette hypothèse de diagonalité demande une simple contrainte sur les $Z_i^{(t)}$ durant la phase d'entraînement pendant laquelle les représentations $Z_i^{(t)}$ sont apprises. Le modèle reste très flexible.

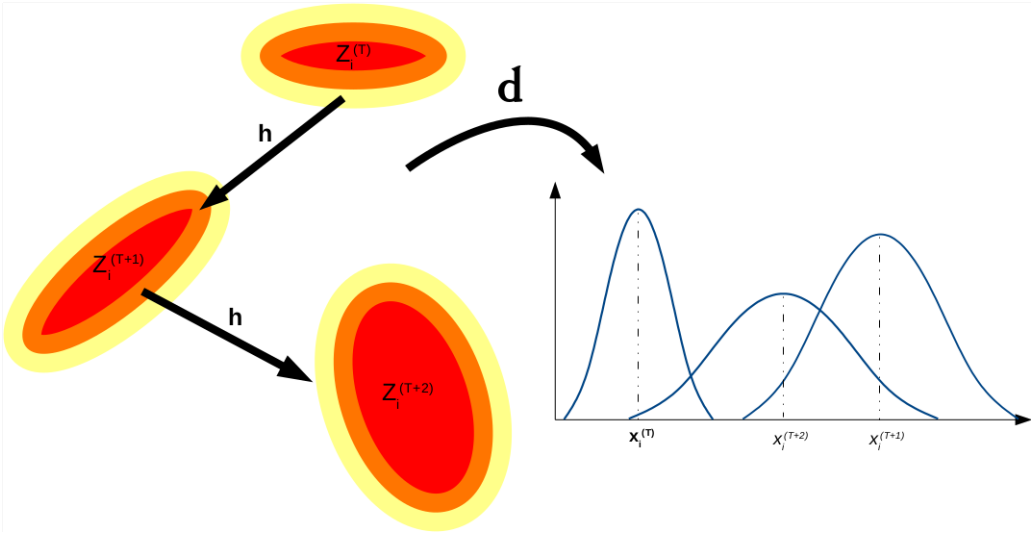

FIGURE 4.9 – La représentation gaussienne $Z_i^{(t)}$, la fonction dynamique h et la fonction de décodage d sont apprises. On calcule $Z_i^{(t+1)}$ et $Z_i^{(t+2)}$ en utilisant h et $Z_i^{(t)}$. $X_i^{(t+1)}$ et $X_i^{(t+2)}$ sont obtenus à partir de $Z_i^{(t+1)}$ et $Z_i^{(t+2)}$ respectivement en utilisant d.

Nous définissons une fonction de coût globale $\mathscr{L}(\mu, \Sigma, f, h)$ où μ et Σ dénotent les moyennes et les matrices de covariance pour toutes les séries et à tous les pas de temps entre 1 et T. Cette fonction s'écrit comme la somme de trois termes :

$$\begin{aligned}\mathscr{L}(\mu, \Sigma, f, h) = \sum_{i=1}^{n}\sum_{t=1}^{T} \Delta_{\mathrm{De}}(d(Z_i^{(t)}), x_i^{(t)}) \\ + \lambda_{\mathrm{Dy}} \sum_{i=1}^{n}\sum_{t=1}^{T-1} \Delta_{\mathrm{Dy}}(Z_i^{(t+1)}, h(Z_i^{(t)})) \\ + \lambda_{\mathrm{R}} \sum_{j=1}^{n}\sum_{t=1}^{T} e_{i,j}\Delta_{\mathrm{R}}(Z_i^{(t)}, Z_j^{(t)}) \quad (4.2.1)\end{aligned}$$

où λ_{Dy} et λ_{R} sont des hyper-paramètres pondérant l'importance des différents éléments de la fonction de coût.
Le premier terme correspond au terme de décodage et force la fonction d et les distributions des variables Z apprises à "expliquer" les observations. Le second terme, la composante dynamique, encourage h à modéliser les dynamiques temporelles dans l'espace latent tandis que le troisième terme capture les relations entre les séries. Dans la suite, nous utilisons pour d une fonction linéaire et pour h une version linéaire et une non linéaire. La flexibilité du modèle vient essentiellement de sa capa-

cité d'apprendre les représentations Z et utiliser un décodeur non-linéaire n'améliorerait pas forcément son expressivité.

4.2.3.1 Apprentissage

L'apprentissage du modèle se fait en minimisant la fonction de coût $\mathscr{L}(\mu, \Sigma, f, h)$ par rapport à μ, Σ, d et h. Pour simplifier les notations, les paramètres de d et h ne sont pas rendus explicites dans les notations – d et h sont seulement supposés être des fonctions dérivables. À la fin du processus d'apprentissage, toutes les distributions latentes sont connues pour les données d'apprentissage tout comme les fonction h et d. Nous utilisons *ADAM* [67] comme algorithme de descente de gradient. Nous avons testé différents variantes du modèle. Nous donnons d'abord la version permettant d'obtenir les meilleurs résultats. D'autres variantes sont décrites à la fin de cette section.

4.2.3.2 Modélisation du Décodeur

Pour mesurer l'erreur entre la distribution prédite par le modèle $d(Z_i^{(t)})$ et l'observation $x_i^{(t)}$, nous utilisons comme fonction de coût Δ_{De_1} l'espérance de l'erreur quadratique moyenne entre les prédictions et les observations :

$$\Delta_{\text{De}_1}(d(Z_i^{(t)}), x_i^{(t)}) \stackrel{\text{def}}{=} E\left[(d(Z_i^{(t)}) - x_i^{(t)})^2\right] \tag{4.2.2}$$

Où d est une fonction linéaire. Ce coût peut être écrit explicitement ainsi :

$$\Delta_{\text{De}_1}(d(Z_i^{(t)}), x_i^{(t)}) = \sum_{k=1}^{d} \theta_k^2 \sigma_{i,k}^{(t)} + \left(< \theta, \mu_i^{(t)} > - x_i^{(t)}\right)^2 \tag{4.2.3}$$

Minimiser Δ_{De_1} met à jour et la moyenne et la variance de la distribution. Plus spécifiquement, une valeur observée x_i avec Δ_{De_1} fera baisser les variances $\sigma_{i,k}^{(t)}$. C'est une propriété intéressante car les valeurs observées doivent avoir des représentations avec de faibles variances.

4.2.3.3 Modélisation de la Dynamique

La fonction h associe une distribution $\mathcal{N}(\mu_i^{(t)}, \Sigma_i^{(t)})$ à $\mathcal{N}(\mu_i^{(t+1)}, \Sigma_i^{(t+1)})$. En se basant sur [114, 40], nous utilisons une divergence de Kullback-Leibler (notée $D_{KL}(\cdot||\cdot)$) pour comparer la distribution de $Z_i^{(t+1)}$ à la distribution prédite par h, $d(Z_i^{(t)})$. Nous faisons l'hypothèse que la distribution résultante (pour $Z_i^{(t+1)}$) est Gaussienne. La divergence de Kullback Leibler entre deux distributions gaussiennes a une forme analytique simple à partir de laquelle le gradient peut être facilement calculé[114].

Comme pour le modèle RAINSTORM, nous considérons que la représentation latente au temps $(t+1)$ est une transformation non linéaire de celle au temps t. En

pratique, nous utilisons deux perceptrons multi-couche, h^m and h^c ; h^m prédit la moyenne et h^c la variance : plus précisément, à $t+1$ la moyenne est donnée par $\mu_i^{(t+1)} = h^m(\mu_i^{(t)}, \Sigma_i^{(t)})$, et la variance par $\Sigma_i^{(t+1)} = h^c(\mu_i^{(t)}, \Sigma_i^{(t)})$. Ceci donne :

$$\Delta_{\mathrm{Dy}_1}(Z_i^{(t+1)}, h(Z_i^{(t)})) \stackrel{\text{def}}{=} D_{KL}(Z_i^{(t+1)} || \mathcal{N}(h^m(\mu_i^{(t)}, \Sigma_i^{(t)}), h^c(\mu_i^{(t)}, \Sigma_i^{(t)}))) \tag{4.2.4}$$

4.2.3.4 Terme de Régularisation Structurelle

Enfin,le terme Δ_{R} de régularisation structurelle est défini désormais comme :

$$\Delta_{\mathrm{R}}(Z_i^{(t)} || Z_j^{(t)}) = D_{KL}(Z_i^{(t)} || Z_j^{(t)}) \tag{4.2.5}$$

qui a encore, en tant que variable aléatoire gaussienne, une forme analytique simple qui peut être utilisée pendant l'apprentissage [3]. La divergence de Kullback-Leibler est ici également un choix naturel et cette mesure de dissimilarité entre distributions de probabilités a des propriétés intéressantes pour la modélisation des relations :

— Elle est non négative

— Elle est convexe et s'optimise donc facilement

— Elle est non symétrique

En particulier, cette dernière propriété permet de modéliser plus finement les relations entre les différentes représentations. Dans le cas du modèle RAINSTORM, la proximité spatiale (distance euclidienne) était utilisée pour contraindre les représentations de séries connectées dans le graphe de relations.

4.2.4 Variantes

Nous avons expérimenté d'autres variantes du modèle en considérant d'autres fonctions de coût et d'autres fonctions dynamiques h. Pour les fonctions de coût, nous avons testé une différence simple entre l'espérance de d et l'observation en utilisant une erreur quadratique moyenne :

$$\Delta_{\mathrm{De}_2}(d(Z_i^{(t)}), x_i^{(t)}) \stackrel{\text{def}}{=} \left(E\left[d(Z_i^{(t)})\right] - x_i^{(t)}\right)^2 \tag{4.2.6}$$

En considérant une fonction de décodage linéaire telle que $d(\cdot) = < \theta, \cdot >$, θ étant l'ensemble des paramètres de d, Δ_{De_2} peut être réécrit comme :

$$\Delta_{\mathrm{De}_2}(d(Z_i^{(t)}), x_i^{(t)}) = (< \theta, \mu_i^{(t)} > - x_i^{(t)})^2 \tag{4.2.7}$$

3. $\frac{1}{2}(\mathrm{tr}({\Sigma_j^{(t)}}^{-1}\Sigma_i^{(t)}) + (\mu_j^{(t)} - \mu_i^{(t)})^T {\Sigma_j^{(t)}}^{-1}(\mu_j^{(t)} - \mu_i^{(t)}) - d - \log(\frac{|\Sigma_i^{(t)}|}{|\Sigma_j^{(t)}|}))$

Cette fonction de coût modélise uniquement la moyenne de la distribution tandis que celui défini précédemment prend en compte également la variance.

Pour h, nous avons également considéré une fonction linéaire au lieu d'une fonction non linéaire. On considère alors que la représentation latente au temps $(t+1)$ est une transformation linéaire de la représentation latente au temps t. La variable transformée est alors également gaussienne et ses paramètres se calculent simplement. h est une fonction de $\mathbb{R}^d$ dans $\mathbb{R}^d$ qui est représentée par une matrice $\gamma \in \mathcal{M}_{d,d}(\mathbb{R})$:

$$\Delta_{\mathrm{Dy}_2}(Z_i^{(t+1)}, h(Z_i^{(t)})) \stackrel{\text{def}}{=} D_{KL}(Z_i^{(t+1)} || \gamma Z_i^{(t)}) = D_{KL}(Z_i^{(t+1)} || \mathcal{N}(\gamma \mu_i^{(t)}, \gamma \Sigma_i^{(t)} \gamma^T)) \quad (4.2.8)$$

4.2.5 Expériences

4.2.5.1 Jeux de Données

Nous menons des expériences sur quatre jeux de données, extraits respectivement de Google Flu Trends [4], WHO [5] et deux jeux de données issus du Grand Lyon [6] (GL) (les jeux de données sur le trafic et l'occupation des parkings à Lyon). Toutes les séries sont normalisées. Pour tous les jeux de données, les relations utilisées sont binaires indiquant si deux séries sont reliées ou non.

— Le dataset **Google Flu Trend** (GFT) est composé du nombre de requêtes en rapport avec la grippe agrégées chaque semaine dans 29 pays. Les relations entre séries sont définies selon la présence ou non d'une frontière entre deux pays. Il y a 96 relations.
— Le dataset **GL Traffic** (GL-T) correspond au jeu de données sur le trafic à Lyon déjà décrit dans la section précédente. Les relations binaires sont basées sur la proximité des routes. Nous considérons 130 relations.
— Le dataset **GL Park** (GL-P) correspond au jeu de données sur l'occupation des parkings à Lyon. Il y a 74 relations définies.
— Le jeu de données **WHO** donne le nombre de décés dus à la diphtérie au fil du temps dans 91 pays, ce qui donne 91 séries temporelles. Les relations entre séries sont définies selon la présence ou non d'une frontière entre deux pays. Il y a 228 relations.

4. http ://www.google.org/flutrends
5. http ://www.who.int
6. http ://data.grandlyon.com

4.2.5.2 Baselines

Nous comparons notre approche aux cinq baselines suivantes :
— (**AR**), un modèle autorégressif linéaire.
— Feed Forward Neural Network (**FFNN**), qui correspond à un modèle autorégressif non linéaire.
— **RNN**, un réseau de neurones récurrent avec une couche cachée et la fonction $tanh$ en fonction d'activation non linéaire.
— **KF** [62], est un filtre de Kalman classique avec des transformations linéaires d'un état à un autre.
— **DFG** [83], un modèle de l'état de l'art qui apprend des variables latentes déterministes en modélisant la dynamique et la probabilité jointe entre les séries.

Les différentes versions de notre modèle sont notées $\mathbf{RDG}_{i,j}$ avec $i, j \in \{1, 2\}$. $\mathbf{RDG}_{1,1}$ corresponds alors au modèle principal introduit dans la section 4.2.3, $\mathbf{RDG}_{1,2}$ au modèle avec un coût de décodage Δ_{De_1} et une fonction h linéaire, i.e. un coût dynamique Δ_{Dy_2}.

4.2.5.3 Protocole Expérimental

Les hyper-paramètres des différents modèles sont sélectionnés en utilisant une procédure de validation croisée pour séries temporelles appelée "validation avec origine glissante" également utilisée par exemple dans [14, 44].
Ce protocole considère une fenêtre glissante de taille T. La valeur de T est fixée pour être assez grande pour capturer les dynamiques principales de chaque série. Chaque série est normalisée de telle sorte que ses valeurs soient comprises entre zéro et un.
Comme métrique d'évaluation, nous considérons la racine d'une erreur quadratique (RMSE). Les modèles sont évalués en prédiction à $t + 1$.

4.2.5.4 Résultats

Nous présentons d'abord les performances de notre modèle par rapport aux baselines à l'horizon 1 dans la Figure 4.10. Nous avons testé les quatre variantes de notre approche, i.e. les combinaisons de Δ_{De_1} ou Δ_{De_2} avec Δ_{Dy_1} ou Δ_{Dy_2}.
Notre méthode obtient les meilleures performances sur tous les jeux de données excepté GFT où KF est plus efficace. Notre approche surpasse les autres sur deux datasets (GL-P -Grand Lyon Parks- et GFT -Google Flu Trends- sur le tableau) et obtient des résultats similaires à RNN sur les deux autres (GL-T -Grand Lyon Traffic- et WHO). $\mathbf{RDG}_{1,1}$ obtient souvent de meilleurs résultats, i.e. la meilleure combinaison est l'espérance d'une erreur quadratique pour le décodeur et un modèle non linéaire pour la fonction dynamique.

La Figure 4.11 montre la qualité de la prédiction (en termes de RMSE) à $(T + 1)$, $(T + 2)$, $(T + 3)$, $(T + 4)$ et $(T + 5)$ et illustre l'habilité de RDG à prédire correctement

FIGURE 4.10 – Comparaison (en termes de RMSE) à T+1 sur les quatre jeux de données pour les baselines et notre modèle sur la tâche de prédiction. $\text{RDG}_{k,l}$ correspond à la variante avec les coûts $(\Delta_{\text{De}_k}, \Delta_{\text{Dy}_l})$.

Model	GL-T	GL-P	GFT	WHO
AR	0.0752	0.0892	0.0626	0.0832
FFNN	0.0751	0.0894	0.045	0.0838
RNN	0.0709	0.0890	0.0431	**0.0795**
KF	0.0711	0.0833	**0.0388**	0.0799
DFG	0.0712	0.0911	0.0592	**0.0795**
RDG$_{2,2}$	0.0742	0.0902	0.0607	0.0848
RDG$_{2,1}$	**0.0707**	0.0834	0.0434	0.0796
RDG$_{1,2}$	0.0765	0.0896	0.0589	0.0831
RDG$_{1,1}$	0.0718	**0.0828**	0.0429	**0.0795**

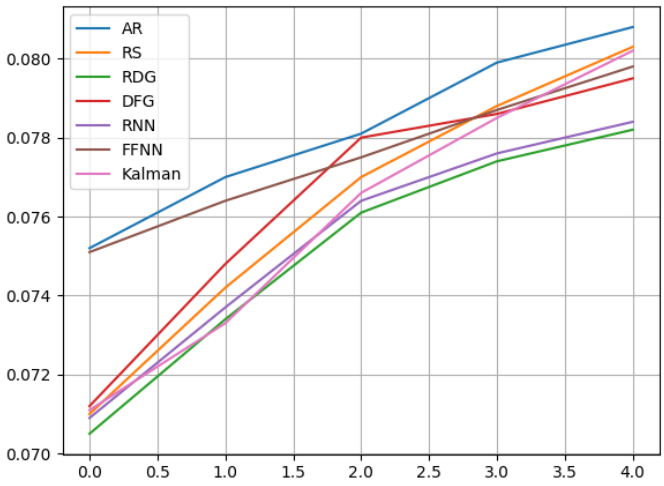

FIGURE 4.11 – Comparaison en termes de RMSE de T+1 à T+5 sur le dataset GL-T entre les baselines et notre modèle (RDG) pour la tâche de prédiction

à différents horizons.
On peut remarquer qu'à $(T + 5)$, KF obtient des performances dégradées par rapport aux autres méthodes de l'état de l'art.

RDG a la propriété additionnelle de modéliser naturellement l'incertitude associée à ses prédiction, ce qui n'est pas le cas pour RNN par exemple. On considère les courbes présentées en Figure 4.12. Elles illustrent les prédictions faites par notre modèle ainsi que les variances associées calculées à partir des représentations gaussiennes. Tout d'abord, on peut remarquer que les valeurs qui correspondent à la vérité terrain sont toujours dans l'intervalle de confiance fourni par notre modèle, ce qui signifie que RDG calcule des minimums et des maximums possibles pertinents. Une autre remarque est que la taille de l'intervalle augmente avec l'horizon de prédiction, ce qui est attendu pour ce genre de modèle.

4.2.6 Régularisation structurelle et incertitude

Nous comparons dans le Tableau 4.1 les résultats entre notre modèle en prenant en compte le graphe de voisinage ($\lambda_R \neq 0$) ou pas ($\lambda_R = 0$) :
Les résultats en prédiction sont uniformément moins bons pour tous les jeux de données lorsque les relations ne sont pas prises en compte ce qui suggère que cette régularisation améliore le modèle (quand la structure donnée a priori est pertinente). De plus, nous donnons les résultats obtenus sans incertitude, ce qui correspond au modèle déterministe RAINSTORM décrit dans le chapitre précédent : ici également, cette version du modèle est meilleure, ce qui montre l'intérêt de considérer des représentations gaussiennes.

