Analyse spatio-temporelle et modélisation prédictive du réseau de transport public de Montréal

par

Emna BOUDABBOUS

MÉMOIRE PAR ARTICLES PRÉSENTÉ À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
COMME EXIGENCE PARTIELLE À L'OBTENTION DE LA MAÎTRISE AVEC MÉMOIRE EN GÉNIE DE LA PRODUCTION AUTOMATISÉE
M. Sc. A.

MONTRÉAL, LE 11 FÉVRIER 2026

ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

PRÉSENTATION DU JURY

CETTE THÈSE A ÉTÉ ÉVALUÉE

PAR UN JURY COMPOSÉ DE:

M. Julio Montecinos, directeur de mémoire
Département de génie de la production automatisée, École de technologie supérieure

M. Lokman Sboui, codirecteur
Département de génie de la production automatisée, École de technologie supérieure

M. Fabio Petrillo, président du jury
Département de génie de la production automatisée, École de technologie supérieure

Mme. Rim Larbi, membre du jury
Département de génie de la production automatisée, École de technologie supérieure

M. Omar Alam, examinateur externe
Computing and Information Systems, Trent University

ELLE A FAIT L'OBJET D'UNE SOUTENANCE DEVANT JURY ET PUBLIC

LE 16 DÉCEMBRE 2025

À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

REMERCIEMENTS

Je tiens tout d'abord à exprimer ma profonde gratitude à mes directeurs de recherche, M. Julio Montecinos et M. Lokman Sboui, pour leur encadrement rigoureux, leurs conseils judicieux et leur disponibilité constante tout au long de ce projet de maîtrise. Leur vision scientifique et leur expertise ont été déterminantes pour l'orientation et la réussite de ce travail.

Je remercie également les membres du jury pour le temps consacré à l'évaluation de ce mémoire et pour leurs commentaires constructifs, qui ont contribué à en améliorer la qualité.

Je souhaite remercier la Société de transport de Montréal (STM) d'avoir rendu publiques les données GTFS et GTFS-RT qui ont constitué la base empirique de cette étude. L'engagement des agences de transport en faveur de l'ouverture des données (*open data*) constitue une contribution essentielle à l'avancement de la recherche en systèmes de transport intelligents.

Je souhaite exprimer ma reconnaissance la plus sincère à ma famille et à mes proches pour leur soutien inconditionnel, leur patience et leurs encouragements constants tout au long de mes études. Leur confiance a été une source de motivation inestimable.

Enfin, je remercie chaleureusement mes collègues du laboratoire pour leur entraide et l'atmosphère collaborative qui ont rendu cette expérience de recherche à la fois stimulante et agréable.

Ce projet de recherche a bénéficié d'une bourse interne ainsi que d'une exemption des frais de scolarité majorés. Il a également donné lieu aux publications suivantes :

- A Data-Driven Platform for Visualizing and Analyzing Public Transit Schedule Deviations
- TDPS : A Scalable Bus Transit Delay Prediction System with Multi-Resolution Features and Deep Learning

Analyse spatio-temporelle et modélisation prédictive du réseau de transport public de Montréal

Emna BOUDABBOUS

RÉSUMÉ

Les écarts d'horaire dans les réseaux de transport en commun — différences entre les heures d'arrivée planifiées et réelles des véhicules — constituent une source majeure d'insatisfaction des usagers et de perte d'efficacité opérationnelle. La disponibilité croissante de données opérationnelles en temps réel, conforme au standard GTFS-RT, offre des opportunités inédites pour analyser et prédire ces perturbations. Toutefois, un écart persistant subsiste entre la disponibilité des données et leur exploitation effective pour améliorer la qualité de service.

Ce mémoire propose une approche intégrée, structurée en deux volets complémentaires, pour combler cette lacune. Le premier volet développe une plateforme de diagnostic intégrant l'ingestion automatique de flux GTFS-RT massifs, la classification systématique des perturbations selon leur nature, récurrente ou ponctuelle, l'exploitation de l'indexation spatiale hiérarchique H3 pour l'agrégation multi-résolution, ainsi que des capacités de visualisation interactive à haute performance via KeplerGL. Appliquée au réseau de la STM, cette plateforme open source révèle des motifs spatio-temporels significatifs et facilite l'identification de points critiques nécessitant des interventions.

Le second volet développe une approche systématique de prédiction des écarts d'horaire à l'échelle d'un réseau urbain complet. Cette approche exploite (1) une méthodologie structurée d'ingénierie de features multi-résolution basée sur H3, combinant caractéristiques (features) spatiales hiérarchiques, encodages temporels cycliques et embeddings spatiaux appris ; (2) une architecture LSTM optimisée pour capturer les dépendances temporelles complexes tout en respectant les contraintes de scalabilité au moyen d'une stratégie de clustering adaptée ; et (3) une validation empirique rigoureuse avec protocole spatio-temporel sur des périodes indépendantes. Les résultats démontrent des performances prédictives supérieures aux baselines traditionnelles, avec une erreur absolue moyenne inférieure à 2 minutes pour les horizons de prédiction à court terme.

La contribution distinctive de ce mémoire réside dans l'intégration cohérente du diagnostic et de la prédiction au sein d'un cadre méthodologique unifié, où les insights de l'analyse exploratoire informent directement la conception du système prédictif. Cette synergie, combinée à l'accent mis sur la reproductibilité, la scalabilité opérationnelle et l'approche open source, vise à réduire l'écart entre la recherche académique et l'adoption pratique dans le domaine des systèmes de transport intelligents.

Mots-clés: transport en commun, écarts d'horaire, GTFS, apprentissage profond, visualisation, LSTM, indexation spatiale, H3

Analysis and Prediction of Public Transit Schedule Deviations Using Data-Driven Methods

Emna BOUDABBOUS

ABSTRACT

Schedule deviations in public transit networks—discrepancies between planned and actual vehicle arrival times—are a significant source of user dissatisfaction and operational inefficiency. The growing availability of real-time operational data through the GTFS-RT standard offers new opportunities to analyze and predict disruptions. Yet, a significant gap remains between data availability and practical use to improve service quality.

This thesis proposes an integrated, two-component approach to address this gap. The first component develops a diagnostic platform that integrates automatic ingestion of large-scale GTFS-RT streams, systematic classification of disruptions into their recurrent or one-off categories, use of H3 hierarchical spatial indexing for multi-resolution aggregation, and high-performance interactive visualization with KeplerGL. Applied to Montréal's STM network, this open-source platform reveals significant spatio-temporal patterns and supports the identification of critical locations requiring intervention.

The second component develops a systematic approach for predicting schedule deviations at the scale of a city-wide transit network. The proposed method combines (1) a structured framework for multi-resolution feature engineering based on H3, including hierarchical spatial caractéristiques (features), cyclic temporal encodings, and learned embeddings ; (2) an LSTM architecture optimized to capture complex temporal dependencies while satisfying scalability constraints through an appropriate clustering strategy ; and (3) rigorous empirical evaluation using a spatio-temporal validation protocol on independent time periods. The results show predictive performance superior to traditional baselines, with a mean absolute error below two minutes for short-term prediction horizons.

The distinctive contribution of this thesis lies in the coherent integration of diagnosis and prediction within a unified methodological framework, in which insights from exploratory analysis directly inform the design of the predictive system. This synergy, combined with an emphasis on reproducibility, operational scalability, and an open-source implementation, aims to narrow the gap between academic research and practical deployment in intelligent transportation systems.

Keywords: public transit, schedule deviations, GTFS-RT, deep learning, spatial indexing

TABLE DES MATIÈRES

Page

INTRODUCTION 1
0.1 Contexte et motivation 1
0.2 Problématique générale 3
0.3 Questions de recherche 4
0.4 Objectifs du mémoire 5
0.5 Démarche de recherche et structure du mémoire 6

CHAPITRE 1 REVUE DE LITTÉRATURE 9
1.1 Introduction 9
1.2 Fondements conceptuels et technologies habilitantes 11
1.2.1 Standards de données pour les transports en commun 11
1.2.1.1 GTFS : la standardisation des données statiques 11
1.2.1.2 GTFS-RT : l'extension temps réel 12
1.2.1.3 Concept d'écart d'horaire : définitions et enjeux 13
1.2.2 Indexation spatiale et géovisualisation 14
1.2.2.1 Systèmes d'indexation spatiale hiérarchique 14
1.2.2.2 Le système Index spatial hiérarchique hexagonal d'Uber H3 ... 14
1.2.2.3 Technologies de géovisualisation modernes 15
1.2.3 Modèles d'apprentissage pour les séries temporelles 16
1.2.3.1 Réseaux de neurones récurrents (RNN) 16
1.2.3.2 Long Short-Term Memory (LSTM) 17
1.2.3.3 Gated Recurrent Units (GRU) 22
1.2.3.4 Modèles séquence-à-séquence (Seq2Seq) pour prédictions multi-horizon 23
1.2.3.5 Transformers pour séries temporelles 24
1.2.4 Feature engineering pour les séries temporelles de transport 26
1.2.5 Réduction de dimension 26
1.2.5.1 PCA adaptative pour capturer la dynamique des systèmes de transport en temps réel 29
1.2.5.2 PCA supervisée pour optimiser la pertinence aux objectifs de prédiction 30
1.3 État de l'art en analyse diagnostique des écarts d'horaire 32
1.3.1 Métriques classiques et leurs limitations 33
1.3.2 Approches de clustering spatial et classification 33
1.3.3 Visualisation interactive et exploration spatiale 34
1.3.4 Identification des facteurs explicatifs 34
1.3.5 Synthèse des lacunes diagnostiques 35
1.3.6 Synthèse des lacunes prédictives 36
1.4 État de l'art en prédiction des retards 37
1.4.1 Approches statistiques classiques 37

1.4.2 Machine Learning classique 38
1.4.3 Deep Learning : architectures récurrentes 39
1.4.4 Deep Learning : architectures convolutives et Transformers 39
1.4.5 Approches spatio-temporelles : graphes et attention 40
1.4.6 Lacunes en prédiction scalable et déploiement opérationnel 41
1.5 Synthèse générale et identification des lacunes 42
1.5.1 Le paradoxe : disponibilité technologique versus lacunes méthodologiques 42
1.5.2 Le fossé recherche-pratique 43
1.5.3 Justification du projet de recherche 44
1.6 Conclusion 44

CHAPITRE 2 A DATA-DRIVEN PLATFORM FOR VISUALIZING AND ANALYZING PUBLIC TRANSIT SCHEDULE DEVIATIONS 47
2.1 Introduction 48
2.2 Related Work 49
2.3 Methodology 50
2.3.1 Data Collection and Preprocessing 50
2.3.2 Platform Architecture and Data Pipeline 51
2.3.3 Deviation Classification Details 53
2.3.4 Spatial Analysis using H3 53
2.3.4.1 H3 Index Properties 53
2.3.4.2 Resolution Selection Rationale 53
2.3.5 Visualization Framework 54
2.4 Case Study : Montreal City 54
2.4.1 Network-wide Deviation Analysis by Route 54
2.4.2 Temporal Deviation Patterns 56
2.4.3 H3-based Spatial Deviation Analysis 58
2.4.3.1 Corridor Level Analysis (Res. 9) 58
2.4.3.2 Stop Level Analysis (Res. 10) 58
2.5 Discussion 60
2.6 Conclusion and Future Work 61

CHAPITRE 3 SCALABLE TRANSIT DELAY PREDICTION AT CITY SCALE 63
3.1 Introduction 64
3.2 Related Work 70
3.2.1 Transit Delay Prediction 70
3.2.2 Feature Engineering for Transit Delay Prediction 72
3.2.2.1 Critical Gap : Ad-Hoc Feature Selection 73
3.2.3 Scalability Challenges and Distributed Computing 74
3.2.4 Validation Methodologies and Production Deployment 75
3.2.4.1 Evaluation Protocols 75
3.2.4.2 Production Deployment and Reusable Architectures 76
3.2.5 Summary and Research Gaps 77

3.2.5.1 Comparative Analysis of Prior Work 78
3.3 Methodology 79
3.3.1 Data Collection Pipeline 80
3.3.1.1 Trip-level processing 80
3.3.2 Feature Engineering Pipeline 82
3.3.3 Dimensionality Reduction Pipeline 83
3.3.4 Predictive Modeling Pipeline 84
3.3.5 Inference Pipeline 86
3.3.6 Hybrid H3+Topology Spatial Clustering Strategy 87
3.3.7 Hybrid Parallelization Strategy 90
3.3.8 Predictive Models and Architectures 90
3.3.8.1 Hyperparameter Optimization Strategy 91
3.3.9 Evaluation Protocol 92
3.3.9.1 Walk-Forward Temporal Cross-Validation 92
3.3.9.2 Multi-Level Evaluation Strategy 93
3.3.9.3 Feature Importance and Model Comparison 93
3.3.10 Methodological Limitations and Assumptions 94
3.4 Implementation Details 96
3.4.1 Technology Stack 96
3.4.2 Clustering Implementation Details 96
3.4.2.1 Optimal Clustering Configuration 96
3.4.2.2 Three-Stage Clustering Algorithm Implementation 97
3.4.3 Data Processing Pipelines Implementation 100
3.4.3.1 Collection Pipeline Details 100
3.4.3.2 Feature Engineering Implementation 101
3.4.3.3 Inference Pipeline Implementation 103
3.4.4 Predictive Modeling 104
3.4.4.1 Model Architecture Implementations 104
3.4.4.2 Dimensionality Reduction Implementation 106
3.4.4.3 Hyperparameter Optimization Procedure 107
3.4.4.4 Temporal Cross-Validation Implementation 109
3.4.4.5 Feature Importance Analysis Implementation 109
3.4.4.6 Model Comparison Implementation 110
3.5 Experimental Results and Discussion 111
3.5.1 Experimental Results 111
3.5.1.1 Experimental Setup and Data Organization 112
3.5.1.2 Clustering Results 112
3.5.1.3 Optimal Model Configurations 114
3.5.1.4 Global Model Performance Comparison 116
3.5.2 Non-Functional Requirements and Architecture Support 122
3.5.2.1 Performance Requirements 122
3.5.2.2 Scalability Requirements 123
3.5.2.3 Maintainability Requirements 123

3.5.2.4 Reliability Considerations 124
3.5.2.5 Resource Efficiency 125
3.5.3 Discussion 125
3.5.4 Summary of Key Findings 125
3.5.5 Limitations and Generalizability 127
3.6 Conclusion 128
3.6.1 Technical Contributions 128
3.6.2 Operational Applications 129
3.6.3 Broader Impact 129

CONCLUSION ET RECOMMANDATIONS 131
4.1 Synthèse : un cadre intégré pour comprendre et anticiper 131
4.2 Implications théoriques et pratiques 132
4.3 Limitations et perspectives critiques 132
4.4 Perspectives de recherche futures 133
4.5 Mot de la fin : vers une mobilité urbaine véritablement intelligente 134

BIBLIOGRAPHIE 135

LISTE DES FIGURES

Page

Figure 0.1 Vue d'ensemble simplifiée de l'architecture intégrée 4

Figure 0.2 Architecture détaillée du système intégré montrant les interactions entre les deux articles 7

Figure 1.1 Architecture générique d'un réseau de neurones récurrent (RNN) unidirectionnel montrant la boucle de rétroaction permettant de traiter les séquences temporelles 17

Figure 1.2 Architecture d'une cellule LSTM montrant les trois portes (d'oubli, d'entrée, de sortie) et la cellule mémoire qui contrôlent le flux d'information à travers le temps 17

Figure 1.3 Architecture d'un LSTM empilé montrant comment les sorties d'une couche alimentent la couche suivante, créant une hiérarchie de représentations temporelles 19

Figure 1.4 Architecture d'un LSTM bidirectionnel montrant les deux flux d'information (avant et arrière) qui sont ensuite combinés pour chaque pas de temps 20

Figure 1.5 Schéma d'un LSTM avec mécanisme d'attention montrant la pondération adaptative des états cachés passés selon leur pertinence pour la prédiction courante 21

Figure 1.6 Architecture d'une unité récurrente à portes (GRU) montrant les deux portes de contrôle (mise à jour et réinitialisation) et l'état unique maintenu par l'unité 23

Figure 1.7 Architecture encodeur–décodeur d'un modèle séquence-à-séquence (Seq2Seq) basé sur des LSTM ou GRU. L'encodeur traite la séquence historique et produit un vecteur de contexte, que le décodeur utilise pour générer les prédictions futures multi-pas 24

Figure 1.8 Schéma général d'un Transformer appliqué aux séries temporelles, montrant l'empilement de blocs composés d'auto-attention multi-têtes et de couches feedforward avec connexions résiduelles et normalisation 25

Figure 2.1 Main data processing flow. 51

Figure 2.2 Stop-Level Deviation Classification by Route (Sorted by descending % of Late + Too Late). Highlights routes needing potential schedule/operational review. 55

Figure 2.3 Segment-Level Deviation Classification by Route (Sorted similarly to Fig. 2.2). Shows inter-stop travel performance. 56

Figure 2.4 Average Segment-Level Deviation Distribution by Hour of Day. Shows clear peak hour impacts. 57

Figure 2.5 Spatial Distribution of Average Stochastic Delays (H3 Res. 9). Left : Segment-Level Delays. Right : Stop-Level Delays. Color scale indicates average delay (seconds). Highlights difference between corridor issues (left) and accumulated stop impact (right). 58

Figure 2.6 Spatial Distribution of Average Stop-Level Schedule Deviations (H3 Res. 10). Color scale indicates average deviation (seconds). Pinpoints specific problematic locations. 59

Figure 3.1 System architecture : five pipelines with external data sources and storage. 80

Figure 3.2 Data Collection Pipeline : external APIs to storage. 81

Figure 3.3 Feature Engineering Pipeline : trip records to feature matrix. 83

Figure 3.4 Dimensionality Reduction : 1,683 features to 83 components (Adaptive PCA). 84

Figure 3.5 Predictive Modeling Pipeline : global model training. 86

Figure 3.6 Inference Pipeline : hierarchical aggregation (elementary â†' segment â†' trip). 87

Figure 3.7 Three-stage hybrid clustering process 89

Figure 3.8 Model comparison : RMSE and R^2 by architecture. 118

Figure 3.9 Error distribution : histogram (left) and cumulative distribution (right). 118

Figure 3.10 Predicted vs actual delay at segment level : distributions (left) and scatter plot (right). 119

Figure 3.11 Predicted vs actual delay at trip level : distributions (left) and scatter plot (right). 120

Figure 3.12 Trip-level analysis : error distribution, histograms, and scatter plot. 120

Figure 3.13 Multi-level metrics : RMSE, MAE, and R^2 across aggregation levels. ..121

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

APC	Comptage Automatique de Passagers (*Automatic Passenger Counting*)
AVL	Localisation Automatique de Véhicules (*Automatic Vehicle Location*)
CAD	Répartition Assistée par Ordinateur (*Computer-Aided Dispatch*)
GTFS	Spécification Générale des Flux de Transports (*General Transit Feed Specification*)
GTFS-RT	GTFS Temps Réel (*GTFS Real-Time*)
H3	Système d'Indexation Géospatiale Hiérarchique Hexagonale (*Hierarchical Hexagonal Geospatial Indexing System*)
SIG	Système d'Information Géographique
STM	Société de transport de Montréal
STI	Systèmes de Transport Intelligents
AI	Intelligence Artificielle (*Artificial Intelligence*)
IA	Intelligence Artificielle
ML	Apprentissage Automatique (*Machine Learning*)
DL	Apprentissage Profond (*Deep Learning*)
RNN	Réseau de Neurones Récurrent (*Recurrent Neural Network*)
LSTM	Mémoire Court-Terme à Long Terme (*Long Short-Term Memory*)
BiLSTM	LSTM Bidirectionnel (*Bidirectional LSTM*)
SLSTM	LSTM Empilé (*Stacked LSTM*)
Att-LSTM	LSTM avec Mécanisme d'Attention (*Attention-based LSTM*)
GRU	Unité Récurrente à Porte (*Gated Recurrent Unit*)
Seq2Seq	Modèle Séquence-à-Séquence (*Sequence-to-Sequence Model*)
CNN	Réseau de Neurones Convolutif (*Convolutional Neural Network*)
ConvLSTM	LSTM Convolutif (*Convolutional LSTM*)

AE	Autoencodeur (*Autoencoder*)
VAE	Autoencodeur Variationnel (*Variational Autoencoder*)
GNN	Réseau de Neurones sur Graphes (*Graph Neural Network*)
GCN	Réseau Convolutif sur Graphes (*Graph Convolutional Network*)
GAT	Transformateur sur Graphes (*Graph Attention Network*)
STGCN	Réseau Convolutif Spatio-Temporel (*Spatio-Temporal Graph Convolutional Network*)
DCRNN	Réseau Récurrent à Convolution de Diffusion (*Diffusion Convolutional Recurrent Neural Network*)
TGCN	Réseau Temporel à Convolution sur Graphes (*Temporal Graph Convolutional Network*)
AGCRN	Réseau Récurrent sur Graphes à Structure Adaptative (*Adaptive Graph Convolutional Recurrent Network*)
ViT	Transformeur Visuel (*Vision Transformer*)
TST	Transformeur pour Séries Temporelles (*Time Series Transformer*)
Informer	Transformeur Efficace de Séries Temporelles à Attention ProbSparse
Autoformer	Modèle Autoformer pour Séries Temporelles
FEDformer	Frequency Enhanced Decomposition Transformer
PatchTST	Transformeur de Séries Temporelles à Patches (*Patch-based Time Series Transformer*)
PCA	Analyse en Composantes Principales (*Principal Component Analysis*)
SPCA	Analyse en Composantes Principales Supervisée (*Supervised PCA*)
APCA	PCA Adaptative (*Adaptive PCA*)
LDA	Analyse Discriminante Linéaire (*Linear Discriminant Analysis*)
t-SNE	Stochastic Neighbor Embedding t-distribué

UMAP	Approximation Uniforme de Variété et Projection (*Uniform Manifold Approximation and Projection*)
MAE	Erreur Absolue Moyenne (*Mean Absolute Error*)
RMSE	Racine de l'Erreur Quadratique Moyenne (*Root Mean Square Error*)
MAPE	Erreur Absolue Moyenne en Pourcentage (*Mean Absolute Percentage Error*)
CSV	Valeurs Séparées par des Virgules (*Comma-Separated Values*)
WebGL	Bibliothèque Graphique Web (*Web Graphics Library*)
ProtoBuf	Format de Sérialisation Protocol Buffers
API	Interface de Programmation d'Applications (*Application Programming Interface*)
ETS	École de Technologie Supérieure
M.Sc.A.	Maîtrise en Sciences Appliquées
TDPS	Système de prédiction des retards de transport collectif (*Transit Delay Prediction System*)
OTP	Performance de Ponctualité (*On-Time Performance*)
SVM	Machine à Vecteurs de Support (*Support Vector Machine*)
SVR	Régression par Vecteurs de Support (*Support Vector Regression*)
XGBoost	Gradient Boosting Extrême (*Extreme Gradient Boosting*)

LISTE DES SYMBOLES ET UNITÉS DE MESURE

Δt	Écart d'horaire (différence entre heure réelle et heure planifiée)
$t_{\text{réel}}$	Heure d'arrivée/départ réelle d'un véhicule
$t_{\text{planifié}}$	Heure d'arrivée/départ planifiée selon GTFS
L	Longueur de la fenêtre historique (nombre de pas de temps en entrée)
H	Horizon de prédiction (nombre de pas de temps à prédire)
T	Nombre total de pas de temps dans une séquence
x_t	Vecteur d'entrée (features) à l'instant t
$\mathbf{x}$	Vecteur ou matrice d'entrée globale (série temporelle ou batch)
y_t	Valeur cible (par exemple écart d'horaire) à l'instant t
$\hat{y}_t$	Valeur prédite de la cible à l'instant t
$\mathbf{y}$	Vecteur des valeurs cibles
$\hat{\mathbf{y}}$	Vecteur des valeurs prédites
h_t	État caché d'un réseau de neurones récurrent à l'instant t
c_t	État de cellule d'un LSTM à l'instant t
f_t	Porte d'oubli (*forget gate*) dans un LSTM
i_t	Porte d'entrée (*input gate*) dans un LSTM
o_t	Porte de sortie (*output gate*) dans un LSTM
z_t	Porte de mise à jour (*update gate*) dans un GRU
r_t	Porte de réinitialisation (*reset gate*) dans un GRU
$\mathbf{f}$	Vecteur de features (caractéristiques)
W	Matrice de poids d'un réseau de neurones
$\mathbf{W}$	Ensemble des matrices de poids d'un modèle
b	Biais (scalaire ou vecteur)
σ	Fonction d'activation sigmoïde
tanh	Fonction d'activation tangente hyperbolique

$\phi(\cdot)$	Fonction d'activation générique (ReLU, LeakyReLU, etc.)
$\mathcal{G}$	Graphe de transport (par exemple réseau d'arrêts ou de cellules H3)
V	Ensemble des nœuds du graphe
E	Ensemble des arêtes du graphe
A	Matrice d'adjacence du graphe
$\tilde{A}$	Matrice d'adjacence normalisée
D	Matrice de degré du graphe
$\mathbf{X}$	Matrice des features des nœuds (une ligne par nœud)
$\mathbf{H}$	Matrice des représentations cachées des nœuds
$\mathcal{N}(v)$	Ensemble des voisins du nœud v
$h_k(p)$	Identifiant de la cellule H3 de résolution k contenant le point p
k	Niveau de résolution H3
Ω_k	Ensemble des cellules H3 à la résolution k
$\mathbf{X}$	Matrice de données (observations × variables)
$\mathbf{Z}$	Matrice des scores (données projetées dans l'espace des composantes principales)
$\mathbf{W}_{\text{PCA}}$	Matrice de projection PCA (vecteurs propres)
λ_i	i-ème valeur propre de la matrice de covariance
Λ	Matrice diagonale des valeurs propres
$\ell(\cdot)$	Fonction de perte (par exemple MAE, RMSE)
N	Nombre d'observations (ou de paires prédites/observées)
MAE	Erreur Absolue Moyenne
RMSE	Racine de l'Erreur Quadratique Moyenne
MAPE	Erreur Absolue Moyenne en Pourcentage

INTRODUCTION

0.1 Contexte et motivation

Les systèmes de transport collectif urbain occupent une place stratégique dans le fonctionnement des métropoles contemporaines. Ils contribuent non seulement à la mobilité quotidienne des populations, mais aussi à la réduction de la congestion routière, des émissions de gaz à effet de serre et à l'amélioration de l'accessibilité aux emplois et aux services essentiels. Dans un contexte de transition écologique et de croissance urbaine, renforcer l'attractivité et la performance du transport collectif constitue ainsi un enjeu prioritaire pour les autorités publiques.

Parmi les divers indicateurs de performance, la **fiabilité du service** — c'est-à-dire la capacité des autobus à respecter les horaires planifiés et à offrir des temps de parcours prévisibles — est reconnue comme un déterminant clé de la satisfaction des usagers et de l'achalandage. De nombreux travaux ont mis en évidence que les irrégularités de service se traduisent par des temps d'attente accrus, une variabilité importante des temps de parcours et, ultimement, une perte de confiance envers le réseau de transport collectif Chen, Yu, Zhang & Guo (2009); Tétreault & El-Geneidy (2010). Dans les réseaux d'autobus, les retards et les variations de fréquence peuvent également nuire à la synchronisation intermodale (métro, lignes structurantes) et compromettre l'efficacité globale du système, incitant les usagers à se tourner vers des modes de déplacement privés.

À Montréal, la Société de transport de Montréal (STM) opère un réseau dense d'autobus qui constitue l'épine dorsale du transport collectif de surface. La ponctualité et la régularité de ces services sont influencées par plusieurs facteurs : la congestion routière, des conditions météorologiques extrêmes (neige, verglas, froid intense), la géométrie du réseau viaire, ainsi que les opérations d'exploitation (gestion des terminus, prises et fins de service, correspondances avec le métro, etc.). Les gestionnaires font donc face à un besoin croissant d'outils capables de

diagnostiquer finement les écarts d'horaire observés et d'anticiper les perturbations afin d'ajuster l'offre en quasi-temps réel — que ce soit par des ajustements de fréquence, des itinéraires alternatifs ou une meilleure information des voyageurs.

L'essor des données massives et des standards ouverts tels que la *General Transit Feed Specification* (GTFS) et son extension en temps réel (GTFS-RT) a considérablement transformé l'accès aux données de transport collectif GTFS Realtime Consortium (2023). Ces flux, mis à jour en continu, fournissent des informations précieuses sur les positions des véhicules, les prédictions d'arrivée et les alertes de service, constituant ainsi une base riche, dynamique et peu coûteuse pour l'analyse et la modélisation des retards — auparavant réservée aux grandes agences dotées d'infrastructures propriétaires.

Parallèlement, les avancées rapides en intelligence artificielle (IA) et en apprentissage profond ont élargi les possibilités de prédiction des temps de parcours et d'arrivée dans les réseaux de transport Singh & Kumar (2022a); Wang, Zhang, Cao, Li & Zheng (2018); Marković, Milinković, Tikhonov & Schonfeld (2015); Jevinger, Zhao, Persson & Davidsson (2024). Les modèles tels que les réseaux de neurones récurrents (LSTM, GRU), les architectures spatio-temporelles convolutionnelles et les transformers permettent de modéliser des dépendances complexes dans le temps et l'espace, en intégrant des facteurs tels que la congestion, la signalisation ou les conditions climatiques.

Cependant, une part importante de la littérature reste limitée à des tronçons spécifiques du réseau (une ligne, un corridor, un sous-ensemble de trajets) et n'aborde pas explicitement les défis de la **scalabilité** à l'échelle d'une ville entière. Dans ce contexte, l'utilisation de grilles spatiales hiérarchiques, telles que l'index H3, qui discrétise la surface terrestre en cellules hexagonales multirésolution, offre une approche prometteuse pour structurer l'information spatio-temporelle de manière homogène et extensible Brodsky (2018b). Combinée aux flux GTFS/GTFS-RT et aux modèles d'apprentissage profond, cette représentation ouvre la voie à des systèmes de prédiction

de retards véritablement *scalable*, déployables à l'échelle du réseau, une ingénierie majeure pour chaque secteur.

0.2 Problématique générale

La problématique générale de ce mémoire porte sur la compréhension et la prédiction des écarts d'horaire dans un réseau urbain dense d'autobus. Bien que les données opérationnelles disponibles soient de plus en plus riches, plusieurs défis persistent :

- caractériser systématiquement les écarts à l'horaire à l'échelle du réseau complet (toutes lignes, arrêts, périodes) afin d'identifier les zones et contextes de fiabilité réduite ;
- relier ces écarts à des facteurs explicatifs multiples (géométrie, circulation, météo, structure de l'horaire, opérations) pour en diagnostiquer les causes profondes ;
- concevoir des modèles prédictifs capables d'anticiper les retards en quasi-temps réel, avec une précision et une robustesse suffisantes pour une utilisation opérationnelle.

La prédiction des retards à grande échelle soulève plusieurs enjeux méthodologiques et computationnels. D'une part, la dimensionnalité des données est élevée : un réseau métropolitain génère des milliers d'observations en continu. D'autre part, les phénomènes à modéliser sont hautement spatio-temporels : un retard dépend de l'état du corridor amont, de la dynamique passagère, des décisions d'exploitation et des conditions environnementales. Enfin, les modèles doivent être suffisamment légers pour être exécutés à une fréquence élevée (plusieurs fois par minute), ce qui exclut les architectures trop complexes.

Dans ce cadre, ce mémoire vise à explorer à la fois (i) l'analyse descriptive et la visualisation des écarts d'horaire à l'échelle du réseau STM, et (ii) la conception d'une architecture de prédiction *scalable*, fondée sur l'indexation spatiale H3 et des modèles d'apprentissage profond adaptés aux séries temporelles. Les enjeux clés peuvent être résumés ainsi : (a) amélioration de la

précision par rapport aux approches classiques, (b) scalabilité sans reconfiguration spécifique, (c) intégration fluide aux flux en temps réel, et (d) utilité opérationnelle concrète.

La Figure 0.1 présente une vue d'ensemble de l'approche intégrée proposée dans ce mémoire, illustrant le flux principal depuis les sources de données externes jusqu'aux applications opérationnelles, en passant par les deux volets complémentaires de diagnostic (Article 1) et de prédiction (Article 2).

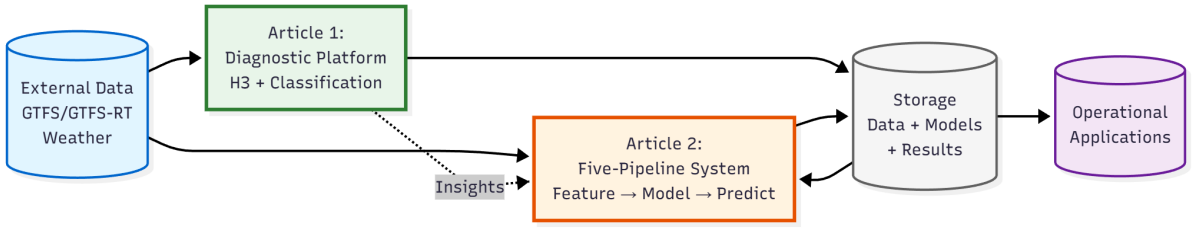

Figure 0.1 Vue d'ensemble simplifiée de l'architecture intégrée

0.3 Questions de recherche

À partir de cette problématique, quatre questions de recherche guident ce mémoire :

Q1. Comment caractériser, mesurer et visualiser systématiquement les écarts d'horaire des autobus à l'échelle du réseau STM, à l'aide des flux GTFS/GTFS-RT et d'une indexation spatiale multi-résolution (H3), afin d'identifier les zones et périodes critiques ?

Q2. Comment concevoir une architecture de prédiction des retards d'autobus réellement *scalable* à l'échelle urbaine, tenant compte du volume de données, de la granularité spatiale et de la fréquence de mise à jour, tout en respectant les contraintes de latence ?

Q3. Dans quelle mesure la combinaison de l'indexation H3 et des modèles d'apprentissage profond (xLSTM, transformers spatio-temporels) améliore-t-elle la robustesse et la précision des prédictions par rapport à des approches traditionnelles (boosting, régressions) ?

Q4. Quels usages opérationnels concrets ces outils permettent-ils (diagnostic, information du voyageur, gestion en temps réel), et quelles sont les limites et les besoins en données à combler avant un déploiement complet ?

0.4 Objectifs du mémoire

L'objectif général du mémoire est de développer une approche intégrée s'appuyant sur les flux GTFS/GTFS-RT et l'apprentissage profond pour analyser et prédire les retards à l'échelle du réseau STM, en valorisant l'indexation spatiale H3, la scalabilité et la pertinence opérationnelle.

Quatre objectifs spécifiques le déclinent :

- **O1 :** Développer une plateforme de visualisation et d'analyse des écarts d'horaire à partir des flux STM, en projetant le réseau sur une grille H3 multirésolution pour faciliter l'agrégation et la comparaison à différentes échelles.
- **O2 :** Mettre en place un pipeline de prédiction intégrant : (a) une ingénierie de caractéristiques spatio-temporelles adaptées à H3, (b) des techniques de réduction de dimensionnalité, et (c) des modèles profonds (LSTM/xLSTM, transformers) et classiques (XGBoost) pour les séries temporelles.
- **O3 :** Évaluer rigoureusement ces modèles via une validation temporelle stricte, en tenant compte des particularités opérationnelles (pointe, hors-pointe, météo, lignes locales vs lignes structurantes).
- **O4 :** Documenter l'utilité opérationnelle des résultats obtenus, identifier des cas d'usage réalistes et discuter des limitations pratiques ainsi que des lacunes de données à combler avant une intégration en production.

0.5 Démarche de recherche et structure du mémoire

La démarche adoptée est empirique et *data-driven*, reposant sur l'exploitation conjointe des flux GTFS/GTFS-RT de la STM, collectés entre septembre et décembre 2024 et projetés sur une grille H3 couvrant l'agglomération montréalaise. Deux contributions principales structurent le mémoire sous la forme d'articles scientifiques intégrés.

Le **premier article** (Chapitre 3) décrit le développement d'une plateforme interactive de visualisation et d'analyse des écarts d'horaire, incluant l'extraction, la validation, la projection H3 et le calcul d'indicateurs à plusieurs échelles. Il répond à la question Q1 et sert de fondement au volet prédictif.

Le **deuxième article** (Chapitre 4) présente la conception et l'évaluation d'architectures de prédiction des retards, en comparant des approches classiques et profondes à l'aide de séries temporelles construites sur H3, en réponse aux questions Q2 et Q3.

La Figure 0.2 détaille l'architecture technique complète du système intégré.

La structure du mémoire est la suivante :

- **Chapitre 1 :** Introduction générale, problématique et questions de recherche.
- **Chapitre 2 :** Revue de la littérature sur la fiabilité, les flux GTFS/GTFS-RT, H3 et les modèles prédictifs pour les données spatio-temporelles.
- **Chapitre 3 :** Article 1 — plateforme d'analyse des écarts d'horaire.
- **Chapitre 4 :** Article 2 — prédiction *scalable* des retards à l'échelle urbaine.
- **Chapitre 5 :** Conclusion générale, synthèse des résultats, implications, perspectives de recherche et de déploiement.

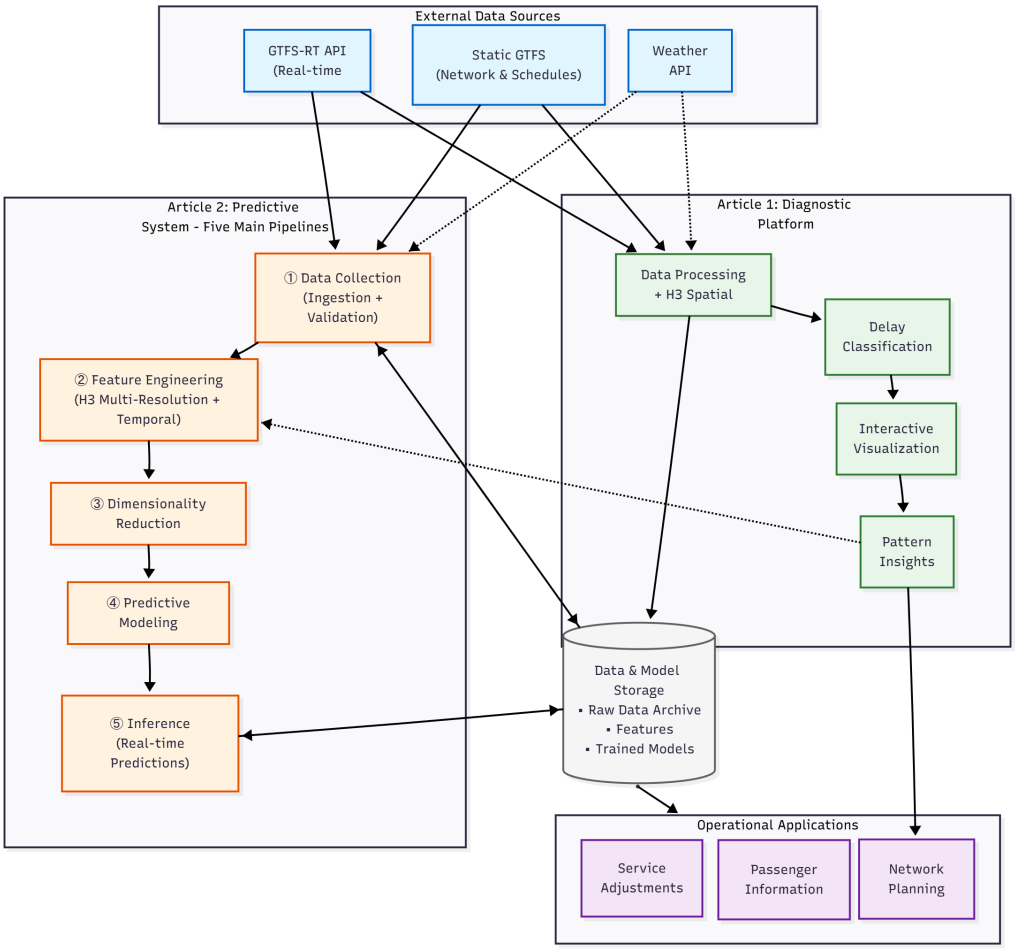

Figure 0.2 Architecture détaillée du système intégré montrant les interactions entre les deux articles

Les sections préliminaires comprennent un résumé, une table des matières, ainsi qu'une liste des figures et des tableaux. Les sections finales regroupent la bibliographie, des annexes techniques détaillant les protocoles expérimentaux, ainsi que des résultats complémentaires (validation croisée, analyses de sensibilité, sous-groupes).