TABLE 4.1 – RMSE à $T + 1$ sur les quatre datasets

Model	GL-T	GL-P	GFT	WHO
Rainstorm	0.0710	0.0886	0.0440	0.0804
RDG ($\lambda_R = 0$)	0.0719	0.900	0.0441	0.0807
RDG	**0.0707**	**0.0828**	**0.0388**	**0.0795**

4.3 Conclusion

Nous avons proposé un nouveau modèle pour prédire des séries temporelles relationnelles. Notre modèle (RDG) est basé sur des représentations latentes gaussiennes, et montre des performances compétitives par rapport à d'autres modèles de l'état de

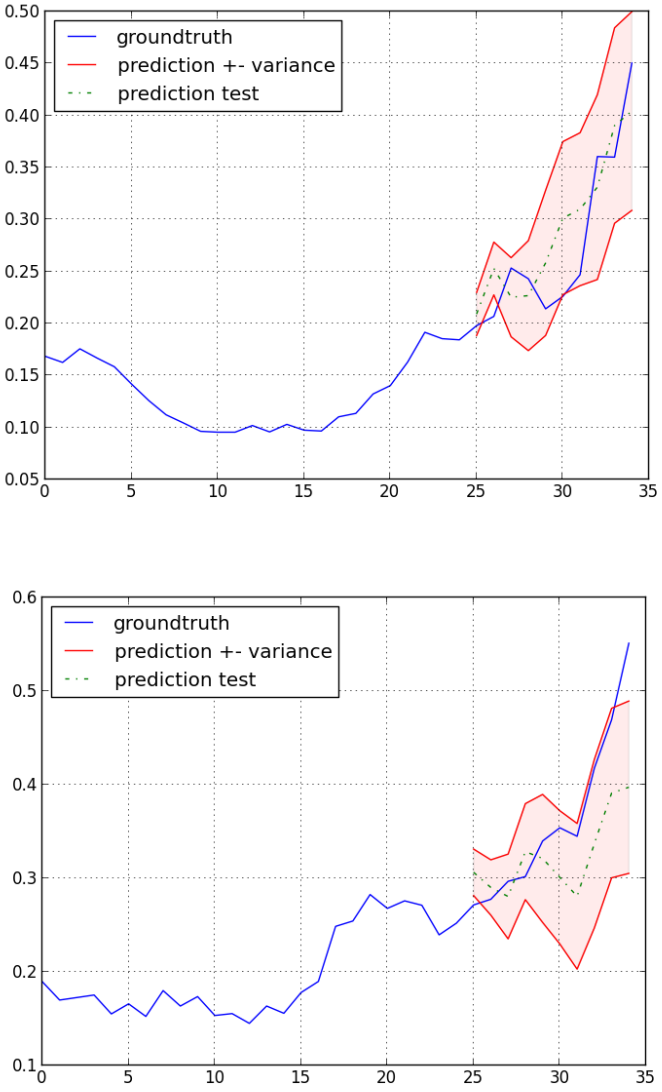

FIGURE 4.12 – Prédictions sur GFT (deux séries temporelles différentes pour le jeu de données) avec le modèle $\text{RDG}_{2,2}$ qui montrent les intervalles de confiance obtenables : $E(d(Z^{(t)})) \pm \text{var}(d(Z^{(t)}))$. La prédiction à $25 + n$ correspond à $d(h^n(Z^{(25)})$.

l'art. De plus, RDG permet de modéliser simplement l'incertitude des prédictions, fournissant par exemple des intervalles de confiance pour chaque prédiction.

CHAPITRE 5

STNN : RÉSEAUX DE NEURONES SPATIO-TEMPORELS

Résumé

Le modèle RAINSTORM présenté dans le chapitre précédent capture un certain type de corrélation entre séries sous la forme d'une régularité entre les représentations latentes de séries voisines et semble particulièrement adapté pour les cas où deux séries proches évoluent "conjointement". Notre deuxième contribution a été de proposer un modèle plus général permettant de capturer des corrélations plus complexes, notamment des corrélations spatio-temporelles très présentes dans les applications visées. Le modèle, que l'on propose et que l'on décline sous plusieurs variantes qui correspondent à différents niveaux d'a priori sur les relations entre séries temporelles, est testé sur plusieurs jeux de données et non seulement ses performances en prédiction sont très satisfaisantes, mais ce modèle est également capable de découvrir des relations spatiales entre plusieurs séries temporelles.

5.1 Introduction

Le modèle RAINSTORM vu dans un chapitre précédent montre l'utilité de prendre en compte les relations connues a priori dans les modèles de représentations. Cependant, le terme de régularisation structurelle impose une contrainte également forte sur toutes les paires de séries reliées. Cette limitation du modèle peut le rendre inadapté à des séries temporelles liées géographiquement représentant par exemple des phénomènes physiques.

Ces séries temporelles qui exhibent des dépendances spatio-temporelles sont présentes dans de nombreux domaines qui incluent l'écologie, la météorologie, la biologie, la médecine, l'économie, le transport et la vision. Une des difficultés est de modéliser correctement la dynamique sous-jacente aux phénomènes spatio-temporels et une approche naturelle est de considérer des modèles dynamiques à variables latentes. Ce type d'approche a été exploité et dans le domaine des statistiques [34] et en apprentissage automatique[9, 69].

Récemment en Deep learning, une séries de modèles ont été développés aboutissant à des représentations des données intéressantes et permettant d'obtenir de bonnes performances pour de nombreuses tâches. Pour les données dynamiques, plusieurs modèles ont récemment rencontré un certain succès dans le traitement de séquences complexes ou l'analyse de séries pour des tâches comme la classification, la prédiction, la génération et beaucoup d'autres [16, 31, 77]. Ces modèles e.g. les réseaux de neurones récurrents (RNN) se focalisent sur l'apprentissage des caractéristiques dynamiques des séquences étudiées, mais la dimension spatiale, essentielle dans de nombreuses applications, a été assez peu étudiée en deep learning. Nous pensons que cette famille de modèle a un fort potentiel pour capturer les dy-

namiques complexes de phénomènes spatio-temporels. Nous nous focalisons sur le problème de la prédiction de séries spatio-temporelles mais ce modèle peut facilement être étendu pour traiter d'autres tâches comme la prédiction spatiale (ou "kriging" [109]) ou la complétion de données manquantes.

Nous proposons dans ce chapitre un nouveau modèle dynamique d'apprentissage de représentation - Spatio-Temporal Neural Network (STNN) - qui a deux composantes : l'une pour apprendre des trajectoires latentes et apprendre la dynamique spatio-temporelle des phénomènes étudiés et l'autre est un décodeur pour prédire les futures observations des séries temporelles modélisées dans l'espace latent du système. Contrairement au modèle RAINSTORM, les relations sont directement prises en compte par le terme dynamique du modèle; ceci permet à cette approche une plus grande flexibilité quant à l'usage de ces relations qui peuvent avoir des importances très variables tandis que dans notre première contribution, elles correspondaient à une égale régularisation des représentations latentes pour toutes les relations.
Ce nouveau modèle est testé et comparé aux méthodes de l'état de l'art incluant des récents réseaux de neurones récurrents sur une série de problèmes impliquant des données spatio-temporelles : la prédiction de propagation de maladies, la prédiction de trafic, la météorologie et l'océanographie.
Ce chapitre est organisé comme suit : le modèle est présenté dans la Section 5.2 avec différentes variantes. Les expériences sont menées et décrites dans la Section 5.3.

5.2 Modèle

5.2.1 Notations et Tâche

Nous considérons à nouveau un ensemble de n séquences temporelles. Nous ne nous intéressons plus ici aux valeurs manquantes dans les séries temporelles; aussi, nous notons désormais m la dimensionnalité de chaque série et T leur longueur (on suppose que toutes les séries ont la même longueur et la même dimensionnalité. C'est usuellement le cas dans les problèmes de prédictions spatio-temporelles mais cette restriction peut être facilement ignorée). Ici, $m = 1$ signifie que nous considérons n séries temporelles monovariées tandis que $m > 1$ correspondra à n séries multivariées avec chacune m composantes. Par commodité, nous introduisons la nouvelle écriture tensorielle suivante : soit X les valeurs de toutes les séries entre le temps 1 et le temps T. X ainsi correspond à un tenseur $\mathbb{R}^{T\times n\times m}$ tri-dimensionnel, tel que $X_{t,i,j}$ est la valeur de la j-ème composante de la série i au temps t. X_t dénotera la "tranche" de X au temps t telle que $X_t \in \mathbb{R}^{n\times m}$ est la valeur de toutes les séries au temps t.
L'organisation spatiale des sources n'est plus capturée par un graphe $\mathcal{G}$ de relations

binaires mais à travers une matrice $W \in \mathbb{R}^{n \times n}$. Idéalement, W devrait indiquer l'influence mutuelle entre les sources; cette information est rarement disponible. En pratique, il peut s'agir de proximité spatiale ou une similarité entre les sources : pour les problèmes géospatiaux, cela peut correspondre à l'inverse d'une distance physique - e.g. distance géodésique- entre les sources. Pour d'autres applications, cela peut venir de connexions locales entre les sources qui utilisent une structure de graphe (e.g. une matrice d'adjacence pour des routes connectées dans une application de prédiction de trafic ou un noyau de graphe pour une application donnée sur le web). Considérons le problème de la prédiction de séries temporelles défini précédemment : pour rappel, il s'agit de prédire le futur de séries temporelles connaissant leur historique. Notre but est d'apprendre un modèle $f : \mathbb{R}^{T \times n \times m} \times \mathbb{R}^{n \times n} \rightarrow \mathbb{R}^{\tau \times n \times m}$ capable de prédire les futures valeurs des séquences à τ pas de temps en se basant sur X et sur les dépendances spatiales connues entre les séries. Dans un second temps, nous montrerons également comment ce modèle entraîné pour la prédiction peut aussi être utilisé pour extraire des corrélations entre séries.

5.2.2 Modélisation de séries temporelles avec des représentations latentes continues

On présente dans un premier temps les idées qui structurent le modèle proposé dans le cas simple de la prédiction de séries temporelles multivariées sans considération de relations spatiales entre les séries. Le modèle a deux composantes. La première est pour la dynamique du processus et est exprimée dans l'espace latent. Soit Z_t la représentation latente d'une série au temps t, le modèle dynamique s'écrit $Z_{t+1} = g(Z_t)$. Le second composant est un décodeur qui associe une représentation latente Z_t avec une prédiction qui correspond à la valeur réelle de la série considérée au temps t, i.e. $d(Z_t) = \tilde{X}_t$, $\tilde{X}_t$ étant la prédiction calculée au temps t. Dans ce modèle et de façon similaire à celui proposé dans la première partie de cette thèse, à la fois les représentations Z_t et les paramètres des composantes dynamiques et du décodeurs devront être appris. Cette définition est formellement similaire à celle proposée dans [83].
Ce modèle diffère des formulations classiques pour les réseaux de neurones récurrents [53, 28]. La composante de l'état d'un réseau de neurone récurrent (dont l'état caché possède une boucle vers lui-même) s'écrit $Z_{t+1} = g(Z_t, X'_t)$, avec X'_t étant la valeur réelle à prédire durant l'entraînement, et X'_t étant la valeur prédite $\tilde{X}_t$ durant l'inférence. Dans notre approche, les représentations Z_t sont directement apprises durant l'entraînement : elles ne prennent pas d'entrées, les dynamiques sont capturées entièrement dans l'espace latent et les données $\tilde{X}_t$ sont générées par la variable Z_t. Ceci est similaire dans l'esprit à des modèles de Markov cachés ou à des filtres de Kalman. Pour le problème de la prédiction de séries temporelles, ce modèle est plus

flexible que des réseaux de neurones récurrents usuels et n'a pas besoin de prendre des entrées comme dans les réseaux de neurones classiques. Un argument similaire est utilisé dans [83].
Une autre différence significative est que d'habitude, dans un RNN (réseaux de neurones récurrent) il y a un unique vecteur (ou état caché) $Z_t \in \mathbb{R}^N$, N étant la dimension de l'espace latent, pour modéliser une séquence temporelle (voir Figure 5.1). Dans notre approche, comme nous considérons plusieurs sources, il y a un vecteur latent associé à chaque source. La composante dynamique de notre modèle est alors entraînée pour capturer les dépendances temporelles et spatiales entre les représentations latentes des différentes sources. Cet aspect est manquant dans les RNNs.

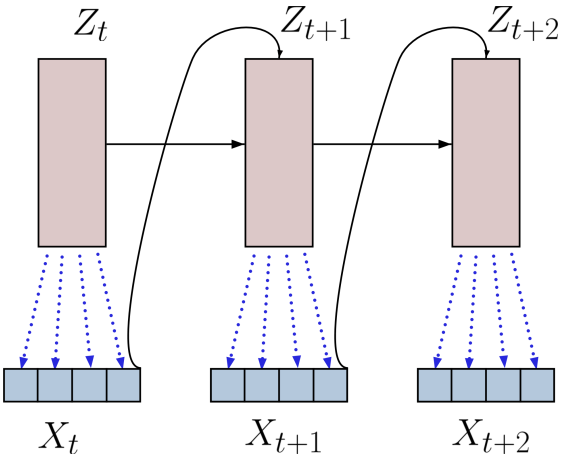

FIGURE 5.1 – Architecture d'un réseau de neurones récurrent classique

.

5.2.2.1 Formulation avec Contraintes Dures

On formule tout d'abord le problème d'apprentissage comme un problème d'optimisation.

— **Problème d'apprentissage** : dans cette formulation, le problème consiste à trouver d^*, g^* and Z^* tels que :

$$\begin{aligned} d^*, g^*, Z^* &= \arg\min_{d,g,Z} \frac{1}{T} \sum_t \Delta(d(Z_t), X_t) \\ &\text{sous contrainte } \forall t \in [1;\ T-1],\ Z_{t+1} = g(Z_t) \end{aligned} \tag{5.2.1}$$

où Δ est une fonction de coût mesurant l'écart entre la prédiction $d(Z_t)$ et la vérité observée X_t.

Ici, la dynamique est traitée à travers l'ensemble des contraintes sur les représentations latentes, $Z_{t+1} = g(Z_t)$ pour tout t. Les valeurs Z_t pour $t > 1$ sont entièrement déterminées par g, d et Z_1.

— **Inférence** : Le problème de prédiction qui nous intéresse vise à prédire le futur d'une série temporelle étant donnés T pas de temps observés. Durant le processus d'inférence, le modèle n'a pas accès aux nouvelles observations. On note $\tilde{Z}_t$ les facteurs latents prédits au temps $T + t$, le processus d'inférence calcule $\tilde{Z}_t$ en appliquant successivement la fonction g t fois sur le vecteur appris Z_t, puis calcule la prédiction en sortie $d(\tilde{Z}_t)$.

5.2.2.2 Formulation avec Contraintes Molles

La composante dynamique dans le modèle décrit précédemment est restreinte à cause des contraintes dures sur les Z_t successifs et ne peut modéliser que des phénomènes assez simples. Cette formulation peut rendre l'apprentissage difficile à cause des problèmes d'explosion/disparition du gradient. Nous introduisons ici un modèle plus flexible qui peut être vu comme une formulation relâchée du premier. Ce modèle correspond à la formulation suivante :

$$\begin{aligned} &\arg\min_{d,g,Z} \frac{1}{T} \sum_t \Delta(d(Z_t),\, X_t) \\ &+\lambda \frac{1}{T} \sum_{t=1}^{T-1} ||Z_{t+1} - g(Z_t)||^2 \end{aligned} \tag{5.2.2}$$

où la valeur de λ correspond à l'ampleur de la contrainte molle et est choisie par validation croisée. La différence cruciale ici est que pour une séquence d'apprentissage, tous les termes Z_t sont maintenant appris alors que seulement Z_1 était appris dans la première version du modèle. Ceci résulte en un modèle plus flexible et plus puissant qui est également plus facile à entraîner. L'apprentissage est accompli par descente de gradient stochastique (pour les expériences décrites par la suite, la méthode de Nesterov du gradient accéléré est utilisée [110]) où, à chaque itération, une paire Z_t et Z_{t+1} est échantillonnée, le gradient étant calculé sur cette paire et sur les valeurs correspondantes X_t et X_{t+1}.
Cette version du modèle permet à l'apprentissage d'être plus stable et de converger vers des solutions satisfaisantes plus facilement. Pour cette version molle du modèle, l'apprentissage peut également être fait par une implémentation de descente de gradient stochastique par mini-batch si, à chaque itération un sous-ensemble de paires (Z_t, Z_{t+1}) est sélectionné au lieu d'une seule ; il en découle un important gain de temps en apprentissage lorsque l'on fait usage de GPUs. Ce gain ne peut pas être atteint avec la formulation "dure" du problème d'apprentissage à cause du fait que les interdépendances de tous les Z_t au Z_1 ne permette pas la parallélisation du cal-

cul du gradient. On remarque que, comme tous les Z_t doivent être appris (ou estimés), ceci augmente le nombre de paramètres. Cependant, à cause des contraintes qui forcent Z_{t+1} à être proche de $g(Z_t)$, le nombre de degrés de liberté est limité et le problème d'apprentissage est résoluble facilement et permet l'obtention de résultats satisfaisants (voir section expérimentale). Dans la suite de ce chapitre, le modèle considéré est la version molle du problème.

5.2.3 Modélisation de Séries Spatio-Temporelles

5.2.3.1 Modélisation l'ensemble des séries

Nous introduisons désormais une composante spatiale dans le modèle défini précédemment. Dans le modèle RAINSTORM, cette information était introduite à travers un terme de régularisation structurelle dans la fonction de coût à minimiser. Cette contrainte imposait que deux séries reliées devaient être proches dans l'espace latent. Cette hypothèse est forte et ne permet par exemple pas au modèle d'ignorer des relations dont la prise en compte ne serait pas bénéfique. L'idée ici est de proposer une formulation qui puisse prendre en compte l'information structurelle directement dans la fonction dynamique. On rappelle que chaque série a sa propre représentation latente à chaque pas de temps. Z_t est alors une matrice de taille $n \times N$ telle que $Z_{t,i}$ est le facteur latent de la série i au temps t, N étant la dimension de l'espace latent. Ceci diffère des approches classiques comme pour [83] ou dans les réseaux de neurones récurrents pour les séries temporelles multivariées où Z_t serait un vecteur commun à toutes les séries. Les fonctions de décodage et dynamique, d et g associent respectivement $\mathbb{R}^{n\times N}$ à $\mathbb{R}^{n\times m}$ et $\mathbb{R}^{n\times N}$ à $\mathbb{R}^{n\times N}$:

5.2.3.2 Intégration de l'Information Spatiale

L'information spatiale est intégrée dans la composante dynamique de notre modèle à travers une matrice $W \in \mathbb{R}_+^{n\times n}$ représentant une information a priori sur l'influence mutuelle entre les séries. La représentation latente d'une série au temps $t+1$ dépendra et de la représentation latente de cette série au temps t (intra-dépendance) et des représentations latentes des autres séries (inter-dépendance). L'intra-dépendance sera modélisée par un mapping linéaire dénoté $\Theta^{(0)} \in \mathbb{R}^{N\times N}$ et l'inter-dépendance sera capturée en moyennant les vecteurs latents des séries voisines en utilisant la matrice W, et en calculant une combinaison linéaire de cette moyenne que l'on notera $\Theta^{(1)} \in \mathbb{R}^{N\times N}$. Formellement, le modèle dynamique $g(Z_t)$ est conçu comme suit :

$$Z_{t+1} = h(Z_t\Theta^{(0)} + WZ_t\Theta^{(1)}) \tag{5.2.3}$$

Ici, h est une fonction non linéaire simple. Dans les expériences réalisées, nous utilisons $h = tanh$. Avec cette fonction dynamique,le problème d'apprentissage défini

en Équation 5.2.1 peut être réécrit de la manière qui suit :

$$\begin{gathered} d^*, Z^*, \Theta^{(0)*}, \Theta^{(1)*} = \\ \underset{d,Z,\Theta^{(0)},\Theta^{(1)}}{\arg\min} \frac{1}{T} \sum_t \Delta(d(Z_t), X_t) \\ +\lambda \frac{1}{T} \sum_{t=1}^{T-1} ||Z_{t+1} - h(Z_t\Theta^{(0)} + WZ_t\Theta^{(1)})||^2 \\ \text{avec } Z_t \in \mathbb{R}^{n\times N} \end{gathered} \tag{5.2.4}$$

Dans cette version du modèle, nous faisons l'hypothèse que W est fournie comme a priori sur les relations spatiales entre les séries. Cette information peut également être apprise comme nous le verrons dans la Section 5.2.5.

5.2.4 Relations basées sur des graphes :

W peut être définie de multiples manières - par exemple un noyau de graphe peut être utilisé pour W dans l'Équation 5.2.4. Dans de nombreux cas, il n'y a pas d'information a priori sur l'influence mutuelle de toutes les différentes sources, mais uniquement une information sur la proximité locale de ces sources comme une matrice d'adjacence de sources voisines selon un graphe. C'est le cas dans plusieurs problèmes décrits en section expérimentale. Par exemple, pour la prédiction de trafic routier, le graphe correspondra à la matrice d'adjacence des routes ; pour les données sur l'épidémiologie, il correspondra à la matrice de voisinage des territoires considérés. Nous pourrions déduire une matrice de similarité globale entre sources, e.g. un noyau de graphe, mais pour de nombreuses applications, il n'y en a pas de naturel et nous avons fait le choix d'utiliser plusieurs noyaux locaux sous la forme de puissance de la matrice d'adjacence. Dans ces cas, nous considérons une formulation légèrement différente du modèle précédent où les voisins à différentes distances (k-hop distance) ont différentes influences sur une série donnée. Notons A la matrice d'adjacence d'un graphe pour des séries données, le modèle décrit par l'Équation 5.2.4 est modifié de la façon suivante :

$$\begin{gathered} d^*, Z^*, \Theta^{(0)*}, .., \Theta^{(K)*} = \\ \underset{d,g,Z,\Theta^{(0)},..,\Theta^{(K)}}{\arg\min} \frac{1}{T} \sum_t \Delta(d(Z_t), X_t) \\ +\lambda \frac{1}{T} \sum_{t=1}^{T-1} ||Z_{t+1} - h(\sum_{k=0}^{K} A^k Z_t \Theta^{(k)})||^2 \\ \text{avec } Z_t \in \mathbb{R}^{n\times N} \end{gathered} \tag{5.2.5}$$

où $A^k Z_t$ vise à calculer la représentation latente moyenne des voisins $k-hop$ et $\Theta^{(k)} \in \mathbb{R}^{N\times N}$ est un modèle de l'influence à distance k à apprendre. La valeur K

est définie manuellement. L'architecture du modèle correspondante est illustrée en Figure 5.2. Elle montre comment $Z_{t+1,i}$ est calculée à partir de Z_t pour une série i particulière pour $K = 2$. Le modèle calcule d'abord l'agrégation des représentations voisines, $(A^k Z_t \Theta^{(k)})_i$ étant cette agrégation des voisins $k - hop$ des séries i, $i = 1, 2$. Ces vecteurs sont ensuite sommés et une transformation non linéaire (tangente hyperbolique) est appliquée sur le résultat.

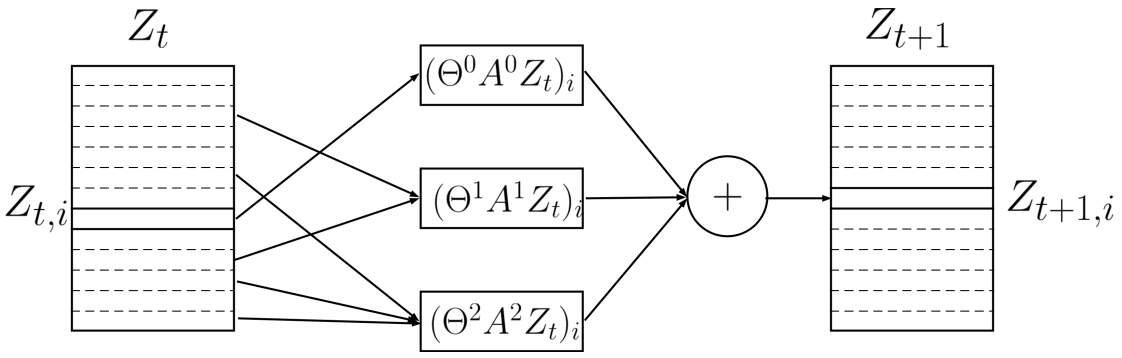

FIGURE 5.2 – Architecture du modèle STNN tel que décrit dans la section in Section 5.2.4

.

Il s'agit du modèle utilisé dans la partie expérimentale et décrit dans la suite de ce chapitre. Il est dénoté STNN pour *Spatio-Temporal Neural Network*. Dans la partie qui suit, nous montrons comment il peut être étendu pour apprendre des dépendances spatiales.

5.2.5 Découverte/Raffinage de Corrélations Spatiales

Tandis que STNN se base sur une connaissance à priori sur la structure du graphe de relations à travers la matrice d'adjacence (A), il peut être utile d'apprendre la force de l'influence entre sources ou d'aller plus loin et de découvrir automatiquement les relations spatiales entre les séries sans aucun a priori. A ces fins, le modèle STNN peut être étendu comme suit :
On note $\Gamma \in \mathbb{R}^{n \times n}$ une matrice de poids *appris* tel que $\Gamma_{i,j}$ est la force de la relation entre la série i et la série j et $\odot$ le résultat du produit de Hadamard (la multiplication terme à terme) entre deux matrices. On remplace le terme A dans un STNN par $A \odot \Gamma$ dans l'Équation 5.2.5. Le cas où il n'y a pas d'information a priori sur les dépendances entre séries correspondra à $A \odot \Gamma$ étant simplement remplacé par Γ. Le problème d'apprentissage correspondant s'écrira :

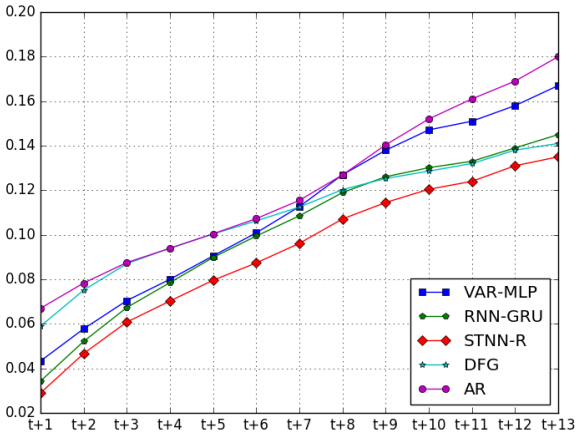

FIGURE 5.3 – RMSE sur le dataset Google Flu à l'horizon $T+1$ à $T+13$

$$
\begin{aligned}
&d^*, Z^*, \Theta^{(0)*}, .., \Theta^{(K)*}, \Gamma^* = \\
&\arg\min_{d,Z,\Gamma} \frac{1}{T} \sum_t \Delta(d(Z_t), X_t) + \gamma|\Gamma| \\
&+\lambda \frac{1}{T} \sum_{t=1}^{T-1} ||Z_{t+1} - h(\sum_{k=0}^{K} (A \odot \Gamma)^k Z_t \Theta^{(k)})||^2 \\
&\text{avec } Z_t \in \mathbb{R}^{n \times N}
\end{aligned}
\tag{5.2.6}
$$

Où $|\Gamma|_1$ est un terme de régularisation L_1 pour rendre la matrice Γ parcimonieuse. Ce modèle sera noté dans la suite de ce chapitre $STNN-R$(efining) quand A est fournie et $STNN-D$(iscovering) quand A ne l'est pas et donc que la structure est uniquement découverte par Γ.

5.3 Expériences

Nous avons comparé les trois variantes de notre modèle *STNN, STNN-R* et *STNN-D* apprises selon l'Équation 5.2.6 aux baselines suivantes :
(i) **Mean** : Une heuristique simple qui prédit les futures valeurs d'une série avec la moyenne des valeurs passées observées calculées sur les T valeurs de chaque fold d'apprentissage.
(ii) **AR** : Un modèle classiquement utilisée en prédiction de séries temporelles et qui correspond à un modèle autorégressif univarié. Pour chaque série et à chaque pas

Dataset	Tâche	n	m	timestep	longueur	fenêtre en train	#folds
Google Flu	Tendances sur la grippe	29	1	semaines	≈ 10 ans	2 ans	50
GHO (25 datasets)	Nombre de morts	91	1	ans	45 ans	35 ans	5
Wind	Vitesse et direction du vent	500	2	heures	30 jours	10 jours	20
PST	Temperature	2520	1	mois	≈ 33 ans	10 ans	15
Bejing	Prédiction de trafic	5000	1	15min	1 semaine	2 jours	20

TABLE 5.1 – Statistiques sur les datasets. n est le nombre de séries, m est la dimension de chaque série, $timestep$ correspond à la durée d'un pas de temps et *#folds* correspond au nombre d'échantillon d'expérience d'apprentissage utilisé en validation. Pour chaque fold, une évaluation a été faite pour les 5 prochains pas de temps $T+1$, $T+2$, ..., $T+5$

de temps, la prédiction est une fonction linéaire de R "lags" de la variable étudiée, R étant un hyper-paramètre.
(iii) **VAR-MLP** pour "vectorial auto-regressive" modèle où les valeurs prédites d'une série au temps $t+1$ dépendent des valeurs passées sur toutes les séries pour un lag de taille R. Le modèle prédictif est un perceptron multi-couche (MLP) avec une couche cachée. Sa performance est uniformément meilleure que celle un VAR-linéaire. Ici encore, le nombre de neurones sur la couche cachée et la valeur de R sont des hyper-paramètres choisis par validation croisée.
(iv) **RNN-tanh** : un réseau de neurone récurrent avec une couche récurrente cachée et la fonction tangente hyperbolique comme fonction d'activation. Comme pour le modèle **VAR-MLP**, toutes les séries sont considérées simultanément c'est-à-dire qu'au temps t, le RNN reçoit en entrée X_{t-1} qui est la valeur de toutes les séries au temps $t-1$ et prédit X_t. De manière similaire à notre modèle, ce réseau de neurone récurrent est un modèle à espace latent avec une dynamique mais sa représentation Z_t dépend explicitement et des valeurs prédites au temps précédent X_{t-1}, et de l'état latent précédent Z_{t-1}. Notons que si ce modèle peut capturer des dépendances spatiales entre séries car elles sont toutes considérées simultanément, il n'a pas de modélisation explicite de ces dépendances.
(v) **RNN-GRU** : idem que le modèle précédent mais les unités récurrentes sont remplacées par des "gated recurrent units" (GRU) qui sont considérées comme l'état de

l'art pour de nombreux problèmes de prédiction de séquence aujourd'hui [1].
(vi) **Dynamic Factor Graph (DFG)** : modèle proposé dans [83]. Le plus proche de notre approche mais utilise une représentation vectorielle pour toutes les séries et, comme un RNN, ne prend pas en compte de relations explicites entre les séries.

Quant au modèle STNN, les valeurs γ de la pénalité L_1 dans *STNN-R* et *STNN-D* (voir Equation 5.2.6) ont été fixées à 0 car, au-delà, les performances en prédiction étaient détériorées (ce qui est classiquement observé dans d'autres modèles avec des régularisations L_1 comme les SVMs). L'influence de γ sur la découverte de structure spatiale est discutée dans la suite du chapitre et illustrée dans la Figure 5.5. Le décodeur d est une simple fonction linéaire pour les trois déclinaisons du modèle.

5.3.1 Jeux de données

Les différents problèmes de prédiction et les jeux de données correspondants sont décrits ci-dessous. Les caractéristiques descriptives de ces jeux de données sont fournies dans le Tableau 5.1.

⊙ **Prédiction d'épidémies :**
Le dataset **Google Flu** contient pour 29 pays environ 10 ans d'estimation de l'activité grippale calculée en agrégeant les requêtes google (voir `http://www.google.org/flutrends`).
Le dataset **Global Health Observatory (GHO)**, rendu disponible par le "Global Health Observatory", (à cette adresse `http://www.who.int/en/`) fournit le nombre de morts pour différentes maladies. Nous en sélectionnons 25, aboutissant à **25 datasets différents**, chacun d'entre eux composés de 91 séries temporelles qui correspondent donc à 91 pays (voir Tableau 5.1). Les résultats dans le Tableau 5.2 sont des moyennes sur tous les datasets.
Le dataset **Wind** (`www.ncdc.noaa.gov/`) consiste en un résumé heure par heure de données météorologiques. Nous prédisons la vitesse du vent et son orientation pour environ 500 stations météorologiques situées aux états unis.
Le dataset **Pacific Sea Temperature (PST)** représente une grille (à une résolution de 2 degrés sur 2 degrés et correspondant à 2520 séries temporelles) mensuelle de la température à la surface de l'eau dans le pacifique pour 399 mois consécutifs entre janvier 1970 et mars 2003. Le but est de prédire les températures futures aux différentes locations géographiques. Les données sont partagées par la "Climate Data Library" de l'université Columbia (`http://iridl.ldeo.columbia.edu/`).

⊙ **Prédiction de trafic routier :** Le but est de prédire le trafic automobile sur un réseau routier.
Nous utilisons le dataset **Beijing dataset** fourni dans [126, 127] qui consiste en des

1. Nous avons également testé les LSTM et obtenu des résultats similaires

Models	Disease		Car Traffic	Geographical		
	Google Flu	GHO [2]	Beijing	Speed	Direction	PST
MEAN	.175	.335	.201	0.191	0.225	.258
AR	.101±.004	.299 ± .008	.075 ± .003	.082 ± .005	0.098±.016	.15 ± .002
VAR-MLP	.095 ± .004	.291 ± .004	.07 ± .002	.071 ± .005	0.111±0.14	.132 ± .003
DFG	.095 ± .008	.288 ± .002	.068 ± .005	.07 ± .004	.092 ± .006	.99 ± .019
RNN-tanh	.082 ± .008	.287 ± .011	.075 ± .006	.064 ± .003	.09 ± .005	.141 ± .01
RNN-GRU	.074 ± .007	.268 ± .07	.074 ± .002	.059 ± .009	.083 ± .005	.104 ± .008
STNN	.066 ± .006	**.261** ± .009	.056 ± .003	**.047** ± .008	**.061** ± .008	.095 ± .008
STNN-R	**.061** ± .008	**.261** ± .01	**.055** ± .004	**.047** ± .008	**.061** ± .008	**.08** ± .014
STNN-D	.073 ± .007	.288 ± .09	.069 ± .01	.059 ± .008	.073 ± .008	.109 ± .015

TABLE 5.2 – Moyenne des RMSE pour différents datasets avec $\tau = 5$ (Erreur de prédiction moyenne pour T+1, T+2,...,T+5). La déviation standard est calculée pour les meilleurs hyper-paramètres.

trajectoires GPS de 10500 taxis pendant une semaine, pour un total de 17 millions de points correspondant à des segments de route sur Pékin. A partir de ce dataset, nous extrayons le volume de trafic agrégé sur des fenêtres de 15min pour 5000 segments de routes. L'objectif est de prédire le trafic pour chaque segment.

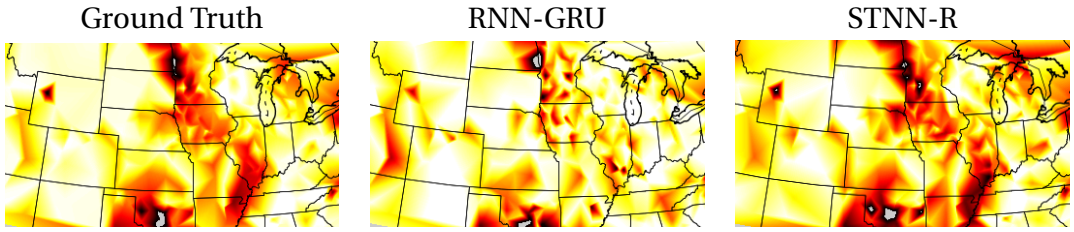

FIGURE 5.4 – Prédiction de la vitesse du vent pour environ 500 stations sur le territoire américain. La prédiction est illustrée pour le timestep $T + 1$ pour les modèles RNN-GRU (centre) et STNN-R (droite).

Pour tous ces datasets, la proximité entre les sources des séries temporelles est fournie à travers un graphe d'adjacence défini par les liens entres sources voisines. Pour la prédiction de trafic routier, deux routes sont connectées si elles partagent une intersection ou si elles aboutissent au même carrefour. Pour la prédiction de propagation d'épidémies, les sources sont connectées si les pays ou états correspondants ont une frontière en commun. Pour les datasets géospatiaux, la distance entre sources était disponible mais par simplification nous utilisons le même protocole que pour les autres datasets. Étant donnée une valeur de seuil d, deux sources sont

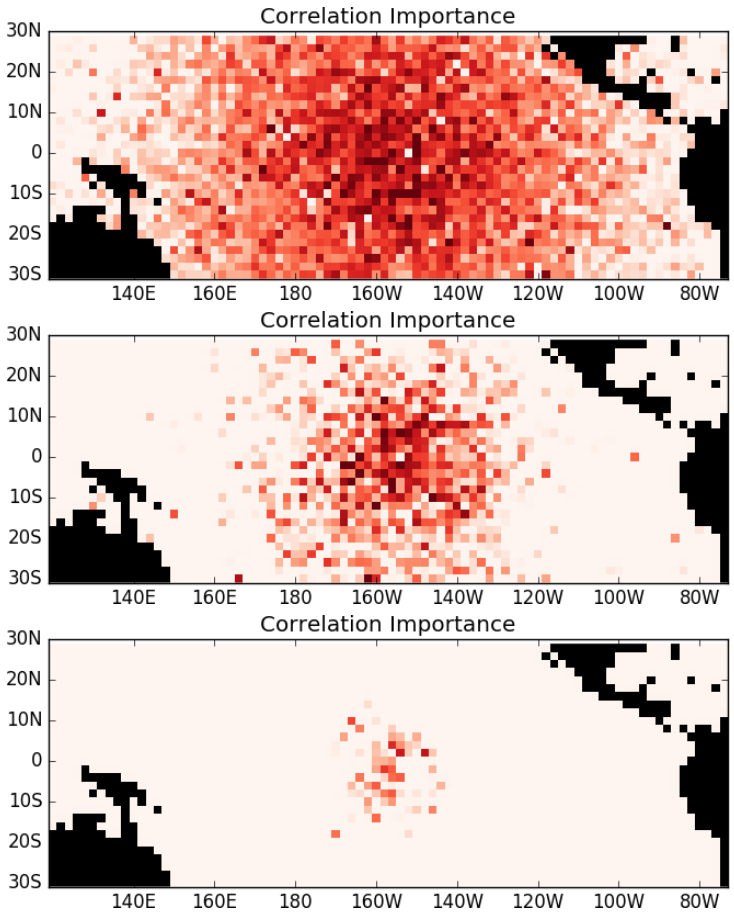

FIGURE 5.5 – Illustrations des corrélations Γ découvertes par le modèle STNN-D, avec γ dans {0.01, 0.1, 1} (de haut en bas).

connectées si la distance qui les sépare est inférieure à d et non connectées dans le cas contraire. Pour toutes les expériences présentées ici et avec les trois variantes de notre modèle (STNN, STNN-R, STNN-D) nous utilisons la formulation d'apprentissage qui correspond à l'Équation 5.2.6.