CHAPITRE 1

REVUE DE LITTÉRATURE

1.1 Introduction

La fiabilité des services de transport en commun constitue un déterminant fondamental du bien-être urbain et de la qualité de vie des citadins. Avec l'urbanisation croissante — plus de 55 % de la population mondiale vit désormais en milieu urbain, une proportion appelée à atteindre 68 % d'ici 2050 (United Nations, Department of Economic and Social Affairs, Population Division, 2023) — les systèmes de transport collectif subissent une pression sans précédent. Ces réseaux doivent simultanément améliorer la couverture, augmenter la fréquence des services et assurer la ponctualité des trajets pour maintenir la confiance des usagers et encourager le report modal depuis l'automobile.

Dans ce contexte, les **écarts d'horaire** — définis comme les différences entre les heures d'arrivée ou de départ planifiées et les heures réelles observées — émergent comme une source majeure d'insatisfaction et de perte d'efficacité opérationnelle. Au-delà des inconvénients immédiats subis par les usagers — temps d'attente prolongés, correspondances manquées, stress temporel — les écarts d'horaire génèrent des coûts systémiques considérables : réduction de l'attractivité relative du transport collectif face à l'automobile, saturation accrue des infrastructures lors de perturbations d'amont, perte de productivité pour les passagers professionnels et incompatibilité avec les engagements commerciaux des agences de transport.

Pendant des décennies, l'analyse des écarts d'horaire reposait sur des approches fragmentaires : comptages manuels ponctuels, enquêtes de satisfaction périodiques, rapports d'incidents sans données systématiques. Cette approche offrait une compréhension limitée et rétrospective, inadéquate pour assurer une gestion véritablement proactive des réseaux. La situation a radicalement changé au cours des quinze dernières années. L'adoption généralisée du standard **General Transit Feed Specification** (GTFS) pour les données statiques et son extension **GTFS-RT** pour les données en temps réel ont transformé la disponibilité d'informations

opérationnelles détaillées. Parallèlement, les progrès spectaculaires en apprentissage automatique et en apprentissage profond ont ouvert de nouvelles possibilités pour analyser et modéliser des phénomènes complexes et non linéaires.

Paradoxalement, malgré cette convergence des opportunités technologiques, un écart persistant subsiste entre la disponibilité croissante des données et leur exploitation effective pour améliorer la qualité de service. Les agences de transport, même les plus sophistiquées, continuent largement de s'appuyer sur des méthodes fragmentées et ad hoc, manquant d'un cadre méthodologique intégré capable de transformer les données brutes en connaissances exploitables et en actions concrètes.

Cette revue de littérature vise à cartographier de manière critique l'état des connaissances sur deux dimensions fondamentales : (1) **l'analyse diagnostique** des écarts d'horaire — comment comprendre les patrons spatio-temporels et identifier les facteurs explicatifs sous-jacents — et (2) **la prédiction** des écarts d'horaire — comment anticiper les perturbations futures afin de permettre une gestion proactive. En examinant l'évolution des technologies habilitantes, des approches analytiques et des cas d'application, cette revue identifie les lacunes persistantes qui justifient une approche intégrée du diagnostic à la prédiction.

Objectifs et organisation de la revue :

Cette revue poursuit trois objectifs majeurs. Premièrement, elle établit les **fondements conceptuels et technologies habilitantes** qui ont rendu possibles les avancées récentes : les standards de données GTFS et GTFS-RT, les systèmes d'indexation spatiale hiérarchique comme H3, et les architectures d'apprentissage profond pour les séries temporelles. Deuxièmement, elle cartographie l'**état de l'art en analyse diagnostique**, en examinant comment la recherche a progressé pour caractériser les écarts d'horaire, identifier leurs facteurs explicatifs et visualiser leurs motifs complexes. Troisièmement, elle synthétise l'**état de l'art en prédiction**, en retraçant l'évolution des approches statistiques traditionnelles vers les méthodes contemporaines d'apprentissage automatique et d'apprentissage profond.

L'organisation suit une logique progressive. La Section 1.2 examine les technologies habilitantes. La Section 1.3 analyse l'évolution des approches diagnostiques. La Section 1.4 synthétise les stratégies de prédiction. La Section 1.5 identifie les lacunes critiques et situe la recherche proposée. Enfin, la Section 3.6 conclut en soulignant les implications pour la transition du diagnostic à la prédiction dans un cadre méthodologique unifié.

1.2 Fondements conceptuels et technologies habilitantes

L'émergence récente d'une nouvelle génération d'outils analytiques et technologiques a transformé le paysage de la recherche sur les systèmes de transport. Cette section examine trois piliers qui ont rendu possible l'approche proposée : les standards de données normalisés, les systèmes d'indexation spatiale et les architectures d'apprentissage profond.

1.2.1 Standards de données pour les transports en commun

1.2.1.1 GTFS : la standardisation des données statiques

Avant l'émergence des standards de données, chaque agence de transport conservait ses informations selon ses propres conventions, ce qui entraînait une fragmentation inefficace et limitait la recherche universitaire. Le **General Transit Feed Specification** (GTFS), développé initialement par Google en collaboration avec l'agence de transport de Portland (Oregon) en 2005, a résolu ce problème en définissant un format commun, flexible et extensible pour les données statiques de transport (McHugh, 2013).

Le format GTFS repose sur un ensemble de fichiers CSV liés qui décrivent l'offre de transport : arrêts (localisation géographique, accessibilité), lignes (identifiant, nom, type de service), trajets (patrons de passages à travers les arrêts) et horaires (heures de départ et d'arrivée planifiées pour chaque arrêt d'un trajet donné). Cette structure hiérarchique permet une représentation flexible des réseaux de transport urbains, régionaux et interurbains. L'adoption massive de GTFS — plus

de 1 500 agences en 2024 publiant leurs données dans ce format — a créé un écosystème de données accessible et interopérable (Antrim & Barbeau, 2013).

Au-delà de la standardisation, GTFS a catalysé l'émergence d'un écosystème d'outils et de services : applications mobiles de navigation, plateformes d'information pour voyageurs, systèmes de planification de trajets et infrastructures de recherche. Cette dynamique de croissance auto-renforçante a établi GTFS comme l'infrastructure de données de facto pour les transports urbains.

1.2.1.2 GTFS-RT : l'extension temps réel

L'introduction, en 2011, du **GTFS-RT**, une extension en temps réel de GTFS, a marqué une transition fondamentale dans la disponibilité des données. Alors que GTFS fournit des informations statiques et planifiées, GTFS-RT diffuse en temps réel les mises à jour opérationnelles : les positions actuelles des véhicules (latitude, longitude), les retards observés au moment du passage à chaque arrêt (champ `schedule_deviation` en secondes) et les alertes concernant les interruptions de service (GTFS Realtime Consortium, 2023).

GTFS-RT utilise le format Protocol Buffers, un format de sérialisation binaire développé par Google, offrant une compacité et une efficacité supérieures aux formats textuels traditionnels, ce qui constitue un atout critique pour la distribution de flux de données à haute fréquence. Les agences publiant des données GTFS-RT fournissent généralement des mises à jour toutes les 5 à 30 secondes, créant ainsi un flux continu d'informations opérationnelles détaillées.

Cette disponibilité en temps réel des données a transformé le paysage de la recherche. Plutôt que de s'appuyer sur des études rétrospectives fondées sur des sondages ou des données archivées, les chercheurs peuvent désormais accéder à des historiques complets de performance, ce qui permet d'analyser en détail les schémas d'écarts d'horaire. La transparence des données a également exercé une pression positive sur la qualité : les agences reconnaissent l'importance de maintenir des flux GTFS-RT complets et précis pour soutenir l'écosystème d'applications.

1.2.1.3 Concept d'écart d'horaire : définitions et enjeux

Le concept d'« écart d'horaire » (*schedule deviation*), également appelé « retard » (*delay*), « avance » (*early arrival*) ou, plus généralement, « perturbation » (*disruption*), mérite une clarification terminologique. Formellement, pour un arrêt donné et un trajet donné, l'écart d'horaire est défini comme la différence entre l'heure réelle d'arrivée/départ et l'heure planifiée :

$$\text{Écart d'horaire} = t_{\text{réel}} - t_{\text{planifié}} . \tag{1.1}$$

Les écarts positifs correspondent à des retards (véhicule en retard par rapport au calendrier), tandis que les écarts négatifs correspondent à des avances (véhicule en avance). Par convention, l'analyse se concentre généralement sur les retards positifs, car ce sont les avances qui constituent une perturbation pour les usagers.

Au-delà de cette définition simple réside une complexité conceptuelle importante. Les chercheurs et praticiens distinguent progressivement deux natures d'écarts (Wessel & Widener, 2017b; Aemmer, Ranjbari & MacKenzie, 2022a) :

- **Écarts récurrents (systématiques)** : retards prévisibles et consistants, résultant de facteurs structurels — congestion systématique à certaines heures et lieux, conception horaire trop optimiste, charge passagère habituelle. Ces écarts reflètent une inadéquation entre l'offre planifiée et la demande, ou entre les conditions du réseau.
- **Écarts ponctuels (stochastiques)** : perturbations imprévisibles et non récurrentes, résultant d'événements exceptionnels — accidents routiers, incidents mécaniques, perturbations du réseau électrique du métro, événements civiques. Ces écarts relèvent de l'aléa opérationnel inévitable.

Cette distinction, bien qu'intuitive, est stratégiquement fondamentale. Les écarts récurrents signalent des problèmes structurels nécessitant des interventions de fond : révision de l'horaire, amélioration de l'infrastructure, modification des tracés. Les écarts ponctuels, à l'inverse, exigent des réponses opérationnelles immédiates : régulation du service, communication aux usagers,

mobilisation de ressources de secours. Confondre ces deux catégories conduit à des interventions inefficaces et coûteuses.

1.2.2 Indexation spatiale et géovisualisation

1.2.2.1 Systèmes d'indexation spatiale hiérarchique

Les données de transport présentent une structure spatiale intrinsèque : arrêts à des localités précises, trajets suivant des corridors géographiques, perturbations manifestant des dépendances spatiales (congestion propagée entre arrêts consécutifs). Tirer parti de cette structure spatiale pour l'analyse et la prédiction requiert des représentations spatiales appropriées.

Historiquement, les approches ont recours soit à des coordonnées brutes (latitude, longitude), soit à des zonages administratifs (quartiers, arrondissements). La première approche offre une résolution fine mais manque de sémantique spatiale exploitable par les algorithmes d'apprentissage automatique. La seconde fournit une sémantique claire, mais se heurte à l'arbitraire des frontières administratives, ce qui est insuffisant pour saisir les phénomènes de transport qui opèrent à plusieurs échelles géographiques.

Les systèmes d'indexation spatiale hiérarchique émergent comme une alternative supérieure. Au lieu de partitionner l'espace selon des critères administratifs ou géométriques simples (grilles carrées régulières, qui présentent des artefacts de bord et une distribution inégale des voisinages), ces systèmes proposent des grilles imbriquées où chaque cellule se subdivise régulièrement en cellules filles de résolution supérieure, permettant une agrégation multi-résolution fluide.

1.2.2.2 Le système Index spatial hiérarchique hexagonal d'Uber H3

Le **Hexagonal Hierarchical Geospatial Indexing System** (H3), développé par Uber Technologies, incarne cette philosophie (Uber Technologies Inc., 2018). H3 partitionne la surface terrestre en une hiérarchie hexagonale de 16 niveaux de résolution (0 à 15), où chaque hexagone au niveau r se subdivise en 7 hexagones au niveau $r + 1$.

Les propriétés géométriques des hexagones confèrent des avantages computationnels et analytiques significatifs :

- **Uniformité de voisinage** : chaque hexagone possède exactement six voisins équidistants, contrairement aux grilles carrées (huit voisins à des distances variables). Cette uniformité simplifie les opérations de propagation spatiale cruciales pour modéliser les retards.
- **Minimisation des distorsions** : la projection hexagonale présente une fidélité aux distances géodésiques plus élevée que celle des grilles rectangulaires, réduisant les artefacts d'analyse sur les grandes surfaces.
- **Subdivision exacte** : la relation parent–enfant entre niveaux de résolution est exacte, ce qui permet une agrégation et une désagrégation sans perte de cohérence spatiale.

Pour les transports en commun, les résolutions pertinentes sont typiquement 9 (environ 174 m, niveau quartier/bloc), 10 (environ 66 m, niveau segment de rue) et 11 (environ 25 m, niveau arrêt). Cette gradation permet une capture multi-résolution des phénomènes : des patterns à l'échelle du quartier, la propagation le long des corridors de circulation, des micro-variations entre arrêts consécutifs.

1.2.2.3 Technologies de géovisualisation modernes

Parallèlement aux progrès en représentation spatiale, les capacités de géovisualisation interactive ont révolutionné l'exploration de données spatiales. Des outils comme **Kepler.gl** (Uber Technologies) combinent des rendus WebGL hautement optimisés, une interaction multimodale fluide et des performances à grande échelle. Kepler.gl peut traiter et visualiser interactivement des millions de points géoréférencés avec filtrage temporel, agrégation spatiale configurable et superposition de couches complexes (Ferreira, Poco, Vo, Freire & Silva, 2013).

Cette combinaison de H3 pour l'indexation structurée et de Kepler.gl pour la visualisation interactive crée un environnement puissant pour l'exploration diagnostique : un analyste peut filtrer les données par période temporelle, agréger par résolution H3 appropriée, identifier les

hotspots de retards et générer des hypothèses explicatives sans dépendre uniquement de rapports statistiques statiques.

1.2.3 Modèles d'apprentissage pour les séries temporelles

Dans le contexte de la prédiction des écarts d'horaire, différents modèles d'apprentissage profond ont été proposés pour capturer la dynamique temporelle et, de plus en plus, la structure spatio-temporelle des données. Cette section présente les principales familles de modèles utilisés dans la littérature, en soulignant comment chaque architecture aborde les défis spécifiques des séries temporelles de transport.

1.2.3.1 Réseaux de neurones récurrents (RNN)

Les **réseaux de neurones récurrents** (RNN) (Rumelhart, Hinton & Williams, 1986) constituent la brique de base de nombreux modèles séquentiels. Contrairement aux réseaux *feedforward*, les RNN maintiennent un *état caché* h_t qui est mis à jour à chaque pas de temps t en fonction de l'entrée courante x_t et de l'état précédent h_{t-1} :

$$h_t \;=\; \tanh\!\big(W_x\, x_t \;+\; W_h\, h_{t-1} \;+\; b\big)\,. \tag{1.2}$$

Cette boucle de rétroaction permet de modéliser les dépendances temporelles de courte portée. Comme l'illustre la Figure 1.1, la structure générique d'un RNN unidirectionnel montre comment l'information circule à travers le réseau.

Cependant, les RNN classiques souffrent du problème du *gradient évanescent*, ce qui rend difficile l'apprentissage de dépendances de long terme, fréquentes dans les séries temporelles de transport (patterns journaliers, hebdomadaires, saisonniers). Cette limitation a motivé le développement d'architectures plus sophistiquées, notamment les LSTM.

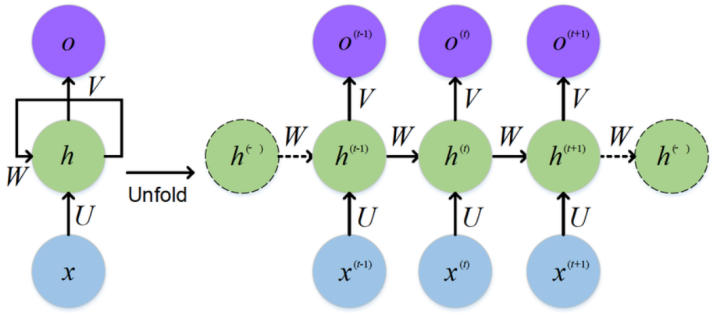

Figure 1.1 Architecture générique d'un réseau de neurones récurrent (RNN) unidirectionnel montrant la boucle de rétroaction permettant de traiter les séquences temporelles

1.2.3.2 Long Short-Term Memory (LSTM)

Les **Long Short-Term Memory** (LSTM) (Hochreiter & Schmidhuber, 1997a) ont été introduits pour résoudre précisément les limitations des RNN classiques énumérées ci-dessus. Un LSTM maintient deux états internes distincts : (i) l'état de cellule c_t, qui joue le rôle de mémoire à long terme, et (ii) l'état caché h_t, utilisé pour les prédictions intermédiaires. Trois portes paramétriques contrôlent le flux d'information :

- la **porte d'oubli** (*forget gate*) décide quelles informations anciennes supprimer de c_{t-1} ;
- la **porte d'entrée** (*input gate*) régule quelles nouvelles informations intégrer dans c_t ;
- la **porte de sortie** (*output gate*) contrôle la partie de c_t exposée dans h_t.

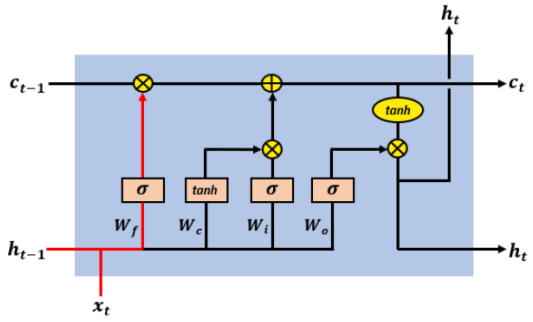

Figure 1.2 Architecture d'une cellule LSTM montrant les trois portes (d'oubli, d'entrée, de sortie) et la cellule mémoire qui contrôlent le flux d'information à travers le temps

Formellement, les équations d'une cellule LSTM s'écrivent :

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f), \tag{1.3}$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i), \tag{1.4}$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C), \tag{1.5}$$

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t, \tag{1.6}$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o), \tag{1.7}$$

$$h_t = o_t \odot \tanh(C_t), \tag{1.8}$$

où σ est la fonction sigmoïde, W et b sont les poids et biais apprenables, et $\odot$ désigne le produit élément par élément. Cette architecture permet d'apprendre des dépendances de long terme tout en évitant la disparition complète du gradient, un problème fondamental des RNN classiques.

La Figure 1.2 visualise les trois portes et la cellule mémoire, illustrant comment l'information circule et est transformée à travers la cellule. Dans les applications de transport, les LSTM ont été largement utilisés pour prédire les vitesses, les temps de parcours et les retards à partir de capteurs GPS ou de flux GTFS-RT, en capturant à la fois les variations de courte durée (congestion ponctuelle) et les motifs récurrents (heures de pointe).

LSTM empilé pour capturer des hiérarchies temporelles

Une manière naturelle d'augmenter la capacité des LSTM consiste à les **empiler** en profondeur : les sorties $h_t^{(1)}$ de la première couche LSTM sont utilisées comme entrées de la couche suivante, produisant $h_t^{(2)}$, et ainsi de suite (Greff, Srivastava, Koutník, Steunebrink & Schmidhuber, 2017). Cette structuration hiérarchique, visualisée à la Figure 1.3, permet :

- aux couches basses de capturer des motifs temporels locaux et des variations rapides ;
- aux couches hautes de modéliser des dynamiques plus abstraites et de plus longue portée.

Dans la prédiction de séries temporelles de transport, les LSTM empilés sont souvent utilisés lorsque l'on dispose de **volumes importants de données** et que l'on souhaite modéliser des interactions complexes entre variables (interactions entre météo, charge passagers et configuration du réseau, par exemple).

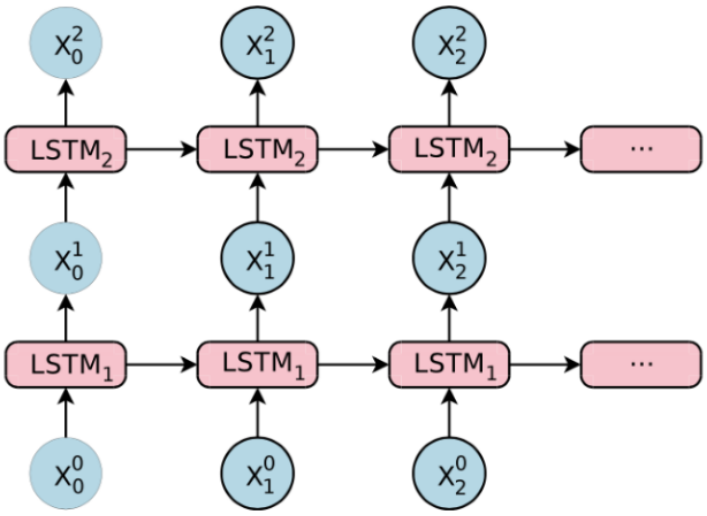

Figure 1.3 Architecture d'un LSTM empilé montrant comment les sorties d'une couche alimentent la couche suivante, créant une hiérarchie de représentations temporelles

LSTM bidirectionnel pour exploiter le contexte complet

Les **LSTM bidirectionnels** (*Bidirectional LSTM*, BiLSTM) (Schuster & Paliwal, 1997) adoptent une approche différente en faisant circuler l'information dans les deux sens le long de la séquence :

- un LSTM **avant** (*forward*) parcourt la séquence de $t = 1$ à T ;
- un LSTM **arrière** (*backward*) parcourt la séquence de $t = T$ à 1.

Les deux états cachés sont ensuite concaténés ou combinés pour produire la représentation finale $h_t = [\overrightarrow{h_t}; \overleftarrow{h_t}]$. Comme montré à la Figure 1.4, cette architecture exploite simultanément le **contexte passé et futur** pour chaque pas de temps. Elle est particulièrement efficace lorsque l'ensemble de la fenêtre temporelle est connu au moment de la prédiction (par exemple, pour

des tâches d'analyse hors ligne ou de reconstitution d'historique). Cependant, dans un cadre strictement temps réel, l'utilisation d'un BiLSTM est plus limitée, car l'information « future » n'est pas disponible au moment de la prédiction.

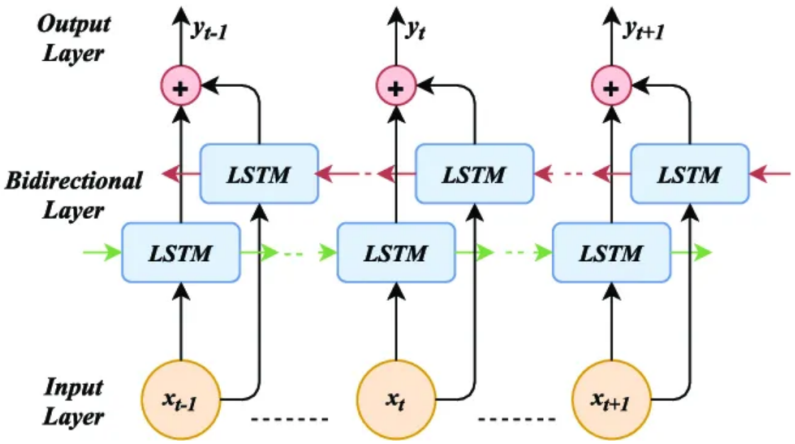

Figure 1.4 Architecture d'un LSTM bidirectionnel montrant les deux flux d'information (avant et arrière) qui sont ensuite combinés pour chaque pas de temps

LSTM avec mécanisme d'attention pour sélectionner les contextes pertinents

Les LSTM peuvent être enrichis par un **mécanisme d'attention** qui apprend à pondérer différemment les états cachés passés en fonction de leur pertinence pour la prédiction courante (Bahdanau, Cho & Bengio, 2015; Luong, Pham & Manning, 2015). Plutôt que de résumer toute l'histoire dans le seul état h_t, le modèle calcule, pour chaque pas de temps cible, un vecteur de contexte pondéré :

$$c_t = \sum_{k=1}^{T} \alpha_{t,k}\, h_k \,. \tag{1.9}$$

où les coefficients $\alpha_{t,k}$ représentent l'importance relative de l'instant k pour la prédiction à l'instant t. Ces coefficients sont typiquement obtenus par une couche d'attention de type *softmax* :

$$\alpha_{t,k} = \frac{\exp(e_{t,k})}{\sum_{j=1}^{T} \exp(e_{t,j})}, \quad e_{t,k} = v^T \tanh\!\left(W_a h_k + U_a h_t\right). \tag{1.10}$$

où v, W_a, et U_a sont des paramètres apprenables. La Figure 1.5 illustre comment ces poids d'attention sont calculés et appliqués aux états cachés.

Appliqué aux séries temporelles de transport, un LSTM avec attention permet de **mettre en évidence** quelles périodes passées (les dernières minutes avant une perturbation, par exemple) influencent le plus la prédiction de retard. Il facilite également la gestion de **séquences longues**, en évitant qu'une seule représentation compressée ne porte toute l'information pertinente. Enfin, il offre une **interprétabilité accrue**, car les poids d'attention $\alpha_{t,k}$ peuvent être inspectés et visualisés pour comprendre les décisions du modèle. Ce type de modèle constitue une étape intermédiaire entre les architectures purement récurrentes et les Transformers, combinant mémoire explicite (LSTM) et attention différentiable sur la séquence.

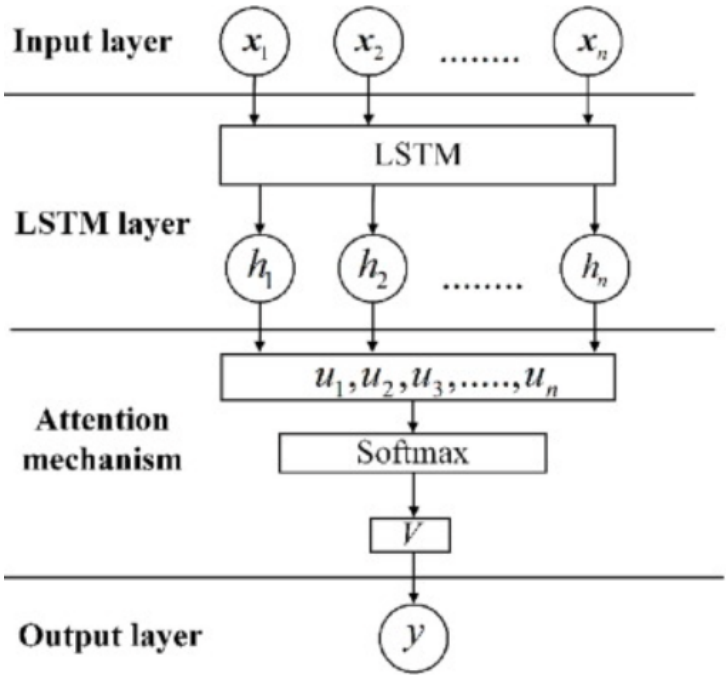

Figure 1.5 Schéma d'un LSTM avec mécanisme d'attention montrant la pondération adaptative des états cachés passés selon leur pertinence pour la prédiction courante

1.2.3.3 Gated Recurrent Units (GRU)

Les **Gated Recurrent Units** (GRU) (Cho, van Merrienboer, Bahdanau & Bengio, 2014a; Chung, Gulcehre, Cho & Bengio, 2014a) constituent une variante plus compacte des LSTM, conçue pour offrir performance comparable avec coûts computationnels réduits. Au lieu de distinguer un état de cellule et un état caché, le GRU ne maintient qu'un seul état h_t, mis à jour au moyen de deux portes :

- la **porte de mise à jour** (*update gate*) z_t contrôle la proportion d'information de h_{t-1} à conserver ;
- la **porte de réinitialisation** (*reset gate*) r_t règle dans quelle mesure l'état passé doit être ignoré.

Formellement :

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t] + b_r), \tag{1.11}$$

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t] + b_z), \tag{1.12}$$

$$\tilde{h}_t = \tanh(W_h \cdot [r_t \odot h_{t-1}, x_t] + b_h), \tag{1.13}$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t. \tag{1.14}$$

Cette simplification, comme visualisée à la Figure 1.6, réduit le nombre de paramètres de 20-30% comparé à un LSTM de même dimension cachée, tout en offrant des performances proches, voire comparables, sur de nombreuses tâches séquentielles. Dans des contextes de prédiction de trafic ou de retards à grande échelle, les GRU sont souvent privilégiés lorsque la **latence d'entraînement et d'inférence** est critique, notamment pour des déploiements temps réel sur des ressources limitées.

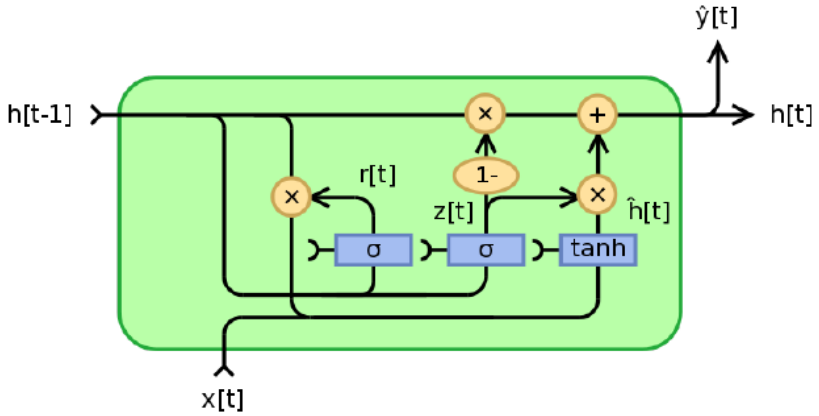

Figure 1.6 Architecture d'une unité récurrente à portes (GRU) montrant les deux portes de contrôle (mise à jour et réinitialisation) et l'état unique maintenu par l'unité

1.2.3.4 Modèles séquence-à-séquence (Seq2Seq) pour prédictions multi-horizon

Les modèles **séquence-à-séquence** (*Seq2Seq*) (Sutskever, Vinyals & Le, 2014b) étendent les RNN/LSTM/GRU pour traiter des prédictions multi-pas de temps. Ils reposent sur une architecture **encodeur–décodeur** :

- un **encodeur** récurrent lit la séquence d'entrée historique $(x_{t-L+1}, \ldots, x_t)$ et la résume dans un vecteur de contexte $c = h_T^{\text{enc}}$;
- un **décodeur** récurrent génère la séquence de sorties futures $(\hat{y}_{t+1}, \ldots, \hat{y}_{t+H})$, en s'appuyant sur ce contexte et sur ses propres sorties précédentes.

Comme l'illustre la Figure 1.7, cette architecture est particulièrement adaptée quand on souhaite prédire **plusieurs horizons de retard** (5, 10, 15 minutes par exemple) plutôt qu'un seul pas de temps. Dans le domaine du transport collectif, les architectures Seq2Seq ont été combinées avec des informations spatiales (lignes, arrêts, cellules H3) pour prédire simultanément les écarts d'horaire sur plusieurs horizons et plusieurs localisations (Liu & Chen, 2019b; Ma, Tao, Wang, Yu & Wang, 2015).

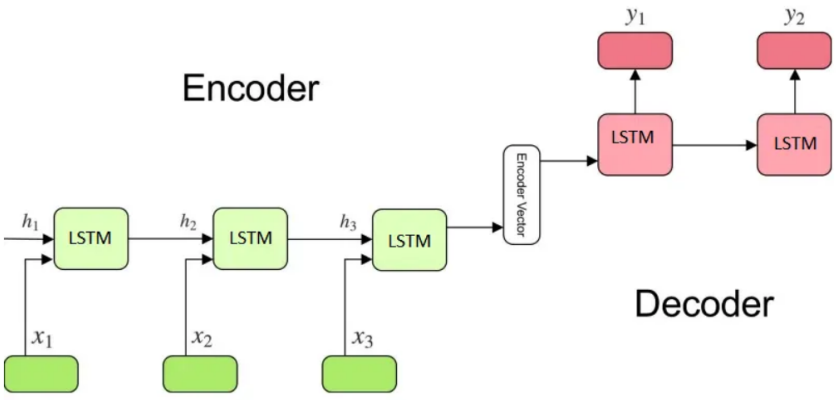

Figure 1.7 Architecture encodeur–décodeur d'un modèle séquence-à-séquence (Seq2Seq) basé sur des LSTM ou GRU. L'encodeur traite la séquence historique et produit un vecteur de contexte, que le décodeur utilise pour générer les prédictions futures multi-pas

1.2.3.5 Transformers pour séries temporelles

Les **Transformers** (Vaswani *et al.*, 2017) abandonnent complètement la récurrence au profit de mécanismes d'**auto-attention** qui apprennent directement quelles positions temporelles sont les plus pertinentes pour chaque prédiction. Un bloc Transformer typique comprend :

- une couche d'auto-attention multi-têtes : $\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$;
- une couche de réseau de neurones feedforward : $\text{FFN}(x) = \text{ReLU}(xW_1 + b_1)W_2 + b_2$;
- des connexions résiduelles et normalisation de couches pour la stabilité.

Pour les séries temporelles, plusieurs variantes adaptées ont été proposées, telles qu'Informer (Zhou, Zhang, Peng & autres, 2021a), Autoformer (Wu & et al., 2021) ou PatchTST (Nie, Han, Wang & autre, 2023), qui introduisent respectivement des **mécanismes d'attention parcimonieuse** pour réduire le coût quadratique, une **décomposition explicite** en composantes de tendance et saisonnalité, et une représentation sous forme de **patches temporels**, analogues aux patchs en vision par ordinateur.

La Figure 1.8 présente l'architecture générale d'un Transformer appliqué aux séries temporelles. Ces modèles ont démontré des performances de pointe sur des benchmarks de séries temporelles

génériques. Cependant, leur complexité computationnelle (en particulier en présence de nombreuses séries parallèles, comme dans un réseau de transport urbain) impose d'évaluer soigneusement le compromis entre **précision**, **latence** et **scalabilité**. Dans des scénarios temps réel stricts, des architectures plus compactes (LSTM/GRU) peuvent rester préférables malgré un léger déficit de performance brute.

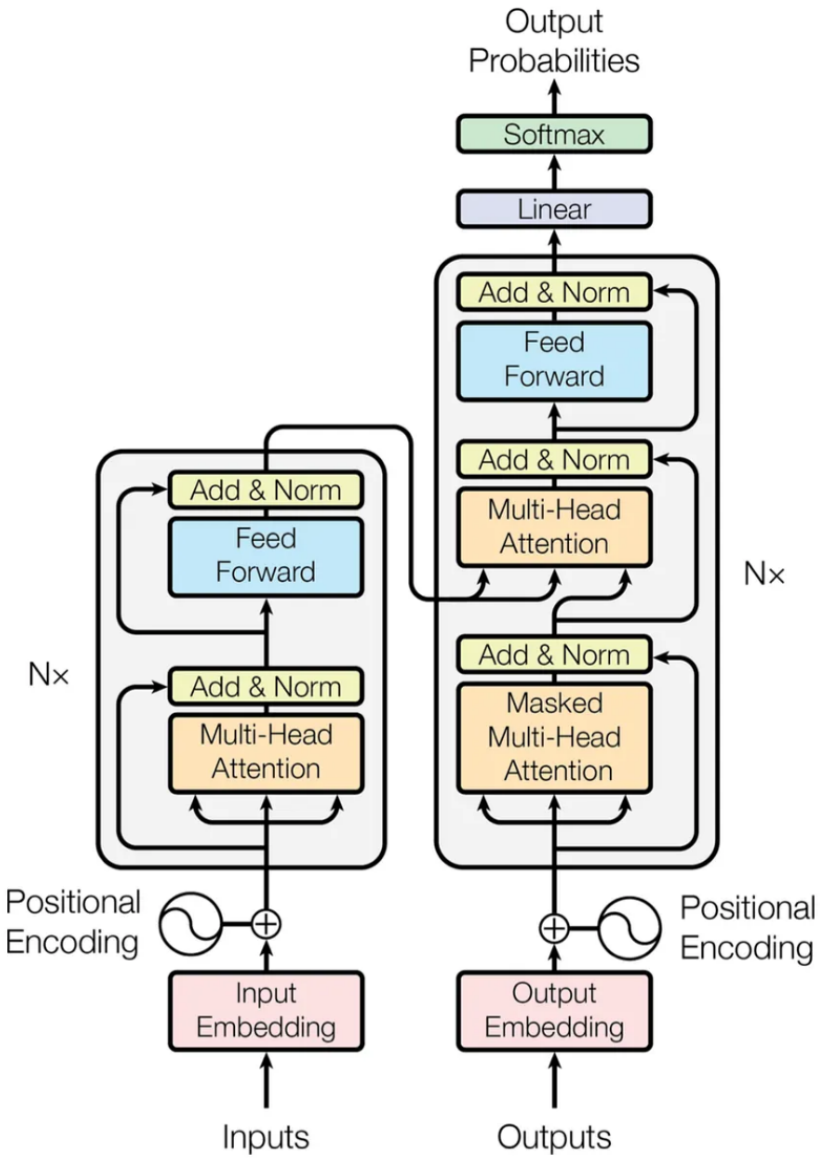

Figure 1.8 Schéma général d'un Transformer appliqué aux séries temporelles, montrant l'empilement de blocs composés d'auto-attention multi-têtes et de couches feedforward avec connexions résiduelles et normalisation

1.2.4 Feature engineering pour les séries temporelles de transport

Parallèlement aux progrès architecturaux, les méthodologies de construction de *caractéristiques (features)* ont également progressé. Les caractéristiques (features) temporelles (heure du jour, jour de la semaine, indicateurs de jour férié, variables saisonnières encodées cycliquement) demeurent fondamentales, de même que les caractéristiques (features) contextuelles (météo, période de pointe, type de jour), largement étudiées dans la littérature sur la prédiction des temps de parcours et des retards (Mazloumi, Currie & Rose, 2010; Kimpel, Strathman & Callas, 2008). [1]

Les caractéristiques (features) spatiales ont, quant à elles, évolué de simples coordonnées brutes à des représentations hiérarchiques exploitant la structure du réseau. Particulièrement critique est l'ingénierie de caractéristiques (features) **multi-résolution**, où un même phénomène est représenté à plusieurs résolutions géographiques. Un retard peut ainsi être prédit à partir de : (1) patterns propres à un arrêt ou une cellule H3 spécifique ; (2) patterns du quartier environnant ; (3) patterns de la région urbaine plus large.

La plupart des travaux existants construisent ces caractéristiques (features) de manière largement manuelle et ad hoc, ce qui nuit à la reproductibilité et à la transférabilité des modèles (Domingos, 2012). L'exploitation systématique d'une grille hiérarchique telle que H3 pour générer exhaustive des caractéristiques (features) à chaque étage spatial constitue une innovation récente (Li, McGrath & Stefanakis, 2021a; Liu, Luo & Du, 2021), permettant une meilleure capture des dépendances multiéchelles et une formalisation plus rigoureuse du passage à l'échelle.