5.3.2 Résultats

Une évaluation quantitative pour une série de tests sur différents datasets est fournie dans le Tableau 5.2. Une première observation est que les modèles STNN et STNN-R qui utilisent un a priori sur l'information spatiale surpassent significativement tous les autres modèles sur tous les datasets considérés. Le gain de performance relatif en comparaison avec les autres méthodes de l'état de l'art est plus

important sur les datasets géospatiaux et les datasets de trafic routier où le nombre de séries est plus important que pour les datasets de propagation d'épidémie. Chez ces derniers, la matrice d'adjacence fournie étant basée sur la présence ou non d'une frontière en commun, il est probable qu'elle ne soit pas parfaitement représentative des liens réels qui peuvent exister entre deux pays lorsqu'il s'agit de propagation de maladies, le flux de voyageurs pouvant par exemple être important entre deux pays très éloignés. Cependant, les modèles STNN ont pu profiter dans tous les cas de l'information de voisinage apportée par le graphe de proximité.

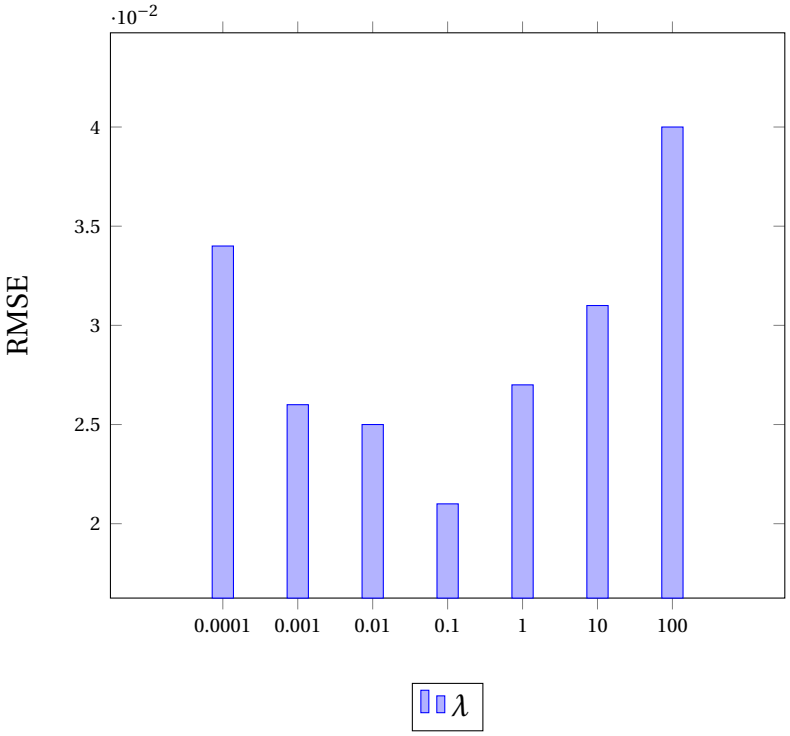

FIGURE 5.6 – RMSE sur le dataset Google Flu en fonction de λ

Les modèles STNN-D et RNN-GRU obtiennent des performances équivalentes. Les deux modèles n'utilisent pas d'information a priori sur la structure des séries temporelles explicitement. STNN utilise une formulation plus compacte que RNN-GRU pour exprimer les dépendances mutuelles entre séries mais les résultats sont comparables. Le modèle "Vectorial AR" obtient logiquement de meilleures performances que le modèle monovarié AR (non montré ici) et la version non linéaire (MLP-VAR) bat la version linéaire.

Les Figures 5.4 et 5.8 illustrent les capacités prédictives des modèles *STNN-R* et *RNN-GRU* sur les datasets des problèmes de météorologie et d'océanographie en

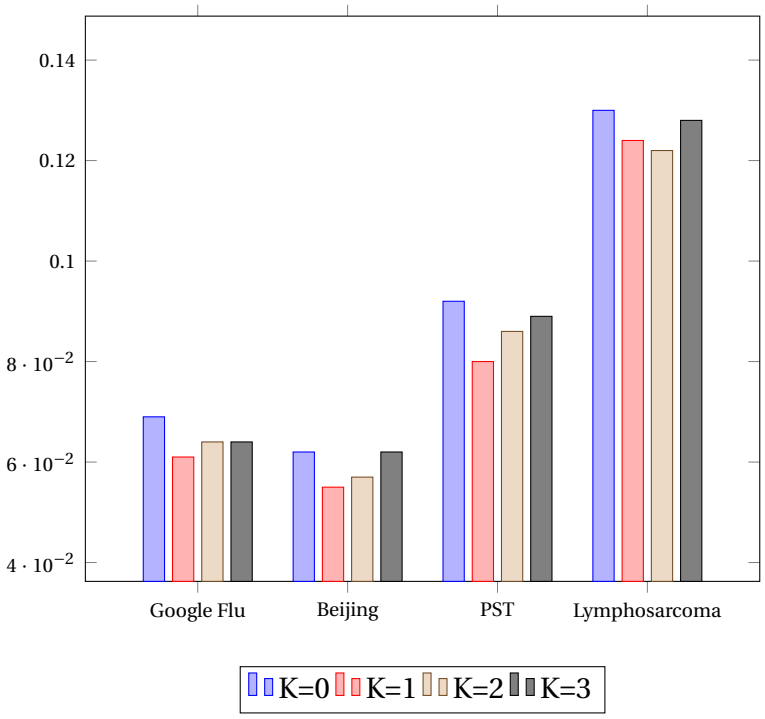

FIGURE 5.7 – RMSE pour le modèle STNN-R sur quatre différents datasets en fonction de K

comparaison avec la vérité constatée (ou mesurée). Clairement, sur ces datasets, STNN obtient de meilleures performances qualitatives que les modèles RNNs en utilisant explicitement une information spatiale. STNN est capable de prédire des détails assez fins correspondant à des interactions locales tandis que les RNNs produisent une prédiction plus bruitée. Ces illustrations sont représentatives des comportements généraux des deux modèles.

Nous donnons également les performances des modèles à différents horizons de prédiction $T+1$, $T+2$, ...$T+13$ – Figure 5.3 pour le dataset Google Flu. Les résultats sont sans équivoque et montrent que STNN obtient de meilleures performances que les autres approches à tous les horizons de prédiction et est capable de mieux capturer les dynamiques et dépendances à long terme. La 5.8 illustre qualitativement l'habilité du modèle à prédire plusieurs time-steps sur un autre dataset.

La Figure 5.6 illustre la valeur du RMSE pour le modèle STNN-R en prédiction à un horizon de $T+1$ pour le dataset Google Flu pour différentes valeurs de λ. On peut voir que la meilleure performance est obtenue pour une valeur intermédiaire de λ : des valeurs basses correspondent à des contraintes temporelles molles et ne

permettent pas au modèle d'apprendre correctement la dynamique des séries. Des valeurs trop fortes à l'inverse (i.e. des contraintes dures) contraignent trop fortement le modèle.

L'influence des voisins à distance (K) est illustrée en Figure 5.7. Elle montre la performance du modèle STNN-R sur différents datasets et pour différentes valeurs de K. On peut observer que considérer un voisinage $1-hop$ est suffisant pour obtenir de bonnes performances dans la majorité des datasets. Pour certaines entrées particulières (e.g *Lymphosarcoma* dataset), de meilleurs résultats sont obtenus avec $K = 2$.
Enfin, nous illustrons dans la Figure 5.5 les valeurs de Γ obtenues avec STNN-D lorsqu'aucune structure (e.g. matrice d'adjacence A) n'est fournie au modèle. Dans cette figure, chaque pixel correspond à une série temporelle particulière et la figure montre les corrélations $\Gamma_{i,j}$ découvertes entre chaque série j avec une série i située au centre de l'image. Plus un pixel est rouge foncé, plus la valeur absolue de $\Gamma_{i,j}$ (les pixels noirs correspondent à des territoires terrestres et non maritimes) est élevée. Différents degrés de sparsité sont illustrés de bas en haut. Ils correspondent à différentes paramétrisations pour le paramètre γ dans l'Équation 5.2.6 et cette figure montre que notre approche est capable de "découvrir" des dépendances spatiales, correspondant ici à des proximités entre les séries.

5.4 Conclusion

Nous avons proposé un nouveau modèle à variables latentes pour répondre efficacement au problème de la prédiction de multiple séries spatio-temporelles. Avec la méthode proposée, les dynamiques des séquences sont capturées dans l'espace latent et les prédictions se font à l'aide d'un mécanisme de mapping entre les représentations et les observations par l'intermédiaire d'une fonction de décodage. De larges expériences sont menées sur des jeux de données représentatifs de différents domaines et nous montrons que ce modèle est capable de capturer des dynamiques spatiales et temporelles et d'obtenir de meilleures performances que les modèles concurrents de l'état de l'art. En termes d'application, nous nous focalisons sur la prédiction mais comme pour le premier modèle présenté dans cette thèse, il peut être utilisé pour la complétion de données manquantes ainsi que pour l'interpolation spatiale de données (kriging). Ce modèle pourrait également être testé dans d'autres domaines comme la complétion ou la prédiction de vidéo [107, 81, 94].

Disease / Model	AR	VAR-MLP	RNN-GRU	Mean	DFG	STNN-R
All causes	0.237	0.228	0.199	0.35	0.291	**0.197**
Tuberculosis	0.407	0.418	**0.37**	0.395	0.421	0.377
Congenital syphilis	0.432	0.443	0.417	0.459	0.422	**0.409**
Diphtheria	0.406	0.396	0.387	0.404	0.419	**0.385**
Malignant neoplasm of esophagus	0.355	**0.341**	**0.341**	0.363	0.372	0.345
Malignant neoplasm of stomach	0.44	0.434	0.431	0.455	0.452	**0.43**
	0.267	0.254	0.282	0.303	0.301	**0.253**
Malignant neoplasm of intestine	0.281	0.29	0.278	0.314	0.305	**0.275**
Malignant neoplasm of rectum	0.501	0.499	**0.481**	0.504	0.509	0.498
Malignant neoplasm of larynx	0.321	0.313	0.32	0.314	0.329	**0.310**
Malignant neoplasm of breast	0.375	0.375	0.382	0.394	0.38	**0.36**
Malignant neoplasm of prostate	0.111	0.113	**0.109**	0.184	0.138	**0.109**
Malignant neoplasm of skin	0.253	0.243	0.227	0.264	0.256	**0.221**
Malignant neoplasm of bones	0.103	0.099	0.097	0.204	0.173	**0.08**
Malignant neoplasm of all other and unspecified sites	0.145	0.157	**0.147**	0.164	0.169	0.156
Lymphosarcoma	0.15	0.132	0.13	0.231	0.135	**0.122**
Benign neoplasms	0.366	0.362	0.332	0.398	0.331	**0.331**
Avitaminonsis	0.492	0.474	0.449	0.571	0.58	**0.414**
Allergic disorders	0.208	0.217	0.221	0.342	0.24	**0.202**
Multiple sclerosis	0.061	0.057	0.061	0.242	0.152	**0.056**
Rheumatic fever	0.325	0.31	0.287	0.345	0.313	**0.256**
Diseases of arteries	0.302	0.301	0.269	0.345	0.328	**0.238**
Influenza	0.141	0.141	0.155	0.23	0.217	**0.125**
Pneumonia	0.119	0.128	**0.1**	0.187	0.187	**0.1**
Pleurisy	0.246	0.246	0.247	0.29	0.272	**0.245**
Gastro-enteritis	0.386	0.369	**0.291**	0.394	0.398	0.295
Disease of teeth	0.344	0.312	0.305	0.413	0.361	**0.302**

TABLE 5.3 – RMSE pour le modèle STNN-R sur les 25 datasets de GHO

TABLE 5.4 – Nous donnons la RMSE moyenne pour les 25 maladies du dataset GHO (Erreur de prédiction moyenne pour T+1, T+2, T+3, T+4,T+5). On peut remarquer que le modèle STNN-R surpasse les modèles de l'état de l'art dans 20 datasets sur 25, et est très proche du modèle RNN-GRU sur les 5 maladies restantes.

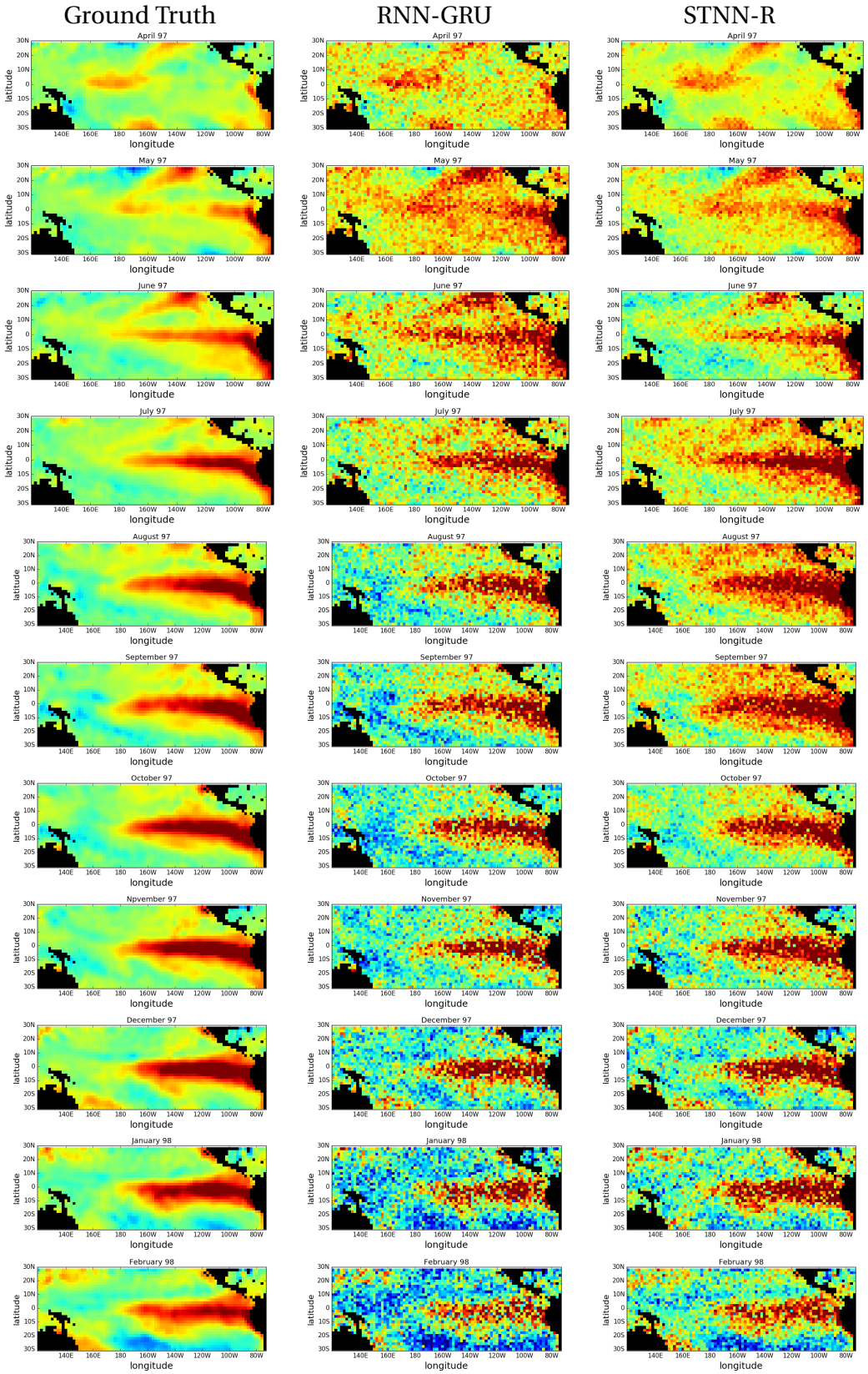

FIGURE 5.8 – Exemple de prédiction pendant 11 mois pour le dataset "température du pacifique". La colonne de gauche est la vérité terrain; les colonnes centrale et celle de droite correspondent respectivement au modèle RNN-GRU et STNN-R. Les valeurs prédites le sont aux temps $T + 1$, $T + 2$, ..., $T + 11$

CHAPITRE 6

CLASSIFICATION DE SÉRIES TEMPORELLES

Résumé

Nous nous intéressons au problème de la classification de séries temporelles qui est un problème clé dans un grand nombre d'applications : transport, finance, santé, bio-informatique, sciences environnementales, etc. Deux familles de techniques sont largement représentées dans la littérature : les algorithmes traditionnels sont basés soit sur des mesures de similarités entre séries temporelles - typiquement avec des distances définies a priori e.g. Dynamic Time Warping- soit sur l'extraction de caractéristiques discriminantes à partir des séries temporelles en les donnant ensuite en entrée d'un classifieur (e.g. SVM). Plus récemment, des architectures profondes ont été appliquées, essentiellement en utilisant des architectures convolutionnelles ou récurrentes. Ces familles de modèles sont connues pour obtenir de bonnes performances lorsque suffisamment de séries temporelles d'entraînement sont disponibles mais peuvent être moins efficaces dans le cas contraire. Dans ce contexte, les réseaux de neurones siamois se sont avérés performants. Pour tirer profit des avantages de ces approches, nous proposons une nouvelle architecture profonde pour la classification de séries temporelles qui combine la puissance discriminante des architectures profondes [42] aux avantages des modèles d'apprentissage de métrique. Nous menons un ensemble d'expériences sur des données artificielles et réelles venant de divers domaines et montrons que notre approche hybride est susceptible d'obtenir de bons taux de classification.

6.1 Introduction

Nous nous intéressons à la tâche de classification de séries temporelles qui correspond à l'assignation d'une série temporelle à une étiquette parmi un ensemble d'étiquettes prédéfini. Deux familles d'approches ont été proposées pour résoudre ce problème (voir Chapitre 2). La première se base sur des mesures de similarités définies à la main, dans l'espace des observations e.g. Dynamic Time Wrapping (DTW). La deuxième famille d'approches se base sur l'extraction de caractéristiques discriminantes à partir des données qui sont ensuite classifiées par un modèle défini a priori. Ainsi, les méthodes existantes font des a priori forts et ne sont pas adaptées à la distribution observée des données.

Nous proposons dans ce chapitre une architecture qui combine des paradigmes de l'apprentissage de métrique avec des réseaux de neurones siamois et des réseaux de neurones profonds. Les algorithmes classiques d'apprentissage profond sont souvent peu efficaces lorsqu'ils doivent faire face à des données supervisées en faible quantité (trop peu d'exemples d'apprentissage)[43][68]. Celà peut être particulièrement problématique dans des applications comme la détection d'évènements rou-

tiers où les exemples sont rares.

Récemment, les réseaux de neurones siamois ont été largement utilisés dans le contexte de l'apprentissage "one-shot", i.e. lorsqu'un seul exemple d'apprentissage est disponible, et il a été montré qu'ils avaient de meilleures capacité de généralistion [116][68]. L'idée de notre modèle est de bénéficier des capacités discriminantes des réseaux de neurones profonds, mais aussi de la capacité des approches d'apprentissage de métrique d'apprendre à partir d'un petit nombre d'exemples. Pour cela, nous combinons un coût de classification avec un coût de minimisation de distance intra-classe dans le but d'apprendre une représentation sur laquelle la classification pourra s'opérer plus facilement.

Cette approche est instanciée avec deux architectures de réseaux de neurones : des réseaux de neurones convolutionnels qui peuvent détecter des formes complexes dans les séries temporelles et des réseaux de neurones récurrents qui sont adaptés aux cas où la présence d'une forme n'est pas la seule information discriminante, mais également un ordre d'apparition par exemple. À notre connaissance, notre modèle est le premier à explorer la combinaison de fonction de classification et d'apprentissage de métrique. Les contributions présentées dans ce chapitre sont les suivantes :

1. Nous proposons un modèle hybride d'apprentissage profond et d'apprentissage de métrique

2. Nous montrons comment ce modèle peut résoudre la tâche de classification de séries temporelles

3. Nous introduisons une famille de problèmes de classification de séries temporelles spécifique et nous donnons des jeux de données artificiels l'illustrant

4. Nous faisons un ensemble d'expérimentations important et comparons notre approche avec d'autres modèles classiques de l'état de l'art.

Ce chapitre est organisé comme suit : La Section 6.2.1 donne brièvement quelques notations. La Section 4.2.3 décrit notre approche hybride. La Section 6.4 définit le protocole expérimental et présente les résultats obtenus. Enfin, la Section 6.5 conclut cette contribution.

6.2 Modèle hybride apprentissage de métrique/classification

6.2.1 Notations

On considère un dataset de n séries temporelles étiquetées dans k différentes classes : $\mathscr{X} = (\mathbf{x}, \mathbf{y})$, où $\mathbf{x}$ est l'ensemble des séries temporelles brutes x_1, x_2, ..., x_n et $\mathbf{y}$ est l'ensemble des étiquettes y_1, y_2, ...y_n avec $\forall i$, $y_i \in \Omega$, Ω étant l'ensemble de toutes les étiquettes possibles $\Omega = l_1, l_2, ..., l_k$. On se concentre sur la tâche de classification qui consiste à assigner une étiquette à chaque exemple d'un ensemble de test. Le pourcentage de bonne classification est utilisé comme métrique pour comparer les différents modèles dans la section 6.4.

L'idée principale de notre approche est de mapper une séquence dans un espace de représentation à grande dimension, et d'utiliser cette représentation latente pour décider à quelle classe d'étiquette assigner la série. On considère un espace de représentation $\mathbb{R}^N$, N étant la taille de l'espace, tel que chaque série en entrée du modèle puisse être représentée par un point particulier dans cet espace. Dans ce qui suit, nous notons $f_\theta(x) \in \mathbb{R}^N$ la fonction qui mappe les séries dans l'espace de représentation, θ étant l'ensemble des paramètres associés à f, et qui seront appris sur l'ensemble d'apprentissage. La forme de la fonction f_θ est décrite dans la Section 6.3.

6.2.2 Approche Discriminante

Comme la fonction f_θ permet de mapper une série temporelle dans $\mathbb{R}^N$, une approche simple consiste à apprendre une fonction de classification g_γ sur l'espace latent telle que $g_{\gamma,k}(f_\theta(x)) = -\log P(l_k|x, \theta, \gamma)$ est la log-probabilité d'assigner le label l_k à une série en entrée x. La fonction f_θ et la fonction $g\gamma$ peuvent être apprises conjointement en minimisant le coût empirique $\mathscr{L}_D$ défini par :

$$\begin{aligned} \mathscr{L}_D(\theta, \gamma) &= \frac{1}{n} \sum_{i=1}^{n} g_{\gamma, y_i}(f_\theta(x_i)) \\ &= \frac{1}{n} \sum_{i=1}^{n} -\log P(y_i|x_i, \theta, \gamma) \end{aligned} \tag{6.2.1}$$

où f_θ et g_γ sont des fonctions dérivables (e.g. des réseaux de neurones). Cette fonction de coût peut être optimisée en utilisant des techniques de descente de gradient usuelles.

6.2.3 Approche Apprentissage de Métrique

Plusieurs modèles se basant sur des mesures de similarité ont été proposés. Ces méthodes reposent typiquement sur une mesure $\Phi(x, x') \in \mathbb{R}^+$ tel que plus Φ est grand, plus x et x' sont similaires. En considérant Φ, la classification d'une nouvelle série x peut être faite en comparant x avec toutes les séries temporelles d'apprentissage, et en choisissant l'étiquette en se basant sur les étiquettes des exemples d'apprentissage les plus similaires en utilisant par exemple une approche plus proche voisin.

Nous proposons d'utiliser une approche basée sur des réseaux de neurones siamois de manière similaire à ce qui est fait dans [49] qui minimise un coût sur des paires d'exemples. L'idée sous-jacente est de forcer le modèle de représentation à apprendre des représentations proches pour des séries qui appartiennent à la même classe et distantes si elles sont de classes différentes. On écrit la fonction coût $\mathscr{L}_M(\theta)$ telle que :

$$\mathscr{L}_M(\theta) = \begin{cases} \frac{1}{2}||f_\theta(x_i) - f_\theta(x_j)||_2^2 & \text{si } y_i = y_j \\ \frac{1}{2}max(0,\, m - ||f_\theta(x_i) - f_\theta(x_j)||_2^2) & \text{sinon} \end{cases} \tag{6.2.2}$$

où l'hyper-paramètre m peut être interprété comme une marge. Dans la partie expérimentale, sa valeur a été fixée à 1. Ce coût permet de minimiser les variations (en termes de distance) intra-classes, c'est-à-dire lorsque les exemples appartiennent à la même classe, et à l'inverse de maximiser la variation inter-classes.

6.2.4 Fonction Objectif Finale

Précédemment sont définis un coût discriminant $\mathscr{L}_D$ et un coût d'apprentissage de métrique $\mathscr{L}_M$. Nous proposons de joindre ces deux fonctions tel que la fonction objectif finale $\mathscr{L}(\theta, \gamma)$ de notre modèle soit un compromis entre les coûts discriminant et d'apprentissage de métrique :

$$\mathscr{L}(\theta, \gamma) = \mathscr{L}_D(\theta, \gamma) + \alpha \mathscr{L}_M(\theta) \tag{6.2.3}$$

où α est un hyper-paramètre qui contrôle l'équilibre entre les deux termes. Dans la partie expérimentale, l'importance de la valeur de cet hyper-paramètre est explorée et discutée (voir Section 6.4). L'architecture globale est illustrée en Figure 6.1. Les rectangles gris correspondent aux paramètres des fonctions f et g qui sont appris respectivement grâce aux "feedbacks" des fonctions de classification et d'apprentissage de métrique pour la fonction f et de la fonction de classification exclusivement pour la fonction g.

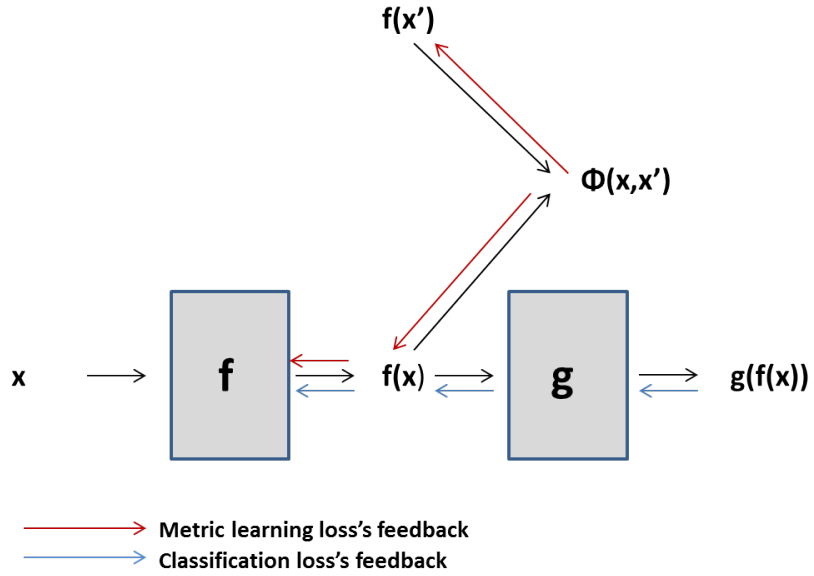

FIGURE 6.1 – Une illustration du modèle que l'on propose. Une séquence x est passée en entrée d'un modèle profond (la fonction f) pour obtenir un vecteur $f(x)$. Ce vecteur est à la fois passé en entrée d'une fonction de classification notée g et comparé par paire avec la représentation $f(x')$ qui correspond à une autre série temporelle x'. Les paramètres des deux fonctions sont appris de manière "end-to-end" en rétro-propageant les deux loss.

6.2.5 Inférence

Pour inférer la classe d'un exemple de test, une représentation est calculée à partir de la fonction f puis cette dernière est traitée par la fonction g qui en sortie donne l'étiquette la plus probable. L'apprentissage de métrique peut donc être vu comme terme de régularisation car la métrique apprise n'est pas utilisée en inférence.

6.2.6 Apprentissage

La fonction objectif est composée de deux fonctions dérivables $\mathscr{L}_D$ et $\mathscr{L}_M$; elle peut donc être optimisée par des approches de descente de gradient. En considérant une descente de gradient stochastique, le premier terme de la fonction de coût $\mathscr{L}_D$ demande de tirer un échantillon des séries d'apprentissage tandis que le second terme requiert de tirer des paires de séries d'apprentissage. L'algorithme de descente de gradient correspondant est illustré dans l'Algorithme 3. Il prend en entrée un ensemble d'apprentissage de séries étiquetées, un hyper-paramètre α pondérant

Algorithm 3 Learning algorithm

1: **input** :
2: Un ensemble d'apprentissage $\mathcal{X} = (\mathbf{x}, \mathbf{y})$
3: Un hyper-paramètre α
4: Un pas d'apprentissage ϵ
5: Deux tailles de mini-batchs B_1 et B_2
6: Paramètres γ et θ initialisés aléatoirement
7: **procedure** APPRENTISSAGE(θ, γ)
8: **for** nombre d'itérations **do**
9: % Descente de gradient pour le terme de classification
10: Tirer B_1 exemples.
11: $\nabla\theta \leftarrow \frac{1}{B_1} \sum_{i=1}^{B_1} \nabla_\theta \Delta(g_\gamma(f_\theta(x_i)), y_i)$
12: $\nabla\gamma \leftarrow \frac{1}{B_1} \sum_{i=1}^{B_1} \nabla_\gamma \Delta(g_\gamma(f_\theta(x_i)), y_i)$
13: $\theta \leftarrow \theta - \epsilon\nabla\theta$
14: $\gamma \leftarrow \gamma - \epsilon\nabla\gamma$
15: % Descente de gradient pour le terme d'apprentissage de métrique
16: Tirer B_2 paires d'exemples.
17: $\nabla\theta \leftarrow \frac{1}{B_2} \sum_{i,j=1}^{B_2} \nabla_\theta \Delta(f_\theta(x_i), f_\theta(x_j), y_i, y_j)$
18: $\theta \leftarrow \theta - \epsilon\alpha\nabla\theta$
19: **endfor**
20: **end procedure**

le terme d'apprentissage de métrique, un pas d'apprentissage ϵ, la taille des mini-batchs B_1 et B_2 [1] et des paramètres θ et γ initialisés aléatoirement (ligne 2-6). Puis, itérativement, les paramètres du gradient sont calculés et leurs valeurs sont mises à jour : les gradients de θ et α relatifs au terme de classification (ligne 10-14) et le gradient de θ par rapport au terme d'apprentissage de métrique (ligne 17-18). L'algorithme produit en sortie deux ensembles de paramètres θ et γ qui permettent de classifier un ensemble donné de séries temporelles. Dans la section suivante, nous décrivons succinctement les deux architectures de réseaux de neurones profonds que l'on propose.

1. En français "mini-lots". Au lieu de calculer le gradient d'un seul exemple comme c'est le cas lors d'une descente de gradient stochastique ou le gradient de tous les exemples, cette méthode calcule à chaque itération le gradient sur un "lot" d'exemples (un sous-ensemble des exemples d'apprentissage). Cette version est plus efficace que la méthode de descente de gradient stochastique, notamment parce que l'usage de GPU permet un gain de temps important dû à la parallélisation. En pratique et dans les expériences, nous utilisons $B_1 = B_2$

6.3 Architectures des réseaux de neurones

Les fonctions f et g décrites précédemment peuvent être instanciées par n'importe quelles fonctions dérivables et en particulier par n'importe quelle architecture de réseaux de neurones. Nous nous focalisons sur des architectures convolutionnelles et récurrentes respectivement pour leur puissance discriminante et leur capacité à traiter des données temporelles. Nous proposons les deux instanciations suivantes pour notre modèle :

6.3.0.1 Réseau de neurones convolutionnel

Comme illustré en Figure 6.2, la série entière est traitée par un réseau de neurones convolutionnel (CNN). Le réseau consiste en plusieurs couches de convolution (leur nombre et leurs caractéristiques sont spécifiés en Section 6.4), suivies par une couche entièrement connectée. Par simplicité, nous nous référons à la partie convolutionnelle en parlant de fonction $f_\theta(x)$, qui prend en entrée une série temporelle x et produit un vecteur en sortie.

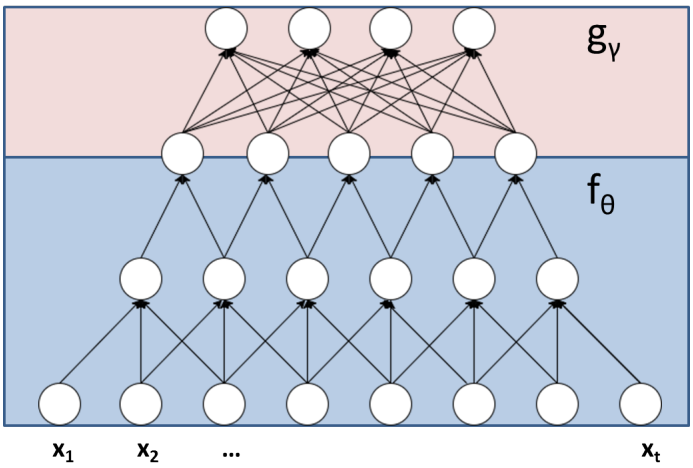

FIGURE 6.2 – Illustration schématique d'une architecture de réseau de neurones convolutionnel profond : elle consiste en deux couches de convolution suivies d'une couche entièrement connectée. Le réseau prend une série temporelle de taille t en entrée. Cette dernière est traitée par la fonction f_θ qui produit un vecteur en sortie qui est passé à la fonction de classification g_γ.

Une telle architecture est capable de détecter efficacement une forme discriminante présente dans les séries temporelles appartenant à une même classe grâce aux couches successives de convolution. La fonction de classification g est ici constituée

d'une couche entièrement connectée qui en sortie donne les probabilités respectives d'appartenance à chaque classe.

6.3.0.2 Réseau de neurones récurrent et convolutionnel

Une autre instanciation du modèle que l'on propose est donnée en Figure 6.3. Une couche récurrente est ajoutée en sortie de couches convolutionnelles. Elle permet de calculer un vecteur en sortie qui sera passé à la fonction de classification g.

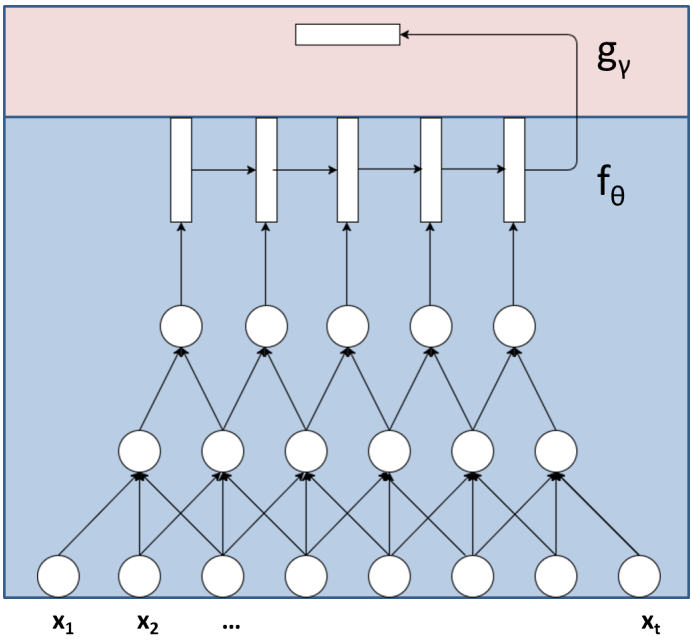

FIGURE 6.3 – Illustration d'une architecture récurrente convolutionnelle profonde qui est composée de deux couches de convolution suivies d'une couche récurrente. L'entrée consiste en une série de taille t. Cette dernière est traitée par la fonction f_θ qui produit un vecteur en sortie qui est passé à la fonction de classification g_γ.

Ces deux architectures sont proposées pour répondre à la large variété de problèmes différents de classification de séries temporelles. Si les réseaux convoluitonnels seuls sont capables d'obtenir des performances très satisfaisantes si les classes sont caractérisées par la présence d'une forme donnée dans les séries temporelles, ils peuvent s'avérer moins efficaces dans le cas où la dynamique d'apparition de ces formes influent sur la classe. A l'inverse, les réseaux de neurones récurrents peuvent perdre de leur efficacité dans le cas de longues séries caractérisées par la présence de formes données.

6.4 Expériences

6.4.1 Jeux de données

Nous testons les performances de notre approche sur les jeux de données publics suivants [2] :

— **UCR** : nous avons selectionné vingt jeux de données sur le " UCR time series repository" [26] qui est un ensemble de datasets de référence pour la tâche de classification de séries temporelles. Tous les jeux de données UCR sont téléchargeables librement (`http://timeseriesclassification.com`). Les données sont séparées en ensemble d'apprentissage/test.

— **PLAID** : ce dataset [45] consiste en 1074 signatures provenant de 11 appareils électroménagers. Il s'agit de la puissance électrique consommée de divers appareils au fil de la journée (ventilateurs, lampes, réfrigérateurs, machines à laver...). Chaque appareil mène à une forme particulière en fonction de ses horaires habituels d'usage, son nombre d'usages dans la journée, etc.

— **Arabic Spoken Digit** : ce dataset [51] comprend 8 800 prononciations de nombres produits par 88 différentes personnes. Chaque personne prononce chaque chiffre dix fois. Les données consistent en une série temporelle de taille 13 MFCCs. Elles sont échantillonnées à 11025Hz et 16 bits en utilisant des fenêtres de Hamming. Nous utilisons deux versions différentes de ce jeu de données :
  — Une "digit version" dans laquelle le chiffre prononcé est l'étiquette de classe
  — Une "voice version" dans laquelle l'étiquette correspond au locuteur.

6.4.2 Modèles Concurrents

Nous donnons et comparons les résultats en classification pour les modèles de l'état de l'art suivants. Notez que de nombreuses autres méthodes existent et que par manque de place, il n'est pas possible d'en donner une liste exhaustive. Cependant, leurs performances sont connues et publiques [27] et on peut voir que notre approche obtient les meilleures performances pour la majorité des jeux de données considérés ici.