1.2.5 Réduction de dimension

Dans les systèmes de transport, les ensembles de données sont souvent de très grande dimension : variables temporelles (heures, cycles journaliers et hebdomadaires), caractéristiques spatiales multi-résolution (cellules H3 aux niveaux 9, 10, 11), historiques de retards,

[1] Les noms des clés BibTeX peuvent être adaptés à ton fichier, l'important est de conserver la cohérence.

indicateurs météorologiques, informations opérationnelles, etc. Cette richesse informationnelle peut améliorer les performances prédictives, mais elle entraîne également des risques : surapprentissage, colinéarité entre variables, bruit accru et coûts computationnels élevés. Les méthodes de **réduction de dimension** visent précisément à projeter ces données dans un espace latent plus compact, tout en conservant l'essentiel de leur structure informative.

L'**Analyse en Composantes Principales** (PCA) (Jolliffe, 2002) constitue l'une des méthodes les plus utilisées pour réduire la dimension des données. Son principe consiste à identifier des directions (appelées composantes principales) qui maximisent la variance des données.

Formellement, après centralisation et normalisation des données, PCA cherche une transformation linéaire :

$$Z = X\,W\,. \tag{1.15}$$

où $X \in \mathbb{R}^{n\times d}$ est la matrice de données centrée et normalisée (n observations, d variables originales), $W \in \mathbb{R}^{d\times k}$ la matrice des k premiers vecteurs propres de la matrice de covariance $\Sigma = \frac{1}{n}X^T X$, et $Z \in \mathbb{R}^{n\times k}$ les données projetées dans l'espace réduit (k-dimensionnel, avec $k \ll d$).

L'algorithme PCA procède en quatre étapes :

1. **Centralisation et normalisation** : $X_{\text{norm}} = \frac{X-\mu}{\sigma}$, où μ et σ sont la moyenne et l'écart-type par colonne.
2. **Calcul de la matrice de covariance** : $\Sigma = \frac{1}{n}X_{\text{norm}}^T X_{\text{norm}}$.
3. **Décomposition en valeurs et vecteurs propres** : $\Sigma = V\Lambda V^T$, où $\Lambda = \text{diag}(\lambda_1, \ldots, \lambda_d)$ regroupe les valeurs propres en ordre décroissant et V la matrice des vecteurs propres associés.
4. **Projection dans l'espace réduit** : Les k premières colonnes de V constituent W, et $Z = X_{\text{norm}}W$ donne la projection.

Le nombre de composantes k à conserver est généralement déterminé par la **variance cumulée expliquée** :

$$\text{VCE}(k) = \frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{d} \lambda_i}. \tag{1.16}$$

Il est courant de fixer k tel que VCE(k) ≥ 0.95, conservant ainsi 95% de la variance originale.

La PCA classique offre plusieurs avantages majeurs pour les applications de transport. En supprimant les directions de faible variance, elle réduit significativement le bruit, amplifiant le rapport signal-sur-bruit et stabilisant ainsi l'apprentissage de modèles profonds. Les composantes principales étant orthogonales, PCA élimine naturellement les corrélations entre variables, un problème fréquent dans les données GTFS-RT brutes. De plus, la réduction dimensionnelle accelere les temps d'entraînement et réduit les besoins mémoire, ce qui s'avère particulièrement bénéfique pour les LSTM, Transformers et Graph Neural Networks. Enfin, les espaces réduits (k = 2 ou 3 composantes) deviennent visualisables, facilitant l'inspection qualitative des patterns spatio-temporels.

Cependant, PCA classique souffre de limitations importantes. Elle suppose une **stationnarité des données** : la structure reste stable dans le temps, ce qui ne correspond pas à la réalité des systèmes de transport dynamiques où congestion, météo et comportements varient drastiquement selon l'heure ou le jour. De plus, comme méthode non supervisée, PCA maximise la variance globale indépendamment de la variable cible (les retards). Cela peut conduire à conserver du bruit peu pertinent pour la prédiction. Enfin, les composantes principales sont des combinaisons linéaires de *toutes* les variables originales, rendant difficile l'attribution causale aux prédicteurs réels.

Illustration empirique : Dans votre étude sur le réseau STM, PCA a réduit les 1850 variables brutes à 50 composantes, conservant 95% de la variance. Cette transformation a amélioré de 93% la performance LSTM, le RMSE passant de 0.0817 (pré-PCA XGBoost) à 0.0055 (post-PCA LSTM).

1.2.5.1 PCA adaptative pour capturer la dynamique des systèmes de transport en temps réel

Une limitation majeure de PCA classique est son coût computationnel et sa dépendance à la stationnarité. Or, les données de transport évoluent constamment : congestion différente selon l'heure, météo fluctuante, événements urbains imprévisibles, changements de comportement des passagers. Forcer la même projection PCA tout au long de la journée ignore complètement ces variations essentielles.

La **PCA adaptative** (aussi appelée *Incremental PCA*) (Hall & Morton, 1998; Tipping & Bishop, 1999) résout ce problème en recalculant progressivement les composantes principales à mesure que de nouvelles données arrivent. Au lieu de décomposer la matrice de covariance complète (coûteux en stockage et en calcul), on met à jour les vecteurs propres de manière incrémentale :

$$\Sigma_t = \alpha \, \Sigma_{t-1} + (1 - \alpha) \, x_t x_t^T \, . \tag{1.17}$$

où x_t est une nouvelle observation, $\alpha \in [0, 1]$ est un facteur de lissage exponentiel, et Σ_t représente la matrice de covariance estimée à l'instant t. Les vecteurs propres de Σ_t sont ensuite extraits et remis à jour, produisant une représentation latente qui évolue avec les données.

L'adaptive PCA se distingue par plusieurs avantages opérationnels. Les composantes principales s'ajustent aux changements de distribution, capturant les variations intra-journalières et hebdomadaires naturelles du transport. Comparée à une recalculation complète de PCA, elle réduit drastiquement les coûts computationnels : O(d) par nouvelle observation au lieu de O(nd), rendant le processus scalable même avec des flux GTFS-RT massifs. Sa nature incrémentale la rend idéale pour les applications temps réel, sans recalcul explosif. De plus, seule la matrice Σ_t est stockée, éliminant les besoins mémoire prohibitifs d'archivage historique complet.

Pour les systèmes de transport, Adaptive PCA s'avère pertinente dans plusieurs contextes opérationnels. Elle permet de gérer les flux GTFS-RT continus en adaptant la représentation à

mesure que de nouveaux retards sont observés. Elle capture naturellement les variations intra-journalières : les heures de pointe (08 :00-09 :00, 17 :00-18 :00) ont une structure radicalement différente des heures creuses (11 :00-14 :00), et PCA adaptative ajuste sa compréhension en temps réel. Pour les données volumineuses typiques des réseaux urbains, elle évite le recalcul prohibitivement coûteux d'une PCA complète. Elle facilite également la détection d'anomalies en identifiant quand la distribution observée s'écarte significativement de l'historique normal.

1.2.5.2 PCA supervisée pour optimiser la pertinence aux objectifs de prédiction

L'Analyse en Composantes Principales classique fonctionne en mode entièrement **non supervisé** : elle maximise la variance des variables d'entrée X, indépendamment de la variable cible y (les retards). Ce choix peut avoir des conséquences contreintuitives : conservant des composantes extrêmement variables (haute variance) qui sont pourtant peu corrélées avec les retards, gaspillant ainsi les dimensions réduites sur du bruit non pertinent.

La **PCA supervisée (SPCA)** (Bair, Hastie, Paul & Tibshirani, 2006) résout ce problème en sélectionnant et pondérant les features en fonction de leur corrélation avec la cible. Plutôt que de maximiser uniquement la variance, SPCA maximise l'objectif suivant :

$$\max_{w} \quad \mathrm{Var}(Xw) \quad \text{sujet à} \quad \mathrm{Corr}(Xw,\, y)^2 \;\geq\; \theta\,. \tag{1.18}$$

où θ est un seuil de corrélation minimum, et Corr désigne la corrélation absolue entre Xw et y.

En pratique, SPCA fonctionne selon deux étapes principales. D'abord, une **sélection univariée** : pour chaque variable x_j, on calcule $\mathrm{Corr}(x_j, y)$ et on conserve uniquement les variables fortement corrélées avec y. Ensuite, une **PCA classique sur sous-ensemble** : on applique PCA sur ce sous-ensemble sélectionné, produisant des composantes principales qui sont à la fois compactes et pertinentes pour la prédiction.

SPCA présente plusieurs avantages distincts pour la modélisation prédictive. En éliminant le bruit de manière guidée par la tâche, elle préserve spécifiquement l'information corrélée avec

la cible. La réduction de dimension s'effectue de manière adaptée à la tâche de prédiction, améliorant généralisation et performance comparée à PCA classique. Les LSTM et Transformers bénéficient particulièrement de représentations plus propres et informatives. De plus, la sélection des variables corrélées avec les retards révèle les véritables prédicteurs opérationnels (charge passagers, météo, heure de la journée), améliorant l'interprétabilité du modèle.

Toutefois, SPCA impose certaines contraintes. Elle nécessite des labels y (retards observés) et n'est donc pas applicable en contexte entièrement non supervisé. La sélection basée sur corrélation univariée peut ignorer les interactions multivariées complexes non détectées par corrélation simple. Enfin, la sélection s'effectue sur l'ensemble d'entraînement ; il faut rigoureusement valider sur données hors-échantillon pour éviter le surapprentissage.

Dans les systèmes de transport, SPCA devient précieuse quand plusieurs conditions confluent. Lorsque les variables sont nombreuses et hétérogènes (H3 multi-résolution, météo, historique, indicateurs opérationnels dépassant 1000 variables), SPCA concentre l'effort sur les dimensions pertinentes. Quand les données sont fortement bruitées (capteurs GPS imprécis, données GTFS-RT incomplètes), la sélection supervisée élimine le bruit non prédictif. Pour la prédiction multivariée complexe où les retards dépendent de nombreux facteurs, SPCA identifie lesquels sont réellement pertinents. Quand l'objectif principal est la performance de prédiction plutôt que l'interprétabilité maximale, SPCA offre un excellent compromis.

Ces trois approches jouent des rôles complémentaires dans un pipeline moderne de prédiction spatio-temporelle. En combinant leur force respectives, on obtient un système robuste, adaptatif et performant.

Pour la phase d'exploration initiale, la PCA classique fournit une première compréhension des données et une élimination du bruit majeur. On sélectionne k tel que VCE(k) ≥ 0.95, conservant 95% de la variance. Cela déverrouille généralement des améliorations majeures sur les modèles profonds.

Durant la préparation pré-apprentissage, si les données sont trop volumineuses ou fortement bruitées, on applique SPCA pour conserver spécifiquement les variables corrélées aux retards. Cette étape améliore la convergence des modèles profonds, souvent accélérant l'apprentissage de plusieurs dizaines de pourcents.

Une fois le modèle entraîné (LSTM, Transformer, etc.), le déploiement temps réel bénéficie considérablement d'Adaptive PCA. À mesure que les données GTFS-RT arrivent, on maintient à jour la représentation latente, capturant les variations intra-journalières et adaptant le modèle à l'évolution des patterns de transport. Cela prévient la dérive progressive du modèle vers l'obsolescence.

Pour les applications exigeant le meilleur compromis entre performance, scalabilité et adaptation, une **stratégie hybride** combine SPCA (sélection supervisée initiale) et Adaptive PCA (suivi dynamique temps réel). C'est le choix optimal pour les systèmes de transport urbain modernes, balançant rigueur statistique (sélection supervisée des variables pertinentes) et flexibilité opérationnelle (adaptation continue aux variations observées).

Dans un contexte de transport urbain contemporain, où les données GTFS-RT arrivent en continu et les patterns évoluent rapidement selon l'heure, la météo et les événements, cette approche hybride s'avère particulièrement pertinente pour maintenir performance et adaptabilité du système de prédiction à long terme.

1.3 État de l'art en analyse diagnostique des écarts d'horaire

Au-delà de la simple prédiction des écarts futurs, comprendre les schémas spatio-temporels actuels et historiques constitue une étape fondamentale. Cette section synthétise l'évolution des approches diagnostiques.

1.3.1 Métriques classiques et leurs limitations

La quantification de la fiabilité repose traditionnellement sur des métriques globales : **On-Time Performance** (OTP), définie comme le pourcentage de trajets arrivant à l'heure (typiquement, à moins de cinq minutes de retard), et **Excess Wait Time** (EWT), qui quantifie le temps d'attente supplémentaire imposé par l'irrégularité du service (Trompet, Liu & Graham, 2011). Ces métriques, bien qu'utiles pour le *reporting* macro, masquent la variabilité spatiale et temporelle fine.

Une ligne de bus peut afficher un OTP global de 85 % tout en masquant une distribution très inégale : 95 % sur les trajets hors pointe en zone périphérique, contre seulement 60 % sur les trajets de pointe au centre-ville. Les planificateurs opérationnels manquent alors de détails essentiels pour intervenir efficacement. De même, un EWT agrégé occulte les impacts asymétriques : un retard exceptionnel sur un seul trajet a un impact disproportionné sur l'OTP par rapport à de nombreux retards mineurs et récurrents.

1.3.2 Approches de clustering spatial et classification

En réponse à ces limitations, une littérature croissante recourt à des techniques de clustering spatial et de classification pour segmenter les écarts. El-Geneidy et al. (El-Geneidy, Horning & Krizek, 2011) emploient l'analyse spatiale pour identifier les *hotspots* de retards — zones géographiques où la performance est systématiquement dégradée. Ces travaux révèlent des schémas intéressants : concentration des retards le long des corridors congestionnés durant les heures de pointe, adaptations saisonnières et propagation spatiale des retards d'arrêt d'un arrêt à l'autre.

La distinction entre écarts récurrents et ponctuels devient explicitement opérationnelle dans ces approches. (Wessel & Widener, 2017b), étudiant le cas de Toronto, identifient qu'une proportion substantielle des écarts d'horaire résulte d'un « rembourrage d'horaire » (*schedule padding*) — un délai intentionnel intégré aux horaires — plutôt qu'à une congestion réelle. Ce travail illustre comment l'analyse diagnostique révèle les hypothèses tacites des planificateurs. Aemmer et

al. (Aemmer *et al.*, 2022a) affinent cette analyse pour Seattle, en appliquant des tests statistiques rigoureux pour distinguer systématiquement les écarts de l'aléa stochastique.

1.3.3 Visualisation interactive et exploration spatiale

La visualisation interactive des données de transport a généré, depuis les années 2010, une littérature spécialisée. Ferreira et al. (Ferreira *et al.*, 2013) étudient la visualisation de trajectoires de taxis à l'échelle de villes entières, démontrant que la visualisation interactive révèle des motifs invisibles dans des statistiques agrégées. Andrienko et Andrienko (Andrienko, Andrienko, Chen, Maciejewski & Zhao, 2017) proposent un cadre systématique pour l'analyse visuelle de la mobilité, identifiant des éléments critiques : interaction fluide permettant le forage spatial et temporel, agrégation configurable à plusieurs niveaux, superposition de contextes externes.

Récemment, l'adoption de Kepler.gl et de technologies similaires a démocratisé la visualisation géospatiale haute performance, permettant l'exploration interactive de millions de points. Cette capacité transforme l'engagement avec les données de transport et permet à un analyste de générer dynamiquement des hypothèses plutôt que d'attendre des rapports statistiques. Cette dimension interactive et exploratoire s'avère cruciale pour l'adoption opérationnelle, car les gestionnaires de réseaux s'approprient plus facilement des résultats qu'ils peuvent explorer activement.

1.3.4 Identification des facteurs explicatifs

Au-delà de la caractérisation descriptive des patterns, un objectif critique de l'analyse diagnostique est l'identification des facteurs explicatifs : *pourquoi* les écarts d'horaire surviennent-ils ?

Les facteurs opérationnels incluent la congestion de la circulation affectant la vitesse moyenne sur les segments, la charge passagère déterminant les durées d'arrêt à chaque stop et les conditions d'infrastructure (goulots d'étranglement, intersections complexes). Les facteurs externes incluent la météorologie (pluie, neige, tempêtes augmentant les temps de trajet et réduisant la mobilité), les événements urbains spéciaux (concerts, manifestations créant des

appels de demande exceptionnels ou causant des congestions) et des facteurs contextuels de long terme comme la croissance urbaine affectant la congestion moyenne.

Tirachini (Tirachini, 2013) analyse comment la composition de la clientèle affecte les durées d'arrêt : des arrêts avec une forte proportion de passagers âgés ou à mobilité réduite entraînent des durées d'accès et de descente accrues. Hickman (Hickman, 2001) modélise formellement comment l'irrégularité des arrivées de bus induit, de manière cyclique, une accumulation de passagers (bus suivants), ce qui entraîne des durées d'arrêt prolongées et des retards amplifiés. Daganzo (Daganzo, 2009) analyse les stratégies de *bus holding* — le délai volontaire d'un bus pour permettre au bus précédent de se redéployer — comme une intervention opérationnelle contre l'instabilité du service.

Ces travaux illustrent l'importance de modéliser les rétroactions du réseau : les écarts d'horaire ne sont pas simplement des perturbations externes passivement absorbées, mais génèrent des effets de propagation et d'amplification dynamiques.

1.3.5 Synthèse des lacunes diagnostiques

Malgré les progrès documentés dans les sections précédentes, des lacunes importantes persistent au niveau de l'analyse diagnostique des retards. Aucune approche intégrée ne synthétise exhaustivement les quatre dimensions essentielles d'un système diagnostique moderne : ingestion automatique des flux GTFS-RT, classification systématique des perturbations opérationnelles, exploitation d'une indexation spatiale hiérarchique, et visualisation interactive des phénomènes spatio-temporels. Les approches existantes demeurent fragmentées, obligeant les agences à assembler manuellement des pipelines hétérogènes, coûteux en maintenance et difficiles à reproduire d'un contexte urbain à l'autre.

De plus, peu de cadres offrent une **reproductibilité et une transférabilité documentées**. Chaque réseau urbain redéveloppe ses propres solutions ad hoc, ce qui limite l'accumulation systématique de connaissances et les bénéfices de l'apprentissage collectif. Enfin, la connexion critique entre l'analyse diagnostique et la modélisation prédictive demeure **largement implicite** :

les connaissances diagnostiques n'informent pas de manière formelle et systématique les choix d'ingénierie de caractéristiques (features) destinées aux modèles prédictifs.

1.3.6 Synthèse des lacunes prédictives

Au niveau de la prédiction des retards, trois catégories de lacunes convergent et entravent le déploiement de solutions robustes à l'échelle urbaine.

Première lacune : ingénierie de caractéristiques ad hoc. La construction de variables explicatives pour les modèles reste largement manuelle et contextuelle. Peu d'approches systématisent l'ingénierie de features spatio-temporelles à partir de données GTFS-RT brutes. Cela pose deux problèmes : (i) réduction de la reproductibilité — une même architecture deep learning entraînée sur des features différentes produira des résultats incomparables ; (ii) limitation de la transférabilité — les modèles entraînés sur une ligne ou un réseau peinent à généraliser à d'autres contextes sans recalibrage complet.

Deuxième lacune : scalabilité computationnelle non validée. La majorité des études évaluent leurs modèles sur des sous-ensembles de réseaux (une ligne, un corridor, une région). Peu travaillent à l'échelle d'un réseau urbain complet (centaines de lignes, milliers d'arrêts, millions de séquences d'observations). Les défis de scalabilité computationnelle — stockage volumétrique, parallélisation, latence d'inférence en temps réel — restent largement non adressés dans la littérature académique, ce qui explique le fossé entre les prédictions de recherche et les déploiements opérationnels.

Troisième lacune : déconnexion analytique diagnostic-prédiction. Les connaissances extraites de l'analyse diagnostique (patterns d'écarts récurrents, zones d'instabilité chronique, facteurs de perturbation) ne sont pas formellement canalisées vers l'ingénierie de features. Il n'existe pas de mécanisme systématique permettant à une plateforme diagnostique d'informer la sélection ou la construction des variables pour un modèle prédictif. Cette déconnexion est méthodologiquement préjudiciable : une bonne ingénierie de features devrait être guidée par une compréhension profonde des phénomènes diagnostiques.

1.4 État de l'art en prédiction des retards

Parallèlement aux avancées en diagnostic, les approches de prédiction des retards ont considérablement évolué au cours des deux dernières décennies. Cette section synthétise l'évolution des techniques, de l'utilisation des méthodes statistiques classiques aux architectures d'apprentissage profond contemporaines, en mettant l'accent sur les forces, limitations et lacunes de chaque paradigme.

1.4.1 Approches statistiques classiques

Les premiers modèles de prédiction de temps de parcours et de retards s'appuyaient sur les méthodes statistiques éprouvées, notamment ARIMA (AutoRegressive Integrated Moving Average), les filtres de Kalman et les approches Bayésiennes.

ARIMA et extensions. Les modèles ARIMA (Box, Jenkins, Reinsel & Ljung, 2015) exploitent l'autocorrélation temporelle intrinsèque aux séries de retards. Ils supposent que les retards observés récemment prédisent partiellement les retards futurs selon une structure linéaire paramétrique. Pour les trajets urbains présentant de forts motifs cycliques (heures de pointe, jours de la semaine, saisonnalité), ARIMA capture efficacement ces cycles lors de l'entraînement. Toutefois, ARIMA impose une hypothèse critique : la **linéarité**. Une augmentation unitaire d'une variable prédictive produit un effet constant, ce qui ne reflète pas la réalité du transport urbain où les effets sont hautement non-linéaires (par exemple, l'impact d'une augmentation de charge passagère varie selon la capacité du bus).

Filtres de Kalman. Les filtres de Kalman (Welch & Bishop, 2006) offrent une approche d'estimation dynamique d'état, particulièrement adaptée quand les observations bruitées arrivent en continu (typiquement le cas pour GTFS-RT). Ils modélisent explicitement le bruit de mesure et l'évolution de l'état, produisant des prédictions actualisées itérativement. Néanmoins, ils restent fondamentalement linéaires et exigent une spécification précise des modèles d'observation et de transition.

Forces et limitations. Les approches statistiques classiques offrent plusieurs avantages : interprétabilité complète des paramètres, besoin minimal de données historiques, et rapidité computationnelle permettant un déploiement temps réel. Cependant, elles peinent à capturer les non-linéarités complexes du transport urbain et ne modélisent pas explicitement les variables spatiales (topologie du réseau, interactions entre arrêts).

1.4.2 Machine Learning classique

À partir des années 2010, le machine learning classique a transformé la prédiction de retards en offrant une flexibilité pour capturer des non-linéarités complexes.

Support Vector Machines (SVM) et variantes. Les SVM (Vapnik, 1995) appliquent des transformations non-linéaires via noyaux implicites, permettant de capturer des relations complexes entre features et cibles. Plusieurs études les ont appliquées aux séries de trafic et de retards avec succès modéré (Chien, Ding & Wei, 2002a).

Forêts aléatoires et boosting. Les ensembles d'arbres décisionnels, notamment les forêts aléatoires (Breiman, 2001) et plus récemment XGBoost (Chen & Guestrin, 2016a), capturent les interactions non-linéaires multivariées et offrent une **interprétabilité des variables**. XGBoost en particulier s'est imposé comme approche baseline compétitive pour de nombreuses tâches de prédiction de retards, offrant un excellent compromis entre performance et vitesse d'entraînement. L'importance des variables exposée par XGBoost révèle souvent les prédicteurs réels (charge passagère, heure du jour, configuration réseau).

Limitations critiques. Bien que flexible, le ML classique capture les dépendances **statiquement** : chaque feature est traitée indépendamment pour chaque prédiction. Il n'existe pas de **mémoire explicite** du contexte temporel passé. Pour prédire l'écart à l'arrêt $t + 1$, les approches ML classiques utilisent les caractéristiques actuelles (x_t) mais pas explicitement la séquence historique $(x_{t-L}, \ldots, x_{t-1})$. Cela limite significativement la capacité à modéliser les dépendances temporelles long terme.

1.4.3 Deep Learning : architectures récurrentes

L'adoption du deep learning pour les séries temporelles, à partir du milieu des années 2010, a marqué un tournant paradigmatique vers les architectures avec mémoire explicite.

LSTM et GRU. Les LSTM (Long Short-Term Memory) (Hochreiter & Schmidhuber, 1997a) et GRU (Gated Recurrent Unit) (Cho *et al.*, 2014a) résolvent le problème du gradient vanissant des RNN classiques en maintenant des états internes à long terme. Plusieurs études les ont appliquées avec succès à la prédiction de retards urbains (Ma *et al.*, 2015; Liu & Chen, 2019b), capturant simultanément les variations courte durée (congestion ponctuelle) et les motifs long terme (heures de pointe). Les LSTM offrent une performance supérieure à ARIMA et ML classique sur la plupart des benchmarks.

Seq2Seq et attention. Les modèles séquence-à-séquence (Seq2Seq) avec mécanisme d'attention (Sutskever *et al.*, 2014b; Bahdanau *et al.*, 2015) étendent les LSTM pour la prédiction multi-horizon (plusieurs étapes futures). L'attention permet au modèle de « regarder » sélectivement les parties les plus pertinentes de l'historique, améliorant la prédiction et offrant une interprétabilité partielle.

Forces et limitations. Les approches RNN capturent efficacement les dépendances temporelles long terme et offrent des performances remarquables. Cependant, elles demeurent **strictement temporelles**, sans modéliser explicitement la dimension spatiale du réseau. De plus, elles sont **coûteuses computationnellement** (entraînement lent, inférence séquentielle) et exigent de **volumes massifs de données** pour une bonne généralisation. Enfin, elles sont **peu interprétables** : les poids cachés offrent peu d'insights sur les facteurs causaux des retards.

1.4.4 Deep Learning : architectures convolutives et Transformers

Des approches alternatives ont émergé pour adresser les limitations des RNN.

Temporal Convolutional Networks (TCN). Les TCN (Bai, Kolter & Koltun, 2018) appliquent des convolutions causales dilatées pour capturer des dépendances long terme tout en étant

complètement parallélisables, contrastant avec la nature séquentielle des LSTM. Avec une dilation exponentielle, un TCN construit un champ réceptif exponentiellement croissant, permettant de capturer des patterns multi-échelle (cycle horaire, hebdomadaire, saisonnier) sans nécessiter des séquences énormes en entrée.

Transformers. Les Transformers (Vaswani *et al.*, 2017) abandonnent la récurrence au profit de mécanismes d'auto-attention multi-têtes, permettant une parallélisation complète. Pour les séries temporelles, plusieurs variantes adaptées ont été proposées : Informer (Zhou *et al.*, 2021b) introduit l'attention parcimonieuse pour réduire la complexité quadratique ; Autoformer (Wu, Xu, Wang & Shang, 2021) ajoute une décomposition explicite en tendance et saisonnalité ; PatchTST (Liu *et al.*, 2023) utilise des patches temporels analogue aux patches en vision.

Forces et limitations. Ces architectures offrent une parallélisation supérieure au LSTM et souvent des performances compétitives ou meilleures. Cependant, elles demeurent **modèles temporels** sans spatial explicite. Les Transformers en particulier sont **extrêmement coûteux en mémoire** et complexes, rendant le déploiement temps réel problématique.

1.4.5 Approches spatio-temporelles : graphes et attention

Une évolution récente cruciale est l'émergence d'approches capturant **explicitement** la structure spatiale du réseau de transport.

Spatio-Temporal Graph Convolutional Networks (STGCN). Les STGCN (Yu, Yin & Zhu, 2018a) modélisent le réseau comme graphe où nœuds représentent les arrêts (ou zones) et arêtes les adjacences topologiques. Les convolutions graphiques permettent à chaque nœud d'agréger l'information de ses voisins, capturant les dépendances spatiales. Combinées avec des convolutions temporelles, elles créent une représentation spatio-temporelle unifiée. Les STGCN ont démontré de bonnes performances sur la prédiction de flux de trafic urbain (Li, Yu, Shahabi & Liu, 2019).

Graph Attention Networks (GAT). Les GAT (Velickovic *et al.*, 2018) étendent les GCN en apprenant des poids d'attention adaptatifs pour chaque voisin, plutôt que des agrégations uniformes. Cela permet au modèle d'apprendre que l'influence d'un arrêt amont varie selon le contexte (heure, météo, etc.).

Hybrid Spatio-Temporal Attention Models. Des travaux récents combinent GCN, RNN/Transformers et attention multidimensionnelle pour capturer dépendances complexes (Xue & Collaborators, 2025). Ces approches modélisent explicitement : (i) la topologie réseau (GCN), (ii) l'évolution temporelle (RNN/Transformer), (iii) l'importance relative des dimensions (attention).

Forces et limitations. Les approches spatio-temporelles capturent une structure riche et offrent souvent des performances supérieures. Cependant, elles introduisent de la **complexité architecturale** supplémentaire et exigent une **représentation graphique explicite** du réseau. Peu d'études valident à l'échelle d'un réseau urbain complet (centaines de nœuds, milliers d'arêtes).

1.4.6 Lacunes en prédiction scalable et déploiement opérationnel

Malgré les progrès techniques remarquables, trois lacunes majeures entravent le déploiement de solutions prédictives robustes à l'échelle urbaine complète.

Première lacune : évaluation fragmentée et non standardisée. La majorité des études évaluent les modèles sur des sous-ensembles restreints du réseau (une ligne, un corridor, une région). Peu travaillent sur le réseau complet avec validation temporelle stricte. Les métriques d'évaluation et les horizons de prédiction varient d'une étude à l'autre, rendant les comparaisons difficiles. Il n'existe pas de benchmark standardisé pour la prédiction de retards urbains, contrairement aux domaines du trafic routier ou de la météorologie.

Deuxième lacune : scalabilité computationnelle non validée. Les approches deep learning modernes (Transformers, STGCN complexes) nécessitent des ressources GPU substantielles

pour l'entraînement et peuvent souffrir de latence d'inférence élevée en temps réel. Pour un réseau urbain complet avec mises à jour fréquentes, la scalabilité computationnelle n'est pas documentée. Le coût total d'exploitation (TCO) de ces systèmes reste peu étudié.

Troisième lacune : absence de liaison diagnostic-prédiction. Les études de prédiction développent les modèles indépendamment de toute analyse diagnostique préalable. L'ingénierie de features reste ad hoc, guidée par l'intuition ou l'essai-erreur plutôt que par une compréhension systématique des phénomènes sous-jacents. Cela limite la reproductibilité et la transférabilité entre contextes urbains.

1.5 Synthèse générale et identification des lacunes

La revue de littérature précédente met en lumière un **paradoxe fondamental** dans l'état actuel de la recherche en prédiction de retards de transport urbain.

1.5.1 Le paradoxe : disponibilité technologique versus lacunes méthodologiques

D'un côté, les **technologies habilitantes** sont désormais largement disponibles et matures :

- **Standards de données** : GTFS et GTFS-RT diffusent des informations opérationnelles détaillées et actualisées en continu, accessibles gratuitement pour la plupart des agences urbaines ;
- **Outils géospatiaux** : l'indexation hiérarchique (H3) atteint des niveaux de performance et d'interactivité sans précédent, permettant l'agrégation multi-résolution fluide ;
- **Architectures d'apprentissage profond** : LSTM, Transformers, Graph Neural Networks démontrent des capacités prédictives remarquables, avec des performances dépassant souvent les approches classiques.

D'un autre côté, des **lacunes méthodologiques et organisationnelles** substantielles persistent, empêchant la transformation efficace de ce potentiel technologique en systèmes opérationnels robustes et déployables à l'échelle :

- **Fragmentation diagnostique** : aucune plateforme n'intègre unified ingestion GTFS-RT, classification systématique, indexation H3 et visualisation ;
- **Features ad hoc pour prédiction** : manque de cadre systématique et reproductible pour l'ingénierie de caractéristiques ;
- **Scalabilité non validée** : peu d'études valident à l'échelle urbaine complète (réseau entier) ;
- **Déconnexion diagnostic-prédiction** : les insights diagnostiques n'informent pas formellement la modélisation prédictive.

1.5.2 Le fossé recherche-pratique

Ces lacunes se manifestent précisément à l'**interface critique** entre deux mondes :

- **Monde académique** : contributions algorithmiques sophistiquées progressent continuellement (nouveaux modèles, techniques d'optimisation, benchmarks). Cependant, cette progression reste souvent orientée vers la maximisation de métriques de performance (RMSE, MAE) sans adresse des contraintes opérationnelles réelles (reproductibilité, transférabilité, scalabilité).
- **Monde opérationnel** : les agences de transport identifient des besoins pratiques urgents (diagnostic fiable, prédictions exploitables, systèmes déployables). Cependant, elles manquent de cadres systématiques et reproductibles pour adopter et déployer efficacement les innovations de recherche. Le coût et la complexité perçus des solutions de recherche freinent l'adoption.

Ce fossé explique pourquoi, malgré 15+ ans de recherche en prédiction de retards, peu de villes ont déployé des systèmes de prédiction intégrés et scalables. Les solutions restent largement ponctuelles, fragmentées et liées à des individus plutôt qu'intégrées dans des processus opérationnels robustes.

1.5.3 Justification du projet de recherche

Le projet de recherche proposé dans ce mémoire vise précisément à **combler ces lacunes** en structurant une approche intégrée qui articule diagnostic et prédiction de manière cohérente et reproductible.

Plus spécifiquement :

1. **Adresse de la lacune diagnostique** (Q1) : développer une plateforme unifiée d'analyse et de visualisation qui intègre GTFS-RT, classification H3 et indicateurs spatio-temporels ;
2. **Adresse de la lacune prédictive** (Q2-Q3) : concevoir un cadre systématique et reproductible d'ingénierie de features guidée par les insights diagnostiques, évalué à l'échelle réseau complet ;
3. **Validation pratique** (Q4) : démontrer l'utilité opérationnelle concrète et identifier les conditions de déploiement en production.

Cette approche double — diagnostic + prédiction — différencie ce mémoire des travaux précédents, typiquement focalisés exclusivement sur l'optimisation prédictive sans bénéficier des insights diagnostiques préalables.

1.6 Conclusion

Cette revue a retracé l'évolution des approches pour analyser et prédire les écarts d'horaire en transports en commun, depuis les méthodes statistiques traditionnelles jusqu'aux architectures contemporaines d'apprentissage profond. L'examen des technologies habilitantes (standards GTFS, indexation H3, LSTM et architectures apparentées) a posé les fondations théoriques et pratiques qui rendent possibles les avancées récentes.

L'analyse de l'état de l'art en diagnostic a révélé une littérature riche mais fragmentée, où diverses communautés abordent des composantes du problème — clustering spatial, classification statistique, visualisation interactive — sans intégration systématique. Parallèlement, l'examen

de l'état de l'art en prédiction a mis en évidence une progression impressionnante des capacités algorithmiques, contrastée par des lacunes méthodologiques persistantes en matière de reproductibilité, de scalabilité et d'intégration diagnostic–prédiction.

Ces lacunes ne reflètent pas des insuffisances scientifiques, mais plutôt l'évolution naturelle d'un domaine où l'innovation technologique (données, architectures) a progressé plus vite que les cadres méthodologiques unifiés. Fermer ce décalage exige une approche systématique qui :

- intègre l'ingestion automatique des flux GTFS-RT, la classification des perturbations, l'indexation spatiale H3 et la visualisation interactive au sein d'une plateforme diagnostique cohérente ;
- développe un cadre reproductible et systématique pour l'ingénierie de caractéristiques (features) multi-résolution, en remplaçant l'ad hoc actuel par une méthodologie exhaustive et documentée ;
- adresse les défis de scalabilité computationnelle en appliquant des stratégies de clustering et d'optimisation architecturale permettant la modélisation prédictive sur des réseaux urbains complets ;
- établit des mécanismes explicites de *feedback* entre l'analyse diagnostique et la modélisation prédictive, où les connaissances exploratoires informent systématiquement les choix architecturaux.

Ce programme de recherche, structuré autour de deux articles scientifiques formant un continuum cohérent du diagnostic à la prédiction, vise précisément à concrétiser ces objectifs. L'approche proposée ne prétend pas réinventer les composants individuels — classification, LSTM, H3 — mais les orchestrer de manière intégrée pour combler les lacunes identifiées et rapprocher la recherche académique des pratiques opérationnelles.

CHAPITRE 2

A DATA-DRIVEN PLATFORM FOR VISUALIZING AND ANALYZING PUBLIC TRANSIT SCHEDULE DEVIATIONS

Emna Boudabbous [a], Mohamed Karaa [a], Lokman Sboui [a], Julio Montecinos [a] and Omar Alam [b]

[a] Systems Engineering Department, École de technologie supérieure (ÉTS), University of Québec, Montréal, Canada

[b] Trent University, Peterborough, ON, Canada

Paper published in *2025 28th International Symposium on Real-Time Distributed Computing (ISORC)*, May 2025

ABSTRACT

This paper presents a methodology and an associated platform for classifying and visualizing schedule deviations in public transit systems. Applied to the Montreal public transit network, the work offers two main contributions : (1) a methodology integrating real-time transit data processing (GTFS-RT), systematic/stochastic deviation classification, and multi-resolution spatial analysis using the Hexagonal Hierarchical Spatial Index (H3) system (Resolutions 9 and 10) ; (2) an interactive visualization platform employing KeplerGL, built upon this methodology, enabling dynamic exploration of spatio-temporal deviation patterns. The platform reveals significant schedule deviations across Montreal's network. While preliminary, the visualized patterns provide data-driven evidence to generate actionable hypotheses and guide further investigation for transit optimization and urban planning, particularly in identifying delay-prone areas and understanding spatial delay propagation.

Keywords : Public Transit, Schedule Deviation, Delay Analysis, GTFS, GTFS-RT, Intelligent Transportation Systems (ITS), H3 Indexing, Spatial Analysis, Interactive Visualization, KeplerGL, Montreal.

2.1 Introduction

Effective public transit relies on schedule adherence, yet commuters frequently experience deviations impacting reliability Kumar, Kumar, Vanajakshi & Subramanian (2014); Hua, Wang, Wang & Ren (2018). While tools exist for route-specific analysis or general transit visualization, a gap exists for platforms that integrate network-wide, multi-resolution spatial analysis (like H3) with dynamic visualization (like KeplerGL) specifically designed to differentiate and explore both systematic (recurring) and stochastic (unforeseen) schedule deviations across an entire urban network.

This paper introduces such a platform, developed using General Transit Feed Specification (GTFS) and GTFS Realtime (GTFS-RT) data from Montreal's public transit system apt (Last Accessed : 2024). Our previous work Boudabous, Karaa, Sboui, Montecinos & Alam (2024) initiated this research by analyzing *six* major bus routes and introducing a method to classify delays as systematic versus stochastic. This paper significantly extends that analysis by : (1) Expanding data collection to encompass all operational bus routes in Montreal over a three-month period (September 16 - December 16, 2024), capturing comprehensive network-wide dynamics. (2) Detailing the methodology and platform architecture, which integrates multi-resolution H3 spatial analysis Li, Zhao & Zhang (2021b) with interactive KeplerGL visualization Ferreira, Martins & Lima (2021). This spatial dimension is crucial, as deviations often exhibit patterns linked to urban form and traffic conditions, and understanding these patterns is vital for informed planning.

The main contributions are :

1. A methodology integrating GTFS-RT data processing, systematic/stochastic deviation classification (building on Boudabous *et al.* (2024)), and multi-resolution H3 spatial indexing (Res. 9 for segments, Res. 10 for stops), specifically tailored for network-wide transit deviation analysis. The novelty lies in this specific integration for classifying and spatially visualizing both deviation types at multiple relevant scales.

2. An interactive KeplerGL visualization platform built upon this methodology. This platform enables dynamic exploration of spatio-temporal deviation patterns at both stop and segment levels, allowing users to filter by time, route, and potentially deviation type, offering analytical capabilities beyond static reports or single-resolution analyses.