— **DTW** : Dynamic Time Warping standard, avec une distance euclidienne utilisée pour calculer la distance paire à paire entre les séquences alignées. La classification est ensuite effectuée à l'aide de la méthode des plus proches voisins.

— **1-NN** : Approche plus proche voisin, avec une distance euclidienne.

— **MCNN** : Pour la méthode "Multi-scale CNN". Ce modèle et l'architecture utilisée sont décrits dans [35]. Les données sont prétraitées manuellement en uti-

2. Nous proposons également des expériences sur des jeux de données synthétiques par la suite

lisant du "down sampling" et du "skip sampling" pour permettre l'approche multi-échelle. Cette approche est l'état de l'art pour nombre de jeu de données UCR.

— **CNN et CNN+met** : Nous donnons les résultats pour l'architecture convolutionnelle que nous utilisons, avec et sans [3] le terme d'apprentissage de métrique pour illustrer le gain obtenu en généralisation grâce à son usage.
— **RCNN and RCNN+met** : De la même manière, nous donnons les résultats avec et sans l'apprentissage de métrique pour la variante récurrente et convolutionnelle.
— **RNN** : Nous donnons aussi les résultats pour un réseau de neurones récurrent classique (LSTM) utilisé en classification de séquence. Cette variante obtient presque systématiquement des performances inférieures à la version RCNN.

6.4.3 Protocole expérimental

Les jeux de données sont séparés en ensemble d'apprentissage et ensemble de test, en suivant la séparation usuellement utilisée [4]. Le seul prétraitement utilisé dans nos expériences est la "z-normalisation" [5] sur les deux ensembles d'apprentissage et de test, avec les moyennes et déviations standard de l'ensemble d'apprentissage pour chaque jeu de données. Les hyper-paramètres sont fixés par validation croisée (avec 4 validations) sur l'ensemble d'apprentissage, et les résultats en test sont reportés. L'ensemble des hyper-paramètres testés et choisis par validation croisée est décrit dans le Tableau 6.1.

6.4.4 Résultats expérimentaux

Une évaluation quantitative pour une série de tests sur différents jeux de données est fournie dans le Tableau 6.2. Une première observation est que les modèles hybrides CNN+met et RCNN+met qui font de l'apprentissage de métrique obtiennent de meilleures performances que les autres modèles testés sur presque tous les jeux de données. Le gain par rapport à des méthodes comme DTW ou MCNN est par exemple de 27% en taux de bonne classification sur le jeu de données "Beef" où le nombre de séries est petit. L'apprentissage de métrique aide ici considérablement à apprendre des représentations discriminantes. Sur d'autres datasets, le modèle est

3. Ce qui correspond au modèle que l'on propose dans le cas particulier où $\alpha = 0$

4. Les jeux de données utilisés sont connus et publiquement disponibles ; ils sont par défaut répartis en ensemble d'apprentissage et ensemble de test. Pour les jeux de données artificiels que nous utilisons et décrivons, nous les rendons disponibles sur GitHub ainsi que la façon de les séparer en entraînement/test.

5. Procédure standard de normalisation de séries temporelles de façon à ce que la moyenne de chaque série soit approximativement 0 et son écart type de 1.

Modèles	Hyper-paramètres	Valeurs testées
DTW	Aucun	Aucun
1-NN	Aucun	Aucun
MCNN	Taille de la fenêtre glissante	(0.5, 0.9)
	Taille des filtres	(0.05, 0.1, 0.2)
	Facteur de pooling	(2, 3, 5)
RNN	Taille de l'état caché	(20, 40, 80, 150)
	Nombre de couches récurrentes	(1, 2, 3)
CNN	Nombre de filtres	(5, 8, 16, 32)
	Taille des filtres de convolution	(3, 5, 8)
	Nombre de couches de convolution	(1, 2)
RCNN	Architecture de la partie convolutionnelle	Meilleurs hyper-paramètres pour CNN
	Taille des couches récurrentes	(40, 80, 150)
	Nombre de couches récurrentes	(1, 2)
CNN + met	idem que CNN	Meilleurs hyper-paramètres pour CNN
	α	(0.0001, 0.001, 0.01, 0.1, 1, 10)
RCNN + met	idem que RCNN	Meilleurs hyper-paramètres pour RCNN
	α	(0.0001, 0.001, 0.01, 0.1, 1, 10)

TABLE 6.1 – Valeurs d'hyper-paramètres testées

également capable de faire usage favorablement du terme d'apprentissage de métrique, améliorant les capacités de généralisation des deux architectures proposées presque systématiquement. Plus généralement, la précision que notre approche atteint est la plus grande pour la majorité des jeux de données utilisés dans ce papier, ce qui montre son intérêt. Dans la sous-section suivante, un ensemble de problèmes particuliers est donné pour mieux illustrer les différences entre les architectures CNN et RCNN et l'habileté du dernier à résoudre des problèmes pour lesquelles l'ordre ou le nombre de formes particulières apparaissant dans les séries temporelles a une importance.

6.4.5 Comptage et ordonnancement de formes particulières

Pour montrer les différences qui peuvent exister entre les modèles RCNN et CNN, nous introduisons deux familles de problèmes spécifiques. Nous montrons que des algorithmes de l'état de l'art comme DTW ne sont pas capables de faire face aux particularités présentées ici. Nous réalisons et partageons des jeux de données qui correspondent à ces problèmes auxquels nous nous référons par "Ordre et comptage de formes dans les séries temporelles" ; des résultats expérimentaux sont donnés.

Dataset/model	DTW	1-NN	MCNN	RNN	CNN	RCNN	CNN met	RCNN met
PLAID	0.758±0	0.559±0	0.84±0.014	0.822±0.008	0.852±0.007	0.847±0.011	**0.933**±0.005	0.925±0.003
Arabic (digit)	0.727±0	0.85±0	0.9±0.008	0.864±0.01	0.94±0.003	0.955±0.004	0.94±0.004	**0.975**±0.009
Arabic (speaker)	0.771±0	0.81±0	0.87±0.002	0.841±0.001	0.86±0.008	0.9±0.01	0.898±0.004	**0.92**±0.003
Fish	0.823±0	0.783±0	**0.891**±0.013	0.683±0.025	**0.891**±0.016	0.74±0.018	**0.891**±0.01	0.74±0.012
Beef	0.633±0	0.633±0	0.633±0	0.566±0	0.766±0	0.566±0	**0.9**±0	0.6±0
ChlorineConcentr	0.65 ±0	0.65±0	0.797±0.015	0.75±0.023	0.851±0.009	0.8±0.01	**0.906**±0.012	0.828±0.011
Coffee	**1.0** ±0	**1.0** ±0	0.964±0.007	0.964±0.015	**1.0** ±0	**1.0** ±0	**1.0** ±0	**1.0** ±0
Adiac	0.604±0	0.611±0	0.769±0.011	0.655±0.003	0.755±0.008	0.655±0.008	**0.782**±0.018	0.655±0.004
TwoPatterns	**1.0** ±0	0.91±0	0.998±0.002	**1.0**±0	0.911±0.018	**1.0**±0	0.911±0.013	**1.0**±0
50words	0.69±0	0.631 ±0	**0.81**±0.02	0.631±0.014	0.607±0.006	0.582±0.005	0.607±0.005	0.608±0.006
OSULeaf	0.591±0	0.521±0	0.572±0.012	0.572±0.01	0.55±0.012	0.587±0.009	0.591±0.01	**0.611**±0.011
Lightening2	**0.869**±0	0.754±0	0.607±0.002	0.754±0.01	0.607±0.003	0.754±0.009	0.689±0.002	0.754±0.007
Lightening7	0.726±0	0.575±0	0.726±0.007	0.712±0.005	0.712±0.002	**0.74**±0.001	0.712±0.003	**0.74**±0.003
SwedishLeaf	0.792±0	0.789±0	0.893±0.004	0.873±0.004	0.853±0.004	0.883±0.005	0.912±0.005	**0.925**±0.006
MALLAT	0.936±0	0.914±0	0.914±0.015	0.914±0.01	0.936±0.012	0.914±0.016	**0.937**±0.006	0.914±0.013
NonInvThorax1	0.79±0	0.829±0	0.936±0.008	0.908±0.012	0.925±0.012	0.936±0.006	**0.938**±0.005	**0.938**±0.005
NonInvThorax2	0.865±0	0.88±0	0.927±0.007	0.94±0.008	0.94±0.009	0.939±0.007	**0.942**±0.008	0.941±0.004
MoteStrain	0.835±0	0.879±0	**0.921**±0.011	0.853±0.014	0.862±0.008	0.866±0.004	0.888±0.004	0.873±0.009
SonyRobotSurface	0.725±0	0.695±0	0.77±0.009	0.847±0.005	0.755±0.01	0.902±0.006	0.755±0.001	**0.92**±0.007
GunPoint	0.907±0	0.913±0	0.96±0.008	0.953±0.006	0.96±0.005	0.973±0.004	0.966±0.014	**0.989**±0.005
SonyRobotSurface2	0.831±0	0.859±0	0.93±0.004	0.924±0.019	0.859±0.011	0.924±0.012	0.859±0.002	**0.947**±0.007
ItalyPowerDemand	0.95±0	0.955±0	0.97±0.011	0.962±0.004	0.972±0.003	0.964±0.004	**0.974**±0.002	0.972±0.006
Haptics	0.377±0	0.37±0	**0.47**±0.005	0.464±0.014	0.464±0.012	0.464±0.014	0.464±0.009	0.464±0.012
Précision Moyenen	0.776	0.755	0.827	0.813	0.819	0.82	**0.845**	0.836
Classement moyen	6.65	6.30	4.91	4.47	3.65	4.82	**2.30**	2.86
# de datasets gagnés	3	1	4	1	2	3	10	**11**

TABLE 6.2 – Taux de bonne classification en Test pour tous les jeux de données décrits et pour toutes les baselines. L'écart type est obtenu sur 5 runs avec les hyperparamètres fixés.

6.4.5.1 Comptage

Nous considérons qu'un "événement" est l'apparition d'une forme spécifique dans une série temporelle. On se place dans le cadre d'un problème de classification binaire où deux séries temporelles appartiennent à la classe 1 si cet évènement apparait deux fois, et à la classe -1 s'il apparait un plus grand nombre de fois (e.g. 3). Nous illustrons une instance de ce problème sur la Figure 6.4. On voit facilement qu'un algorithme de DTW dans de nombreux cas trouvera un alignement entre des séries

au nombre d'évènements différents. A l'inverse, les réseaux de neurones récurrents se sont montrés efficaces lorsqu'ils étaient confrontés à des problèmes de comptage [93], tandis que les réseaux de neurones convolutionnels sont connus pour leur pouvoir discriminant lorsqu'il s'agit de détecter des formes [76]. Ainsi, l'usage d'un RCNN (voire Figure 6.3) est ici naturel : les premières couches de convolution doivent être capables de détecter une forme particulière et la partie récurrente du modèle peut compter son nombre d'occurrences. Cette intuition est vérifiée dans la section expérimentale (Section 6.4).

6.4.5.2 Ordonnancement

Nous gardons la définition précédente d'un "évènement". Nous considérons l'existence de plusieurs formes particulières. Le problème d'ordonnancement qui nous intéresse se penche sur l'ordre d'apparence de deux formes différentes au moins, où différentes classes correspondent à différents ordres entre ces évènements. Nous illustrons une instance de ce problème sur la Figure 6.5.

6.4.5.3 Description des jeux de données artificiels

Nous proposons dans ce papier trois jeux de données synthétiques qui implémentent des instances des problèmes décrits précédemment. Ils sont les suivants :

- **2v3** : Un problème de comptage d'évènements à deux classes, l'un avec deux occurrences d'un évènement donné et l'autre avec trois. Il consiste en 1000 séries de taille 235 chacune.
- **2v4** : Un problème de comptage d'évènements à deux classes, l'un avec deux occurrences d'un évènement donné et l'autre avec quatre. Il consiste en 1000 séries de taille 255 chacune.
- **2v5** : Un problème de comptage d'évènements à deux classes, l'un avec deux occurrences d'un évènement donné et l'autre avec cinq. Il consiste en 1000 séries de taille 275 chacune.
- **3Shapes** : Un problème d'ordonnancement à deux classes : trois formes sont présentes dans deux ordres différents (ce qui est le critère discriminant). Il consiste en 1000 séries de taille 180 chacune.

Pour chacun de ces jeux de données, les ensembles d'apprentissage sont fixés à 800 exemples tandis que les ensembles de tests sont fixés à 200 exemples. Les motifs générés consistent en des formes "triangulaires" formées par une série de valeurs croissantes puis décroissantes symétriquement pour les jeux de données 2v3, 2v4 et 2v5. Cette même forme est reprise pour le jeu de données 3shapes qui voit également apparaître comme motif particulier une série de valeur constante et une série de valeur croissante puis décroissante en suivant une courbe.

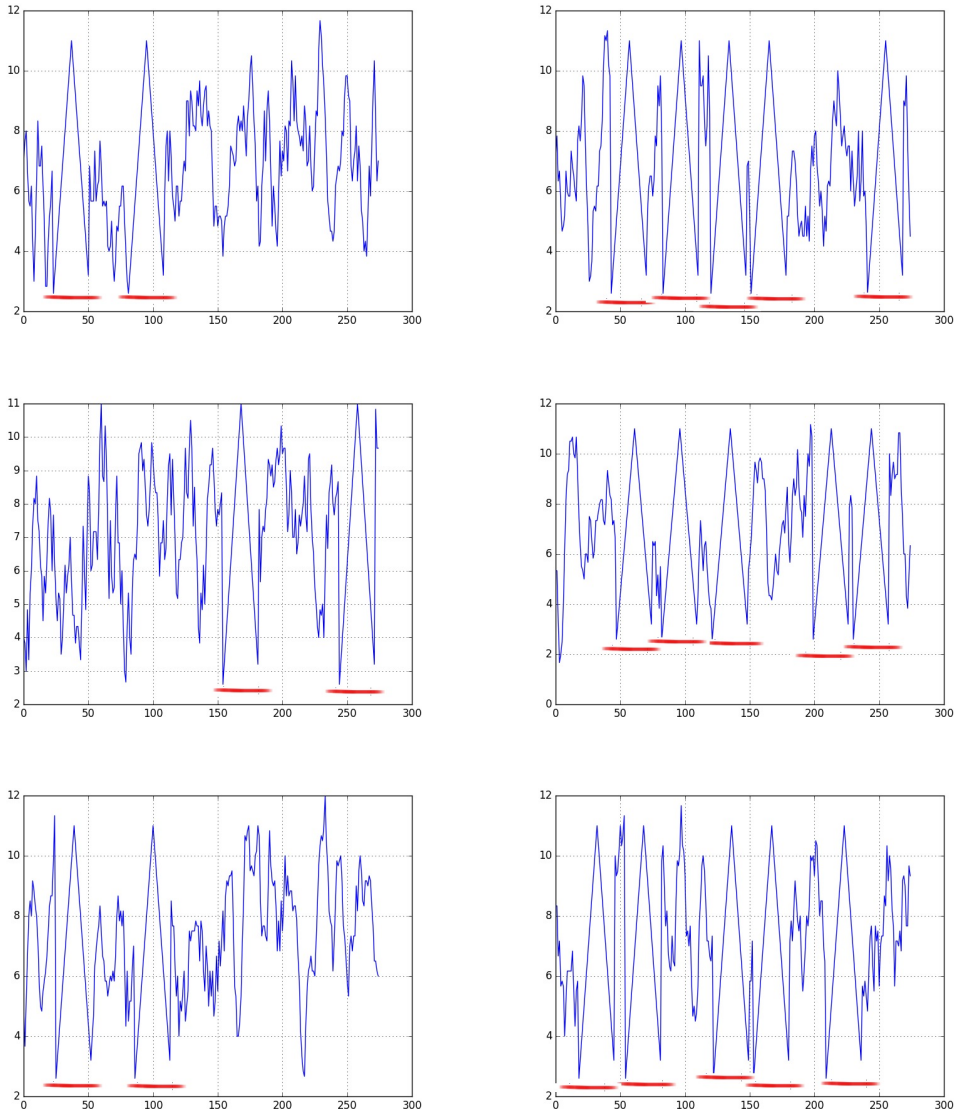

FIGURE 6.4 – Exemple d'un problème de comptage dans les séries temporelles avec deux classes présentes : une forme spécifique montrée à l'aide d'une marque rouge est présente soit deux fois, soit cinq fois selon son appartenance de classe. Ces séries viennent du jeu de données "2v5" que nous proposons.

6.4.6 Résultats sur les jeux de données artificiels

On peut voir que sur les données synthétiques, le modèle RCNN+met atteint presque toujours 100% de précision ce qui montre la capacité des réseaux de neurones ré-

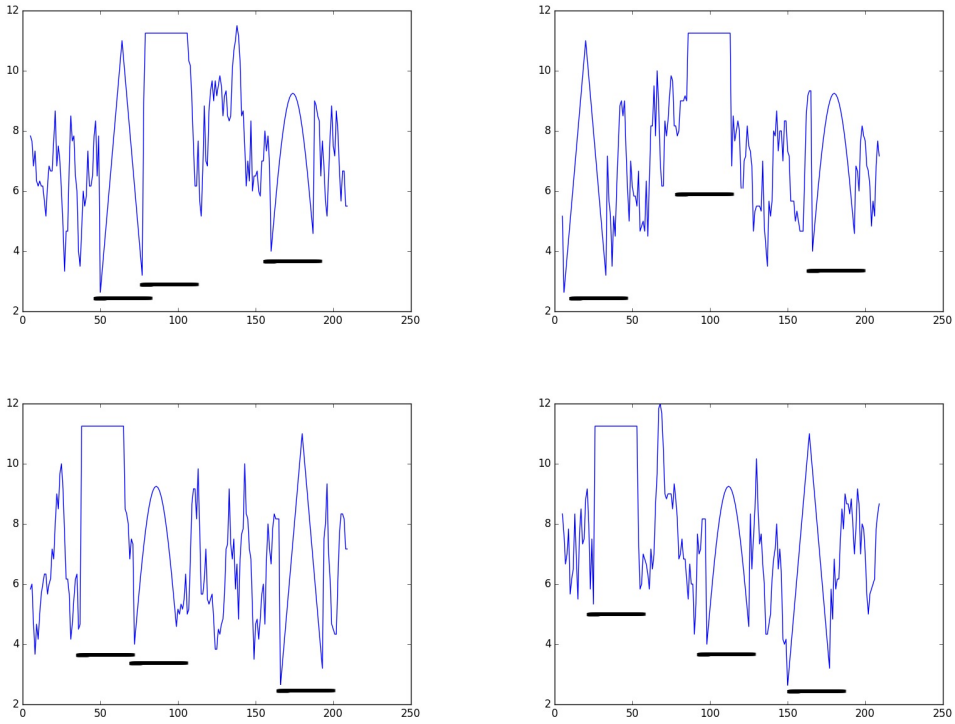

FIGURE 6.5 – Exemple d'un problème d'ordonnancement dans les séries temporelles avec deux classes présentes : trois formes données sont présentes, montrées à l'aide d'une marque noire. Elles sont réparties en deux classes, selon l'ordre d'apparition des formes. Ces séries viennent du jeu de données "3shapes" que nous proposons.

currents à compter ou à détecter un ordre parmi des formes particulières; d'autres modèles comme le DTW ne sont pas aussi efficaces dans ce domaine. Ici encore, le terme d'apprentissage de métrique permet d'atteindre la meilleure précision pour les données en test. On peut également constater que la combinaison de réseaux convolutionnels et récurrents est en général meilleure que des réseaux purement convolutionnels ou purement récurrents.