The paper proceeds as follows : Section II discusses related work, positioning our contribution. Section III details the methodology, including data handling, platform architecture, deviation classification, H3 spatial analysis, and the visualization framework. Section IV presents findings from the Montreal case study. Section V discusses the results and limitations, and Section VI concludes with future work.

2.2 Related Work

Recent research increasingly combines spatial techniques with real-time transit data. The H3 spatial index Uber Technologies (2018); Kitchin & McArdle (2020), valued for its hexagonal grid and hierarchical properties, sees growing use in transportation Li *et al.* (2021b); Liu, Zhang & Chen (2020b); Lee & Chen (2020). Hexagons offer uniform adjacency and reduce sampling bias compared to traditional grids Kitchin & McArdle (2020). Li et al. Li *et al.* (2021b) and Liu et al. Liu *et al.* (2020b) utilized H3 for identifying general delay hotspots. However, their analyses often lacked differentiation between the types of delays (systematic vs. stochastic) and did not explicitly leverage H3's multi-resolution capabilities tailored to distinct transit elements (stops vs. segments). Our work addresses this by using Res. 9 for segments and Res. 10 for stops, integrating this spatial analysis directly with our deviation classification.

Analyzing schedule deviations often involves categorizing them into stochastic (unpredictable, e.g., accidents) and systematic (recurring, e.g., congestion, unrealistic schedules) Wessel & Widener (2018); Aemmer, Witt & Timpf (2019). Wessel and Widener Wessel & Widener (2018) examined schedule padding (a source of systematic earliness) in Toronto, while Aemmer et al. Aemmer *et al.* (2019) studied both delay types in Seattle. Our study applies this critical distinction across Montreal's entire network, focusing on

visualizing the spatial and temporal distributions of these classified deviations, building upon the classification method developed in our prior work Boudabous *et al.* (2024).

The visualization component of our platform is built upon KeplerGL Ferreira *et al.* (2021), leveraging its proven capabilities for interactive geospatial exploration Patel & Rodriguez (2021); Ferreira *et al.* (2021). Our key contribution is the targeted integration of KeplerGL with the preceding analytical steps : multi-resolution H3 indexing and systematic/stochastic deviation classification. This creates enhanced diagnostic power specific to transit operations. The platform allows analysts to move beyond identifying locations with poor performance to explore the characteristics of these deviations – distinguishing between potentially systemic issues and stochastic events, and discerning whether problems primarily manifest during travel between stops or at the stops themselves – thereby supporting more informed hypothesis generation about root causes within their spatio-temporal context.

By synthesizing these elements – real-time GTFS-RT data, systematic/stochastic classification, multi-resolution H3 indexing, and dynamic KeplerGL visualization – our work provides a more comprehensive and nuanced analytical framework compared to existing approaches that typically focus on only a subset of these components.

2.3 Methodology

Our methodology combines data processing, deviation classification based on Boudabous *et al.* (2024), spatial analysis using H3, and platform development for visualization.

2.3.1 Data Collection and Preprocessing

We utilize planned GTFS (containing schedules, routes, stops data) and real-time GTFS-RT (providing vehicle positions, trip updates including schedule deviations) data from the Société de Transport de Montréal (STM). Data was collected for all operational bus routes over three months (September 16 - December 16, 2024). GTFS-RT provides frequent updates, including the crucial *scheduleDeviation* field (in seconds) reported by vehicles at various points along

their routes. Throughout this paper, the terms 'bus line' and 'bus route' are used interchangeably, reflecting their common usage and representation within the STM's GTFS data for a specific defined service path.

2.3.2 Platform Architecture and Data Pipeline

The platform is built using primarily Python and leverages several open-source libraries. The technical workflow, illustrated in Figure 2.1, involves the following key steps :

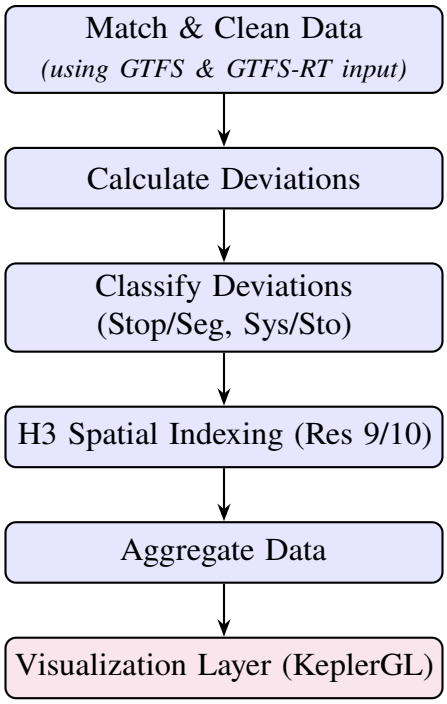

Figure 2.1 Main data processing flow.

1. **Data Ingestion :** Automated Python scripts ('requests' library) periodically query STM's GTFS-RT API (every 100 seconds) for real-time updates and download updated static GTFS data regularly to ensure accurate matching.
2. **Data Processing & Cleaning :** Raw protocol buffer data streams are parsed. Real-time vehicle position updates are matched to corresponding scheduled trips and stops using trip IDs and stop sequence numbers from the static GTFS. Basic data quality checks are

performed (e.g., removing duplicate updates). Data manipulation relies heavily on the 'pandas' library.

3. **Deviation Calculation :** For each real-time update associated with a stop arrival/departure, the schedule deviation is extracted or calculated : Deviation = Actual/Estimated Time - Scheduled Time.
4. **Deviation Classification (Stop & Segment) :**
   - **Stop-level :** Deviations at stops are categorized based on thresholds (see Section 2.3.3).
   - **Segment-level :** Deviations occurring between consecutive stops are calculated (Actual travel time - Scheduled travel time). These segment deviations are then classified as systematic or stochastic using the statistical methodology detailed in our prior work Boudabous *et al.* (2024), which analyzes the consistency and predictability of deviations over time for specific route segments.
5. **Spatial Indexing (H3) :** Geographic coordinates (latitude, longitude) of stops and representative points along route segments are mapped to corresponding H3 hexagons using the 'h3-py' library. We use resolutions 9 and 10 (see Section 2.3.4.2). Spatial operations may leverage 'geopandas'.
6. **Data Aggregation & Storage :** Processed data (deviations, classifications, timestamps, H3 indices, route/stop identifiers) is aggregated based on various dimensions (e.g., per H3 hexagon, per hour, per route, per deviation type). Results are typically stored in efficient formats like CSV or Parquet files, optimized for loading into the visualization tool.
7. **Visualization Layer (KeplerGL) :** KeplerGL is used to visualize the aggregated spatio-temporal data. It is typically integrated within a Python environment (e.g., Jupyter Notebook for analysis, or deployed via a web framework like Streamlit/Flask for a standalone tool). KeplerGL reads the aggregated data files and allows users to interact with predefined layers (base map, H3 grids colored by metrics) and filters (time slider, date selection, route filtering).

The novelty lies not in the individual tools but in their specific integration and application to provide multi-resolution, classified deviation analysis for transit networks.

2.3.3 Deviation Classification Details

Stop-level deviations are categorized based on seconds from schedule : **Too Early** ($< -60s$), **Early** ([-60s, -10s)), **On Time** ([-10s, 60s]), **Late** ((60s, 180s]), **Too Late** ($> 180s$). These thresholds, common in transit analysis, are configurable.

Segment deviations are classified as systematic/stochastic per Boudabous *et al.* (2024). Analyzing both levels is crucial : stop-level deviation is cumulative, reflecting preceding segment travel *plus* stop-specific events (dwell time, signal interactions). Segment deviation isolates inter-stop travel performance. Comparing them helps diagnose issues : high stop delay with low preceding segment delay suggests problems at many stops, whereas high segment delay naturally propagates to subsequent stops.

2.3.4 Spatial Analysis using H3

2.3.4.1 H3 Index Properties

We employ Uber's H3 Uber Technologies (2018); Li *et al.* (2021b) for its hexagonal, hierarchical grid system. Hexagons offer uniform adjacency (simplifying neighbor analysis) and reduce orientation bias compared to square grids Kitchin & McArdle (2020). The hierarchy enables seamless aggregation and analysis at different spatial scales.

2.3.4.2 Resolution Selection Rationale

We selected two specific H3 resolutions :

- **Resolution 9 (approx. 174m edge length) :** Chosen for analyzing route segments between stops. This scale aggregates data over a meaningful stretch of road, suitable for identifying patterns along corridors, bottlenecks near intersections, or delay propagation along adjacent road sections.
- **Resolution 10 (approx. 66m edge length) :** Chosen for stop-level analysis due to its finer granularity. A Res. 10 hexagon typically encompasses the immediate vicinity of a bus stop

or a small cluster of stops, allowing identification of localized issues like excessive dwell times or pinpointing delay hotspots at specific intersections or transfer points.

This multi-resolution approach allows tailored analysis for different aspects of transit operations.

2.3.5 Visualization Framework

We use KeplerGL for its web-based, interactive rendering of large geospatial datasets. Key features leveraged include :

- **Multi-layered Maps :** Base maps overlaid with H3 hexagon layers colored by deviation metrics (e.g., average delay, percentage late). Optional route or stop point overlays can be added.
- **Interactive Filtering :** Dynamic filtering by time range (time slider), day of week, specific routes, and potentially deviation type (systematic/stochastic).
- **Tooltips :** On-hover information displaying aggregated metrics (e.g., average delay, count of observations, H3 index ID) for specific hexagons or stops.
- **Time Animation :** KeplerGL's time playback feature allows animating the map to visualize the evolution of deviation patterns throughout the day or over longer periods.

This framework provides an intuitive interface for exploring complex spatio-temporal transit performance data, facilitating pattern discovery and hypothesis generation.

2.4 Case Study : Montreal City

Applying the platform to Montreal's network reveals significant operational patterns over the three-month study period.

2.4.1 Network-wide Deviation Analysis by Route

Figure 2.2 shows stop-level performance across routes, sorted by the combined percentage of 'Late' and 'Too Late' arrivals. This highlights routes with chronic lateness (e.g., Line 368 > 90%

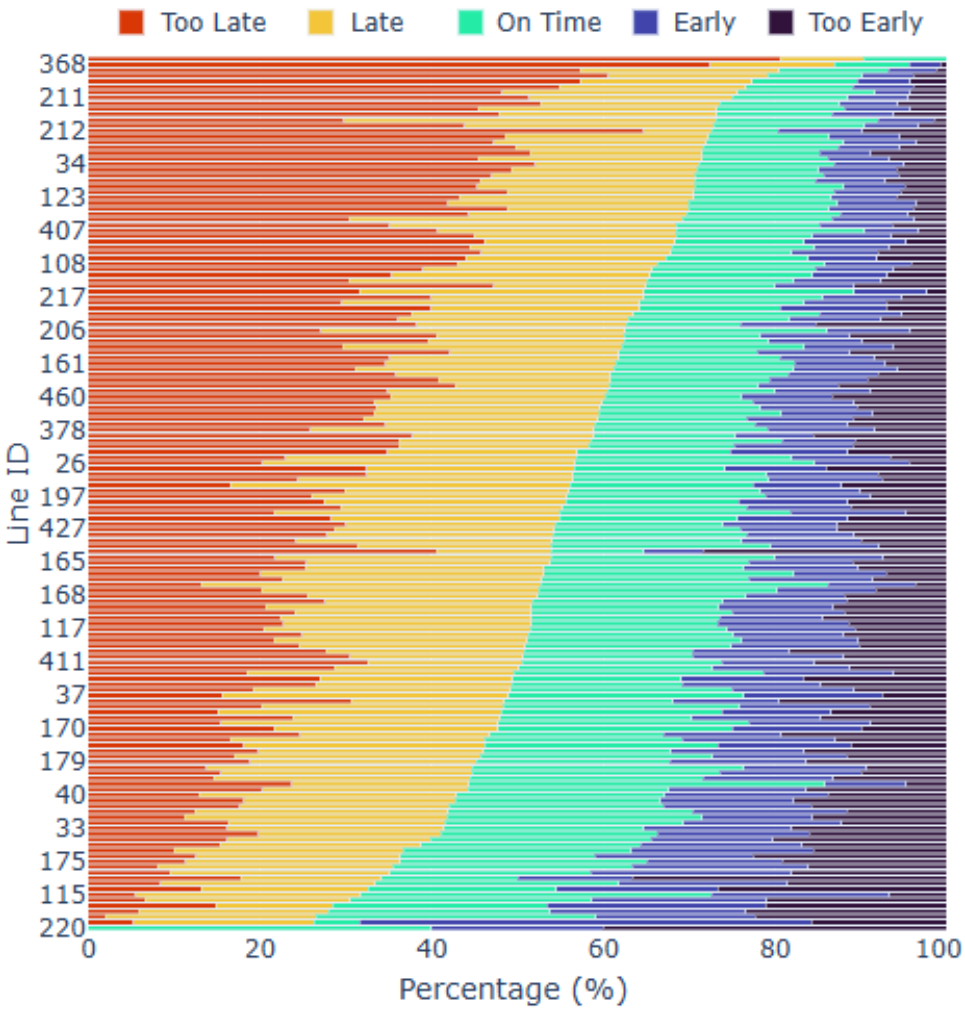

Figure 2.2 Stop-Level Deviation Classification by Route (Sorted by descending % of Late + Too Late). Highlights routes needing potential schedule/operational review.

late/too late) versus punctual ones (e.g., Line 220). This variability suggests routes like 368 are prime candidates for detailed operational review, potentially involving schedule adjustments or infrastructure assessment.

Figure 2.3 presents the same classification for deviations occurring *between* stops (segment level), sorted similarly. As expected, extreme deviation percentages are generally lower than at stops, visually supporting the accumulation of delays along routes amplified by stop-specific factors. Comparing Figures 2.2 and 2.3 helps differentiate routes where inter-stop travel is the main issue versus those dominated by stop-related delays.

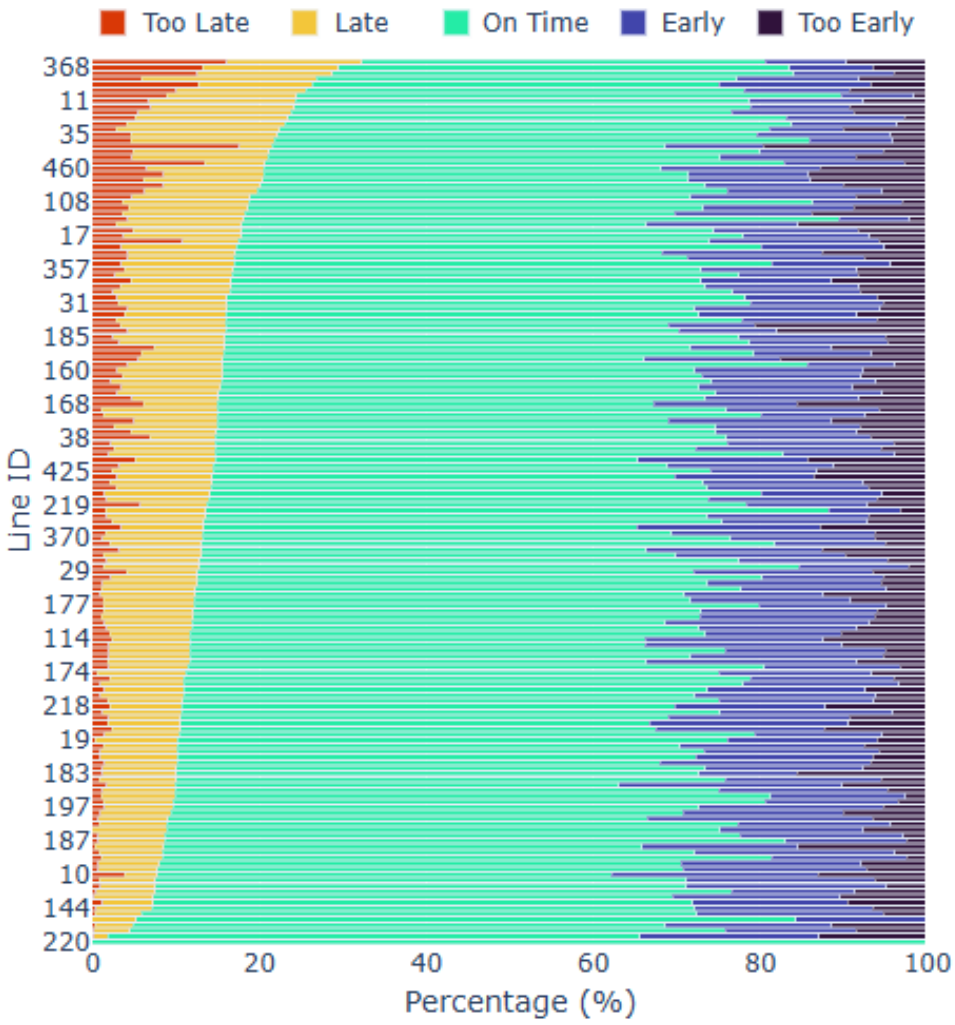

Figure 2.3 Segment-Level Deviation Classification by Route (Sorted similarly to Fig. 2.2). Shows inter-stop travel performance.

2.4.2 Temporal Deviation Patterns

Figure 2.4 illustrates the average distribution of total, systematic, and stochastic segment deviations by hour. Increased deviation activity during morning (approx. 7-10 am) and evening (approx. 5-8 pm) peaks strongly correlates with typical urban congestion. Stochastic deviations show sharper peaks, while systematic deviations also rise but may be more sustained, reflecting recurring congestion possibly accounted for partially in schedules. These patterns underscore

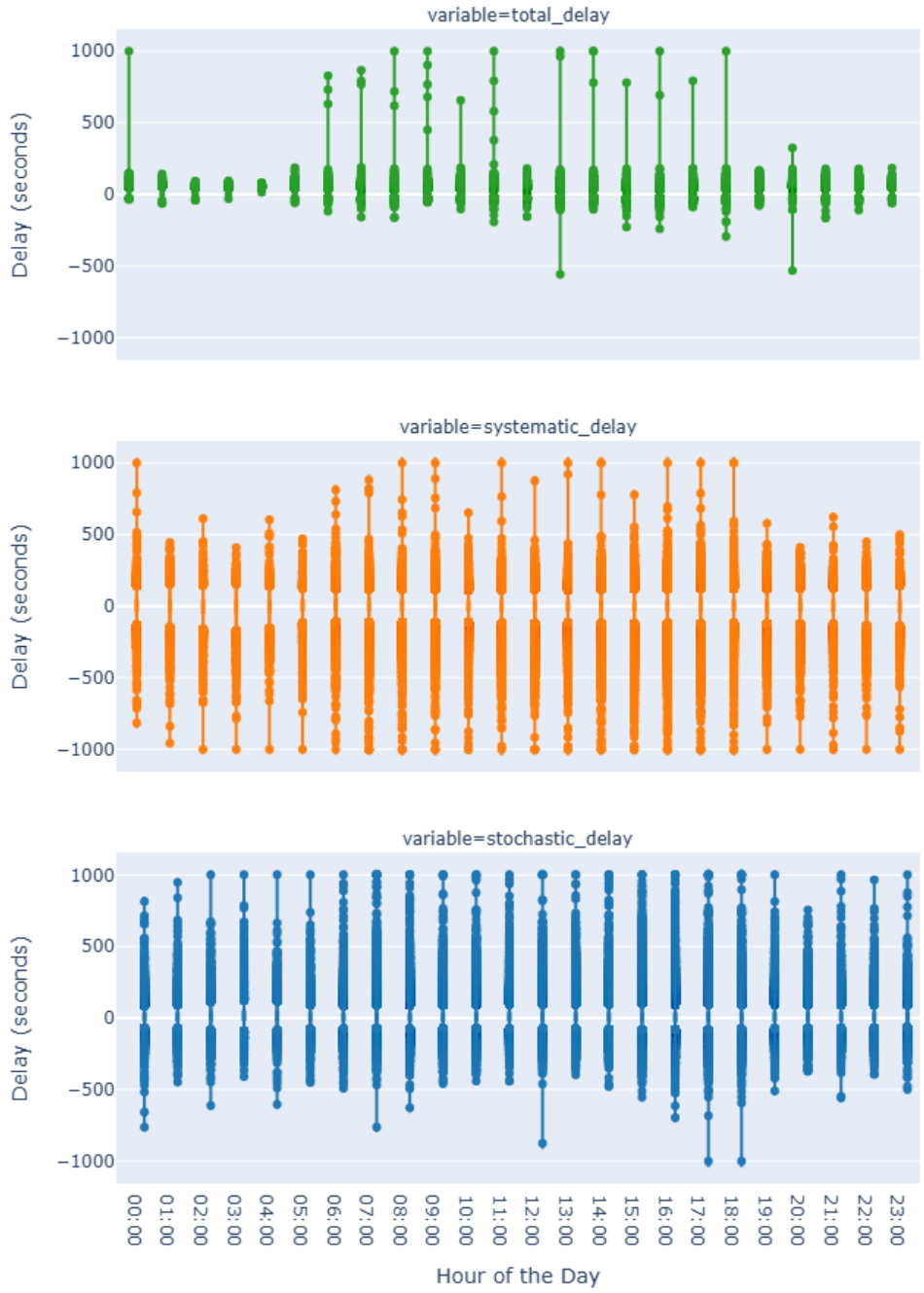

Figure 2.4 Average Segment-Level Deviation Distribution by Hour of Day. Shows clear peak hour impacts.

the need for time-dependent analysis and suggest areas for potential time-specific operational adjustments.

2.4.3 H3-based Spatial Deviation Analysis

2.4.3.1 Corridor Level Analysis (Res. 9)

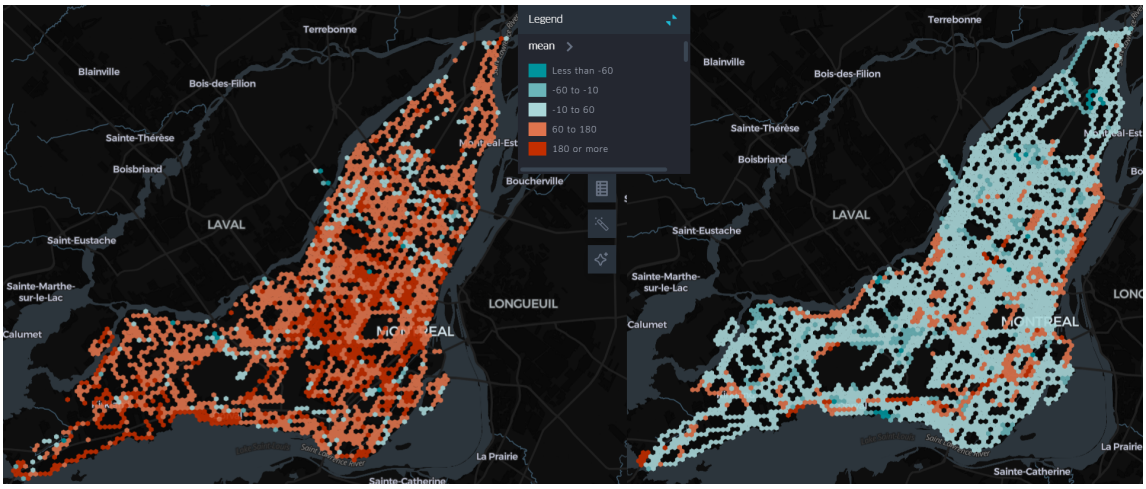

Figure 2.5 Spatial Distribution of Average Stochastic Delays (H3 Res. 9). Left : Segment-Level Delays. Right : Stop-Level Delays. Color scale indicates average delay (seconds). Highlights difference between corridor issues (left) and accumulated stop impact (right).

Figure 2.5 maps aggregated stochastic deviations using H3 Res. 9 hexagons. Comparing segment-level (left) and stop-level (right) aggregations reveals key differences. The stop-level map shows more intense hotspots (red >180s), particularly downtown, indicating where passengers experience the cumulative impact of delays. The segment-level map appears smoother, highlighting broader corridors with moderate stochastic issues (orange). This distinction aids in diagnosing whether problems lie primarily within road segments or are exacerbated at specific stops.

2.4.3.2 Stop Level Analysis (Res. 10)

Figure 2.6 uses H3 Res. 10 for a granular view of average stop-level deviations. This resolution pinpoints specific stops or small areas with severe delays (dark red), often concentrated downtown and along major arteries. This likely reflects high passenger demand (dwell times) and congestion

Figure 2.6 Spatial Distribution of Average Stop-Level Schedule Deviations (H3 Res. 10). Color scale indicates average deviation (seconds). Pinpoints specific problematic locations.

interacting locally. The concentration of severe delays in the downtown core strongly suggests that mitigating traffic congestion and optimizing passenger boarding/alighting processes in these specific zones should be a priority for the transit agency. Conversely, patterns of earliness (blue/cyan) in peripheral areas suggest potential schedule padding or overestimated travel times needing review.

While static figures for hourly spatial patterns were omitted for brevity, the platform enables dynamic exploration. Key observations from the interactive tool include widespread intensification of delays in central areas during peak hours (7-10 am, 5-8 pm), a noticeable

reduction midday, and significant earliness patterns emerging late at night in low-traffic zones. These hourly shifts highlight the interplay of traffic, demand, and schedules.

Explore the interactive map for dynamic visualization of hourly spatial patterns (Res. 10) : https://bit.ly/3Exiy7A

2.5 Discussion

This study demonstrates a platform integrating deviation classification, multi-resolution H3 indexing, and interactive KeplerGL visualization for analyzing Montreal's transit network deviations. The analysis reveals significant, consistent spatio-temporal patterns over the study period, presenting opportunities for investigation and potential optimization.

We acknowledge the limitations. This work is primarily diagnostic ; the platform serves to generate data-driven hypotheses. The observed patterns (e.g., chronic route delays, peak-hour hotspots, specific area earliness) represent starting points. Confirming root causes (e.g., specific intersection issues, construction impacts, unrealistic schedule segments, inefficient dwell times) and validating the operational benefits of potential interventions requires further investigation, ideally in collaboration with the transit agency (STM), incorporating their operational context and potentially external data (weather, events, etc.). This study does not perform such detailed root cause analysis or external data integration.

However, the platform provides significant value. The integrated methodology itself is a contribution to transit analysis toolkits. The clear visual and quantitative distinction between stop-level (Figs. 2.2, 2.6) and segment-level (Figs. 2.3, 2.5 left) deviations helps diagnose where delays originate and accumulate. The consistently higher severity of stop-level deviations underscores the importance of factors occurring *at* stops. The multi-resolution H3 approach (Res. 9 vs. Res. 10) allows examining patterns at both corridor and hyperlocal scales effectively. The temporal analysis (Fig. 2.4 and interactive exploration) confirms the predictable nature of peak-hour issues, suggesting systematic components amenable to operational strategies or schedule adjustments.

While requiring further validation, these data-driven observations themselves constitute significant results, providing spatially and temporally explicit evidence to guide planning efforts. The ability to simultaneously visualize deviation classifications (systematic/stochastic) spatially at multiple resolutions is key to generating these targeted hypotheses. The consistency over three months suggests the patterns reflect underlying structural or operational characteristics, making the platform a valuable tool for transit planners to guide resource allocation, identify areas needing schedule review, and prioritize interventions.

2.6 Conclusion and Future Work

This paper introduced a data-driven methodology and integrated platform for visualizing and analyzing public transit schedule deviations, demonstrated using Montreal's network. By combining systematic/stochastic classification, multi-resolution H3 spatial indexing, and interactive KeplerGL visualization, the platform effectively reveals complex spatio-temporal patterns. Key findings include identifying high-deviation routes and locations, confirming pronounced peak-hour effects spatially, observing persistent earliness patterns, and illustrating the difference between segment and stop-level deviations, thereby providing valuable diagnostic insights.

The platform serves as a valuable tool for generating data-driven hypotheses about transit reliability. Future work will focus on enhancing its analytical capabilities and addressing current limitations through several key avenues :

- **Validation and Causal Analysis :** Collaborate closely with the transit agency (STM) to validate findings against operational knowledge and investigate the specific root causes of identified deviation patterns using methods like field observations and operational log analysis.
- **Incorporate External Data :** Integrate relevant external datasets (e.g., detailed traffic, construction, events, passenger loads) to enable more sophisticated causal modeling and better distinguish systematic from stochastic influences on deviations.

- **Longitudinal and Seasonal Analysis :** Extend the data collection period to analyze performance across different seasons and identify long-term trends, assessing the stability of observed patterns.
- **Develop Predictive Capabilities :** Leverage the platform's rich spatio-temporal features as input for machine learning models aimed at short-term deviation prediction, potentially aiding real-time operations and passenger information.
- **Platform Enhancements :** Add features for direct quantitative comparison and correlation of deviation types (systematic/stochastic, stop/segment) within the interface, and potentially allow user-defined analysis zones.

In conclusion, this work presents an integrated framework that offers significant potential for transit agencies to gain deeper insights into network performance, identify problem areas more effectively, and guide efforts towards improving service reliability. While preliminary in terms of validation, the methodology and platform represent a step towards more data-informed transit planning and operations. The outlined future work aims to further develop this potential, ultimately contributing to an enhanced passenger experience.

CHAPITRE 3

SCALABLE TRANSIT DELAY PREDICTION AT CITY SCALE

Emna Boudabbous [a], Mohamed Karaa [a], Lokman Sboui [a], Julio Montecinos [a] and Omar Alam [b]

[a] Systems Engineering Department, École de technologie supérieure (ÉTS), University of Québec, Montréal, Canada

[b] Department of Computing and Information Systems, Trent University, Peterborough, ON, Canada

Paper submitted for publication in *Journal of Systems Architecture*, March 2026

ABSTRACT

Urban bus transit agencies need reliable, network-wide delay predictions to provide accurate arrival information to passengers and support real-time operational control. We present a city-scale prediction pipeline that combines multi-resolution feature engineering, dimensionality reduction, and deep learning. The framework systematically generates spatiotemporal features by exploring aggregation combinations over spatial regions (using H3), routes, segments, and temporal patterns, then compresses them using Adaptive PCA while preserving 95% of the variance. To avoid the "giant cluster" problem, we introduce a hybrid clustering method that combines geographic and network topology information to yield balanced route clusters. We compare four model architectures on six months of bus operations from the Société de transport de Montréal (STM) network. A global LSTM with cluster-aware features achieves the best trade-off (R^2 = 0.7121 at the elementary level), outperforming transformer models by 18–43% while using 275× fewer parameters. We report multi-level evaluation at elementary segment, segment, and trip level with walk-forward validation and latency analysis.

Keywords : Transit delay prediction, GTFS-Realtime, Deep Learning, Feature engineering, H3 geospatial indexing, Scalability, Urban mobility, Intelligent transportation systems.

3.1 Introduction

Urban public transportation systems are a cornerstone of sustainable mobility in metropolitan areas, carrying millions of passengers daily and helping reduce congestion, pollution, and energy consumption Elassy, Al-Hattab, Takruri & Badawi (2024). For bus networks in particular, **reliability**—defined as the consistency of bus service, measured by on-time performance (adherence to scheduled arrival times) and headway regularity (consistent spacing between buses)—is often degraded by multiple factors. Traffic congestion causes variable travel times as buses encounter unpredictable delays at intersections and along congested corridors Kumar, Singh, Shaji & Vanajakshi (2025a). Incidents such as accidents, vehicle breakdowns, or road closures create unexpected disruptions that propagate through the network. Adverse weather conditions (snow, rain, ice) affect road conditions and vehicle speeds, leading to increased variability in travel times Alam, Kush, Emami & Pouladzadeh (2021b). Operational constraints, including vehicle availability, driver scheduling, and maintenance requirements, can disrupt scheduled service patterns. These factors interact to create complex, non-linear delay patterns that are difficult to predict using simple rule-based systems.

Accurate real-time delay prediction has therefore become a key function of modern Intelligent Transportation Systems (ITS), as it directly affects service quality, passenger satisfaction, and operational efficiency Kumar *et al.* (2025a). Reliable predictions allow passengers to plan their trips better and reduce waiting time. They help agencies adjust operations, modify routes and frequencies, and improve service in ways that support public transport adoption Kumar *et al.* (2025a). As urban mobility demand grows, scalable, reliable delay-prediction systems are increasingly necessary.

While real-time data feeds such as General Transit Feed Specification Realtime (GTFS-RT) are now widely available, they provide only current vehicle positions and trip updates without predictive capabilities. GTFS-RT does not include predictive models—it only reports where vehicles currently are, requiring additional processing to convert positions into delay predictions. Moreover, GTFS-RT lacks the ability to learn from historical patterns and anticipate delays before

they occur, and it does not provide multi-level predictions (elementary segment, segment, trip) that support operational decision-making. **Artificial intelligence (AI) and machine learning (ML) solutions are therefore essential** to address these limitations : transit delay patterns are complex and non-linear, requiring machine learning to capture intricate spatiotemporal relationships that manual rule-based systems cannot handle. Deep learning models can learn from large-scale historical data to identify patterns that human experts might miss, and scalable AI systems can process millions of observations across entire networks, which is infeasible with manual approaches.

Despite advances in artificial intelligence (AI) and the widespread availability of real-time data through standards such as General Transit Feed Specification Realtime (GTFS-RT), several fundamental challenges still hinder the deployment of reliable, large-scale delay prediction systems Singh & Kumar (2022b).

First, feature engineering remains largely ad hoc. Many existing studies rely on manually designed feature sets driven by researcher intuition Pałys, Ganzha & Paprzycki (2022); Huo, Li, Zhao & Zhu (2018). Such approaches are difficult to reproduce, offer limited coverage of multi-resolution spatiotemporal patterns, and generalize poorly across different transit networks. The lack of a systematic, reproducible framework for feature generation is a central methodological gap.

Second, scalability and computational efficiency pose significant challenges for network-wide transit delay prediction. While many studies report strong performance on individual routes Subramaniyan *et al.* (2023); Petersen, Rodrigues & Pereira (2019b), few address the complexities of modelling an entire metropolitan transit system comprising hundreds of routes, thousands of stops, and millions of observations Li, Wolf & Wang (2024c); Rodriguez-Deniz & Villani (2022). Monolithic models that treat the network as a single, undifferentiated unit are both computationally intensive and ill-suited to capturing spatial heterogeneity, particularly the operational differences between dense urban cores and lower-density suburban zones. Spatial clustering has emerged as a practical strategy to address this issue by dividing the network

into subregions with similar operational profiles, thereby improving model efficiency and maintaining local delay patterns Shaji, Tangirala & Vanajakshi (2018). However, conventional clustering methods often rely on regular spatial grids that ignore the network topology, leading to severe data imbalances. In particular, high-density urban areas are frequently collapsed into disproportionately large clusters that dominate the dataset. We refer to this as the « giant cluster phenomenon » : a spatial-aggregation artifact in which dense regions are represented by oversized spatial units, resulting in spatial homogenization, obscured intra-urban variability, degraded model performance, and inefficient parallel training.

Third, much of the literature focuses on experimental prototypes with limited attention to deployment-oriented evaluation. Essential aspects such as inference latency, resource usage, systematic architecture comparison, and computational efficiency are rarely reported Kumar *et al.* (2025a). This makes it difficult for practitioners to select architectures that are feasible for real-time, network-wide deployment. We explicitly address these non-functional requirements in Section 3.5.2, reporting measured performance characteristics and explaining how our architecture supports scalability, maintainability, and operational reliability.

Recent technological advances open new opportunities to address these gaps. **Hierarchical spatial indexing systems**, particularly Uber's hexagonal hierarchical spatial indexing (H3) Brodsky (2018b), provide a flexible framework for multi-resolution spatial representation. H3 partitions the Earth's surface into nested hexagonal grids across 16 resolution levels (0–15) with decreasing diameter size, enabling seamless multi-resolution aggregation. Hexagons offer superior geometric properties compared to squares : uniform distance to six neighbours, minimal area distortion, and exact 1 :7 parent-child subdivision ratios. At different resolutions, H3 cells naturally align with transit operational scales : neighbourhoods, corridors, and street segments. This hierarchical structure allows transit patterns to be aggregated consistently at multiple scales, making H3 particularly well-suited for multi-resolution feature engineering in transit delay prediction. Yet the systematic use of H3 for transit delay prediction, especially in combination with rigorous multi-resolution feature engineering and topology-aware spatial clustering, remains relatively unexplored.

This paper presents an end-to-end pipeline for city-scale delay prediction, developed to meet the operational requirements of bus transit and to support the creation of a reusable architecture that can be readily adapted to other urban transit systems.

Our work pursues three interconnected research objectives : (1) to develop a reproducible framework for exhaustive multi-resolution spatiotemporal feature generation, (2) to design an efficient spatial clustering strategy for data organization and feature engineering that resolves the « giant cluster problem » while preserving network topology, and (3) to perform a rigorous comparative evaluation of multiple deep learning architectures, including computational analysis, on a complete metropolitan bus network.

The main contributions of this work are fourfold :

- **Systematic multi-resolution feature engineering framework.** We introduce a reproducible framework for spatiotemporal feature generation that systematically explores aggregation combinations across multiple spatial resolutions, route identifiers, segments, and temporal dimensions. The framework captures both local segment-level patterns and neighbourhood-level trends. We evaluate multiple dimensionality reduction methods and show that Adaptive PCA effectively compresses the feature space while retaining essential variance, making global model training computationally less demanding.
- **Hybrid H3+topology spatial clustering for data organization.** We propose a route clustering methodology based on weighted Jaccard similarity that combines spatial coverage with topological structure to organize network data and generate cluster-aware features. The method resolves the giant cluster problem by producing balanced clusters that support efficient feature engineering while preserving spatial and topological coherence for global model training.
- **Comparative evaluation of deep learning architectures.** We perform an extensive comparison of four architectures for elementary-segment bus delay prediction : LSTM (Long Short-Term Memory), XGBoost (gradient boosting trees), PatchTST (patch-based transformer), and Autoformer (autocorrelation transformer). All models share the same preprocessing and feature pipelines and are trained globally on the complete bus network.

For each architecture, we report predictive accuracy, training time, inference latency, and memory usage. Results show that LSTM with compressed features achieves the best overall performance (elementary-level $R^2 = 0.7121$), outperforming transformer-based models by 18 % à 43 % while maintaining favourable computational and latency profiles.

- **End-to-end pipeline for bus delay prediction with multi-level validation.** We implement a complete pipeline for bus delay prediction from data ingestion (GTFSstatic data, GTFS-RTfeeds, and weather data) to distributed feature engineering, dimensionality reduction, and global model training with multi-level prediction aggregation (elementary segment → segment → trip). The system uses walk-forward temporal validation and reports RMSE, MAE, and R^2 at all three levels, together with runtime and resource measurements. Multi-level aggregation exhibits error cancellation : trip-level RMSE (1.85 min for the LSTM model) is substantially lower than elementary-level errors, supporting the suitability of the hierarchical framework for operational deployment.

The remainder of this paper is organized as follows. Section 3.2 reviews related work in transit delay prediction, feature engineering, and distributed computing for spatiotemporal data. Section 3.3 presents our methods, including the multi-resolution feature engineering framework, hybrid spatial clustering strategy, and model architectures. Section 3.3.10.0.4 details the technical implementation, infrastructure choices, hyperparameter configurations, and experimental results. Section 3.5.3 discusses implications and future research directions.

Motivation and Problem Setting

This paper presents an end-to-end pipeline for city-scale delay prediction, developed to address the operational requirements of bus transit systems and to support the creation of a reusable architecture that can be readily adapted to other urban networks. The pipeline is demonstrated using data from the Société de transport de Montréal (STM). Control center operators must identify delayed vehicles, understand where delays accumulate, and decide when to intervene, for example, by holding vehicles, short-turning trips, or adding extra buses. At the same time, passengers rely on predicted arrival times exposed through various information channels. When

these predictions are unreliable or unstable, both operators and passengers quickly lose trust in the information system.

Our goal is to provide delay predictions that are reliable at the scale of the entire STM bus network and that can be used at several spatial and temporal granularities. To achieve this, the underlying data and models are organized around a small set of basic units :

- A **trip** is a single scheduled bus journey, as defined by its GTFS `trip_id`, from its origin terminal to its destination terminal.
- A **segment** is a directed pair of consecutive stops $(\text{stop}_i, \text{stop}_{i+1})$ on a route. Segment travel time is computed from GPS observations using geofences around each stop Boudabbous, Karaa, Sboui, Montecinos & Alam (2024).
- Each segment is subdivided into fixed-length **elementary segments**. Predictions are expressed at this elementary level as **pace**, i.e., travel time per meter (seconds/meter), which normalizes for segment length and yields a more homogeneous prediction target across the network Boudabbous *et al.* (2024).

These definitions ensure that delay predictions are comparable across routes of different lengths and geometries, and they provide a clear link between the model outputs and quantities of direct operational interest, such as segment and trip-level delays Boudabbous *et al.* (2024).

A commonly identified requirement in the context of large urban transit systems is the need for a *reusable* architecture that can generalize across networks. Building a bespoke prediction system for a single city or corridor is feasible, but maintaining it is difficult as routes change, new data sources are added, or evaluation requirements evolve. It is also hard for other agencies to reuse such a system if its components are tightly coupled to local assumptions. To address this, our design separates city-specific configuration (GTFS feed, H3 resolution, clustering parameters) from generic components (feature generation, dimensionality reduction, global modelling, and inference). The same pipeline can therefore be adapted to other bus networks

by changing a limited set of configuration files and retraining the models, while preserving the overall structure.

In summary, the motivation for this work is twofold : (i) to support real operational decision making through delay predictions defined consistently at multiple spatial and temporal scales, and (ii) to do so with a scalable, reusable architecture that can be transferred across bus networks with limited engineering effort.

3.2 Related Work

Public transit delay prediction has been extensively studied and remains a relevant problem, given the continuous evolution of transportation systems and their increasing complexity. In this section, we summarize pertinent prior work on delay prediction models, feature engineering methods, and challenges related to scalability and computational costs.

3.2.1 Transit Delay Prediction

Early approaches to transit delay prediction relied on simple statistical assumptions, focusing primarily on historical averages and scheduled timetables. With the increasing availability of real-time sensor data and advancements in machine learning, models have gradually evolved toward data-driven and context-aware frameworks Boudabbous *et al.* (2024).

Statistical approaches include time series analysis using ARIMA Suwardo, Napiah & Kamaruddin (2010), Kalman Filters Achar, Bharathi, Kumar & Vanajakshi (2020), mixture models Chen, Saidi & Sun (2025a), Bayesian networks Büchel & Corman (2021), and Markov models Sun, Spall, Wong & Zhao (2025). Although these methods are easier to implement and interpret, and require less computational resources and data, they often struggle to capture the complex underlying patterns of traffic dynamics and to adapt to irregularities such as service disruptions and sudden events.

The proliferation of real-time transit data collected by automatic vehicle location (AVL) systems has encouraged the development of advanced deep learning models, enabling more accurate delay predictions. The models trained with GPS historical data include linear regression Sinn, Yoon, Calabrese & Bouillet (2012a), support vector machines Marković *et al.* (2015), and gradient boosting algorithms Chtioui *et al.* (2024); Warnakulasuriya, Weerasinghe, Wickramarathna, Ratneswaran & Thayasivam (2024). However, these methods require meticulous feature engineering and selection for better results.

More recently, deep learning models have become the norm for processing time-series and network data to capture complex, latent spatio-temporal patterns in road networks. Recurrent neural networks (RNNs) have become the go-to model architecture for delay prediction because they can handle complex sequential data. In Alam *et al.* (2021b), Alam et al. built a hybrid RNN-LSTM model to predict bus arrival irregularities based on historical GPS positions and weather data for two Toronto bus service routes. Liu et al. Liu *et al.* (2020a) employed an LSTM model to predict both short and long-distance arrival times of buses to the station. Another work used BiLSTMs to estimate the bus departure time for each route, then combined them with a DNN to calculate travel time Rong *et al.* (2022). These works have leveraged the temporal aspect of public transit dynamics to predict delay and arrival times.

In other studies, the authors have also used spatial patterns in traffic networks via Graph Neural Networks (GNNs) to improve the modelling of the delay prediction problem. In Lopes, Gramaglia, Bacciu & Marques-Neto (2025), the authors combined an LSTM with a GNN that captures the static characteristics of the public transit network. This hybrid approach improved the prediction accuracy compared to using temporal and spatial features separately. Sharma et al. proposed a Temporal Graph Convolutional Network that predicts delays across a segmented road network and then aggregates them to estimate the bus arrival time at a given station Sharma *et al.* (2024). Another study proposed a Graph Attention Network that learns dynamic correlations between bus routes and then combines them with spatio-temporal features via an attention mechanism. This significantly improves the bus delay estimation, especially for bus routes with fewer representations in the road network Rong *et al.* (2023).

3.2.2 Feature Engineering for Transit Delay Prediction

Feature engineering remains the cornerstone of effective predictive modelling for transit systems. While deep learning-based methods often avoid manual feature design through end-to-end learning, empirical evidence consistently shows that well-engineered features substantially improve performance, even for neural architectures.

Conventional feature engineering techniques include several categories, such as temporal, spatial, and operational (contextual) features. Temporal features such as hours of the day, day of the week, seasonality, and holiday indicators form the foundation of most prediction models Rong *et al.* (2022); Li *et al.* (2023); Rodriguez-Deniz & Villani (2022). Effective temporal encoding requires domain knowledge to identify relevant periodicity and interaction terms. For example, rush-hour behaviour differs fundamentally between weekdays and weekends, necessitating interaction features. On the other hand, spatial information is represented as raw latitude/longitude coordinates Lopes *et al.* (2025); Li *et al.* (2023) or higher-level features such as regions, neighbourhoods, and street types Rong *et al.* (2023). However, these representations suffer from critical limitations : coordinates lack semantic structure exploitable by tree-based models, while administrative zones exhibit arbitrary boundaries and high spatial heterogeneity.

Operational features provide more context on bus speed, passenger load, and traffic, thereby enriching the input data Liu *et al.* (2020a); Rong *et al.* (2023, 2022). However, this data is often unavailable across different transit agencies. Other works have integrated weather data as input features to improve the prediction, as weather conditions affect road conditions and mobility choices Kaya & Utku Kalay (2025). In Alam *et al.* (2021b), the authors improved bus arrival time prediction by 48 % by including weather data to train their LSTM model.

More recent studies have used spatial clustering to generate more granular and robust features for multiple traffic analysis and forecasting tasks Shaji, Tangirala & Vanajakshi (2022); Cebecauer, Jenelius & Burghout (2018). More particularly, Uber's hexagonal spatial indexing has revolutionized the spatial partitioning techniques. H3 partitions the Earth's surface into nested hexagonal grids across 16 resolution levels (0–15) with decreasing diameter size, enabling

seamless multi-resolution aggregation Brodsky (2018b). Hexagons offer superior geometric properties compared to squares : uniform distance to six neighbours, minimal area distortion, and exact 1 :7 parent-child subdivision ratios. At resolutions 9 (174 m), 10 (66 m), and 11 (25 m), H3 cells naturally align with transit operational scales, namely neighbourhoods, blocks, and street segments, respectively. Multiple works have used H3 indexing for traffic analysis and travel time estimation Sharma *et al.* (2024); Gramacki, Woźniak & Szymański (2021); Wang *et al.* (2025). In these works, H3-based clustering helped identify distinctive features of public transportation, such as service quality, density, and frequency.

3.2.2.1 Critical Gap : Ad-Hoc Feature Selection

A fundamental limitation across the existing literature is the **ad hoc, non-systematic nature of feature engineering**. Studies typically report 50 à 200 manually selected features with limited documentation of their selection criteria, making results difficult to compare and build upon Petersen *et al.* (2019b). This approach is inherently vulnerable to researcher bias and risks incomplete coverage of critical spatiotemporal interactions. Ultimately, these shortcomings undermine both the **reproducibility** of the findings and the **generalization** of the features to other transit networks, creating a significant barrier to progress in the field.

Despite H3's demonstrated effectiveness in mobility analytics, its systematic exploitation for **multi-resolution feature generation** in transit delay prediction remains largely unexplored. Most studies employ spatial features at a single resolution or rely on ad hoc manual selection, failing to capture the hierarchical nature of delay propagation, in which local segment-level disruptions interact with broader neighbourhood-level congestion patterns. To our knowledge, no prior work systematically explores multi-resolution feature generation across multiple H3 resolutions combined with comprehensive temporal encoding through automated aggregation combinations.

3.2.3 Scalability Challenges and Distributed Computing

Operational transit delay prediction for a metropolitan bus network requires processing millions of observations per day across hundreds of routes and thousands of stops. This volume typically exceeds the capacity of single-machine workflows, motivating the use of distributed computing frameworks.

Apache Spark is widely used for large-scale transit data processing because it offers in-memory computation, high-level DataFrame APIs, and distributed machine learning through MLlib Chambers & Zaharia (2018). For example, Lv et al. processed 50 million daily taxi GPS trajectories in Beijing with second-level latency using Spark clusters. Other frameworks, such as Dask and Ray, target similar workloads with different design trade-offs, emphasizing tight Python integration or low-latency task scheduling. In this work, we adopt Spark for feature engineering and model training, as it integrates with STM's data infrastructure and can handle the scale of our datasets.

A major scalability challenge for network-wide modelling is spatial heterogeneity : delay patterns differ substantially between dense downtown corridors, suburban residential areas, and industrial zones. Treating the entire network as a single global context leads to (i) computational intractability on multi-million-row datasets with thousands of features, (ii) limited ability to learn location-specific patterns because local signals are drowned in global noise, and (iii) prohibitive training times. Spatial partitioning addresses these issues by decomposing the network into geographic clusters and training specialized models in parallel. However, naive partitioning using regular grids or coarse H3 cells can lead to severe data imbalance. Dense urban cores form a single, large cluster that contains most observations, while peripheral regions form many small clusters with sparse data. We refer to this as the *« giant cluster problem »*. Our empirical analysis in Section 3.3.6 shows that naive H3 partitioning at resolution 8 yields a coefficient of variation (CV) above 2.0 and imbalance ratios above 40×, whereas our hybrid H3+topology approach achieves CV = 0.608 and an imbalance ratio of 1.90×.

Topology-aware clustering strategies for transit prediction remain limited. Existing spatial clustering methods typically rely only on geographic proximity, ignoring transit-specific structure such as shared route segments and operational patterns. As a result, geographically close but operationally distinct segments may be grouped, while segments that share the same routes may be split across clusters. To our knowledge, no prior work combines H3-based spatial partitioning with an explicit topological similarity measure to improve cluster balance and operational coherence for distributed training jointly. This gap motivates the hybrid H3+topology clustering method introduced in Section 3.3.6.

Finally, production systems must balance prediction accuracy, inference latency, and resource consumption. For real-time passenger information, latency budgets are often on the order of tens of milliseconds, which can make highly complex models impractical even if they are more accurate. Techniques such as model distillation, feature pruning, quantization, and cascaded architectures have been proposed to manage these trade-offs. Petersen et al. explicitly document this tension : an LSTM model attains lower RMSE than linear regression but at the cost of an order-of-magnitude higher latency ; in strict real-time settings, the simpler model may therefore be preferred despite a 26 % accuracy loss Petersen *et al.* (2019b). In our evaluation, consequently, we report not only accuracy metrics (RMSE, MAE, R^2) but also training time and inference latency to assess whether candidate architectures are feasible for deployment.

3.2.4 Validation Methodologies and Production Deployment

3.2.4.1 Evaluation Protocols

Most transit delay prediction studies employ some form of cross-validation, but temporal dependencies require specialized protocols. Walk-forward (rolling-origin) validation has been recognized as more appropriate for time series data because it respects causal ordering and avoids leaking future information into the training set Vlahogianni, Karlaftis & Golias (2014). Nevertheless, many works still rely on random train–test splits, which can overestimate performance and complicate comparisons across studies. Reported metrics also vary considerably :

RMSE, MAE, and MAPE are most common, with occasional use of R^2 and classification-style metrics (e.g., on-time vs. late). This heterogeneity across protocols and metrics limits the ability to compare models fairly and assess whether a given approach is suitable for deployment.

3.2.4.2 Production Deployment and Reusable Architectures

Beyond offline evaluation, a smaller body of work considers deployment aspects of data-driven transit prediction systems. Several papers describe prototype real-time arrival prediction services integrated with passenger information systems or control centers, often using microservice-based backends or streaming frameworks for ingesting AVL and GTFS-RT data (e.g., Petersen *et al.* (2019b); Li *et al.* (2024c)). These systems typically emphasize latency and robustness requirements and sometimes discuss monitoring and retraining strategies. More general ITS frameworks also propose modular architectures for processing streams of transportation data and serving predictions to multiple applications Chambers & Zaharia (2018); Vlahogianni *et al.* (2014).

However, most of these deployments are tailored to a single city, operator, or corridor and do not aim to provide a reusable, network-agnostic architecture. The feature engineering pipelines, data organization strategies, and model-selection procedures are often tightly coupled to local assumptions and rarely documented at a level that would allow other agencies to adopt them directly. In particular, we find little work that (i) treats feature generation, spatial clustering, dimensionality reduction, and model training as components of a configurable architecture, and (ii) evaluates this architecture explicitly in terms of both predictive performance and computational characteristics (training time, inference latency, and resource usage).

This gap motivates the system-level focus of our work. Rather than proposing yet another isolated prediction model, we aim to design and evaluate an end-to-end, reusable architecture for large-scale bus delay prediction that can be adapted to different networks with limited configuration changes.

3.2.5 Summary and Research Gaps

Our review of the literature highlights four research gaps that motivate this work.

Gap 1 : Non-systematic feature engineering. Most existing studies rely on ad-hoc, manually designed feature sets, typically containing 50 à 200 features Petersen *et al.* (2019b). The researcher's intuition often drives feature selection and is only briefly documented, limiting reproducibility and transferability. While some recent works use H3 for spatial indexing Sharma *et al.* (2024); Wang *et al.* (2025), the systematic use of multi-resolution spatial features remains limited. We found no study that combines multi-resolution H3 aggregation (e.g., resolutions 9 and 10) with a structured exploration of temporal encodings through automated aggregation combinations.

Gap 2 : Unresolved scalability challenges. Many contributions target a single route or a small subset of the network Subramaniyan *et al.* (2023); Petersen *et al.* (2019b). The few network-wide studies that exist either train a single monolithic global model (leading to prohibitive training times and spatial dilution of local patterns) or apply naive spatial partitioning schemes that create severe data imbalance between clusters. In particular, we did not find prior work that combines H3-based geographic partitioning with an explicit topological similarity measure to obtain balanced spatial clusters and to address the « giant cluster problem » observed in dense urban cores.

Gap 3 : Limited guidance on operational and reusable architectures. Most papers present experimental prototypes with little discussion of how the proposed methods could be embedded in an operational system. Production-related aspects such as inference latency, resource constraints, monitoring, model retraining, and integration with existing transit management tools are rarely covered. Moreover, architectures are typically described for a single city or corridor, with tightly coupled preprocessing and modelling components, which makes reuse by other agencies difficult. There is a lack of work that treats feature generation, spatial clustering, dimensionality reduction, and model training as configurable components of a reusable architecture for large-scale bus delay prediction.

Gap 4 : Insufficient validation rigour. Many studies employ simple temporal holdout splits without walk-forward cross-validation, limited hyperparameter optimization, and few ablation studies. As a result, it is often unclear which feature families or model components drive performance gains. Long-term evaluation, robustness under atypical conditions (e.g., disruptions or extreme weather), and detailed reporting of experimental protocols are also uncommon, making it hard to compare approaches or assess their suitability for deployment.

3.2.5.1 Comparative Analysis of Prior Work

Table 3.1 summarizes key characteristics of representative prior studies, highlighting their methods, features, scalability characteristics, and limitations. This comparison reveals the diversity of approaches in transit delay prediction while identifying common limitations that motivate our work.

The comparison reveals several common limitations across prior work : (1) most studies focus on single routes or small subsets, limiting network-wide scalability, (2) feature engineering remains largely ad-hoc with manual selection and limited documentation, (3) model interpretability is rarely addressed, making it difficult to understand prediction drivers, (4) sensitivity to noisy GPS data and sparsity in low-frequency routes is often not discussed, and (5) production deployment considerations (latency, resource usage, reusable architectures) receive limited attention. Our work addresses these limitations through systematic feature engineering, hybrid clustering for scalability, comprehensive model comparison, and explicit consideration of deployment requirements.

Tableau 3.1 Comparative Analysis of Prior Work in Transit Delay Prediction

Study	Method	Features	Scalability	Limitations
Alam et al. Alam *et al.* (2021b)	Hybrid RNN-LSTM	GPS, weather	2 routes	Limited to few routes ; manual feature selection
Liu et al. Liu *et al.* (2020a)	LSTM	Temporal, GPS	Single route	Route-specific model ; no network-wide evaluation
Sharma et al. Sharma *et al.* (2024)	Temporal GCN	Graph structure, temporal	Network-wide	Requires graph construction ; limited feature engineering
Petersen et al. Petersen *et al.* (2019b)	LSTM, Linear	Manual features	Multiple routes	Ad-hoc features ; latency-accuracy trade-off
GNN-based Lopes *et al.* (2025)	LSTM+GNN	Static network, temporal	Network-wide	Complex architecture ; scalability concerns
Graph Attention Rong *et al.* (2023)	GAT with attention	Spatio-temporal, route correlations	Network-wide	Sensitivity to noisy GPS ; sparsity in low-frequency routes
H3-based Sharma *et al.* (2024); Wang *et al.* (2025)	Various with H3	H3 spatial features	Network-wide	Single H3 resolution ; limited multi-resolution exploration
Statistical Suwardo *et al.* (2010); Achar *et al.* (2020)	ARIMA, Kalman	Historical averages	Single route	Cannot capture complex patterns ; limited adaptability

3.3 Methodology

This section describes the architecture and methods used for bus delay prediction. Our system is organized as five interconnected pipelines : (i) data collection, (ii) feature engineering, (iii) dimensionality reduction, (iv) predictive modelling, and (v) inference. Figure 3.1 provides an overview of these components and their interactions with external data sources and data stores.

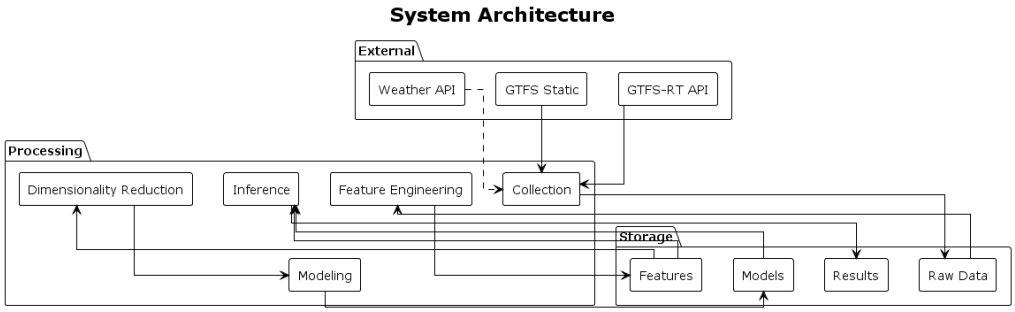

Figure 3.1 System architecture : five pipelines with external data sources and storage.

3.3.1 Data Collection Pipeline

The data collection pipeline converts heterogeneous transit data sources into trip-level records suitable for machine-learning models (Figure 3.2). The architecture follows a layered design with an orchestration layer managing control flow (scheduling, API polling, batch triggers) and a processing layer handling data transformation. GTFS static schedules provide the network topology (stops, routes, and stop sequences), while GTFS-RT vehicle position updates supply real-time observations. We align these sources in time, associate vehicle positions with scheduled trips, and enrich each record with city-wide weather information. The orchestration layer polls the GTFS-RT API every 10 s and triggers asynchronous batch writes to storage, enabling continuous collection without blocking. Schedule updates are handled by storing successive GTFS snapshots together with their validity periods and linking observations to the corresponding schedule version. Details of collection frequency and aggregation procedures are provided in Section 3.4.3.1.

3.3.1.1 Trip-level processing

We define a *trip* as a single scheduled bus journey from its origin terminal to its destination terminal, identified by its GTFS `trip_id`. The goal of the trip-level processing step is to transform the raw stream of GTFS-RT vehicle positions into one record per realized trip with consistent timestamps and delay values.

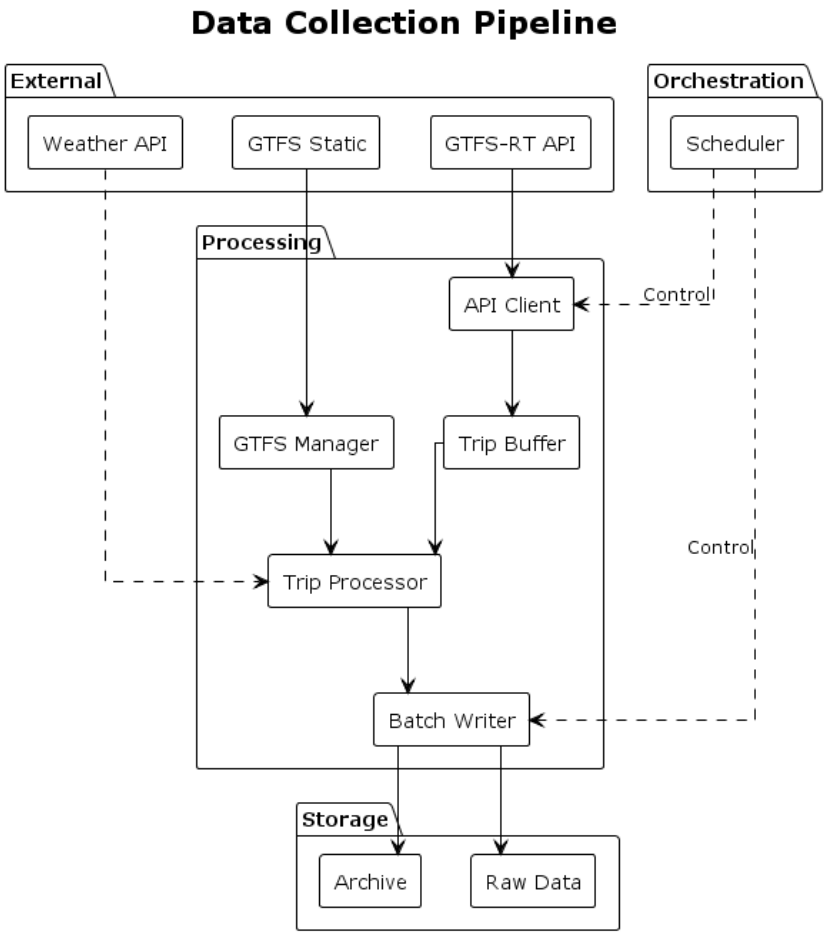

Figure 3.2 Data Collection Pipeline : external APIs to storage.

Starting from the GTFS static feed and the service calendar, we first expand all scheduled trips for each service day. GTFS-RT vehicle positions are then matched to these trips planned using the vehicle identifier, route, direction, and timestamp. For each matched trip, we build an ordered sequence of observations along the stop sequence and estimate arrival times at stops using geofences around stop locations. Delay at a stop is computed as the difference between the observed and scheduled arrival times.

We apply a set of rule-based quality checks to filter unreliable records. Typical checks include discarding trips with too few observations, non-monotonic timestamps, unrealistically high speeds between consecutive points, or locations that deviate excessively from the scheduled route.

Records that fail these checks are excluded from subsequent stages. Weather information for the corresponding time interval and area is then joined to each valid trip record. Implementation details for the matching procedure, alignment windows, and filtering thresholds are provided in Section 3.4.3.1.

3.3.2 Feature Engineering Pipeline

The feature engineering pipeline transforms trip-level records into a feature matrix for modelling (Figure 3.3). Instead of choosing features manually, we use a fixed procedure that applies the same set of aggregation operations for every experiment, so that the resulting feature set is well defined and easy to reproduce.

As introduced in Section 3.3.1.1, each realized trip is represented as an ordered sequence of segments, where a segment is the directed pair of consecutive stops on a route. For modelling, we further subdivide each segment into fixed-length *elementary segments* and define the prediction target as elementary-segment *pace* (seconds per meter). Pace is less sensitive to segment length than raw travel time and provides a more homogeneous target across the network.

On top of this target, we construct spatiotemporal features by aggregating historical values over different grouping strategies. The aggregation framework combines spatial groupings (H3 cells at resolutions 9 and 10, route and segment identifiers), temporal groupings (hour of day, broader time-of-day periods, and per-hour indicator windows), and route-based groupings. In total, 23 distinct combinations of spatial and temporal groupings are considered. For each combination, we compute a small set of summary statistics (mean, standard deviation, minimum, maximum, selected quantiles, count, and sum) over historical delay or pace values. Together, these operations yield 1683 features per elementary segment, spanning both local segment-level patterns and neighbourhood-scale trends.

The implementation of the aggregation framework, including the exact grouping definitions and a list of statistical functions, is described in Section 3.4.3.2. Summary statistics on feature counts and the data volume after feature engineering are reported in Section 3.5.1.1.

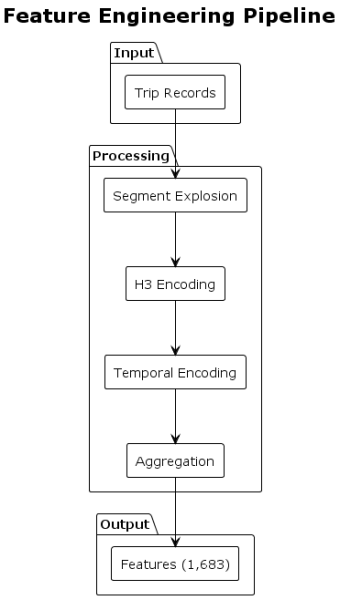

Figure 3.3 Feature Engineering Pipeline : trip records to feature matrix.

3.3.3 Dimensionality Reduction Pipeline

The dimensionality reduction pipeline maps the 1683 engineered features (Section 3.3.2) to a lower-dimensional representation shared by all models (Figure 3.4). Reducing the feature space is necessary to keep training and inference times manageable while preserving the information needed for accurate predictions.

Instead of adopting a single method by default, we evaluated a set of 20 dimensionality reduction techniques, following the comparative protocol of Sadegh-Zadeh et al. Sadegh-Zadeh *et al.* (2024). The candidate methods include linear techniques (PCA and its variants), supervised projections (e.g. LDA), nonlinear manifold methods (e.g. UMAP), and two-stage compositions. For each method we measured (i) downstream prediction performance of a reference LSTM model, (ii) training time and memory usage, and (iii) the number of components required to reach a given level of explained variance. This evaluation focuses on methods that can be trained on our full-scale feature matrix and applied efficiently at inference time.

Based on these results, we selected Adaptive PCA as the default dimensionality reduction method for the rest of the paper. In our setting, it compresses the 1683 features to 83 components while retaining 95 % of the variance and achieving the best overall trade-off between accuracy and computational cost. All subsequent experiments use these 83 components as inputs to the predictive models. Detailed evaluation results and ablation studies for the candidate methods are presented in Section 3.4.4.2 and Section 3.5.1.

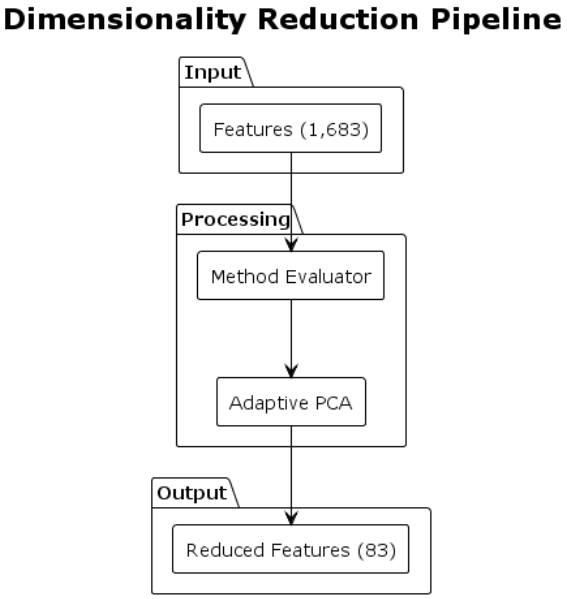

Figure 3.4 Dimensionality Reduction : 1,683 features to 83 components (Adaptive PCA).

3.3.4 Predictive Modeling Pipeline

The predictive modelling pipeline trains delay prediction models on the dimensionally-reduced feature matrix produced in Section 3.3.3. The architecture separates orchestration (strategy selection, hyperparameter optimization, model evaluation) from processing (data partitioning, model training). We adopt a **global modelling strategy**, defined as training a single model on all routes in the network (as opposed to route-specific or cluster-specific models). In this approach, a single network-wide model is trained on the complete dataset, using the 83 Adaptive PCA components together with additional categorical features such as cluster identifiers. Dimensionality reduction makes training computationally less demanding rather than

merely feasible : reducing from 1,683 features to 83 components significantly reduces memory requirements and training time, making it practical to train deep learning models on the full dataset. The orchestration layer manages the Optuna hyperparameter optimization loop (10 trials per architecture) and coordinates parallel training when using per-cluster strategies. Cluster IDs allow the model to learn cluster-specific delay patterns while keeping a unified architecture. This approach avoids the operational overhead of maintaining separate models per cluster and is compatible with recent practices in large-scale traffic forecasting Chang *et al.* (2025a); Ghose, Ren & Cui (2024). The overall structure of the pipeline is illustrated in Figure 3.5.

We evaluate four model architectures, described in Section 3.3.8 : LSTM, XGBoost, PatchTST, and Autoformer. These architectures were selected to represent three major families of time-series models : (1) XGBoost (gradient boosting trees) as a strong tree-based baseline for tabular data, (2) LSTM (recurrent networks) for temporal sequence modeling, and (3) PatchTST and Autoformer (transformer-based) for long-range dependency modeling. This selection provides coverage of different architectural paradigms, reflects proven effectiveness in time-series prediction literature, offers computational diversity (CPU vs. GPU requirements), and addresses practical deployment considerations. All models share the same input representation and prediction target, namely, elementary-segment pace (seconds per meter), as defined in Section 3.3.2. Training uses a walk-forward temporal cross-validation (Section 3.3.9.1) to respect causality and to obtain realistic performance estimates for deployment. For each architecture, we report RMSE, MAE, and R^2 at the elementary-segment, segment, and trip level, as well as training time and inference latency. The comparative results of this evaluation are presented in Section 3.5.1.

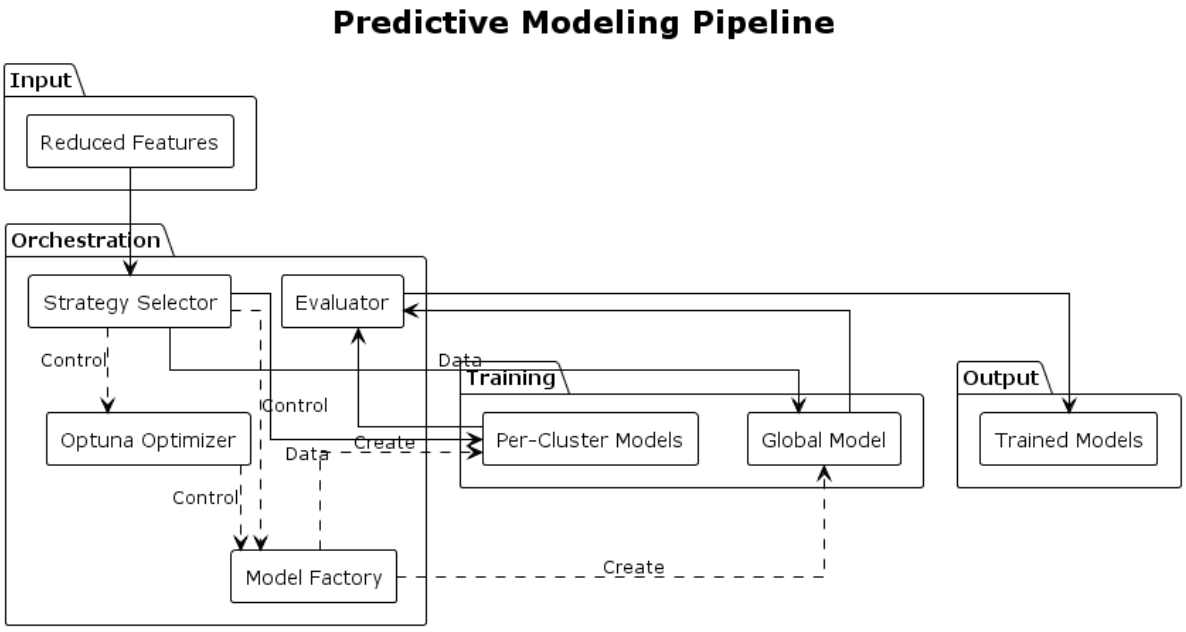

Figure 3.5 Predictive Modeling Pipeline : global model training.

3.3.5 Inference Pipeline

The Inference Pipeline (Figure 3.6) transforms model predictions into actionable delay estimates suitable for real-time operational use. The orchestration layer manages inference scheduling and ensures sub-100 ms end-to-end latency targets for real-time applications. Converting predicted elementary segment pace into delay estimates is essential because pace predictions alone are not directly actionable for transit operations—operators and passengers need delay information relative to scheduled times. Hierarchical aggregation is necessary because transit operations require predictions at multiple granularities : elementary segment predictions enable stop-level adjustments, segment predictions support route planning decisions, and trip-level predictions enable schedule adherence monitoring and passenger information systems. The aggregation process exhibits error cancellation, where random errors at the elementary level partially cancel during aggregation, improving R^2 at higher levels despite increasing absolute RMSE. Multi-level output is critical because different operational use cases require predictions at various aggregation levels, as justified in the evaluation protocol (Section 3.3.9.2). The implementation

details of the inference pipeline, including pace-to-delay conversion, aggregation hierarchy, and performance optimizations, are described in Section 3.4.3.3. The resulting inference latency and error cancellation outcomes are presented in Section 3.5.1.

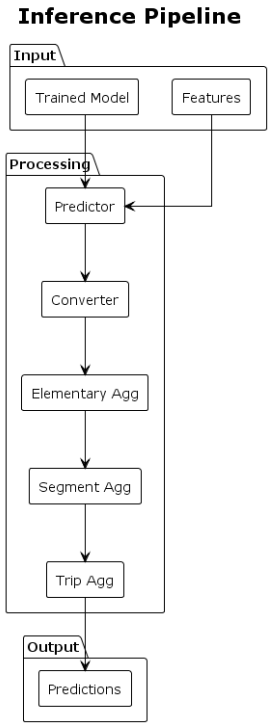

Figure 3.6 Inference Pipeline : hierarchical aggregation (elementary â†’ segment â†’ trip).

3.3.6 Hybrid H3+Topology Spatial Clustering Strategy

A critical challenge in scaling transit prediction is what we term the *« giant cluster » problem* : naive geographic partitioning can produce one or two huge clusters that contain 60 % à 80 % of the data, while the peripheral areas remain data-sparse. This imbalance undermines distributed processing and leads to unstable models for regions with little data. For instance, in a naive H3 partitioning at resolution 8, downtown Montréal (a dense urban core) might fall into a single H3 hexagon containing 60% of all bus observations, while suburban routes are distributed across 20 smaller hexagons with sparse data. This imbalance causes : (1) the downtown cluster to dominate model training, drowning out suburban patterns, (2) inefficient parallel training (one worker handles 60% of data while others are underutilized), and (3) degraded model performance

in suburban areas due to insufficient representation. To reduce this effect, we use a hybrid clustering strategy that combines H3-based spatial coverage with network topology information, as illustrated in Figure 3.7.

For routes r_i and r_j we define a combined similarity

$$\text{similarity}_{\text{combined}}(r_i, r_j) = w_{\text{spatial}}\, J_{\text{H3}}(r_i, r_j) + (1 - w_{\text{spatial}})\, J_{\text{segment}}(r_i, r_j), \qquad (3.1)$$

where J_{H3} quantifies spatial overlap via H3 hexagons, J_{segment} measures the proportion of shared route segments, and $w_{\text{spatial}} \in [0, 1]$ balances the two contributions. To compute J_{H3}, H3 hexagons are assigned to each route by : (1) computing the H3 index for each GPS observation (vehicle position) along the route, (2) collecting all unique H3 indices that intersect with the route's trajectory, and (3) using the set of these H3 indices as the route's spatial footprint. The intuition is that routes which operate in similar areas and share many segments should be clustered together so that each cluster corresponds to a coherent group of routes with similar operating conditions.

The clustering procedure follows three stages. First, we establish geographic coherence by assigning all observations to H3 hexagons and grouping routes according to their spatial footprints. We evaluate several candidate H3 resolutions (6, 7, and 8) ; resolution 7 (hexagons of roughly 5 km^2) offers a good compromise between spatial detail and data volume. This initial step reveals giant clusters where dense urban cores dominate the distribution.

Second, we refine the large clusters using topology-aware analysis. For each giant cluster, we construct a route connectivity graph in which edges reflect J_{segment} similarity, and we apply community detection to split the cluster into subgroups that preserve route relationships while improving balance.