6.4.7 Discussion et pondération du terme d'apprentissage de métrique

Nous illustrons l'influence de la valeur du terme d'apprentissage de métrique sur la précision en classification, en faisant varier α de 0 à $+\infty$. Lorsque $\alpha = 0$, le signal d'apprentissage de métrique disparaît et seul le coût de classification est minimisé. Quand $alpha$ augmente, le terme d'apprentissage de métrique devient plus

Dataset/model	DTW	1-NN	MCNN	RNN	CNN	RCNN	CNN met	RCNN met
2v5	0.85±0	0.62±0	0.79±0.011	0.99±0.005	0.785±0.009	0.995±0.005	0.795±0.014	**1.0**±0
2v4	0.65±0	0.595±0	0.73 ±0.003	0.985±0.005	0.695±0.005	0.99±0	0.695±0.05	**1.0** ±0
2v3	0.59±0	0.515±0	0.65±0.001	0.98±0.002	0.64±0.008	0.995±0	0.64±0.01	**1.0** ±0
3Shapes	0.94±0	0.93±0	975±0.001	0.98±0.002	0.93±0.003	0.975±0.001	0.94±0.004	**0.985** ±0.002

TABLE 6.3 – Taux de bonne classification en Test pour tous les jeux de données décrits et pour toutes les baselines. L'écart type est obtenu sur 5 runs avec les hyperparamètres fixés.

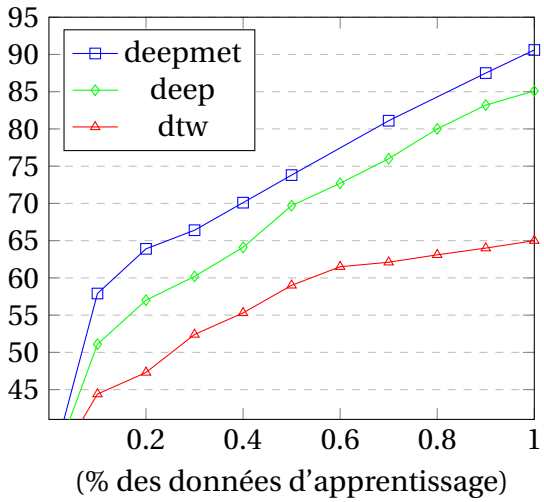

FIGURE 6.6 – Précision en test en fonction du pourcentage de données d'apprentissage utilisé pour le dataset ChlorineConcentration

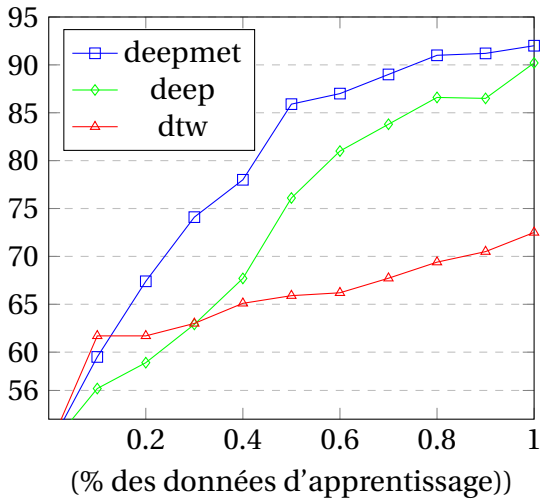

FIGURE 6.7 – Précision en test en fonction du pourcentage de données d'apprentissage utilisé pour le dataset SonyAIBORobotSurface

important et prend progressivement le pas devant le terme de classification jusqu'à $\alpha = +\infty$ où seul l'apprentissage de métrique persiste. Une visualisation qualitative du processus est montrée dans la Figure 6.8 pour donner une intuition du phénomène sous-jacent. Le Tableau 6.2 montre que le terme d'apprentissage de métrique améliore quasiment systématiquement la précision en classification. Plus précisément, les Figures 6.7 et 6.6 illustrent le fait que le terme d'apprentissage de métrique améliore surtout les performances quand le jeu de données d'apprentissage est de petite taille, les réseaux de neurones siamois étant capables d'apprendre à partir de très peu d'exemples. Le gain en performance peut être moins important quand le nombre d'exemples d'apprentissage est très important mais reste quand même ob-

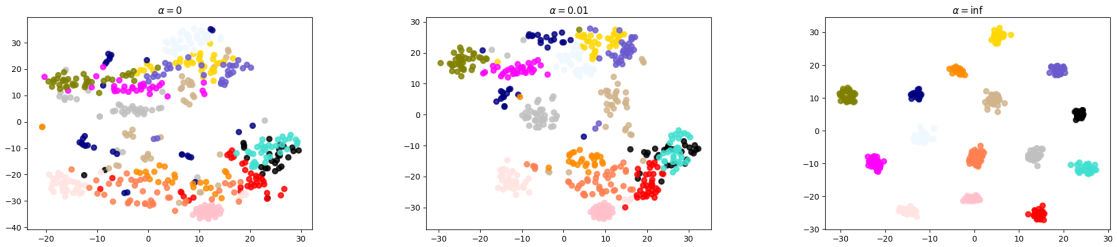

FIGURE 6.8 – Visualisation t-SNE[80] des caractéristiques extraites avec un modèle RCNN siamois pendant l'apprentissage pour différentes valeurs de α pour le jeu de données ChlorineConcentration.

servé. Le terme d'apprentissage de métrique peut alors être vu comme un terme de régularisation permettant de généraliser à partir de peu d'exemples mais également un équilibre entre séparabilité des données et minimisation de la variation intra-classe. les Figure 6.9 et 6.10 montrent comment la valeur du paramètre α influence le score en précision.

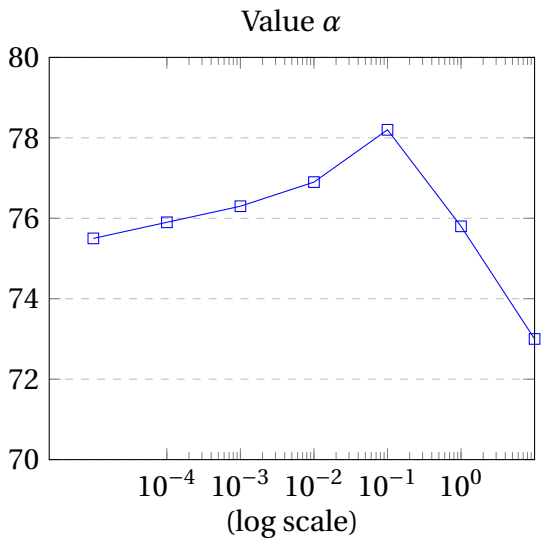

FIGURE 6.9 – Précision en test en fonction de la valeur de l'hyper-paramètre α sur le jeu de données Adiac (échelle logarithmique)

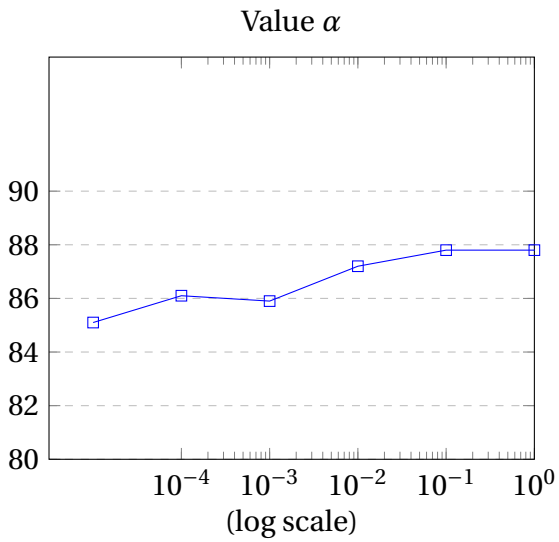

FIGURE 6.10 – Précision en test en fonction de la valeur de l'hyper-paramètre α sur le jeu de données ChlorineConcentration (échelle logarithmique)

6.5 Conclusion

Ce chapitre a présenté une méthode pour apprendre simultanément une métrique et un modèle de classification pour les séries temporelles. Nous montrons que considérer deux critères - l'un de classification et l'autre d'apprentissage de métrique- permet d'améliorer le pouvoir de généralisation de réseaux de neurones profonds dans le cas des données temporelles. Deux différentes architectures sont proposées, l'une convolutionnelle et l'autre convolutionnelle et récurrente. Les deux coûts sont appris en partageant les paramètres entre des réseaux de neurones siamois et une architecture profonde. Des expériences sont menées sur des jeux de données artificiels et réels et la méthode que nous proposons obtient les meilleures performances dans la majorité des cas. Cette méthode est particulièrement appropriée dans le cas où les données d'apprentissage sont peu nombreuses. Une extension possible de ces travaux pourrait impliquer des données de différentes natures.

CHAPITRE 7

CONCLUSION

7.1 Conclusion

Dans cette thèse, nous nous sommes intéressés à deux tâches principales : la prédiction de séries temporelles multivariées et relationnelles et la classification de séries temporelles. Dans le cas de la prédiction de séries multivariées, nous disposons de connaissances structurelles sur les séries temporelles sous la forme d'un graphe de voisinage entre elles.

Nous proposons deux modèles pour résoudre le problème de la prédiction de séries temporelles. Le modèle RAINSTORM dans le Chapitre 3, ainsi que des variantes du modèle dans le chapitre 4. Dans le Chapitre 5, nous proposons un autre modèle capable non seulement de prédire mais aussi d'apprendre les relations entre séries. Dans le Chapitre 6, nous donnons un modèle hybride d'apprentissage de métrique et de classification de séries temporelles.

Dans le Chapitre 3, nous proposons un modèle de prédiction et de complétion de séries temporelles multivariées capable de prendre en compte un ensemble de relations sous la forme d'un graphe entre les séries. Les séries temporelles sont projetées dans un espace latent de représentation et les séries reliées selon le graphe donné en entrée ont des projections proches dans l'espace de représentation. Nos expériences montrent qu'un tel modèle bénéficie de l'information structurelle et témoigne de l'intérêt de prendre en compte ces a priori pour la tâche de prédiction de séries temporelles, mais aussi en complétion. Nous menons également des expériences décrivant une situation où la nature des informations est hétérogène : à chaque pas de temps, il peut s'agir de valeurs réelles correspondant à la quantité modélisée par les séries temporelles ou alors d'étiquettes discrètes. Des expériences montrant que le modèle obtient des performances satisfaisantes dans ce contexte sont décrites.

Dans le chapitre suivant, nous décrivons la façon dont nous avons étendu cette première approche pour dans un premier temps être capable de prédire des séries temporelles de natures différentes : nous menons des expérimentations sur des données fournies par la ville de Lyon où nous prédisons conjointement le trafic automobile ainsi que la disponibilité des places de stationnement. Les séries des deux types sont projetées dans le même espace latent dans lequel elles maintiennent une influence mutuelle par le biais d'un terme de régularisation structurelle. Un décodeur spécifique à chaque série est en revanche appris, ce qui permet une meilleure prise en compte des caractéristiques de chaque phénomène à prédire. Nous montrons l'intérêt de considérer les prédictions de trafic et de la disponibilité des places de stationnement comme des phénomènes joints plutôt que de les traiter séparément.

Nous présentons également une autre version du modèle où les séries temporelles sont projetées dans un espace de représentation gaussien au lieu d'un espace de représentation déterministe. Cette extension permet d'avoir des prédictions avec une incertitude exprimée naturellement ; les expériences montrent également que quantitativement, les performances peuvent être plus satisfaisantes.

La contribution suivante est une formulation plus générale d'un modèle pouvant s'appliquer à la prédiction de séries spatio-temporelles mais également à la découverte de structure entre ces séries. Un graphe de voisinage est également fourni en entrée mais est pris en compte directement dans la fonction dynamique à la place du terme de régularisation structurelle que nous proposions précédemment. Les relations qui peuvent être de natures et d'intensités très diverses sont ainsi modélisées spécifiquement, ce dont le modèle précédent était incapable. Plusieurs variantes sont données :

— Une version avec les relations connues.
— Une version où les relations sont connues mais dont l'intensité est à raffiner.
— Une version où les relations sont inconnues et donc à découvrir.

Les capacités prédictives et de découverte de structure du modèle sont illustrées sur des problèmes venant de domaines différents et la supériorité de notre approche par rapport aux méthodes de l'état de l'art est montrée.

Dans le Chapitre 6, nous proposons un modèle de classification de séries temporelles : un coût d'apprentissage de métrique est appris conjointement à un coût de classification sur des architectures de réseaux de neurones profonds. Nous montrons l'amélioration des performances en termes de taux de bonne classification lorsque ce terme est appris ; celà est particulièrement observé pour les jeux de données de taille réduite. Nous donnons deux architectures différentes, l'une convolutionnelle et l'autre récurrente et convolutionnelle pour résoudre la tâche de classification de séries. Des expériences sur des données réelles et artificielles sont menées pour mettre en lumière l'intérêt d'utiliser notre approche.

7.2 Perspectives

Notre travail sur la prise en compte d'une information structurelle lors de la prédiction de séries temporelles montre l'intérêt de cette famille d'approches et ouvre un certain nombre de perspectives de recherche.

Tout d'abord, plusieurs variantes spécifiques concernant la formulation et l'influence de l'information spatiale et temporelle sont envisagées. Par exemple, l'aspect

multi-relationnel n'est pas abordé dans nos travaux. En effet, plusieurs types de relations peuvent exister entre les différentes séries temporelles et peuvent être prises en compte différemment par les modèles d'apprentissage de représentation. Ainsi, la prédiction de diffusion des épidémies pourrait être améliorée en prenant en compte - en plus du voisinage géographique entre pays - d'autres facteurs pouvant influer sur les relations entre les séries temporelles produites, comme les flux de voyageurs échangés ou des caractéristiques environnementales propres aux diverses régions. Des modèles capables de tenir compte de ces différentes informations relationnelles pourraient être dérivés facilement des modèles que nous avons proposés.

Un autre aspect qui pourrait être exploré serait de considérer des relations dynamiques alors que nos approches ne considèrent que des relations statiques. Par exemple, les flux de voyageurs entre plusieurs pays varient constamment. Dans le cas du trafic automobile, cela pourrait signifier que les modèles prédictifs sont capables de s'adapter à la fermeture temporaire de certaines voies et d'anticiper les congestions à venir.

Plus généralement, les modèles prédictifs que nous proposons pourraient également être testés sur des données structurées spécifiques de natures différentes. La prédiction ou la complétion de vidéo serait un des champs d'application possible et de futures recherches pourraient par exemple s'intéresser à l'intégration dans nos modèles des méthodes de prise en compte de la structure des images (convolutions). D'autres tâches de prédiction dans des données structurées peuvent également être mentionnées comme la prédiction d'activité dans les réseaux sociaux ou la prédiction de liens.

Dans un registre différent, nous avons également pu montrer l'utilité de considérer des modèles joints d'apprentissage de métrique et de classification de séries temporelles pour améliorer les performances en classification de ces approches. Une extension naturelle à ces travaux serait d'explorer les performances de ces modèles hybrides sur des tâches habituellement visées par l'apprentissage de métrique comme l'identification de paires (appliquée par exemple à la reconnaissance de signature). Les performances obtenues laissent aussi envisager que considérer des modèles hybrides de classification et d'apprentissage de métrique pourrait être bénéfique sur d'autres familles de données temporelles comme les vidéos, ou statiques comme les images...

Une autre direction de recherches envisagée est le développement de modèles capables d'apprendre à partir du plus petit nombre d'exemples disponibles. Une formulation hybride d'apprentissage de métrique et de classification sous contrainte de

budget pourrait être proposée. L'acquisition d'exemples étiquetés étant souvent coûteuse ou possible uniquement dans des quantités très limitées, ces méthodes économes sont de plus en plus investiguées (cf : apprentissage "one-shot").

BIBLIOGRAPHIE

[1] I.e.e.e icdm contest : Tomtom traffic prediction for intelligent gps navigation.

[2] John Aach and George M Church. Aligning gene expression time series with time warping algorithms. *Bioinformatics*, 17(6) :495–508, 2001.

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

[4] Alexander C Aitken. Iv.—on least squares and linear combination of observations. *Proceedings of the Royal Society of Edinburgh*, 55 :42–48, 1936.

[5] Muhammad Tayyab Asif, Justin Dauwels, Chong Yang Goh, Ali Oran, Esmail Fathi, Muye Xu, Menoth Mohan Dhanya, Nikola Mitrovic, and Patrick Jaillet. Spatiotemporal patterns in large-scale traffic speed prediction. *Intelligent Transportation Systems, IEEE Transactions on*, 15(2) :794–804, 2014.

[6] Muhammad Tayyab Asif, Nikola Mitrovic, Lalit Garg, Justin Dauwels, and Patrick Jaillet. Low-dimensional models for missing data imputation in road networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013.

[7] Anthony Bagnall, Jason Lines, Aaron Bostrom, James Large, and Eamonn Keogh. The great time series classification bake off : a review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery*, pages 1–55, 2016.

[8] Anthony Bagnall, Jason Lines, Jon Hills, and Aaron Bostrom. Time-series classification with cote : the collective of transformation-based ensembles. *IEEE Transactions on Knowledge and Data Engineering*, 27(9) :2522–2535, 2015.

[9] Mohammad Taha Bahadori, Qi Rose Yu, and Yan Liu. Fast multivariate spatiotemporal analysis via low rank tensor learning. In *Advances in neural information processing systems*, pages 3491–3499, 2014.

[10] Yu BAI, Kun XUE, and Xiao-guang YANG. Forecasting method of parking-demand based on capacity-of-network [j]. *Journal of Traffic and Transportation Engineering*, 4(4) :49–52, 2004.

[11] Marta Bańbura and Michele Modugno. Maximum likelihood estimation of factor models on datasets with arbitrary pattern of missing data. *Journal of Applied Econometrics*, 29(1) :133–160, 2014.

[12] Gustavo EAPA Batista and Maria Carolina Monard. An analysis of four missing data treatment methods for supervised learning. *Applied Artificial Intelligence*, 17(5-6) :519–533, 2003.

[13] Mustafa Gokce Baydogan, George Runger, and Eugene Tuv. A bag-of-features framework to classify time series. *IEEE transactions on pattern analysis and machine intelligence*, 35(11) :2796–2802, 2013.

[14] Souhaib Ben Taieb and Rob Hyndman. Boosting multi-step autoregressive forecasts. In *Proceedings of The 31st International Conference on Machine Learning*, pages 109–117, 2014.

[15] Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 1171–1179, 2015.

[16] Yoshua Bengio. Neural net language models. *Scholarpedia*, 3(1) :3881, 2008.

[17] Donald J Berndt and James Clifford. Using dynamic time warping to find patterns in time series. In *KDD workshop*, volume 10, pages 359–370. Seattle, WA, 1994.

[18] Simon Bourigault, Cedric Lagnier, Sylvain Lamprier, Ludovic Denoyer, and Patrick Gallinari. Learning social network embeddings for predicting information diffusion. In *Proceedings of the 7th ACM international conference on Web search and data mining*. ACM, 2014.

[19] George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. *Time series analysis : forecasting and control*. John Wiley & Sons, 2015.

[20] Jane Bromley, James W. Bentz, Léon Bottou, Isabelle Guyon, Yann LeCun, Cliff Moore, Eduard Säckinger, and Roopak Shah. Signature verification using a "siamese" time delay neural network. *IJPRAI*, 7(4) :669–688, 1993.

[21] Felix Caicedo, Carola Blazquez, and Pablo Miranda. Prediction of parking space availability in real time. *Expert Systems with Applications*, 39(8) :7281–7290, 2012.

[22] Kanad Chakraborty, Kishan Mehrotra, Chilukuri K Mohan, and Sanjay Ranka. Forecasting the behavior of multivariate time series using neural networks. *Neural es maladresses ou des imprudences. Le vol de données personnelles, l'usurpation d'identité, les intrusions et les piratages sont de plus en plus cités dans l'actualité.Pour évitenetworks*, 5(6), 1992.

[23] Chris Chatfield. *Time-series forecasting*. CRC Press, 2000.

[24] Huanhuan Chen, Fengzhen Tang, Peter Tino, and Xin Yao. Model-based kernel for efficient time series analysis. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 392–400. ACM, 2013.

[25] Yanping Chen, Bing Hu, Eamonn Keogh, and Gustavo EAPA Batista. Dtw-d : time series semi-supervised learning from a single example. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 383–391. ACM, 2013.

[26] Yanping Chen, Eamonn Keogh, Bing Hu, Nurjahan Begum, Anthony Bagnall, Abdullah Mueen, and Gustavo Batista. The ucr time series classification archive. *URL www. cs. ucr. edu/˜ eamonn/time_series_data*, 2015.

[27] Yanping Chen, Eamonn Keogh, Bing Hu, Nurjahan Begum, Anthony Bagnall, Abdullah Mueen, and Gustavo Batista. The ucr time series classification archive, July 2015. `www.cs.ucr.edu/~eamonn/time_series_data/`.

[28] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv :1406.1078*, 2014.

[29] Sumit Chopra, Raia Hadsell, and Yann LeCun. Learning a similarity metric discriminatively, with application to face verification. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 539–546. IEEE, 2005.

[30] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv :1412.3555*, 2014.

[31] Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C Courville, and Yoshua Bengio. A recurrent latent variable model for sequential data. In *Advances in neural information processing systems*, pages 2962–2970, 2015.

[32] Jerome T Connor, R Douglas Martin, and Les E Atlas. Recurrent neural networks and robust time series prediction. *Neural Networks, IEEE Transactions on*, 5(2) :240–254, 1994.

[33] Gabriella Contardo, Ludovic Denoyer, Thierry Artieres, and Patrick Gallinari. Learning states representations in pomdp. *arXiv preprint arXiv :1312.6042*, 2013.

[34] Noel A. C. Cressie and Christopher K. Wikle. *Statistics for spatio-temporal data*. Wiley series in probability and statistics. Hoboken, N.J. Wiley, 2011.

[35] Zhicheng Cui, Wenlin Chen, and Yixin Chen. Multi-scale convolutional neural networks for time series classification. *arXiv preprint arXiv :1603.06995*, 2016.

[36] Jason V Davis, Brian Kulis, Prateek Jain, Suvrit Sra, and Inderjit S Dhillon. Information-theoretic metric learning. In *Proceedings of the 24th international conference on Machine learning*, pages 209–216. ACM, 2007.

[37] Corrado De Fabritiis, Roberto Ragona, and Gaetano Valenti. Traffic estimation and prediction based on real time floating car data. In *Intelligent Transportation Systems, 2008. ITSC 2008. 11th International IEEE Conference on*, pages 197–203. IEEE, 2008.

[38] Jan G De Gooijer and Rob J Hyndman. 25 years of time series forecasting. *International journal of forecasting*, 22(3) :443–473, 2006.

[39] Guido Dornhege, Benjamin Blankertz, Matthias Krauledat, Florian Losch, Gabriel Curio, and Klaus robert Muller. Optimizing spatio-temporal filters for improving brain-computer interfacing. In *in NIPS*, 2005.

[40] Ludovic Dos Santos, Benjamin Piwowarski, and Patrick Gallinari. Multilabel classification on heterogeneous graphs with gaussian embeddings. In *ECML-KDD*, 2016.

[41] Mark S Dougherty and Mark R Cobbett. Short-term inter-urban traffic forecasts using neural networks. *International journal of forecasting*, 13(1) :21–31, 1997.

[42] Thibaut Durand, Nicolas Thome, and Matthieu Cord. Weldon : Weakly supervised learning of deep convolutional neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[43] Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, 11(Feb) :625–660, 2010.

[44] Gartheeban Ganeshapillai, John Guttag, and Andrew Lo. Learning connections in financial time series. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 109–117, 2013.

[45] Jingkun Gao, Suman Giri, Emre Can Kara, and Mario Bergés. Plaid : a public dataset of high-resoultion electrical appliance measurements for load identification research : demo abstract. In *proceedings of the 1st ACM Conference on Embedded Systems for Energy-Efficient Buildings*, pages 198–199. ACM, 2014.

[46] Alan Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *IIIE ICASSP*, 2013.

[47] Dimitrios Gunopulos and Gautam Das. Time series similarity measures and time series indexing. In *Acm Sigmod Record*, volume 30, page 624. ACM, 2001.

[48] Valery Guralnik and Jaideep Srivastava. Event detection from time series data. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 33–42. ACM, 1999.

[49] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *Computer vision and pattern recognition, 2006 IEEE computer society conference on*, volume 2, pages 1735–1742. IEEE, 2006.

[50] James Douglas Hamilton. *Time series analysis*, volume 2. Princeton university press Princeton, 1994.

[51] Nacereddine Hammami and Mokhtar Sellam. Tree distribution classifier for automatic spoken arabic digit recognition. In *Internet Technology and Secured Transactions, 2009. ICITST 2009. International Conference for*, pages 1–4. IEEE, 2009.

[52] Johan Himberg, Kalle Korpiaho, Heikki Mannila, Johanna Tikanmaki, and Hannu TT Toivonen. Time series segmentation for context recognition in mobile devices. In *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*, pages 203–210. IEEE, 2001.

[53] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8) :1735–1780, 1997.

[54] James Honaker and Gary King. What to do about missing values in time-series cross-section data. *American Journal of Political Science*, 54(2) :561–581, 2010.

[55] Bing Hu, Yanping Chen, and Eamonn Keogh. Time series classification under more realistic assumptions. In *Proceedings of the 2013 SIAM International Conference on Data Mining*, pages 578–586. SIAM, 2013.

[56] Junlin Hu, Jiwen Lu, and Yap-Peng Tan. Discriminative deep metric learning for face verification in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1875–1882, 2014.

[57] Faridul Islam, Muhammad Shahbaz, Ashraf U Ahmed, and Md Mahmudul Alam. Financial development and energy consumption nexus in malaysia : a multivariate time series analysis. *Economic Modelling*, 30 :435–441, 2013.

[58] Yann Jacob, Ludovic Denoyer, and Patrick Gallinari. Classification and annotation in social corpora using multiple relations. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 1215–1220. ACM, 2011.

[59] SamRoweis JacobGoldberger and RuslanSalakhutdinov GeoffHinton. Neighbourhood components analysis. *NIPS'04*, 2004.

[60] GR Jagadeesh, T Srikanthan, and XD Zhang. A map matching method for gps based real-time vehicle location. *Journal of Navigation*, 57(03) :429–440, 2004.

[61] Young-Seon Jeong, Myong K Jeong, and Olufemi A Omitaomu. Weighted dynamic time warping for time series classification. *Pattern Recognition*, 44(9) :2231–2240, 2011.

[62] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D) :35–45, 1960.

[63] Yiannis Kamarianakis and Poulicos Prastacos. Forecasting traffic flow conditions in an urban network : comparison of multivariate and univariate approaches. *Transportation Research Record : Journal of the Transportation Research Board*, 1857(1), 2003.

[64] Ahmad Kazem, Ebrahim Sharifi, Farookh Khadeer Hussain, Morteza Saberi, and Omar Khadeer Hussain. Support vector regression with chaos-based firefly algorithm for stock market price forecasting. *Applied soft computing*, 13(2) :947–958, 2013.

[65] Eamonn Keogh, Kaushik Chakrabarti, Michael Pazzani, and Sharad Mehrotra. Locally adaptive dimensionality reduction for indexing large time series databases. *ACM SIGMOD Record*, 30(2) :151–162, 2001.

[66] Eamonn Keogh, Selina Chu, David Hart, and Michael Pazzani. An online algorithm for segmenting time series. In *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*, pages 289–296. IEEE, 2001.

[67] Diederik Kingma and Jimmy Ba. Adam : A method for stochastic optimization. *ICLR'15*.

[68] Gregory Koch. *Siamese neural networks for one-shot image recognition*. PhD thesis, University of Toronto, 2015.

[69] Hema Koppula and Ashutosh Saxena. Learning spatio-temporal structure from rgb-d videos for human activity detection and anticipation. In *Proceedings of ICML*, 2013.

[70] Takashi Kuremoto, Shinsuke Kimura, Kunikazu Kobayashi, and Masanao Obayashi. Time series forecasting using a deep belief network with restricted boltzmann machines. *Neurocomputing*, 137 :47–56, 2014.

[71] Christopher G Lamoureux and William D Lastrapes. Persistence in variance, structural change, and the garch model. *Journal of Business & Economic Statistics*, 8(2) :225–234, 1990.

[72] Martin Längkvist, Lars Karlsson, and Amy Loutfi. A review of unsupervised feature learning and deep learning for time-series modeling. *Pattern Recognition Letters*, 42 :11–24, 2014.

[73] Marc T. Law, Nicolas Thome, and Matthieu Cord. Quadruplet-wise image similarity learning. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2013.

[74] Marc T. Law, Nicolas Thome, and Matthieu Cord. Fantope regularization in metric learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[75] Marc T. Law, YaoLiang Yu, Matthieu Cord, and Eric P. Xing. Closed-form training of mahalanobis distance for supervised clustering. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[76] Yann LeCun, Yoshua Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10) :1995, 1995.

[77] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks. *arXiv preprint arXiv :1511.05493*, 2015.

[78] T Warren Liao. Clustering of time series data—a survey. *Pattern recognition*, 38(11) :1857–1874, 2005.

[79] Yisheng Lv, Yanjie Duan, Wenwen Kang, Zhengxi Li, and F-Y Wang. Traffic flow prediction with big data : A deep learning approach. 2014.

[80] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov) :2579–2605, 2008.

[81] Michaël Mathieu, Camille Couprie, and Yann LeCun. Deep multi-scale video prediction beyond mean square error. *CoRR*, abs/1511.05440, 2015.

[82] Martin R Min, Laurens Maaten, Zineng Yuan, Anthony J Bonner, and Zhaolei Zhang. Deep supervised t-distributed embedding. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 791–798, 2010.

[83] Piotr Mirowski and Yann LeCun. Dynamic factor graphs for time series modeling. In *Machine Learning and Knowledge Discovery in Databases*, pages 128–143. Springer, 2009.

[84] Douglas C Montgomery, Cheryl L Jennings, and Murat Kulahci. *Introduction to time series analysis and forecasting*. John Wiley & Sons, 2015.

[85] Klaus Robert Muller, Alexander J Smola, Gunnar Ratsch, Bernhard Scholkopf, Jens Kohlmorgen, and Vladimir Vapnik. Using support vector machines for time series prediction. *Advances in kernel methods—support vector learning*, 1999.

[86] John Paparrizos and Luis Gravano. k-shape : Efficient and accurate clustering of time series. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pages 1855–1870. ACM, 2015.

[87] Irfan Pratama, Adhistya Erna Permanasari, Igi Ardiyanto, and Rini Indrayani. A review of missing values handling methods on time-series data. In *Information Technology Systems and Innovation (ICITSI), 2016 International Conference on*, pages 1–6. IEEE, 2016.

[88] Tooraj Rajabioun and Petros Ioannou. On-street and off-street parking availability prediction using multivariate spatiotemporal models. *Intelligent Transportation Systems, IEEE Transactions on*, 16(5) :2913–2924, 2015.

[89] Thanawin Rakthanmanon, Bilson Campana, Abdullah Mueen, Gustavo Batista, Brandon Westover, Qiang Zhu, Jesin Zakaria, and Eamonn Keogh. Searching and mining trillions of time series subsequences under dynamic time warping. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 262–270. ACM, 2012.

[90] Yuanfang Ren and Yan Wu. Convolutional deep belief networks for feature extraction of eeg signal. In *Neural Networks (IJCNN), 2014 International Joint Conference on*, pages 2850–2853. IEEE, 2014.

[91] Steffen Rendle. Factorization machines with libfm. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 3(3) :57, 2012.

[92] Matt Richtel. Now, to find a parking spot, drivers look on their phones. *The New York Times (Technology)*, 95, 2011.

[93] Paul Rodriguez, Janet Wiles, and Jeffrey L Elman. A recurrent neural network that learns to count. *Connection Science*, 11(1) :5–40, 1999.

[94] Bernardino Romera-Paredes and Massimiliano Pontil. A new convex relaxation for tensor completion. In *Advances in Neural Information Processing Systems*, pages 2967–2975, 2013.

[95] Sam Roweis, Geoffrey Hinton, and Ruslan Salakhutdinov. Neighbourhood component analysis. *Adv. Neural Inf. Process. Syst.(NIPS)*, 17 :513–520, 2004.

[96] Havard Rue and Leonhard Held. *Gaussian Markov random fields : theory and applications*. CRC Press, 2005.

[97] Haşim Sak, Andrew Senior, and Françoise Beaufays. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

[98] Ruslan Salakhutdinov and Geoffrey E Hinton. Learning a nonlinear embedding by preserving class neighbourhood structure. In *AISTATS*, volume 11, 2007.

[99] Patrick Schäfer. The boss is concerned with time series classification in the presence of noise. *Data Mining and Knowledge Discovery*, 29(6) :1505–1530, 2015.

[100] Patrick Schäfer. Scalable time series classification. *Data Mining and Knowledge Discovery*, 30(5) :1273–1298, 2016.

[101] Arthur Schuster. On the periodicities of sunspots. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 206 :69–100, 1906.

[102] Jingbo Shang, Yu Zheng, Wenzhu Tong, Eric Chang, and Yong Yu. Inferring gas consumption and pollution emission of vehicles throughout a city. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014.

[103] Jingbo Shang, Yu Zheng, Wenzhu Tong, Eric Chang, and Yong Yu. Inferring gas consumption and pollution emission of vehicles throughout a city. In *KDD 2014*, 2014.

[104] Weiwei Shi, Yongxin Zhu, S Yu Philip, Tian Huang, Chang Wang, Yishu Mao, and Yufeng Chen. Temporal dynamic matrix factorization for missing data prediction in large scale coevolving time series. *IEEE Access*, 4 :6719–6732, 2016.

[105] Yunlong Song, Min Liu, Shaojie Tang, and Xufei Mao. Time series matrix factorization prediction of internet traffic matrices. In *Local Computer Networks (LCN), 2012 IEEE 37th Conference on*, pages 284–287. IEEE, 2012.

[106] S Sridevi, S Rajaram, C Parthiban, S SibiArasan, and C Swadhikar. Imputation for the analysis of missing values and prediction of time series data. In *Recent Trends in Information Technology (ICRTIT), 2011 International Conference on*, pages 1158–1163. IEEE, 2011.

[107] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhudinov. Unsupervised learning of video representations using lstms. In David Blei and Francis Bach, editors, *Proceedings of the 32nd ICML-15*, 2015.

[108] Anthony Stathopoulos and Matthew G Karlaftis. A multivariate state space approach for urban traffic flow modeling and prediction. *Transportation Research Part C : Emerging Technologies*, 11(2) :121–135, 2003.

[109] Michael L Stein. *Interpolation of spatial data : some theory for kriging*. Springer Science & Business Media, 2012.

[110] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *Proceedings of the 30th ICML*, 2013.

[111] Ilya Sutskever, James Martens, and Geoffrey E. Hinton. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning, ICML 2011*, 2011.

[112] Zaiyong Tang, Chrys de Almeida, and Paul A Fishwick. Time series forecasting using neural networks vs. box-jenkins methodology. *Simulation*, 57(5) :303–310, 1991.

[113] Howell Tong. *Non-linear time series : a dynamical system approach*. 1990.

[114] Luke Vilnis and Andrew McCallum. Word representations via gaussian embedding. *ICLR*, 2015.

[115] Taras K Vintsyuk. Speech discrimination by dynamic programming. *Cybernetics and Systems Analysis*, 4(1) :52–57, 1968.

[116] Oriol Vinyals, Charles Blundell, Tim Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems*, pages 3630–3638, 2016.

[117] Xiao-Wei Wang, Dan Nie, and Bao-Liang Lu. Emotional state classification from eeg data using machine learning approach. *Neurocomputing*, 129 :94–106, 2014.

[118] Swarna Weerasinghe. A missing values imputation method for time series data : an efficient method to investigate the health effects of sulphur dioxide levels. *Environmetrics*, 21(2) :162–172, 2010.

[119] Li Wei, Nitin Kumar, Venkata Nishanth Lolla, Eamonn J Keogh, Stefano Lonardi, and Chotirat (Ann) Ratanamahatana. Assumption-free anomaly detection in time series. In *SSDBM*, volume 5, pages 237–242, 2005.

[120] Christopher K. Wikle. Modern perspectives on statistics for spatio-temporal data. *Wiley Interdisciplinary Reviews : Computational Statistics*, 7(1) :86–98, 2015.

[121] Christopher K Wikle and Mevin B Hooten. A general science-based framework for dynamical spatio-temporal models. *Test*, 19(3) :417–451, 2010.

[122] Billy Williams, Priya Durvasula, and Donald Brown. Urban freeway traffic flow prediction : application of seasonal autoregressive integrated moving average and exponential smoothing models. *Transportation Research Record : Journal of the Transportation Research Board*, (1644) :132–141, 1998.

[123] Xiaopeng Xi, Eamonn Keogh, Christian Shelton, Li Wei, and Chotirat Ann Ratanamahatana. Fast time series classification using numerosity reduction. In *Proceedings of the 23rd international conference on Machine learning*, pages 1033–1040. ACM, 2006.

[124] SHI Xingjian, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. Convolutional lstm network : A machine learning approach for precipitation nowcasting. In *Advances in Neural Information Processing Systems*, pages 802–810, 2015.

[125] Yimin Xiong and Dit-Yan Yeung. Mixtures of arma models for model-based time series clustering. In *Data Mining, 2002. ICDM 2003. Proceedings. 2002 IEEE International Conference on*, pages 717–720. IEEE, 2002.

[126] Jing Yuan, Yu Zheng, Xing Xie, and Guangzhong Sun. Driving with knowledge from the physical world. In *Proceedings of the 17th ACM SIGKDD*, pages 316–324. ACM, 2011.

[127] Jing Yuan, Yu Zheng, Chengyang Zhang, Wenlei Xie, Xing Xie, Guangzhong Sun, and Yan Huang. T-drive : driving directions based on taxi trajectories. In *Proceedings of the 18th SIGSPATIAL*, pages 99–108. ACM, 2010.

[128] G Peter Zhang. Time series forecasting using a hybrid arima and neural network model. *Neurocomputing*, 50 :159–175, 2003.

[129] Yi Zheng, Qi Liu, Enhong Chen, Yong Ge, and J Leon Zhao. Time series classification using multi-channels deep convolutional neural networks. In *International Conference on Web-Age Information Management*, pages 298–310. Springer, 2014.

[130] Yu Zheng. T-drive trajectory data sample, August 2011.