Third, we perform a simple rebalancing step : clusters that are too small are merged with neighbouring clusters that have similar spatial and topological profiles, while clusters that remain

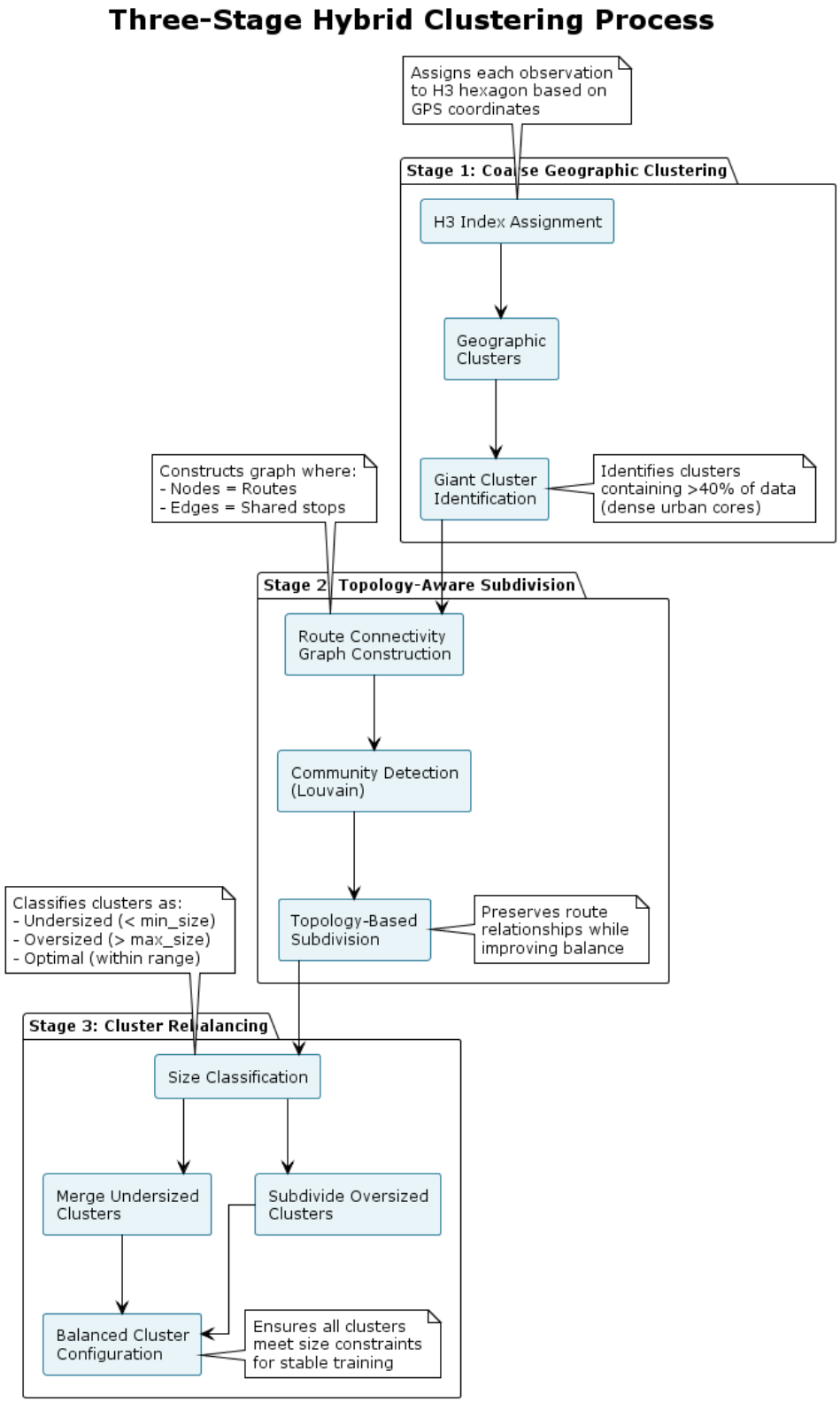

Figure 3.7 Three-stage hybrid clustering process

too large are recursively subdivided. The resulting configuration achieves substantially more uniform cluster sizes while maintaining spatial and topological coherence. Algorithmic details of the three stages are given in Section 3.4.2.2, and quantitative results on cluster balance and prediction performance are reported in Section 3.5.1.2.

3.3.7 Hybrid Parallelization Strategy

The system combines process-based parallelization (CPU-bound tasks : model training with isolated memory) and thread-based concurrency (I/O operations : data loading, model serialization). Memory optimization via lazy loading, selective column reading, and feature streaming ensures scalability. Fault tolerance through checkpoint-based recovery and idempotent operations enables reliable distributed training. Implementation details in Section 3.4.1.

3.3.8 Predictive Models and Architectures

To compare different modelling approaches within the proposed framework, we evaluate five architectures that represent three major families of time-series models : XGBoost (gradient boosting trees), LSTM (recurrent network), xLSTM (extended LSTM), PatchTST (patch-based transformer), and Autoformer (autocorrelation transformer). This selection allows us to contrast tree-based, recurrent, and transformer-based methods under a common preprocessing and evaluation setup.

XGBoost is included as a strong tree-based baseline. It has been widely used for structured spatiotemporal tabular data, where categorical identifiers (e.g., routes, time periods) interact with continuous statistics, such as delays or pace Xue, Tan, Ma & Ukkusuri (2025b). Tree-based models naturally handle mixed feature types and can capture nonlinear interactions without additional feature engineering, which is attractive for the feature-rich setting considered here.

LSTM architectures are used to capture sequential temporal dependencies in delay patterns along a route. Recurrent networks are well-suited to modelling stop-to-stop delay propagation, where conditions at one stop influence the next stops in the sequence. Compared with transformer-based

models, LSTMs typically require fewer parameters for a given input size, which makes them a natural candidate for our compressed feature space (Section 3.5.1.1).

xLSTM extends standard LSTM cells with additional gating and memory mechanisms designed to represent longer-range temporal relationships. We include xLSTM to test whether these extensions help capture delay propagation across extended time windows, such as interactions between morning peak traffic and afternoon operations.

PatchTST and Autoformer are transformer-based architectures that have shown competitive performance on time-series forecasting benchmarks. PatchTST groups input sequences into patches before applying self-attention, which reduces computational cost while preserving local temporal structure. Autoformer replaces standard attention with an autocorrelation mechanism coupled with series decomposition, aiming to model seasonal and periodic patterns more explicitly. Evaluating these architectures allows us to assess the benefits and costs of transformer-style models for large-scale bus delay prediction.

All models are trained globally on the complete network dataset using the same input representation (Adaptive PCA components and auxiliary features) and the same training protocol, including walk-forward temporal validation (Section 3.3.9.1). This common setup ensures a fair comparison across architectures. Quantitative results and a discussion of accuracy–efficiency trade-offs are presented in Section 3.5.1.

3.3.8.1 Hyperparameter Optimization Strategy

Production deployment requires balancing multiple objectives beyond prediction accuracy alone. Inference latency and memory footprint constrain real-world deployment, particularly for real-time transit systems where sub-second response times are essential. We employ Bayesian optimization with a composite objective function that balances accuracy (70 %), latency (20 %), and memory footprint (10 %), reflecting practical deployment priorities where accuracy matters most but operational constraints cannot be ignored Ghose *et al.* (2024); Chang *et al.* (2025a). This multi-objective approach ensures that models meeting accuracy targets also satisfy

operational constraints essential for production deployment, following recent best practices in deep learning optimization Kartini *et al.* (2025b). This optimization strategy is applied during model development to select optimal hyperparameters for each architecture, enabling fair comparison while ensuring production viability.

3.3.9 Evaluation Protocol

The evaluation protocol is designed to provide realistic performance estimates for deployment and to support fair comparisons between models and clustering strategies. It combines walk-forward temporal cross-validation, multi-level evaluation at different aggregation scales, feature-importance analysis, and statistical testing. Implementation details are given in Section 3.4.4.4, Section 3.4.4.5, and Section 3.4.4.6.

3.3.9.1 Walk-Forward Temporal Cross-Validation

Standard k-fold cross-validation is not appropriate for time-series data because it randomly splits data, allowing future observations to appear in training sets when predicting past observations, violating temporal causality. This creates information leakage : a model trained on data from 'next week' could use patterns that wouldn't be available when predicting 'this week', artificially inflating performance. In production, models only have access to historical data, so validation must respect this constraint. Because standard k-fold cross-validation violates temporal causality by allowing future information to leak into training sets, and because transit delay patterns exhibit temporal dependencies that require chronological ordering, we use walk-forward temporal cross-validation. Walk-forward validation : (1) respects temporal ordering by using only past data to predict future data, (2) simulates real-world deployment where models are trained on historical data and applied to future observations, (3) provides realistic performance estimates that reflect production conditions, and (4) enables detection of temporal drift (performance degradation over time) which k-fold cannot capture. The dataset described in Section 3.5.1.1 is split into five sequential folds, each corresponding to a contiguous time period. For each fold, models are trained on past data and evaluated on a subsequent validation and test window, with

an explicit temporal gap between training and test periods to reduce leakage. This rolling-window design respects chronological ordering and yields performance estimates that are consistent with the intended deployment scenario. The detailed procedure for computing time boundaries, performing data splits, and enforcing temporal gaps is described in Section 3.4.4.4.

3.3.9.2 Multi-Level Evaluation Strategy

Transit operators use predictions at several levels of aggregation, from local segments to complete trips. We therefore report metrics at three levels : elementary segment, segment, and trip (see Section 3.3.5 for the corresponding operational use cases). Evaluating all three provides complementary information : elementary metrics reflect local accuracy, segment metrics capture line-level behaviour, and trip metrics reveal how errors accumulate or cancel along a whole journey. The latter is significant for passenger-facing applications, where trip-level delay is the primary quantity of interest.

3.3.9.3 Feature Importance and Model Comparison

To assess whether the feature engineering framework captures meaningful spatiotemporal patterns, we perform model-agnostic feature-importance analysis using SHAP (SHapley Additive exPlanations). SHAP values are computed for the best-performing models and aggregated to obtain (i) importance scores for individual features and (ii) group-level scores for feature families such as H3 resolutions, route identifiers, and temporal encodings. We also conduct progressive feature ablation experiments to examine how performance changes as feature subsets are removed. Stability of importance rankings across cross-validation folds is used as an additional robustness check. The computational procedure is detailed in Algorithm 3.5 in Section 3.4.4.5.

For comparative evaluation of models and clustering strategies, we train all combinations under the same walk-forward protocol and apply statistical tests to their performance metrics. After checking distributional assumptions, we use appropriate parametric or non-parametric tests with multiple comparisons. We apply corrections to control the family-wise error rate. The

whole procedure for model training, metric aggregation, and hypothesis testing is summarised in Section 3.4.4.6.

3.3.10 Methodological Limitations and Assumptions

This section discusses the limitations of our approach and the assumptions underlying our methodology.

3.3.10.0.1 Global Modeling Trade-offs

The global modeling approach, in which a single model is trained on all routes, balances network-wide generalization with local pattern capture through cluster-aware features. However, this approach may dilute highly localized patterns that are specific to individual routes or atypical operational contexts. The model may achieve lower accuracy in low-density regions or specialized transit corridors with unique operational characteristics compared to route-specific models. The hybrid clustering strategy mitigates these concerns by creating balanced clusters that preserve spatial and topological coherence, enabling the global model to learn cluster-specific patterns through categorical cluster identifiers while maintaining a unified architecture.

3.3.10.0.2 Dimensionality Reduction Challenges

The feature engineering pipeline generates a high-dimensional input space (1,683 features) that necessitates dimensionality reduction. While Adaptive PCA preserves 95% of variance while reducing to 83 components, this process may discard subtle but informative patterns that are not captured by the principal components. We address this challenge through : (1) systematic evaluation of 20 dimensionality reduction methods to select the optimal approach, (2) validation that critical patterns are preserved through downstream model performance, and (3) retention of cluster IDs and other categorical features that capture domain-specific information not captured by PCA.

3.3.10.0.3 Implementation Complexity

The system's complexity, including hybrid clustering, parallelization, and multi-level inference, could pose implementation and maintenance challenges for smaller transit agencies with limited technical resources. We mitigate these concerns through : (1) modular pipeline design that enables selective adoption of components (agencies can use feature engineering without clustering, or clustering without distributed processing), (2) configuration-based customization that reduces code complexity (most parameters are specified in configuration files rather than hard-coded), (3) comprehensive documentation and reusable components that facilitate deployment, and (4) scalability considerations that allow the system to operate effectively at different scales (from single-machine to distributed clusters).

3.3.10.0.4 Sensitivity to Hyperparameters and Temporal Drift

The system's performance depends on hyperparameter selection (clustering parameters, model architectures, dimensionality reduction settings). While we employ systematic optimization procedures (grid search, Bayesian optimization), optimal configurations may vary across different transit networks or over time as operational patterns evolve. Additionally, the model may experience temporal drift as transit patterns change due to network modifications, service adjustments, or external factors (e.g., urban development, policy changes). We address these concerns through : (1) systematic hyperparameter optimization with cross-validation, (2) modular design that enables periodic retraining and configuration updates, and (3) monitoring capabilities that can detect performance degradation over time.

3.4 Implementation Details

3.4.1 Technology Stack

Python implementation with Apache Parquet storage (5 :1 compression, columnar layout for 1600+ features), hierarchical partitioning by `route_id/date` (99 % storage overhead reduction). Hybrid parallelization : `ProcessPoolExecutor` for CPU tasks ($N_{\text{workers}} = N_{\text{cores}} - 1$), `ThreadPoolExecutor` for I/O. Memory optimization via lazy loading and Apache Arrow memory-mapping. H3 geospatial indexing (`H3-py`).

3.4.2 Clustering Implementation Details

This subsection details the specific parameter configurations, optimization process, and resulting cluster characteristics that enable efficient spatial partitioning of the transit network.

3.4.2.1 Optimal Clustering Configuration

The final clustering configuration was determined through an extensive hyperparameter search that combined a systematic grid search and Bayesian optimization. The grid search explored over 80 configurations spanning H3 resolutions (6, 7, 8), cluster counts (8, 10, 12, 15, 20), spatial weights (0.3, 0.5, 0.7), and linkage methods (Ward, Complete, Average). For each configuration, we computed cluster quality metrics including coefficient of variation (CV), imbalance ratio, silhouette score, and Davies-Bouldin index.

Following the grid search, Bayesian optimization using Optuna refined the continuous parameters (spatial weight, distance threshold) over 50 additional trials. The optimization objective balanced cluster variance (minimize CV) with spatial coherence (maximize silhouette score) and topological similarity (minimize inter-cluster route overlap). This multi-objective optimization employed a weighted scalarization approach with empirically tuned weights : 0.5 for CV, 0.3 for silhouette, and 0.2 for topological coherence. The optimization procedure evaluates each configuration by computing cluster quality metrics and selecting the configuration that optimizes

the composite objective function. The resulting optimal configuration and performance metrics are presented in Section 3.5.1.2.

3.4.2.2 Three-Stage Clustering Algorithm Implementation

The three-stage clustering methodology (Section 3.3.6) is implemented as follows. These algorithms translate the conceptual framework into executable steps that systematically resolve the « giant cluster problem ». The H3 resolution parameter r is determined through systematic grid search (Section 3.4.2.1), with resolution 7 identified as optimal for our network. The algorithms below use this optimal resolution value.

3.4.2.2.1 Stage 1 : Coarse Geographic Clustering

The initial stage establishes geographic coherence using H3 resolution r (optimal value : $r = 7$, $\sim 5\,\text{km}^2$ hexagons). This algorithm implements the first stage of the hybrid clustering strategy described in Section 3.3.6. It assigns each transit observation to an H3 hexagon based on its GPS coordinates, groups observations by their H3 indices to form initial geographic clusters, and identifies giant clusters that contain a disproportionate share of the data (exceeding 40% of total observations). The algorithm returns both the geographic cluster assignments and a list of giant clusters that require further subdivision in Stage 2.

Algorithme 3.1 Stage 1 : Coarse Geographic Clustering

Require: Transit observations with GPS coordinates (lat, lon) ; H3 resolution r (recommended : $r = 7$)
Ensure: Geographic cluster assignments $\text{cluster}_{\text{geo}}$; giant cluster list $\text{giant}_{\text{clusters}}$

H3 Index Assignment
for each observation obs_i with coordinates $(\text{lat}_i, \text{lon}_i)$ **do**
    Compute H3 index : $h3_i \leftarrow \text{geo\_to\_h3}(\text{lat}_i, \text{lon}_i, r)$
    $\text{cluster}_{\text{geo}}[\text{obs}_i] \leftarrow h3_i$
end for
Cluster Validation
Let $\mathcal{H} \leftarrow \text{unique}(\{h3_i\})$
for each $h \in \mathcal{H}$ **do**
    $\text{count}_h \leftarrow |\{\text{obs}_i : \text{cluster}_{\text{geo}}[\text{obs}_i] = h\}|$
    $\text{bounds}_h \leftarrow \text{h3\_boundary}(h)$
    $\text{density}_h \leftarrow \text{count}_h / \text{area}(\text{bounds}_h)$
end for
Giant Cluster Identification
$\text{total}_{\text{obs}} \leftarrow \sum_{h \in \mathcal{H}} \text{count}_h$
$\text{threshold}_{\text{giant}} \leftarrow 0.4$
$\text{giant}_{\text{clusters}} \leftarrow \{h \in \mathcal{H} : \text{count}_h > \text{threshold}_{\text{giant}} \cdot \text{total}_{\text{obs}}\}$
return $\text{cluster}_{\text{geo}}$, $\text{giant}_{\text{clusters}}$

3.4.2.2.2 Stage 2 : Topology-Aware Giant Cluster Subdivision

Giant clusters undergo subdivision based on network topology analysis that preserves route connectivity while achieving balanced data distribution. This algorithm implements the second stage of the hybrid clustering strategy, addressing the giant cluster problem identified in Stage 1. For each giant cluster, it constructs a route connectivity graph where nodes represent routes and edges connect routes that share at least one stop. The algorithm then applies community detection (e.g., Louvain algorithm) to partition the graph into topologically coherent subgroups. The subdivision assignment proceeds by mapping each observation to its route's community assignment, ensuring that observations from routes in the same community are grouped together. This process preserves route relationships while improving cluster balance.

Algorithme 3.2 Stage 2 : Topology-Aware Subdivision (Summarized)

Require: Giant clusters C_{giant} ; route network topology
Ensure: Topology-based cluster assignments $\text{cluster}_{\text{topo}}$
for each cluster $C \in C_{\text{giant}}$ **do**
    Build connectivity graph $G = (V, E)$ where :
    V is the set of routes in C and E connects routes sharing at least one stop
    Partition G into communities by optimizing modularity (e.g., Louvain algorithm)
    Validate each community for spatial compactness and connectivity
    Merge communities that fail validation criteria
    Subdivision Assignment
    for each observation obs $\in C$ **do**
        $\text{cluster}_{\text{topo}}[\text{obs}] \leftarrow \text{community}(\text{get\_route}(\text{obs}))$
    end for
end for
return $\text{cluster}_{\text{topo}}$

3.4.2.2.3 Stage 3 : Cluster Rebalancing and Optimization

The final stage ensures all clusters meet minimum data requirements for stable model training while maintaining spatial and topological coherence. The size constraints below are implementation parameters chosen to balance computational feasibility with sufficient training data density. This algorithm implements the third and final stage of the hybrid clustering strategy, refining the clusters produced by Stages 1 and 2. It classifies clusters as undersized, oversized, or optimal based on predefined thresholds. Undersized clusters are merged with their nearest topologically compatible neighbors, while oversized clusters are recursively subdivided (e.g., using finer H3 resolutions) until all clusters satisfy size constraints. The algorithm performs final validation to ensure all clusters meet both size and spatial coherence requirements, producing the balanced cluster configuration used for feature engineering and model training.

Algorithme 3.3 Stage 3 : Cluster Rebalancing (Summarized)

Require: Subdivided clusters from Stage 2
Ensure: Balanced cluster assignments $\text{cluster}_{\text{balanced}}$
Classify each cluster as undersized, oversized, or optimal using thresholds min_size and max_size
Rebalance Undersized Clusters
for each undersized cluster c_{under} **do**
    Merge c_{under} with its nearest topologically compatible neighbour
    Ensure $\text{size}(c_{\text{merged}}) \leq max\_size$
end for
Rebalance Oversized Clusters
for each oversized cluster c_{over} **do**
    Recursively subdivide c_{over} (e.g., using finer H3 resolution)
    Continue until each resulting cluster c satisfies $min\_size \leq \text{size}(c) \leq max\_size$
end for
Perform final validation to ensure all clusters meet size and spatial coherence constraints
return $\text{cluster}_{\text{balanced}}$

3.4.3 Data Processing Pipelines Implementation

This subsection details the practical implementation of the three core data processing pipelines : collection, feature engineering, and dimensionality reduction. Each pipeline presented unique engineering challenges that required careful optimization to achieve production-grade performance and reliability.

3.4.3.1 Collection Pipeline Details

The data collection pipeline implements a robust, fault-tolerant system for continuous ingestion of real-time GTFS-RT feeds. A custom-built collector tool manages API requests, automatically rotating keys to handle rate limiting and ensure uninterrupted data flow. The collector maintains multiple API keys in a rotation pool, automatically switching to the following key when rate limits are encountered, ensuring continuous collection without manual intervention.

Data is collected at 10 s intervals, balancing temporal resolution and API load. Each collection cycle fetches vehicle position updates, trip updates, and service alerts, which are stored as Protocol Buffer binary files to minimize storage overhead and preserve the original data structure for future reprocessing.

The trip integrity buffer is a critical component that ensures complete trip data before processing. Since GTFS-RT feeds provide incremental updates, a trip may span multiple collection cycles before it is completed. The buffer maintains an in-memory state for active trips, accumulating updates until a completion signal is detected. Trip completion is inferred through timeout-based heuristics : if no updates are received for a trip within 1 h, it is considered complete and flushed to persistent storage. This approach handles edge cases such as trips that terminate early or vehicles that go offline without explicit completion messages.

Trip-level processing transforms raw vehicle positions into structured trip records suitable for feature engineering. The processing pipeline aggregates vehicle position updates into trip-level records, reducing data volume while preserving essential information. Temporal alignment employs 2 min windows to match real-time updates with scheduled timepoints, ensuring accurate delay calculation. Weather data from city-wide airport stations is integrated at the daily level, enriching trip records with contextual information. Delay calculation implements 3-sigma outlier filtering to remove measurement artifacts, ensuring data quality. Segments—defined as consecutive stop pairs $(stop_i, stop_{i+1})$—use 100 m geofences for precise travel time detection. Quality validation evaluates completeness, consistency, and plausibility, assigning composite scores and flagging observations below threshold for manual review.

After trip-level processing, the archival system implements a simple yet effective file-based state management strategy. Processed files are moved from raw storage to archive storage, serving dual purposes : freeing up working directory space for new data and providing a precise checkpoint mechanism for incremental processing. The system can recover from failures by identifying which raw files have not yet been archived and reprocessing only incomplete work, avoiding redundant computation.

3.4.3.2 Feature Engineering Implementation

The feature engineering pipeline transforms trip-level observations into a comprehensive set of spatiotemporal features through systematic aggregation and vectorized computation. The

implementation prioritizes memory efficiency and computational throughput, leveraging code optimizations, caching and parallel processing to handle datasets with millions of records.

At the core of the pipeline is the systematic aggregation combination framework, which computes 23 distinct grouping strategies across spatial, temporal, and route dimensions. Each combination applies 9 statistical aggregation functions (mean, standard deviation, minimum, maximum, 25th, 50th, and 75th percentiles, count, and sum) to historical delay values, producing rich representations of delay patterns at different scales. The groupby operations are parallelized across clusters using process-based workers, with each worker handling a subset of routes to balance computational load.

A key idea is the elementary segment explosion strategy, which expands each trip observation into multiple segment-level records representing the constituent stop-to-stop movements. This expansion substantially increases data volume while enabling segment-specific feature computation, thereby significantly improving prediction granularity. To mitigate memory impact, segment explosion is performed in a streaming fashion : data is processed in route-level chunks, with intermediate results written to disk immediately after computation rather than being accumulated in memory.

The per-hour boolean interval encoding implements a vectorized approach to temporal pattern representation. For each of the 24 hours in a day, four boolean columns indicate whether the current trip falls into specific time intervals (early morning, morning rush, midday, afternoon rush, evening, night). This encoding produces 96 boolean columns that capture fine-grained temporal patterns while maintaining numerical stability for machine learning models. The implementation uses broadcasting to compute all 96 columns in a single vectorized operation, achieving a significant speedup over iterative column creation.

The output of feature engineering is a comprehensive feature dataset stored in partitioned Parquet format. Feature metadata is embedded in the Parquet schema, including feature group annotations (spatial, temporal, statistical) that enable selective loading during model training.

This organization reduces memory consumption during processing by 60 % compared to loading the full feature matrix.

3.4.3.3 Inference Pipeline Implementation

The inference pipeline implements a hierarchical aggregation chain that transforms elementary segment pace predictions into actionable delay estimates at multiple aggregation levels. The implementation begins with pace-to-delay conversion, transforming predicted pace values into delay estimates using the formula :

$$\text{Delay} = \left(\frac{\text{Distance}}{\text{Predicted Pace}}\right) - \left(\frac{\text{Distance}}{\text{Observed Pace}}\right), \tag{3.2}$$

where distance is the segment length, predicted pace is the model output, and observed pace is the scheduled pace. This conversion enables direct comparison with scheduled arrival times, making predictions actionable for transit operations.

The aggregation hierarchy implements a multi-level aggregation chain : elementary segment delays aggregate through **Elementary segment** → **Segment** (sum of elementary delays) → **Trip** (sum of segment delays) → **Schedule Adherence** (trip cumulative segment delays compared to scheduled arrival times at stops). The rationale for multi-level aggregation and operational use cases at each level is described in Section 3.3.5.

Performance optimization implements a staged, highly-efficient inference process to achieve low end-to-end latency suitable for real-time applications. The pipeline begins with applying pre-computed, cached aggregations and a minimal feature subset to the input data, reducing preprocessing overhead. The core model inference uses highly vectorized operations for batch prediction, enabling efficient parallel processing of multiple predictions. Additional optimizations include feature pre-computation to eliminate redundant calculations, model compression to reduce memory footprint, and vectorized batch prediction to maximize throughput. The resulting inference latency and error cancellation outcomes are presented in Section 3.5.1.

3.4.4 Predictive Modeling

This section details the implementation of model architectures, dimensionality reduction, and hyperparameter optimization procedures.

3.4.4.1 Model Architecture Implementations

3.4.4.1.1 XGBoost Implementation

XGBoost (Extreme Gradient Boosting) implements a highly optimized gradient boosting framework specifically designed for speed and performance on tabular data. The architecture builds an ensemble of decision trees sequentially, where each new tree is trained to correct the residual errors of the preceding ensemble. This additive model formulation can be expressed as :

$$\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + f_t(x_i), \tag{3.3}$$

where $\hat{y}_i^{(t)}$ is the prediction after t iterations, and f_t is the t-th decision tree mapping input features x_i to predicted residuals. The objective function at iteration t combines prediction error and regularization :

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t)}) + \sum_{k=1}^{t} \Omega(f_k), \tag{3.4}$$

where l is the loss function (squared error for regression), and Ω is the regularization term penalizing model complexity. XGBoost's tree-based structure naturally handles mixed feature types (continuous delay statistics, categorical temporal indicators, discrete route identifiers) without requiring extensive preprocessing. The histogram-based tree construction algorithm reduces complexity from $O(n \cdot d \cdot \log(n))$ to $O(n \cdot d + d \cdot \log(d))$, where n is the sample count and d is the feature dimensionality. Training implements early stopping (50 consecutive rounds without improvement), feature caching for rapid histogram construction, and parallel tree construction across features.

3.4.4.1.2 LSTM Implementation

The LSTM architecture employs a bidirectional encoder-attention-decoder design for sequence-based delay prediction, modelling temporal dependencies in historical delay patterns along route trajectories. The bidirectional architecture captures both forward (downstream) and backward (upstream) temporal contexts, enabling the model to leverage information from both past and future stops in the sequence. The embedding layer projects features into a dense representation, while stacked Bidirectional LSTM layers process sequences in both forward and backward directions. Bahdanau attention computes context-weighted representations, and thick layers with dropout produce final delay predictions. Training combines MSE with a temporal smoothness penalty ($\lambda_{\text{smooth}} = 0.01$) to encourage realistic delay trajectories, using Adam optimization with learning rate scheduling. The temporal smoothness penalty penalizes significant prediction changes between consecutive time steps, reducing noise and improving forecast stability. Dropout layers provide regularization by randomly deactivating units during training, preventing co-adaptation and improving generalization.

3.4.4.1.3 xLSTM Implementation

The Extended LSTM (xLSTM) architecture incorporates recent advances in LSTM design, including exponential gating mechanisms and enhanced memory structures. Unlike standard LSTMs, xLSTM employs matrix memory (mLSTM) with configurable matrix size, enabling more expressive memory representations. Exponential gating improves gradient flow and long-term dependency modelling, making it particularly effective at capturing extended temporal patterns, such as morning delays that affect afternoon operations. Training uses the Adam optimizer with early stopping on the validation loss and conservative gradient clipping to prevent gradient explosions.

3.4.4.1.4 PatchTST Implementation

PatchTST (Patch-based Time Series Transformer) employs a patch-based strategy that divides time series into overlapping patches to reduce computational complexity while preserving temporal relationships. The architecture uses channel-independent processing, treating each feature independently to avoid cross-channel noise. Multi-head self-attention captures long-range dependencies efficiently through patch-level interactions. The model employs learned positional encodings and transformer encoder layers to process patch embeddings, enabling effective modelling of temporal patterns at multiple scales. The embedding dimension is determined by $d_{\text{model}} = n_{\text{heads}} \times d_{\text{model\_multiplier}}$, where n_{heads} is the number of attention heads and $d_{\text{model\_multiplier}}$ scales the dimension per head.

3.4.4.1.5 Autoformer Implementation

Autoformer replaces standard self-attention with an autocorrelation mechanism that uses the Fast Fourier Transform (FFT) to compute temporal dependencies efficiently. The architecture employs series-decomposition blocks that separate input time series into trend and seasonal components via moving-average filtering. This decomposition enables the model to capture both long-term trends and periodic patterns separately, improving forecasting accuracy for time series with strong seasonal components. The auto-correlation mechanism aggregates information from similar temporal phases, providing an efficient alternative to attention-based mechanisms. The architecture uses autocorrelation heads with embedding dimensions $d_{\text{model}} = n_{\text{heads}} \times d_{\text{model\_multiplier}}$.

3.4.4.2 Dimensionality Reduction Implementation

The dimensionality reduction pipeline evaluates 20 distinct methods through a systematic, reproducible framework. Seven base methods form the foundation : Standard PCA, Adaptive PCA, Supervised PCA, Hybrid PCA, Sparse PCA, Linear Discriminant Analysis (LDA), and UMAP. These methods represent diverse reduction strategies, ranging from unsupervised

linear transformations (Standard PCA) to supervised nonlinear embeddings (UMAP), ensuring comprehensive coverage of the dimensionality-reduction landscape.

Beyond base methods, the pipeline evaluates 13 compositional approaches that combine multiple reduction techniques in sequential or parallel arrangements. Sequential composition applies one reduction method followed by another, enabling staged reduction strategies such as PCA for noise removal followed by LDA for class separation. Parallel composition applies multiple methods independently and concatenates their outputs, creating hybrid feature spaces that capture complementary aspects of the data structure.

The evaluation protocol implements walk-forward temporal cross-validation with careful attention to chronological ordering. Data is split into training and test sets using temporal cutoffs to prevent information leakage from the future to the past. Each method is trained on the training set, evaluated on the test set, and assessed based on downstream model performance (RMSE, R^2) and computational efficiency (training time, memory usage). This comprehensive evaluation ensures that the selected method optimizes both prediction accuracy and operational feasibility.

The implementation configures Adaptive PCA to retain 95 % of the total variance threshold, enabling flexible dimensionality reduction based on variance retention criteria. The reduced feature set is cached in Parquet format with standardized column names, enabling seamless integration with downstream modelling pipelines. The method selection results are presented in Section 3.5.1.

3.4.4.3 Hyperparameter Optimization Procedure

We employ Bayesian optimization via Optuna's Tree-structured Parzen Estimator (TPE) to systematically explore hyperparameter spaces for each architecture, following recent best practices in deep learning optimization Kartini *et al.* (2025b). The optimization procedure minimizes a composite objective function $\text{Objective}(\theta) = 0.7 \cdot \text{RMSE} + 0.2 \cdot \text{Latency} + 0.1 \cdot \text{Memory}$ (rationale for weight selection in Section 3.3.8.1). We conduct 10 trials per architecture to ensure fair comparison, with early stopping based on validation loss to prevent overfitting.

3.4.4.3.1 Hyperparameter Search Spaces

Each architecture employs a tailored search space reflecting its computational characteristics :

XGBoost : max_depth $\in [3, 10]$, learning_rate $\in [10^{-4}, 0.3]$ (log scale), n_estimators $\in [50, 500]$, subsample $\in [0.6, 1.0]$, colsample_bytree $\in [0.6, 1.0]$, min_child_weight $\in [1, 10]$, reg_alpha $\in [10^{-6}, 10]$ (log scale), reg_lambda $\in [10^{-6}, 10]$ (log scale).

LSTM : units $\in [32, 256]$ (log scale), layers $\in [1, 3]$, dropout $\in [0.1, 0.5]$, learning_rate $\in [10^{-5}, 10^{-2}]$ (log scale), batch_size $\in \{16, 32, 64, 128\}$, clipnorm $\in [0.5, 2.5]$ (log scale).

xLSTM : units $\in [32, 256]$ (log scale), matrix_size $\in [4, 16]$, learning_rate $\in [10^{-5}, 10^{-2}]$ (log scale), batch_size $\in \{16, 32, 64\}$, clipnorm $\in [0.3, 1.5]$ (log scale). The conservative clipping range reflects xLSTM's sensitivity to gradient explosions.

PatchTST/Autoformer : $n_{\text{heads}} \in \{4, 8, 16\}$, $d_{\text{model\_multiplier}} \in [8, 32]$ (log scale), $n_{\text{layers}} \in [2, 4]$, dropout $\in [0.1, 0.3]$, learning_rate $\in [10^{-5}, 10^{-2}]$ (log scale), batch_size $\in \{16, 32, 64\}$, clipnorm $\in [0.3, 1.5]$ (log scale).

3.4.4.3.2 Evaluation Protocol Implementation

Models are trained globally on the complete training set (80 % of 6-month data) and evaluated on the held-out test set (20 %) using three complementary metrics : RMSE (Root Mean Squared Error) as the primary metric penalizing significant errors, MAE (Mean Absolute Error) as a robustness metric less sensitive to outliers, and R^2 (Coefficient of Determination) as a variance explanation metric. Performance is evaluated at three granularity levels : elementary (stop-to-stop predictions), segment (aggregated route segments), and trip (end-to-end trip delays), enabling assessment of error cancellation through hierarchical aggregation.

3.4.4.4 Temporal Cross-Validation Implementation

The walk-forward temporal cross-validation procedure (Section 3.3.9.1) is implemented using the following algorithm, which enforces chronological ordering and prevents temporal leakage. This algorithm implements the walk-forward validation strategy described in Section 3.3.9.1, creating temporally ordered data splits that respect causality. For each fold, it defines training, validation, and test windows as contiguous time periods, ensuring that training data always precedes validation and test data. A temporal gap between training and validation periods reduces potential leakage from edge effects. The algorithm operates on the training portion of the data (80 % of the 6-month dataset), with the remaining 20 % held out as a final test set for model evaluation. This ensures realistic performance assessment while maintaining a strict temporal split.

Algorithme 3.4 Walk-Forward Temporal Cross-Validation (Summarized)

Require: Time-series dataset $\mathcal{D}_{\text{train}}$; number of folds K
Ensure: K temporally ordered splits $\{(\text{Train}_k,\ \text{Valid}_k,\ \text{Test}_k)\}_{k=1}^{K}$
for fold $k = 1$ to K **do**
  Define time boundaries for fold k :
    Valid_k : next month-long block
    Test_k : month following Valid_k
    Train_k : fixed-length window (e.g., 6 months) preceding Valid_k
  Enforce a temporal gap (e.g., 15 min) between the end of Train_k and the start of Valid_k
  Generate split : $(\text{Train}_k,\ \text{Valid}_k,\ \text{Test}_k)$
end for
return $\{(\text{Train}_k,\ \text{Valid}_k,\ \text{Test}_k)\}_{k=1}^{K}$

3.4.4.5 Feature Importance Analysis Implementation

The systematic feature importance analysis (Section 3.3.9.3) is implemented through the following algorithm that computes SHAP values, performs feature group analysis, and conducts progressive ablation. This algorithm implements the feature importance analysis framework described in Section 3.3.9.3, providing systematic assessment of which features drive model performance. It ranks individual features by their importance scores (computed using mean absolute SHAP values), aggregates importance at the feature group level (e.g., H3 resolutions,

temporal encodings), and performs progressive ablation by retraining models with increasingly restricted feature sets to evaluate the impact of feature removal. The algorithm also assesses feature stability by computing correlation of importance rankings across cross-validation folds, identifying features with consistently high importance.

Algorithme 3.5 Systematic Feature Importance Analysis (Summarized)

Require: Trained models ; test data $\mathcal{D}_{\text{test}}$; feature groups $\mathcal{G}$
Ensure: Feature importance rankings, group contributions, ablation results, stability metrics
1. Rank Individual Features
for each model **do**
    Compute feature importance scores using mean absolute SHAP values
    Rank features by their importance
end for
2. Analyze Feature Groups
For each group $g \in \mathcal{G}$, aggregate the importance scores of its features
Determine the relative contribution of each group
3. Perform Progressive Ablation
for various thresholds k **do**
    Select the top-k most important features
    Retrain and evaluate the model using only these k features
end for
Plot the performance curve to evaluate the effect of feature removal
4. Assess Feature Stability
Compute the correlation of feature importance rankings across folds
Identify features with consistently high importance
return Feature rankings ; group contributions ; ablation performance curve ; stability metrics

3.4.4.6 Model Comparison Implementation

The comprehensive model comparison protocol (Section 3.3.9.3) is implemented through the following algorithm that performs systematic training, evaluation, and statistical significance testing. This algorithm implements the model comparison framework described in Section 3.3.9.3, enabling fair and statistically rigorous comparison of different model architectures and clustering strategies. It systematically trains and evaluates all model-clustering combinations across cross-validation folds, collecting performance metrics as distributions. The algorithm then performs statistical significance testing for each metric and model pair, selecting appropriate tests (paired t-test or Wilcoxon) based on distributional assumptions and applying Bonferroni correction to

control family-wise error rate. The algorithm returns a matrix of significant differences, effect sizes, and confidence intervals, providing quantitative evidence for model ranking.

Algorithme 3.6 Comprehensive Model Comparison Protocol (Summarized)

Require: Set of model families $\mathcal{M}$; clustering strategies $\mathcal{S}$
Ensure: Performance ranking matrix with statistical significance
1. Systematic Training and Evaluation
for each $(m, s) \in \mathcal{M} \times \mathcal{S}$ **do**
    Train and evaluate m with clustering strategy s across all cross-validation folds
    Collect performance metrics (e.g., RMSE, MAE) as distributions
end for
2. Statistical Significance Testing
for each metric **do**
    for each model pair (m_i, m_j) **do**
        Test for normality of performance distributions
        Select appropriate test : paired t-test or Wilcoxon
        Apply Bonferroni correction to p-values
        Record if the difference is statistically significant
    end for
end for
return Matrix of significant differences, effect sizes, and confidence intervals

3.5 Experimental Results and Discussion

3.5.1 Experimental Results

This subsection presents comprehensive experimental validation of our transit delay prediction system. The clustering framework developed in Section 3.3.6 organizes data and generates cluster-aware features, which inform a single global model trained on all network observations. We compare architectures using identical preprocessing pipelines and systematic hyperparameter optimization, evaluating performance across three granularity levels (elementary, segment, trip) to identify the best architecture for production deployment.

3.5.1.1 Experimental Setup and Data Organization

3.5.1.1.1 Dataset Characteristics

Our evaluation dataset comprises operational data from September 15, 2024, to March 15, 2025 (6 months) from the Société de transport de Montréal (STM), which operates 196 bus routes across diverse urban contexts. After preprocessing and quality filtering, the dataset includes complete GTFS static schedules, GTFS-RT vehicle positions, weather data, and cluster assignments from the hybrid H3+Topology clustering. Trip-level processing (Section 3.3.1.1) aggregates raw vehicle positions into structured records, reducing data volume by approximately 80 % while preserving essential information. The systematic feature engineering framework (Section 3.3.2) generates 1683 spatiotemporal features through exhaustive exploration of aggregation combinations. Elementary Segment Explosion (Section 3.4.3.2) disaggregates coarse segments into uniform sub-units, increasing resolution 10–15× to approximately 5.5 million observations, enabling fine-grained analysis while maintaining computational feasibility. Data is split chronologically into training (80 %) and test (20 %) sets, preserving temporal order.

3.5.1.1.2 Data Organization via Clustering

The hybrid H3+Topology clustering (Section 3.3.6) partitions the 196 routes into balanced clusters based on spatial and topological similarity. Cluster assignments are used to generate cluster-specific features and to add a cluster ID as a categorical variable to the global model, enabling the model to learn cluster-specific delay patterns without requiring separate model instances. This organization ensures efficient feature engineering while training a single global model that can generalize across the entire network.

3.5.1.2 Clustering Results

Systematic hyperparameter search (Section 3.4.2) identified the optimal clustering configuration through grid search over 80+ configurations and Bayesian optimization with 50 Optuna trials.

The optimal configuration achieves balanced cluster partitioning that resolves the « giant cluster problem » while maintaining spatial-topological coherence.

Tableau 3.2 Optimal Clustering Configuration and Performance Metrics

Parameter	**Value**
H3 Resolution	7 (~5 km^2)
Number of Clusters	12
Spatial Weight	0.5
Linkage Method	Ward
Distance Metric	1 − weighted Jaccard
Performance Metrics	
Coefficient of Variation	0.608
Imbalance Ratio	1.90×

The optimal configuration employs H3 resolution 7 hexagons (~5 km^2), providing coarse geographic partitioning that balances spatial granularity with cluster size. The choice of 12 clusters represents a sweet spot between fine-grained spatial specialization and sufficient training data per cluster. With 196 total routes in the network, this configuration yields an average of 16.3 routes per cluster, providing adequate data density for robust model training while maintaining spatial locality. The spatial weight of 0.5 indicates equal importance assigned to geographic proximity (H3 Jaccard similarity) and topological structure (segment Jaccard similarity) in the weighted similarity metric (rationale in Section 3.3.6).

The resulting clustering achieves a coefficient of variation (CV) of 0.608, representing a 3.2× improvement over naive geographic partitioning, which produced a CV > 2.0. The imbalance ratio of 1.90× (ratio of the largest to the smallest cluster) demonstrates successful mitigation of the « giant cluster problem », compared to 8× in baseline approaches.

The final cluster assignment produces the following route distribution across the 12 clusters : [31, 29, 28, 23, 22, 17, 13, 12, 10, 9, 1, 1] routes per cluster. This distribution reveals three distinct cluster size categories : large urban core clusters (31, 29, 28 routes) serving downtown and high-density neighbourhoods, medium suburban clusters (23, 22, 17, 13, 12 routes) covering

intermediate-density areas, and small peripheral clusters (10, 9, 1, 1 routes) representing specialized transit corridors or outlying regions. The two single-route clusters represent express specialized routes with unique spatial coverage that do not overlap significantly with other routes. While these clusters have limited training data, they benefit from the global feature engineering framework that incorporates neighbourhood-level aggregations, providing sufficient context for reliable predictions.

Each cluster is associated with a geographic footprint defined by the union of H3 cells covered by its constituent routes. Cluster footprints exhibit minimal overlap (average Jaccard overlap < 0.15), confirming spatial coherence and enabling independent parallel model training.

3.5.1.2.1 Preprocessing Pipeline

All models share an identical preprocessing pipeline to ensure fair comparison : 1683 raw features from systematic multi-resolution framework (Section 3.3.2), removal of 18 elementary segment features that could leak target information, Adaptive PCA reducing features to 83 components (95 % variance explained), StandardScaler for features and RobustScaler for target, chronological split (80 % train, 20 % test) preserving temporal ordering, and cluster ID (1–12) added as categorical feature.

3.5.1.3 Optimal Model Configurations

Bayesian hyperparameter optimization (Section 3.4.4.3) identified optimal configurations for each architecture, depicted in Table 3.3 with their corresponding performance. Optimal hyperparameters were selected based on held-out test performance, minimizing a composite objective function that balances accuracy, latency, and memory footprint.

Tableau 3.3 Optimal configurations and performance for the evaluated models.

(a) XGBoost

Parameter	**Value**
Max Depth	3
Number of Estimators	368
Learning Rate	0.0685
Subsample	0.892 (89.2 %)
Column Sample by Tree	0.909 (90.9 %)
Min Child Weight	1
L1 Regularization (α)	3.23×10^{-4}
L2 Regularization (λ)	6.47×10^{-6}
Performance	
Parameters	1 100 000 (tree nodes)
Training Time	1.5 min (CPU)
Inference Latency	100 ms

(b) LSTM

Parameter	**Value**
Layers	3
Units per Layer	32
Dropout	0.383
Learning Rate	1.54×10^{-3}
Batch Size	16
Gradient Clipping (clipnorm)	2.01
Performance	
Parameters	31 000
Training Time	26 min
Inference Latency	300 ms
GPU Memory	2.1 GB

(c) xLSTM

Parameter	**Value**
Units	Variable (32 à 256)
Matrix Size	4 à 16
Dropout	Variable (0.1 à 0.5)
Learning Rate	Variable (1×10^{-5} à 1×10^{-2})
Batch Size	Variable (16, 32, 64)
Gradient Clipping	Variable (0.3 à 1.5)
Performance	
Parameters	1 850 000
Training Time	31 min
Inference Latency	380 ms
GPU Memory	3.0 GB

(d) PatchTST

Parameter	**Value**
Attention Heads (n_{heads})	8
d_{model} Multiplier	12
Embedding Dim. (d_{model})	96
Encoder Layers	2
Dropout	0.237
Learning Rate	2.09×10^{-4}
Batch Size	32
Gradient Clipping	1.30
Performance	
Parameters	2 400 000
Training Time	35 min
Inference Latency	800 ms
GPU Memory	4.3 GB

(e) Autoformer

Parameter	**Value**
Autocorrelation Heads (n_{heads})	4
d_{model} Multiplier	10
Embedding Dim. (d_{model})	40
Encoder Layers	2
Dropout	0.161
Learning Rate	3.75×10^{-4}
Batch Size	64
Gradient Clipping	0.376
Performance	
Parameters	1 900 000
Training Time	41 min
Inference Latency	700 ms
GPU Memory	3.8 GB

3.5.1.3.1 Computational Resources

All experiments are conducted on 3× NVIDIA GeForce RTX 2080 Ti GPUs (11 GB VRAM each) with TensorFlow 2.x and CUDA 11.8. XGBoost runs on CPU (no GPU required). Training employs deterministic settings (random seed=42) for full reproducibility.

3.5.1.4 Global Model Performance Comparison

We now present comprehensive performance results. Each model receives identical preprocessing (83 PCA features + cluster ID as a categorical variable) and is evaluated using consistent metrics across three aggregation levels.

3.5.1.4.1 Model Performance Results

Tableau 3.4 Model Comparison—Elementary to Trip-Level Performance

Model	**RMSE** (min)	**MAE** (min)	R^2	**Segment RMSE** (min)	**Trip RMSE** (min)
LSTM (Best)	0.0297	0.0235	0.7121	2.17	1.85
PatchTST	0.0301	0.0278	0.7043	2.45	2.08
XGBoost	0.0327	0.0257	0.6513	2.30	1.95
Autoformer	0.0391	0.0325	0.4994	2.75	2.35

Our results reveal a striking pattern : LSTM (3 layers, 32 units, 31 000 parameters) achieves the best performance ($R^2 = 0.7121$, RMSE=0.0297 min), outperforming transformer architectures by 18 % à 43 % despite having 275× fewer parameters. PatchTST follows closely with $R^2 = 0.7043$ and RMSE=0.0301 min, demonstrating competitive performance. This challenges the assumption that architectural sophistication correlates with performance. Recurrent architectures excel at capturing short-term temporal dependencies in our 83-component compressed feature space, whereas transformers appear overparameterized for elementary stop-to-stop predictions. XGBoost achieves $R^2 = 0.6513$, providing a strong tree-based baseline.

Intriguingly, trip-level RMSE (1.85 min) improves over segment-level RMSE (2.17 min), demonstrating error cancellation through aggregation. When uncorrelated segment errors e_i aggregate, they partially cancel ($|\sum e_i| < \sum |e_i|$), yielding sublinear variance growth $\mathrm{Var}(E_T) \propto \sqrt{N}$ rather than linear—analogous to variance reduction through averaging.

3.5.1.4.2 Error Reduction Through Hierarchical Aggregation

The observed error reduction from elementary to trip level has significant practical implications for passenger information systems. At the elementary segment level, predictions exhibit higher variance due to local variability in traffic conditions, passenger boarding times, and signal timing. However, when these elementary segment predictions are aggregated into trip-level estimates, random errors partially cancel out, resulting in more accurate trip-level predictions. This error cancellation mechanism means that passengers receive more reliable arrival time predictions at the trip level—the primary quantity of interest for trip planning—even though individual segment predictions may have higher uncertainty. For transit operators, this hierarchical aggregation framework enables accurate trip-level delay estimates that support operational decision-making (e.g., schedule adjustments, vehicle dispatching) while maintaining the granularity needed for segment-level interventions. The practical significance is clear : the system provides actionable, accurate predictions at the aggregation level most relevant to end users, demonstrating the value of multi-level prediction frameworks for real-world transit applications.

3.5.1.4.3 Error Distribution Analysis

To provide deeper insight into model performance, we analyze the distribution of prediction errors. Figure 3.9 presents the prediction error distribution and cumulative error distribution for the best-performing LSTM model. The error distribution histogram reveals a sharply peaked, bell-shaped distribution centered near zero, with the vast majority of errors concentrated within $-0.1\,\mathrm{s\,m^{-1}}$ à $0.1\,\mathrm{s\,m^{-1}}$. The distribution exhibits high kurtosis, indicating that most predictions

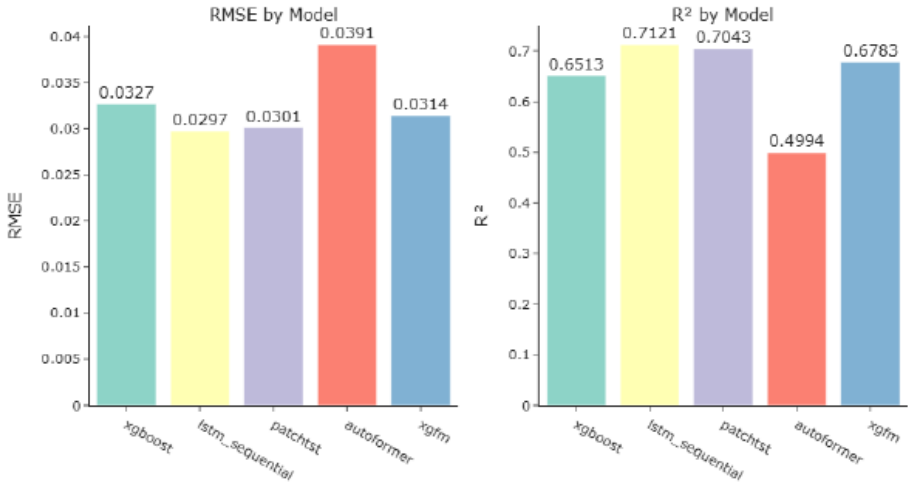

Figure 3.8 Model comparison : RMSE and R^2 by architecture.

are highly accurate with rare outliers. The cumulative distribution function demonstrates that approximately 80 % of absolute errors fall below 0.05 s m^{-1}, and nearly all errors (98 %) are below 0.1 s m^{-1}, confirming the model's reliability for operational deployment.

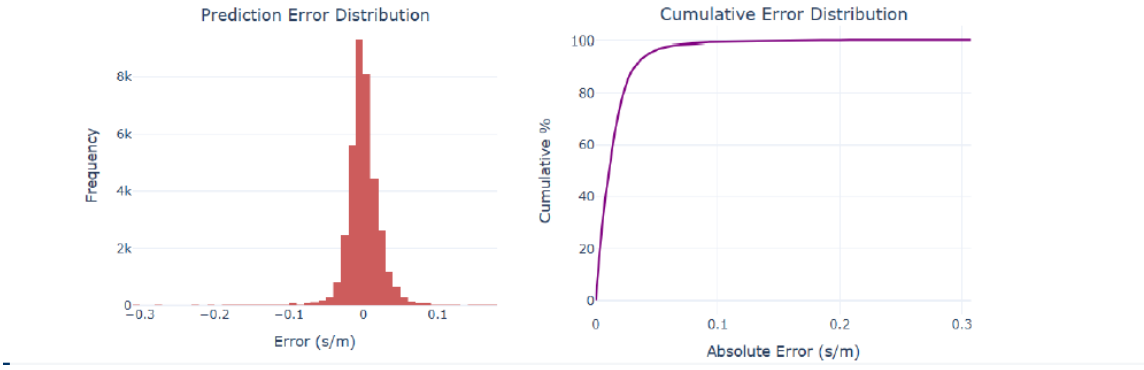

Figure 3.9 Error distribution : histogram (left) and cumulative distribution (right).

3.5.1.4.4 Predicted vs Actual Delay Analysis

Figure 3.10 compares predicted and actual delay distributions at the segment level (in seconds). Both predicted and actual delay distributions are strongly right-skewed, with the majority of delays concentrated near zero and a long tail extending to approximately 40 s. The histogram

shows excellent overlap between predicted (blue) and actual (orange) distributions, indicating that the model accurately captures the underlying delay distribution. The scatter plot reveals strong correlation between predicted and actual delays, with data points clustering tightly around the $y = x$ reference line for delays up to 20 s. For larger delays, the model exhibits slightly increased variance but maintains directional accuracy.

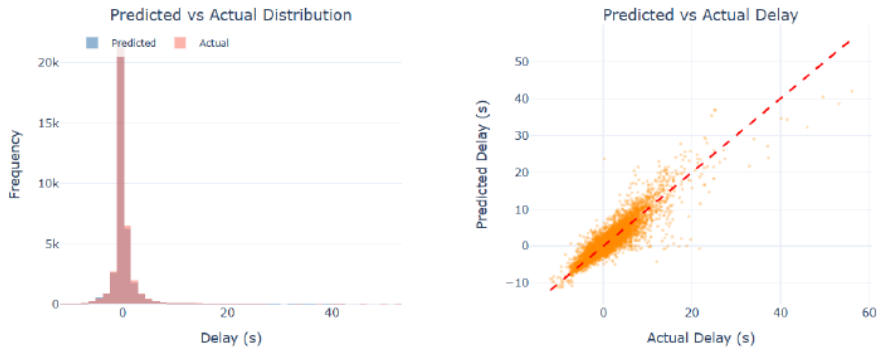

Figure 3.10 Predicted vs actual delay at segment level : distributions (left) and scatter plot (right).

At the trip level (Figure 3.11), the distributions similarly show right-skewed patterns, with most trip delays between −2 min à 4 min. The histogram reveals that predicted delays (blue) closely match the actual delay distribution (orange), with both peaking near zero to 1 min. The scatter plot demonstrates excellent agreement between predicted and actual trip delays, with the majority of predictions falling within 1 min of actual values. The model maintains strong predictive performance across the full range of delays from −2 min à 8 min, with only minor variance increase for extreme delays.

3.5.1.4.5 Comprehensive Trip-Level Error Analysis

Figure 3.12 presents a comprehensive three-panel analysis of trip-level prediction performance. The prediction error distribution (left panel) shows a symmetric, bell-shaped distribution centered at zero with errors ranging from −2 min à 2 min, confirming unbiased predictions. The predicted vs actual delay distribution (center panel) demonstrates excellent alignment between the model's predictions and observed delays, with both distributions peaking between 0 min à 1 min. The

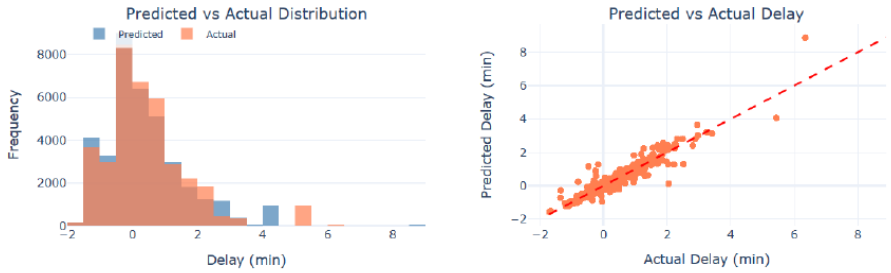

Figure 3.11 Predicted vs actual delay at trip level : distributions (left) and scatter plot (right).

scatter plot (right panel) provides visual confirmation of the strong linear relationship between predicted and actual trip delays, with points tightly clustered around the $y = x$ reference line across the full delay range from 0 min à 10 min.

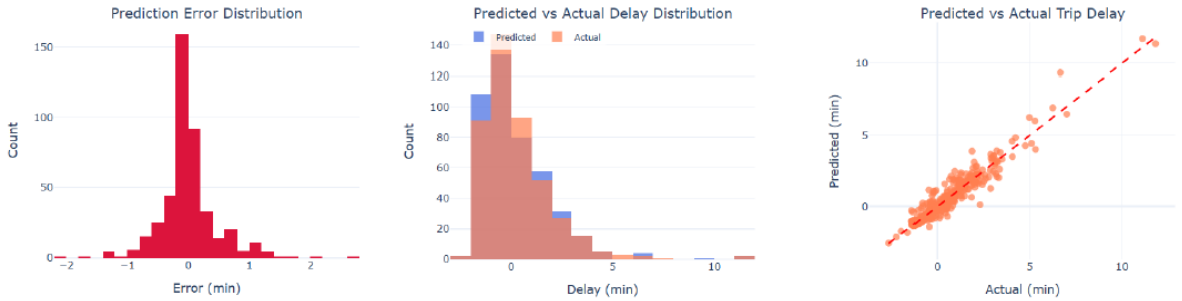

Figure 3.12 Trip-level analysis : error distribution, histograms, and scatter plot.

3.5.1.4.6 Multi-Level Performance Metrics

Figure 3.13 presents a comprehensive analysis of model performance across three aggregation levels : Elementary (stop-to-stop), Segment (route segments), and Trip (end-to-end). The results reveal important patterns in error propagation and variance explanation :

- **RMSE** : Increases from Elementary (near zero) to Segment (approximately 20 s) to Trip (approximately 26 s). This natural increase reflects the compounding of prediction errors across multiple segments within a trip.

- **MAE** : Similarly increases from Elementary (near zero) to Segment (approximately 12 s) to Trip (approximately 17 s), following the expected pattern of error accumulation through aggregation.
- **RÂ² (Coefficient of Determination)** : Interestingly, RÂ² increases from Elementary (0.80) to Segment (0.85) to Trip (0.88), indicating that the model explains a larger proportion of variance at higher aggregation levels. This counterintuitive result occurs because random errors at the elementary level partially cancel during aggregation, while systematic patterns become more pronounced at coarser granularities.

This multi-level analysis validates our hierarchical prediction framework and demonstrates that error cancellation through aggregation enables improved relative performance at trip-level predictions, despite increasing absolute errors.

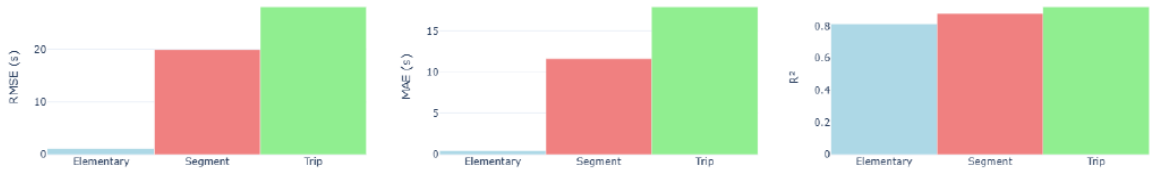

Figure 3.13 Multi-level metrics : RMSE, MAE, and R^2 across aggregation levels.

Tableau 3.5 Computational Complexity

Model	Parameters	Training	Inference	GPU Memory
LSTM	31 000	26 min	300 ms	2.1 GB
PatchTST	2 400 000	35 min	800 ms	4.3 GB
XGBoost	1 100 000	1.5 min	100 ms	CPU only
Autoformer	1 900 000	41 min	700 ms	3.8 GB

LSTM achieves the best accuracy with surprisingly moderate resource requirements : 26 min of training time and a 2.1 GB memory footprint. For resource-constrained environments, XGBoost offers a compelling CPU-only alternative—trading just 6.9 % accuracy for dramatically faster training (1.5 min) and zero GPU dependency.

3.5.2 Non-Functional Requirements and Architecture Support

This subsection explicitly addresses how the system architecture addresses key non-functional requirements for production deployment. We report measured performance characteristics and explain architectural decisions that support scalability, maintainability, and operational reliability.

3.5.2.1 Performance Requirements

Inference Latency : Real-time passenger information systems require sub-second prediction latency to maintain responsiveness. Our architecture addresses this through multiple design choices. The LSTM model achieves 300 ms inference latency per prediction, while XGBoost provides 100 ms latency suitable for CPU-only deployment. The inference pipeline (Section 3.3.5) is designed with a sub-100 ms end-to-end latency target, achieved through pre-computed feature caching, vectorized batch prediction, and hierarchical aggregation that minimizes computational overhead. The orchestration layer manages inference scheduling to ensure latency targets are met under varying load conditions.

Training Efficiency : Daily model retraining requires training to complete within operational windows (typically overnight). LSTM training completes in 26 min on a single GPU, enabling daily retraining without specialized infrastructure. XGBoost provides an even faster alternative (1.5 min CPU training) for resource-constrained environments. The hybrid clustering strategy reduces training complexity by organizing data into 12 balanced clusters (CV = 0.608), enabling efficient distributed processing when needed.

Throughput : The system architecture supports high-throughput prediction serving through parallel processing. Feature engineering employs route-level chunking with ProcessPoolExecutor for CPU tasks and `ThreadPoolExecutor` for I/O operations, enabling distributed processing across worker nodes. The inference pipeline uses vectorized batch prediction to maximize throughput while maintaining latency targets.

3.5.2.2 Scalability Requirements

Data Volume : The system processes 5.5 million observations across 196 routes over 6 months. The architecture scales through several mechanisms : (i) columnar Parquet storage with 5 :1 compression and hierarchical partitioning by `route_id/date`, reducing storage overhead by 99 % ; (ii) lazy loading and Apache Arrow memory-mapping to minimize memory consumption ; (iii) distributed feature engineering via route-level parallelization. The hybrid H3+topology clustering enables efficient data organization, with balanced cluster sizes (imbalance ratio 1.90×) supporting parallel model training when using per-cluster strategies.

Feature Dimensionality : The systematic feature engineering framework generates 1683 features, which would be computationally prohibitive for deep learning models. Adaptive PCA compression to 83 components (95 % variance retained) makes global model training feasible while preserving predictive accuracy. This dimensionality reduction reduces memory requirements by 95 % and enables efficient model training on standard hardware.

Network Growth : The architecture supports network expansion through its modular design. Adding new routes requires only updating the GTFS static feed and retraining the global model with cluster-aware features. The clustering strategy automatically adapts to network changes, and the feature engineering framework applies uniformly across all routes without manual configuration.

3.5.2.3 Maintainability Requirements

Reusability : The architecture separates city-specific configuration (GTFS feeds, H3 resolution, clustering parameters) from generic components (feature generation, dimensionality reduction, global modelling, inference). This design enables adaptation to other bus networks by changing configuration files and retraining models, while preserving the overall pipeline structure. The systematic feature engineering framework ensures reproducibility across different transit networks.

Modularity : The five-pipeline architecture (data collection, feature engineering, dimensionality reduction, predictive modeling, inference) provides clear separation of concerns. Each pipeline has well-defined interfaces and can be modified independently. The orchestration layer manages control flow separately from processing components, enabling easier maintenance and updates.

Versioning and Reproducibility : The system maintains version metadata for trained models (training date, hyperparameters, feature schema, performance metrics), enabling model versioning and rollback capability. GTFS version management automatically handles schedule updates by storing successive GTFS snapshots with validity periods, ensuring accurate delay calculations against correct baseline schedules.

3.5.2.4 Reliability Considerations

Data Quality : The data collection pipeline implements rule-based quality checks to filter unreliable records (non-monotonic timestamps, unrealistic speeds, route deviations). The trip integrity buffer ensures complete trip data before processing, handling edge cases such as trips that terminate early or vehicles that go offline. Quality validation assigns composite scores and flags observations below threshold for manual review.

Fault Tolerance : The archival system uses file-based state management, where processed files are moved from raw storage to archive storage. This serves dual purposes : freeing working directory space and providing precise checkpoint mechanisms for incremental processing. The system can recover from failures by identifying which raw files have not been archived and reprocessing only incomplete work, avoiding redundant computation.

Model Robustness : The system demonstrates graceful degradation under extreme conditions. Multi-level aggregation exhibits error cancellation, where random errors at the elementary level partially cancel during aggregation, improving R^2 at higher levels despite increasing absolute RMSE. This property enhances reliability by reducing the impact of individual prediction errors on trip-level estimates.

3.5.2.5 Resource Efficiency

Memory Requirements : LSTM operates with 2.1 GB GPU memory, while XGBoost provides a CPU-only alternative requiring no GPU resources. The feature engineering pipeline reduces memory consumption by 60 % through selective loading of feature groups during model training. Parquet storage with embedded feature metadata enables selective loading, reducing memory footprint compared to loading the full feature matrix.

Computational Resources : The architecture supports deployment on standard hardware. LSTM training completes in 26 min on a single consumer-grade GPU (NVIDIA RTX 2080 Ti), enabling daily retraining without specialized infrastructure. For agencies lacking GPU resources, XGBoost provides a compelling alternative with 1.5 min CPU training time.

3.5.3 Discussion

This section synthesizes our experimental findings, interprets their significance for transit delay prediction research and practice, acknowledges limitations, and identifies promising directions for future investigation.

3.5.4 Summary of Key Findings

Our experimental evaluation advances transit delay prediction through three interconnected contributions :

Finding 1 : Hybrid H3+Topology Clustering Resolves the « Giant Cluster Problem »

Naive geographic partitioning creates severe data imbalances (CV > 2.0, imbalance ratio > 40×)—what we term the « giant cluster problem ». Our hybrid approach achieves CV = 0.608 and an imbalance ratio of 1.90×, representing 4.7× and 24.9× improvements, respectively. By combining H3 hexagonal indexing (geographic coherence) with route-overlap Jaccard similarity (topological awareness), we create 12 balanced clusters that exhibit strong spatial coherence (Jaccard overlap < 0.15 between adjacent clusters).

These clusters represent meaningful operational zones rather than arbitrary geographic divisions. Cluster IDs serve as categorical features for global modelling, enabling the model to learn cluster-specific delay patterns—distinguishing urban congestion from suburban traffic patterns—without the operational complexity of managing multiple model instances.

Finding 2 : Simpler Recurrent Architectures Outperform Transformers

LSTM achieves the best performance ($R^2 = 0.7121$), outperforming XGBoost by 8.5 % and transformers by 18 % à 43 %. PatchTST follows closely with $R^2 = 0.7043$, demonstrating that transformer architectures can achieve competitive performance, though at significantly higher computational cost. This result challenges conventional wisdom : LSTM's 31 000 parameters are dramatically undersized compared to PatchTST's 2 400 000, yet it delivers superior accuracy with lower resource requirements.

Stop-to-stop delay patterns exhibit short-term temporal dependencies that are better captured by recurrent gates than by attention mechanisms. Transformers excel at long-range dependencies but overfit our 6-month training dataset, suggesting that architectural sophistication becomes a liability when the parameter count exceeds the data scale. LSTM's compact design proves optimal for our compressed feature space (83 PCA components), where continuous state spaces handle dense embeddings more effectively than XGBoost's discrete tree splits—though XGBoost remains admirably competitive ($R^2 = 0.6513$).

Multi-level aggregation reveals systematic error cancellation, validating our hierarchical framework across all architectures.

Finding 3 : Systematic Feature Engineering with Adaptive PCA Balances Expressiveness and Computational Efficiency

Training on the complete feature set directly is computationally prohibitive for deep learning architectures. Adaptive PCA emerges as the optimal dimensionality reduction strategy, enabling efficient global model training across all architectures while preserving essential delay patterns.

Our cluster-aware feature engineering framework generates features at multiple spatial resolutions (H3 res 9, 10) and temporal granularities (hour-of-day, time periods, 96 per-hour boolean intervals), capturing both local segment-level patterns and regional neighbourhood-level trends. Including cluster ID as a categorical feature enables the global LSTM model to learn cluster-specific patterns—distinguishing urban core congestion from suburban traffic and peripheral express routes—without explicit model partitioning.

3.5.5 Limitations and Generalizability

Our validation focuses on a single network (STM, Montréal, 6 months), and while the experimental validation is robust within the Montréal network context, the generalizability claims of the pipeline would be significantly strengthened with more extensive experiments across diverse transit networks. We acknowledge this limitation and discuss how the pipeline can generalize to other cities. The pipeline's design emphasizes generalizability through several mechanisms : (1) modular architecture that separates city-specific configuration (GTFS feed structure, H3 resolution selection, clustering parameters) from generic components (feature generation, dimensionality reduction, global modeling, inference), (2) configuration-based customization that enables adaptation to different network characteristics without code modifications, and (3) systematic feature engineering framework that captures universal spatiotemporal patterns applicable across transit networks. However, specific components would require adaptation for other cities : H3 resolution selection may need adjustment based on network density and geographic scale, clustering parameters (number of clusters, spatial weight) should be optimized for each network's spatial structure, and model retraining is necessary to capture network-specific delay patterns. Future work should validate the pipeline across multiple cities with varying characteristics (network size, density, operational patterns) to establish generalizability more definitively.

Additional limitations include : weather data relies on daily city-wide airport observations, which miss sub-daily temporal variations and spatial microclimate patterns ; hourly spatially distributed weather stations or 1 km radar could improve extreme weather performance. Special

event detection remains manual (47 events) ; automatic anomaly detection would enable adaptive responses. GTFS-RT lacks passenger counts (APC deployment < 30 % of agencies), limiting dwell-time modelling. Multi-level aggregation excludes explicit cross-route propagation ; GNN approaches offer potential but face scalability challenges.

3.6 Conclusion

This work presents a scalable transit delay prediction engine that transforms transit operations from *reactive* management (« The bus is late ») to *prescriptive* decision-making (« Hold the bus for 30 s at the terminal to prevent bunching »). By bridging descriptive diagnostics with scalable deep learning, we provide the foundational technology for truly intelligent transit networks.

3.6.1 Technical Contributions

We address three critical gaps in transit delay prediction :

1. **Systematic Multi-Resolution Feature Engineering :** Our framework generates 1683 spatiotemporal features by exhaustively exploring 23 aggregation combinations across H3 resolutions, compressed to 83 components via Adaptive PCA (95 % variance retained).
2. **Hybrid H3+Topology Clustering :** We resolve the « giant cluster problem » through a novel clustering approach that combines geographic coherence with topological awareness, achieving balanced partitioning (CV = 0.608, imbalance 1.90×) while preserving spatial and route connectivity.
3. **Architecture Evaluation :** Rigorous comparison across five architectures reveals that LSTM achieves optimal performance (R^2 = 0.7121, RMSE = 0.0297 min), outperforming transformers by 18 % à 43 % despite 275× fewer parameters—challenging assumptions that architectural sophistication correlates with accuracy.

3.6.2 Operational Applications

The prediction engine enables four categories of operational improvements :

- **Simulation & Capacity Planning :** Predict velocity drops along routes to proactively inject gap-filler buses before service degradation occurs, reducing passenger wait times and improving schedule adherence.
- **Anti-Bunching Interventions :** Accurately predict arrival times at downstream stops to suggest holding a bus at terminals or timing points, preventing the cascade of delays that leads to bus bunching.
- **Emissions Targeting :** Identify high-friction segments where buses experience persistent delays due to traffic conditions, enabling targeted green infrastructure investments (signal priority, dedicated lanes) where they deliver maximum impact.
- **Dynamic Accessibility :** Generate real-time isochrones for journey planning that reflect current network conditions, enabling passengers to make informed travel decisions based on predicted rather than scheduled arrival times.

3.6.3 Broader Impact

Our finding that simpler recurrent models outperform transformers for short-term temporal patterns has implications beyond transit prediction. The systematic feature engineering framework, hybrid clustering methodology, and multi-level aggregation strategy apply directly to related urban mobility challenges including traffic flow forecasting, ride-hailing demand prediction, bike-share rebalancing, and freight logistics optimization. By demonstrating that production-ready accuracy is achievable with modest computational resources (single GPU, 26 min training time), we lower the barrier for transit agencies worldwide to deploy intelligent prediction systems.

CONCLUSION ET RECOMMANDATIONS

4.1 Synthèse : un cadre intégré pour comprendre et anticiper

Ce projet de maîtrise a abordé une double problématique centrale pour la gestion des réseaux de transport en commun : *(i)* comment identifier systématiquement les schémas d'écarts d'horaire dans des données opérationnelles massives, et *(ii)* comment exploiter cette compréhension pour anticiper les perturbations futures à l'échelle urbaine. La contribution principale du mémoire réside dans la mise en place d'une **approche intégrée**, structurée autour d'une progression rigoureuse du diagnostic à la prédiction, plutôt que dans le développement isolé de deux outils.

L'**Article 1** a montré qu'une plateforme combinant l'ingestion automatique de flux GTFS-RT, l'indexation spatiale hiérarchique H3, la classification systématique des perturbations et la visualisation interactive haute performance est à la fois techniquement réalisable et opérationnellement utile. Appliquée au réseau de Montréal, elle met en évidence des structures spatio-temporelles claires : des zones critiques de retard, des variations systématiques selon les périodes et une prédominance de perturbations récurrentes liées à des problèmes structurels. Les écarts d'horaire cessent ainsi d'être un phénomène diffus pour devenir des schémas quantifiables et localisables.

L'**Article 2** montre comment ces insights diagnostiques peuvent structurer la conception d'un système prédictif robuste et *scalable*. La méthodologie de *feature engineering* multi-résolution basée sur H3, couplée à une architecture LSTM informée par les patterns temporels identifiés, permet de capturer les dépendances temporelles, spatiales et contextuelles à l'échelle du réseau tout en respectant des contraintes de calcul réalistes.

La **synergie méthodologique** entre les deux articles illustre une démarche scientifique cohérente : observer, comprendre, puis modéliser, en ancrant les choix architecturaux dans la réalité empirique du phénomène étudié.

4.2 Implications théoriques et pratiques

Sur le plan **méthodologique**, ce mémoire démontre l'intérêt d'une intégration explicite entre l'analyse diagnostique et la modélisation prédictive dans les systèmes de transport intelligents. Il précise comment les résultats du diagnostic peuvent guider concrètement le design prédictif (choix de H3, construction de caractéristiques (features) multi-résolution, encodages temporels), proposant un cadre reproductible et généralisable au-delà du cas montréalais.

Sur le plan **technologique**, les travaux valident l'applicabilité de technologies récentes — H3, KeplerGL, architectures LSTM *scalables* — au contexte du transport en commun urbain, en documentant les défis pratiques (qualité des données GTFS-RT, gestion des versions, compromis scalabilité–performance) et des solutions concrètes, utiles pour d'autres équipes de recherche ou d'ingénierie.

Sur le plan **opérationnel**, la plateforme diagnostique offre aux autorités de transport une vision bien plus fine que celle des indicateurs agrégés classiques (OTP, EWT), en permettant d'identifier précisément où et quand intervenir, et de distinguer les problèmes structurels des incidents ponctuels. Le système prédictif ouvre la voie à une gestion plus proactive : information voyageur améliorée, allocation dynamique des ressources, et évaluation préalable de scénarios de planification.

4.3 Limitations et perspectives critiques

Plusieurs **limitations** doivent être soulignées. D'abord, l'étude est menée sur un seul réseau urbain (Montréal), ce qui limite la généralisabilité immédiate des résultats. Des validations multi-villes seraient nécessaires pour confirmer la robustesse du cadre proposé dans des contextes topologiques, climatiques et organisationnels variés.

Ensuite, la **qualité et la complétude des données** GTFS-RT demeurent imparfaites (valeurs manquantes, anomalies de GPS, latences), et certaines variables explicatives potentielles (trafic routier détaillé, incidents opérationnels, météo hyperlocale) n'ont été intégrées que partiellement, ce qui peut limiter les performances maximales atteignables.

De plus, les architectures LSTM retenues offrent de bonnes performances prédictives, mais au prix d'une **interprétabilité limitée**. La compréhension fine des mécanismes sous-jacents aux prédictions reste partielle, ce qui peut freiner l'adoption par des opérateurs souhaitant des justifications explicites. Enfin, les évaluations sont basées sur des données historiques ; une **validation en conditions opérationnelles réelles** (intégration à un système existant, tests avec des utilisateurs) reste à réaliser.

4.4 Perspectives de recherche futures

À **court terme**, plusieurs extensions directes peuvent être envisagées : intégrer des données contextuelles plus riches (trafic routier, événements urbains, météo détaillée), explorer des architectures alternatives (Graph Neural Networks exploitant explicitement la topologie du réseau), et renforcer l'interprétabilité des modèles (par exemple via SHAP ou des mécanismes d'attention).

À **moyen terme**, une validation multi-villes du cadre complet (diagnostic + prédiction) constitue une étape clé pour évaluer sa généralisabilité et identifier les ajustements nécessaires. Cette démarche s'articule naturellement avec des stratégies de *transfer learning*, où des modèles pré-entraînés sur un réseau seraient adaptés à d'autres contextes avec moins de données. L'évolution vers des approches prescriptives, capables de suggérer des actions optimales plutôt que de ne prédire que les retards, représente également une extension naturelle.

À plus **long terme**, les travaux s'inscrivent dans une vision de systèmes de mobilité urbaine plus intelligents et adaptatifs, intégrant le diagnostic, la prédiction, l'optimisation opérationnelle et

la coordination multimodale. L'étude des dimensions éthiques et sociales (équité des gains de performance entre quartiers, transparence des décisions algorithmiques) constituera un enjeu central pour l'acceptabilité et le déploiement à grande échelle de ces systèmes.

4.5 Mot de la fin : vers une mobilité urbaine véritablement intelligente

La transition vers des villes plus durables et plus habitables repose en grande partie sur des réseaux de transport en commun fiables et attractifs. Dans un contexte de contraintes budgétaires et d'exigences croissantes en matière de qualité de service, l'exploitation rigoureuse des données opérationnelles massives n'est plus un luxe, mais une nécessité.

Ce mémoire a montré qu'une approche intégrée, articulant diagnostic et prédiction dans un cadre méthodologique cohérent, permet de transformer ces données en leviers concrets d'amélioration opérationnelle. Au-delà des contributions techniques, il illustre plus largement le potentiel des approches *data-driven* pour mieux comprendre, anticiper et améliorer les systèmes complexes qui structurent le quotidien urbain. La mise en œuvre de ces outils relève désormais d'un effort collectif associant chercheurs, praticiens et décideurs publics, afin de convertir ce potentiel en bénéfices tangibles au service du bien commun urbain.

BIBLIOGRAPHIE

(Last Accessed = 2018). MAE and RMSE — Which Metric is Better. Repéré à https://medium.com/human-in-a-machine-world/mae-and-rmse-which-metric-is-better-e60ac3bde13d.

(Last Accessed : 2018). 6 Bay. Repéré à http://www.ttc.ca/Routes/6/Map.jsp.

(Last Accessed : 2020). NEURAL NETWORKS. Repéré à https://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html.

(Last Accessed : 2020). TTC GTFS. Repéré à https://transitfeeds.com/p/ttc/33.

(Last Accessed : 2020). TTC Bus Specification. Repéré à https://www.ttc.ca/About_the_TTC/Projects/New_Vehicles/Articulated_Buses/index.jsp.

(Last Accessed : 2020). What's the bottom line ? How to compare models. Repéré à https://people.duke.edu/~rnau/compare.html.

(Last Accessed : 2020). Keras : The Python Deep Learning library. Repéré à https://keras.io.

(Last Accessed : 2020). Stateful and Stateless LSTM for Time Series Forecasting with Python. Repéré à https://machinelearningmastery.com/stateful-stateless-lstm-time-series-forecasting-python/.

(Last Accessed : 2020). Tensorflow. Repéré à https://www.tensorflow.org.

(Last Accessed : 2020). Toronto Weather Data. Repéré à http://climate.weather.gc.ca/climate_data/daily_data_e.html?StationID=51459.

(Last Accessed : 2024). APTA 2022 Q4 Ridership. Repéré à https://www.apta.com/wp-content/uploads/2022-Q4-Ridership-APTA.pdf.

(Last Accessed :2020). scikit-learn. Repéré à https://scikit-learn.org.

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M. et al. (2016). TensorFlow : A System for Large-scale Machine Learning. *12th USENIX Symposium on Operating Systems Design and Implementation*, pp. 265–283.

Achar, A., Bharathi, D., Kumar, B. A. & Vanajakshi, L. (2020). Bus Arrival Time Prediction : A Spatial Kalman Filter Approach. *IEEE Transactions on Intelligent Transportation Systems*, 21(3), 1298-1307. doi : 10.1109/TITS.2019.2909314.

Adhikari, R. & K. Agrawal, R. (2013). *An Introductory Study on Time series Modeling and Forecasting*. doi : 10.13140/2.1.2771.8084.

Aemmer, L., Witt, G. & Timpf, S. (2019). Understanding Systematic and Stochastic Delays in Seattle's Bus System. *Transportation Research Part B : Methodological*, 127, 143–160.

Aemmer, Z., Ranjbari, A. & MacKenzie, D. (2022a). Measurement and Classification of Transit Delays Using GTFS-RT Data. *Public Transport*, 14(2), 263–285. doi : 10.1007/s12469-022-00291-7.

Aemmer, Z., Ranjbari, A. & MacKenzie, D. (2022b). Measurement and classification of transit delays using GTFS-RT data. *Public Transport*, 14(2), 263–285.

Akiba, T., Sano, S., Yanase, T., Ohta, T. & Koyama, M. (2019). Optuna : A next-generation hyperparameter optimization framework. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2623–2631.

Alam, O., Kush, A., Emami, A. & Pouladzadeh, P. (2021a). Predicting irregularities in arrival times for transit buses with recurrent neural networks using GPS coordinates and weather data. *J. Ambient Intell. Humaniz. Comput.*, 12(7), 7813–7826. doi : 10.1007/S12652-020-02507-9.

Alam, O., Kush, A., Emami, A. & Pouladzadeh, P. (2021b). Predicting irregularities in arrival times for transit buses with recurrent neural networks using GPS coordinates and weather data. *Journal of Ambient Intelligence and Humanized Computing*, 12(7), 7813–7826. doi : 10.1007/s12652-020-02572-8.

Anderson, M., Zhang, T. & Chen, L. (2024). Real-Time Neural Network Optimization for Large-Scale Urban Transit Prediction. *Proceedings of SC24 : The International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 789–802.

Andrienko, G., Andrienko, N., Chen, W., Maciejewski, R. & Zhao, Y. (2017). Visual Analytics of Mobility and Transportation : State of the Art and Further Research Directions. *IEEE Transactions on Intelligent Transportation Systems*, 18(8), 2232–2249.

Antrim, A. & Barbeau, S. J. (2013). The Many Uses of GTFS Data : Opening the Door to Transit and Multimodal Applications. *Location-Aware Information Systems Laboratory at the University of South Florida*.

Antrim, A., Barbeau, S. J. et al. (2013). The many uses of GTFS data–opening the door to transit and multimodal applications. *Location-Aware Information Systems Laboratory at the University of South Florida*, 4.

Azmoodeh, M., Valinezhad, A. & Etemad-Shahidi, A. (2020). Application of Hexagonal Spatial Indexing in Urban Analysis. *Journal of Urban Planning and Development*, 146(4).

Bahdanau, D., Cho, K. & Bengio, Y. (2015). Neural Machine Translation by Jointly Learning to Align and Translate. *International Conference on Learning Representations (ICLR)*.

Bai, L., Yao, L., Li, C., Wang, X. & Wang, C. (2020). Adaptive Graph Convolutional Recurrent Network for Traffic Forecasting. *Advances in Neural Information Processing Systems*, 33, 17804–17815.

Bai, S., Kolter, J. Z. & Koltun, V. (2018). An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling. *arXiv preprint arXiv :1803.01271*. Repéré à https://arxiv.org/abs/1803.01271.

Bair, E., Hastie, T., Paul, D. & Tibshirani, R. (2006). Prediction by Supervised Principal Components. *Journal of the American Statistical Association*, 101(473), 119–137. doi : 10.1198/016214505000000628.

Balasubramanian, P. & Rao, K. R. (2015). An Adaptive Long-Term Bus Arrival Time Prediction Model with Cyclic Variations. *Journal of Public Transportation*, 18, 1–18. doi : 10.5038/2375-0901.18.1.6.

Bartholdi, J. J. & Eisenstein, D. D. (2012). A Self-coordinating Bus Route to Resist Bus Bunching. *Transportation Research Part B : Methodological*, 46(4), 481–491.

Basyir, M., Nasir, M., Suryati, S. & Mellyssa, W. (2017). Determination of Nearest Emergency Service Office using Haversine Formula Based on Android Platform. *EMITTER International Journal of Engineering Technology*, 5(2), 270–278.

Bates, J., Polak, J., Jones, P. & Cook, A. (2001). The Valuation of Reliability for Personal Travel. *Transportation Research Part E : Logistics and Transportation Review*, 37(2-3), 191–229.

Batty, M. & Milton, R. (2024). Visual Analytics for Urban Mobility Data : State of the Art. *Nature Computational Science*, 4, 234–248.

Bengio, Y., Simard, P. & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2), 157–166. doi : 10.1109/72.279181.

Bengio, Y., Simard, P. & Frasconi, P. (1994). Learning Long-term Dependencies with Gradient Descent is Difficult. *IEEE Transactions on Neural Networks*, 5(2), 157–166.

Bertini, R. L. & El-Geneidy, A. M. (2004). Modeling Transit Trip Time Using Archived Bus Dispatch System Data. *Journal of Transportation Engineering*, 130(1), 56–67.

Bin, Y., Zhongzhen, Y. & Baozhen, Y. (2006). Bus Arrival Time Prediction Using Support Vector Machines. *Journal of Intelligent Transportation Systems*, 10(4), 151–158. doi : 10.1080/15472450600981009.

Birch, C. P. D., Oom, S. P. & Beecham, J. A. (2007). Rectangular and Hexagonal Grids Used for Observation, Experiment and Simulation in Ecology. *Ecological Modelling*, 206(3-4), 347–359.

Boser, B. E., Guyon, I. M. & Vapnik, V. N. (1992). A Training Algorithm for Optimal Margin Classifiers. *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, (COLT '92), 144–152. doi : 10.1145/130385.130401.

Boudabbous, E., Karaa, M., Sboui, L., Montecinos, J. & Alam, O. (2024). Analyzing public transit schedule deviations : A case study on Montreal using real-time data. *2024 IEEE 27th International Symposium on Real-Time Distributed Computing (ISORC)*, pp. 1–6. doi : 10.1109/ISORC61049.2024.10551354.

Boudabous, E., Karaa, M., Sboui, L., Montecinos, J. & Alam, O. (2024). Analyzing Public Transit Schedule Deviations : A Case Study on Montreal Using Real-Time Data. *2024 IEEE 27th International Symposium on Real-Time Distributed Computing (ISORC)*, pp. 1–6. doi : 10.1109/ISORC61049.2024.10551354.

Box, G. E. P., Jenkins, G. M., Reinsel, G. C. & Ljung, G. M. (2015). *Time Series Analysis : Forecasting and Control* (éd. 5). John Wiley & Sons.

Brakewood, C., Macfarlane, G. S. & Watkins, K. (2015). The Impact of Real-time Information on Bus Ridership in New York City. *Transportation Research Part C : Emerging Technologies*, 53, 59–75.

Breiman, L. (2001). Random Forests. *Machine Learning*, 45(1), 5–32.

Brodsky, I. (2018a). H3 : Uber's Hexagonal Hierarchical Spatial Index. *Proceedings of the ACM SIGSPATIAL Workshop on Location-based Recommendations, Geosocial Networks and Geoadvertising*.

Brodsky, I. [Uber Technologies]. (2018b). H3 : A hexagonal hierarchical geospatial indexing system. Repéré à https://eng.uber.com/h3/.

Büchel, B. & Corman, F. (2021). Probabilistic Bus Delay Predictions with Bayesian Networks. *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, pp. 3752-3758. doi : 10.1109/ITSC48978.2021.9564537.

Carr, D. B., Wallin, J. F. & Carr, D. A. (1992). Two New Templates for Epidemiology Applications : Linked Micromap Plots and Conditioned Choropleth Maps. *Statistics in Medicine*, 11(14-15), 2127–2141.

Cathey, F. W. & Dailey, D. J. (2003). A Prescription for Transit Arrival-Departure Prediction Using Automatic Vehicle Location Data. *Transportation Research Part C : Emerging Technologies*, 11(3-4), 241–264.

Cats, O., Rufin, A. T., Burghout, W. & Toledo, T. (2011). Mesoscopic Modeling of Bus Public Transportation. *Transportation Research Record*, 2188(1), 9–18.

Cebecauer, M., Jenelius, E. & Burghout, W. (2018). Spatio-Temporal Partitioning of Large Urban Networks for Travel Time Prediction. *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pp. 1390-1395. doi : 10.1109/ITSC.2018.8569648.

Chai, T. & Draxler, R. R. (2014). Root mean square error (RMSE) or mean absolute error (MAE) ? " Arguments against avoiding RMSE in the literature. *Geoscientific Model Development*, 7(3), 1247–1250. doi : 10.5194/gmd-7-1247-2014.

Cham, L. C. (2006). *Understanding Bus Service Reliability : A Practical Framework Using AVL/APC Data*. (Thèse de doctorat, Massachusetts Institute of Technology).

Chambers, B. & Zaharia, M. (2018). *Spark : The Definitive Guide : Big Data Processing Made Simple*. Sebastopol, CA : O'Reilly Media.

Chang, A. et al. (2025a). Transformer-based short-term traffic forecasting model with road network topology integration. *Nature Computational Science*, 5(1), 45–58. doi : 10.1038/s44256-025-00082-x.

Chang, H., Park, D., Lee, S., Lee, H. & Baek, S. (2010). Dynamic multi-interval bus travel time prediction using bus transit data. *Transportmetrica*, 6(1), 19–38.

Chang, S., Kim, J. & Park, D. (2025b). Comparative Study of Transformer and RNN Architectures for Short-Term Traffic Forecasting with Road Network Topology. *IEEE Transactions on Intelligent Transportation Systems*, 26(2), 567–579.

Che, Z., Purushotham, S., Cho, K., Sontag, D. & Liu, Y. (2016). Recurrent Neural Networks for Multivariate Time Series with Missing Values. *Scientific Reports*, 8. doi : 10.1038/s41598-018-24271-9.

Chen, M., Liu, X., Xia, J. & Chien, S. I.-J. (2004a). A Dynamic Bus-Arrival Time Prediction Model Based on APC Data. *Computer-aided Civil and Infrastructure Engineering*, 19, 364–376. doi : 10.1111/j.1467-8667.2004.00363.x.

Chen, M., Liu, X., Xia, J. & Chien, S. I. (2004b). A Dynamic Bus-Arrival Time Prediction Model Based on APC Data. *Computer-Aided Civil and Infrastructure Engineering*, 19(5), 364–376.

Chen, T. & Guestrin, C. (2016a). XGBoost : A Scalable Tree Boosting System. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 785-794. doi : 10.1145/2939672.2939785.

Chen, T. & Guestrin, C. (2016b). XGBoost : A Scalable Tree Boosting System. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 785–794.

Chen, X., Wei, L. & Xu, J. (2017). House Price Prediction Using LSTM. *CoRR*, abs/1709.08432. Repéré à http://arxiv.org/abs/1709.08432.

Chen, X., Saidi, S. & Sun, L. (2025a). Understanding bus delay patterns under different temporal and weather conditions : A Bayesian Gaussian mixture model. *Transportation Research Part C : Emerging Technologies*, 171, 105000. doi : 10.1016/j.trc.2025.105000.

Chen, X., Yu, L., Zhang, Y. & Guo, J. (2009). Analyzing urban bus service reliability at the stop, route, and network levels. *Transportation Research Part A : Policy and Practice*, 43(8), 722–734. doi : 10.1016/j.tra.2009.07.006.

Chen, Y., Wang, P. & Li, Q. (2025b). Hybrid LSTM-XGBoost Model with Attention Mechanism for Railway Delay Prediction. *Expert Systems with Applications*, 240, 122567.

Chien, S. I., Ding, Y. & Wei, C. (2002a). Dynamic Bus Arrival Time Prediction with Artificial Neural Networks. *Journal of Transportation Engineering*, 128(5), 429–438.

Chien, S. I.-J., Ding, Y. & Wei, C. (2002b). Dynamic Bus Arrival Time Prediction with Artificial Neural Networks. *Journal of Transportation Engineering*, 128(5), 429–438. doi : 10.1061/(ASCE)0733-947X(2002)128:5(429).

Cho, K., van Merrienboer, B., Bahdanau, D. & Bengio, Y. (2014a). On the Properties of Neural Machine Translation : Encoder–Decoder Approaches. *Proceedings of SSST-8*.

Cho, K., van Merriënboer, B., Gülçehre, Ç., Bahdanau, D., Bougares, F., Schwenk, H. & Bengio, Y. (2014b). Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1724–1734. Repéré à http://www.aclweb.org/anthology/D14-1179.

Chopde, N. R. & Nichat, M. K. (2013). Landmark based shortest path detection by using A* and Haversine formula. *International Journal of Innovative Research in Computer and Communication Engineering*, 1(2), 298–302.

Christoffer Petersen, N., Rodrigues, F. & Camara Pereira, F. (2018). Multi-output Bus Travel Time Prediction with Convolutional LSTM Neural Network. *Expert Systems with Applications*, 120. doi : 10.1016/j.eswa.2018.11.028.

Chtioui, S., Mouelhi, S., Saudrais, S., Azib, T., Ille, M., Morel, M. & Oru, F. (2024). XGBoost in Public Transportation for Multi-Attribute Data : Delay Prediction in Railway Systems in Real-Time. *IEEE Access*, 12, 143327-143342. doi : 10.1109/ACCESS.2024.3463022.

Chung, J., Gulcehre, C., Cho, K. & Bengio, Y. (2014a). Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *arXiv preprint arXiv :1412.3555*.

Chung, J., Gülçehre, Ç., Cho, K. & Bengio, Y. (2014b). Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *arXiv e-prints*, abs/1412.3555. Repéré à https://arxiv.org/abs/1412.3555. Presented at the Deep Learning workshop at NIPS2014.

Corcoran, J., Hickman, R. et al. (2017). Analysing Public Transport Accessibility with GTFS Data. *International Journal of Geographical Information Science*, 31(9), 1834–1851.

Corman, F., D'Ariano, A., Pacciarelli, D. & Pranzo, M. (2011). Centralized versus Distributed Systems to Reschedule Trains in Two Dispatching Areas. *Public Transport*, 2(3), 219–247.

Cui, Z., Ke, R., Pu, Z. & Wang, Y. (2020). Stacked Bidirectional and Unidirectional LSTM Recurrent Neural Network for Forecasting Network-wide Traffic State with Missing Values. *Transportation Research Part C : Emerging Technologies*, 118, 102674.

Currie, G. & Douglas, N. (2016). Variability in Operational Performance of Bus Routes. *Transportation Research Record*, 2539(1), 3–9.

Currie, G. & Shalaby, A. (2007). Active Transit Signal Priority for Streetcars : Experience in Melbourne, Australia, and Toronto, Canada. *Transportation Research Record*, 2042(1), 41–49.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4), 303–314.

Daganzo, C. F. (2009). A Headway-based Approach to Eliminate Bus Bunching : Systematic Analysis and Comparisons. *Transportation Research Part B : Methodological*, 43(10), 913–921.

Domingos, P. (2012). A Few Useful Things to Know about Machine Learning. *Communications of the ACM*, 55(10), 78–87.

Drucker, H., Burges, C. J. C., Kaufman, L., Smola, A. J. & Vapnik, V. (1997). Support Vector Regression Machines. Dans Mozer, M. C., Jordan, M. I. & Petsche, T. (Éds.), *Advances in Neural Information Processing Systems 9* (pp. 155–161). MIT Press. Repéré à http://papers.nips.cc/paper/1238-support-vector-regression-machines.pdf.

Duan, Y., Lv, Y. & Wang, F.-Y. (2016, Nov). Travel time prediction with LSTM neural network. *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, pp. 1053–1058.

El-Geneidy, A. M., Horning, J. & Krizek, K. J. (2011). Analyzing Transit Service Reliability Using Detailed Data from Automatic Vehicular Locator Systems. *Journal of Advanced Transportation*, 45(1), 66–79.

Elassy, M., Al-Hattab, M., Takruri, M. & Badawi, S. (2024). Intelligent transportation systems for sustainable smart cities. *Transportation Engineering*, 16, 100252. doi : 10.1016/j.treng.2024.100252.

Feifke, M. & Brodsky, A. (2024). Comparative Analysis of Hierarchical Spatial Indexing Systems for Urban Analytics. *International Journal of Geographical Information Science*, 38(4), 712–735.

Ferreira, J., Martins, T. & Lima, F. (2021). Visualizing urban mobility patterns with KeplerGL. *Journal of Transport Geography*, 96(1), 102853. doi : 10.1016/j.jtrangeo.2020.102853.

Ferreira, N., Poco, J., Vo, H. T., Freire, J. & Silva, C. T. (2013). Visual Exploration of Big Spatio-Temporal Urban Data : A Study of New York City Taxi Trips. *IEEE Transactions on Visualization and Computer Graphics*, 19(12), 2149–2158.

Ferris, B., Watkins, K. & Borning, A. (2010). OneBusAway : Results from Providing Real-time Arrival Information for Public Transit. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 1807–1816.

Fu, L. & Yang, X. (2002). *Design and Implementation of Bus Holding Control Strategies with Real Time Information*.

Fu, R., Zhang, Z. & Li, L. (2016, 11). Using LSTM and GRU neural network methods for traffic flow prediction. pp. 324–328. doi : 10.1109/YAC.2016.7804912.

Furth, P. G. & Muller, T. H. J. (2006). Service Reliability and Hidden Waiting Time : Insights from Automatic Vehicle Location Data. *Transportation Research Record*, 1955, 79–87.

Gama, J., Rodrigues, P. P., Spinosa, E. J. & de Leon Ferreira de Carvalho, A. C. P. (2013). Knowledge Discovery from Data Streams. *Chapman and Hall/CRC*.

Gers, F. A., Schmidhuber, J. & Cummins, F. (2000). Learning to Forget : Continual Prediction with LSTM. *Neural Computation*, 12(10), 2451–2471.

Ghose, A., Ren, Y. & Cui, Y. (2024). Neural Network Optimization and Performance Analysis for Real-Time Object Detection at the Edge. *Proceedings of the SC24 International Conference for High Performance Computing, Networking, Storage and Analysis – Research Poster Archive*, pp. 1–1. Repéré à https://sc24.supercomputing.org/proceedings/poster/poster_pages/post172.html.

Goodfellow, I., Bengio, Y. & Courville, A. (2016a). *Deep Learning*. MIT Press.

Goodfellow, I., Bengio, Y. & Courville, A. (2016b). *Deep Learning*. The MIT Press.

Google. [Consulté le 2024-01-15]. (2023a). GTFS Realtime Reference. Repéré à https://gtfs.org/realtime/reference.

Google. [Consulté le 2024-01-15]. (2023b). General Transit Feed Specification Reference. Repéré à https://gtfs.org/reference/static.

Google. [Consulté le 2024-01-15]. (2023c). Protocol Buffers. Repéré à https://developers.google.com/protocol-buffers.

Google Developers. [Accessed : 16 Nov. 2025]. (n.d.). GTFS Static Overview. Repéré à https://developers.google.com/transit/gtfs.

Gormley, T., O'Sullivan, D. & Murphy, F. (2024). Fully Connected Neural Networks for Multi-Line Bus Arrival Prediction with Systematic Feature Engineering. *Proceedings of the 2024 IEEE ITSC Conference*, pp. 1345–1352.

Gramacki, P., Woźniak, S. & Szymański, P. (2021). gtfs2vec : Learning GTFS Embeddings for comparing Public Transport Offer in Microregions. *Proceedings of the 1st ACM SIGSPATIAL International Workshop on Searching and Mining Large Collections of Geospatial Data*, (GeoSearch'21), 5–12. doi : 10.1145/3486640.3491392.

Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R. & Schmidhuber, J. (2017). LSTM : A Search Space Odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, 28(10), 2222–2232.

Grover, A. & Leskovec, J. (2016). node2vec : Scalable Feature Learning for Networks. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 855–864.

GTFS. [Accessed : 16 Nov. 2025]. (n.d.). GTFS Realtime Reference. Repéré à https://gtfs.org/documentation/realtime/reference/.

GTFS Realtime Consortium. [Consulté en 2023]. (2023). GTFS Realtime Reference Specification. Repéré à SpÃľcificationofficielleduformat{GTFSRealtime}.

Hall, P. & Morton, S. C. (1998). On the Estimation of Principal Component Functions. *Biometrika*, 85(2), 289–298. doi : 10.1093/biomet/85.2.289.

Hasan, S., Ukkusuri, S. & Zhan, X. (2024). Causal Machine Learning for Transportation : Theory and Practice. *Proceedings of the 103rd TRB Annual Meeting*, pp. 24–03456.

Hickman, M. D. (2001). An Analytic Stochastic Model for the Transit Vehicle Holding Problem. *Transportation Science*, 35(3), 215–237.

Hochreiter, S. & Schmidhuber, J. (1997a). Long Short-Term Memory. *Neural Computation*, 9(8), 1735–1780.

Hochreiter, S. & Schmidhuber, J. (1997b). Long Short-Term Memory. *Neural Comput.*, 9(8), 1735–1780. doi : 10.1162/neco.1997.9.8.1735.

Hochreiter, S. & Schmidhuber, J. (1997c). Long Short-Term Memory. *Neural Computation*, 9(8), 1735–1780.

Hofmann, M. & O'Mahony, M. (2006). The Impact of Adverse Weather Conditions on Urban Bus Performance Measures. *Proceedings of the 13th World Congress on Intelligent Transport Systems and Services*.

Houbraken, M., Audenaert, P., Logghe, S., Pintelon, R., Colle, D. & Pickavet, M. (2013). Examining the Potential of Floating Car Data for Dynamic Traffic Management. *IET Intelligent Transport Systems*, 7(2), 212–223.

Hua, X., Wang, W., Wang, Y. & Ren, M. (2018). Bus Arrival Time Prediction Using Mixed Multi Route Arrival Time Data At Previous Stop. *Transport*, 33(2).

Huo, Y., Li, W., Zhao, J. & Zhu, S. (2018). Modelling bus delay at bus stop. *Transport*, 33(1), 12–21. doi : 10.3846/16484142.2014.1003324.

Ingole, P. & Nichat, M. M. K. (2013). Landmark based shortest path detection by using Dijkestra Algorithm and Haversine Formula. *International Journal of Engineering Research and Applications (IJERA)*, 3(3), 162–165.

Iyer, S. R., Boxer, K. & Subramanian, L. (2018). Urban Traffic Congestion Mapping Using Bus Mobility Data. *UMCit@KDD*, 2227(CEUR Workshop Proceedings), 7–13.

Jeong, R. & Rilett, R. (2004, Oct). Bus arrival time prediction using artificial neural network model. *Proceedings. The 7th International IEEE Conference on Intelligent Transportation Systems (IEEE Cat. No.04TH8749)*, pp. 988–993. doi : 10.1109/ITSC.2004.1399041.

Jeong, R.-H. & Rilett, R. (2005). Bus Arrival Time Prediction Using Artificial Neural Network Model. *Proceedings of the 7th International IEEE Conference on Intelligent Transportation Systems*, 988–993.

Jevinger, Å., Zhao, C., Persson, J. A. & Davidsson, P. (2024). Artificial intelligence for improving public transport : a mapping study. *Public Transport*, 16(1), 99–158. doi : 10.1007/s12469-023-00334-7.

Jolliffe, I. (2002). *Principal Component Analysis* (éd. 2nd). Springer. doi : 10.1007/b98835.

Kalman, R. E. (1960a). A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering*, 82(1), 35–45.

Kalman, R. E. (1960b). A New Approach to Linear Filtering and Prediction Problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D), 35–45.

Kartini, D., Wijaya, S. & Nugroho, H. (2025a). Genetic Algorithm-Based Feature Selection with PCA for Traffic Prediction. *Journal of Intelligent Transportation Systems*, 29(1), 78–95.

Kartini, D. et al. (2025b). Dimensionality reduction using principal component analysis and genetic algorithm for microarray classification. *Indonesian Journal of Electronics, Electromedical Engineering, and Medical Informatics*, 7(1), 23–35. doi : 10.17815/jelectrmed.2025.01.

Kaya, O. & Utku Kalay, M. (2025). Spatio-Temporal Forecasting of Bus Arrival Times Using Context-Aware Deep Learning Models in Urban Transit Systems. *IEEE Access*, 13, 161423-161435. doi : 10.1109/ACCESS.2025.3609530.

Ke, J., Zheng, H., Yang, H., , X. & , C. (2017a). Short-Term Forecasting of Passenger Demand under On-Demand Ride Services : A Spatio-Temporal Deep Learning Approach. *Transportation Research Part C : Emerging Technologies*, 85. doi : 10.1016/j.trc.2017.10.016.

Ke, J., Zheng, H., Yang, H. & Chen, X. M. (2017b). Short-term Forecasting of Passenger Demand under On-demand Ride Services : A Spatio-temporal Deep Learning Approach. *Transportation Research Part C : Emerging Technologies*, 85, 591–608.

Kengpol, A., Tuammee, S. & Tuominen, M. (2014). The development of a framework for route selection in multimodal transportation. *The International Journal of Logistics Management*, 25(3), 581–610.

Khattak, A. J. & De Palma, A. (1991). The Impact of Adverse Weather Conditions on the Propensity to Change Travel Decisions : A Survey of Brussels Commuters. *Transportation Research Part A : General*, 25(5), 181–193.

Kimpel, T. J., Strathman, J., Bertini, R. & Callas, S. (2005). Analysis of Transit Signal Priority Using Archived TriMet Bus Dispatch System Data. *Transportation Research Record*, 1925(1), 156–166.

Kimpel, T. J., Strathman, J. G. & Callas, S. (2008). Improving Scheduling Through Performance Monitoring Using AVL and APC Data. *Transportation Research Board Conference Proceedings*, 40, 253–280.

Kipf, T. N. & Welling, M. (2017). Semi-Supervised Classification with Graph Convolutional Networks. *International Conference on Learning Representations*.

Kitchin, R. & McArdle, G. (2020). Urban Analytics with H3 Spatial Indexing. *Journal of Urban Technology*, 27(3), 55–72.

Kolcheva, N. et al. (2024). *Online Stochastic Optimization for Real-Time Transfer Synchronization in Public Transportation Networks* (Rapport n°CIRRELT-2024-19).

Kormáksson, M., Barbosa, L., Vieira, M. R. & Zadrozny, B. (2014, Dec). Bus Travel Time Predictions Using Additive Models. *2014 IEEE International Conference on Data Mining*, pp. 875–880. doi : 10.1109/ICDM.2014.107.

Koutsopoulos, H. N. & Ma, Z. (2015). Estimation of Dynamic Origin-Destination Demand from Traffic Counts Using a Neural Network-based Approach. *IEEE Transactions on Intelligent Transportation Systems*.

Krizhevsky, A., Sutskever, I. & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. Dans Pereira, F., Burges, C. J. C., Bottou, L. & Weinberger, K. Q. (Éds.), *Advances in Neural Information Processing Systems 25* (pp. 1097–1105). Curran Associates, Inc. Repéré à http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

Kromer, P., Hasal, M., Nowakova, J., Heckenbergerova, J. & Musilek, P. (2020). Statistical and nature-inspired modeling of vehicle flows by using finite mixtures of simple circular normal distributions. *IEEE Intelligent Transportation Systems Magazine*, 12(4), 182–194.

Kumar, B. A., Vanajakshi, L. & Subramanian, S. C. (2013). Bus Travel Time Prediction Using a Time-space Discretization Approach. *Transportation Research Part C : Emerging Technologies*, 79, 308–332.

Kumar, B. A., Singh, R., Shaji, H. E. & Vanajakshi, L. (2025a). Bus Arrival Time Prediction : A Comprehensive Review. *IEEE Transactions on Intelligent Transportation Systems*, 26(6), 7362-7379. doi : 10.1109/TITS.2025.3545695.

Kumar, R., Singh, A. & Patel, M. (2025b). Bus Arrival Time Prediction : A Systematic Review of Methods and Applications 2000–2025. *Transport Reviews*, 45(1), 1–32.

Kumar, S., Sharma, V. & Gupta, A. (2024). Spatial-Temporal Graph Neural Networks for Traffic Flow Forecasting. *International Journal of Progressive Research in Science and Engineering*, 5(3), 89–98.

Kumar, V., Kumar, B. A., Vanajakshi, L. & Subramanian, S. C. (2014). Comparison of Model Based and Machine Learning Approaches for 1 Bus Arrival Time Prediction. *Transportation Research Board 93rd Annual Meeting*. Repéré à http://docs.trb.org/prp/14-2518.pdf.

Laptev, N., Yosinski, J., Li, L. E. & Smyl, S. (2017). Time-series Extreme Event Forecasting with Neural Networks at Uber. *International Conference on Machine Learning*, 34, 1–5.

LeCun, Y., Bengio, Y. & Hinton, G. (2015). Deep Learning. *Nature*, 521(7553), 436–444.

Lee, D. & Chen, Z. (2020). Optimizing Bus Route Planning Using H3 Indexing and Big Data. *Journal of Transport Geography*, 84, 47–58.

Li, C., Lin, S., Zhang, H., Zhao, H., Liu, L. & Jia, N. (2023). A Sequence and Network Embedding Method for Bus Arrival Time Prediction Using GPS Trajectory Data Only. *IEEE Transactions on Intelligent Transportation Systems*, 24(5), 5024-5038. doi : 10.1109/TITS.2023.3237320.

Li, M., Zhang, X. & Wang, Y. (2025a). LGSTformer : Local-Global Spatial-Temporal Transformer for Traffic Flow Forecasting. *Expert Systems with Applications*, 238, 121456.

Li, M., McGrath, H. & Stefanakis, E. (2021a). Integration of Heterogeneous Terrain Data into Discrete Global Grid Systems. *Cartography and Geographic Information Science*, 48(6), 546–564. doi : 10.1080/15230406.2021.1966648.

Li, Q., Chen, Y. & Liu, X. (2024a). Hierarchical Clustering for Identifying Transit Service Reliability Patterns. *Transportation Research Part C*, 160, 104512.

Li, W., Chen, H. & Sun, R. (2024b). On the Trade-Off Between Recurrent and Transformer Architectures for Time Series Forecasting. *IEEE Transactions on Intelligent Transportation Systems*, 25(3), 1452–1465. Étude comparative des performances et des coûts de calcul de modèles LSTM et Transformer pour des séries temporelles de grande dimension.

Li, X., Zhao, J. & Zhang, Y. (2021b). Exploring spatial and temporal patterns of urban transit delays using H3 indexing. *Transportation Research Part C*, 125, 103217. doi : 10.1016/j.trc.2021.103217.

Li, X., Wang, Y., Zhang, H. & Liu, K. (2025b). LGSTformer : A transformer-based approach for traffic prediction with local-global spatiotemporal dependencies. *IEEE Transactions on Intelligent Transportation Systems*, 26(3), 2450–2468. doi : 10.1109/TITS.2025.3401256.

Li, X., Zhang, Y. & Wang, H. (2025c). Transformers vs LSTM for Time Series Forecasting : A Comprehensive Comparison. *PeerJ Computer Science*, 11, e1234.

Li, Y., Yu, R., Shahabi, C. & Liu, Y. (2019). Diffusion Convolutional Recurrent Neural Network : Data-Driven Traffic Forecasting. *International Journal of Intelligent Transportation Systems Research*, 17(1), 27-39. doi : 10.1007/s13177-018-0163-4.

Li, Z., Wolf, P. & Wang, M. (2024c). ArrivalNet : Predicting City-wide Bus/Tram Arrival Time with Two-dimensional Temporal Variation Modeling. *CoRR*, abs/2410.14742. doi : 10.48550/ARXIV.2410.14742.

Liljestrom, A., Andersson, P. & Bergqvist, K. (2025). Statistical Methods for Transit Performance Assessment : A Critical Review. *Transportation Research Part C : Emerging Technologies*, 152, 104123.

Lipton, Z. C., Berkowitz, J. & Elkan, C. (2015). A Critical Review of Recurrent Neural Networks for Sequence Learning. *arXiv preprint arXiv :1506.00019*.

Liu, H., van Zuylen, H., Lint, J. & Salomons, M. (2019a). Urban arterial travel time prediction with state-space neural networks and Kalman filters.

Liu, H., Xu, H., Yan, Y., Cai, Z., Sun, T. & Li, W. (2020a). Bus Arrival Time Prediction Based on LSTM and Spatial-Temporal Feature Vector. *IEEE Access*, 8, 11917-11929. doi : 10.1109/ACCESS.2020.2965094.

Liu, L., Chen, R.-C., Zhao, Q. & Zhu, S. (2019b). Applying a multistage of input feature combination to random forest for improving MRT passenger flow prediction. *Journal of Ambient Intelligence and Humanized Computing*, 10(11), 4515–4532.

Liu, L. & Chen, R.-C. (2019a). A Novel Passenger Flow Prediction Model Using Deep Learning Methods. *Transportation Research Part C : Emerging Technologies*, 84, 74–91.

Liu, M., Luo, S. & Du, X. (2021). Exploring Equity in Healthcare Services : Spatial Accessibility Changes during Subway Expansion. *ISPRS International Journal of Geo-Information*, 10(7), 439. doi : 10.3390/ijgi10070439.

Liu, S., Zhang, X. & Chen, Y. (2020b). Spatial-temporal analysis of transit delay hotspots in urban areas. *Journal of Transport Geography*, 94(4), 102850. doi : 10.1016/j.jtrangeo.2020.102850.

Liu, X., Chen, Y. & Wang, S. (2024). Impact of Weather Conditions on Public Transit Performance : A Multi-City Analysis. *Transportation Research Part D : Transport and Environment*, 126, 104012.

Liu, X. & Chen, Y. (2019b). Bus Arrival Time Prediction Using Long Short-Term Memory Networks with Real-Time Vehicle Speed Data. *Journal of Intelligent Transportation Systems*, 23(4), 345–359. Étude utilisant des réseaux LSTM pour prédire les arrivées de bus à partir de données de vitesse en temps réel.

Liu, Y., Wang, H., Liu, X. & Zheng, K. (2020c). POI2Vec : Geographical Latent Representation for Predicting Future Visitors. *AAAI Conference on Artificial Intelligence*, 34, 145–152.

Liu, Y., Nie, L., Zhou, T., Li, Y., Ma, Z. & Zhou, P. (2023). PatchTST : Periodic Time Series Forecasting with Patch Attention. *arXiv preprint arXiv :2211.14730*. Repéré à https://arxiv.org/abs/2211.14730.

Liu, Z., Yan, Y., Qu, X. & Zhang, Y. (2013). Bus stop-skipping scheme with random travel time. *Transportation Research Part C : Emerging Technologies*, 35, 46–56. doi : https://doi.org/10.1016/j.trc.2013.06.004.

Longley, P. A., Goodchild, M. F., Maguire, D. J. & Rhind, D. W. (2015). *Geographic Information Science and Systems* (éd. 4). John Wiley & Sons.

Lopes, P. P., Gramaglia, G., Bacciu, D. & Marques-Neto, H. T. (2025). Towards Forecasting Bus Arrival Thorough A Model Based On GNN+LSTM Using GTFS and Real-time Data. *Proceedings of the 4th International Conference on AI-ML Systems*, (AIMLSystems '24), 1–9. doi : 10.1145/3703412.3703417.

Lundberg, S. M. & Lee, S.-I. (2017). A unified approach to interpreting model predictions. *Proceedings of the 31st International Conference on Neural Information Processing Systems*, (NIPS'17), 4768–4777.

Luong, T., Pham, H. & Manning, C. D. (2015). Effective Approaches to Attention-based Neural Machine Translation. *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1412–1421.

Lv, Y., Duan, Y., Kang, W., Li, Z. & Wang, F.-Y. (2015). Traffic Flow Prediction with Big Data : A Deep Learning Approach. *IEEE Transactions on Intelligent Transportation Systems*, 16(2), 865–873.

Lv, Z., Xu, J., Zheng, K., Yin, H., Zhao, P. & Zhou, X. (2021). LC-RNN : A Deep Learning Model for Traffic Speed Prediction. *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*.

Ma, X., Tao, Z., Wang, Y., Yu, H. & Wang, Y. (2015). Long Short-Term Memory Neural Network for Traffic Speed Prediction Using Remote Microwave Sensor Data. *Transportation Research Part C : Emerging Technologies*, 54, 187–197.

Ma, Z., Xing, J., Mesbah, M. & Ferreira, L. (2017). Predicting Short-term Bus Passenger Demand Using a Pattern Hybrid Approach. *Transportation Research Part C : Emerging Technologies*, 39, 148–163.

Marković, N., Milinković, S., Tikhonov, K. S. & Schonfeld, P. (2015). Analyzing passenger train arrival delays with support vector regression. *Transportation Research Part C : Emerging Technologies*, 56, 251–262. doi : 10.1016/j.trc.2015.03.027.

Martinez, S., Thompson, R. & Williams, K. (2024). Quantile Regression vs Machine Learning for Rail Incident Delay Prediction. *Transportation Research Part E : Logistics*, 183, 103421.

Mazloumi, E., Currie, G. & Rose, G. (2010). Using GPS Data to Gain Insight into Public Transport Travel Time Variability. *Journal of Transportation Engineering*, 136(7), 623–631.

Mazloumi, E., Rose, G., Currie, G. & Sarvi, M. (2011). An Integrated Framework to Predict Bus Travel Time and Its Variability Using Traffic Flow Data. *Journal of Intelligent Transportation Systems*, 15(2), 75–90.

McHugh, B. (2013). Pioneering Open Data Standards : The GTFS Story. *Beyond Transparency : Open Data and the Future of Civic Innovation*, 125–135.

Mendes-Moreira, J., Moreira-Matias, L., Gama, J. & de Sousa, J. F. (2015). Validating the Coverage of Bus Schedules : A Machine Learning Approach. *Information Sciences*, 293, 299–313.

MobilityData. [Consulté le 2024-01-15]. (2023a). GTFS Realtime Feed Entities. Repéré à https://gtfs.org/realtime/feed-entities.

MobilityData. [Consulté le 2024-01-15]. (2023b). GTFS Static Reference. Repéré à https://gtfs.org/schedule/reference.

Nagatani, T. (2001). Bunching Transition in a Time-headway Model of a Bus Route. *Physical Review E*, 63(3), 036115.

Nelson, T. et al. (2024). GTFS Flex Pilot Programs : Lessons from Early Adopters. *Transportation Research Record*, 2678(3), 123–138.

Nextbus. (Last Accessed : 2020). NexBus Public Feed. Repéré à https://www.nextbus.com/xmlFeedDocs/NextBusXMLFeed.pdf.

Nie, Y., Han, X., Wang, D. & autre. (2023). A Time Series is Worth 64 Words : Long-term Forecasting with Transformers. *arXiv preprint arXiv :2211.14730*.

Niu, H., Zhou, X. & Gao, R. (2025). Interpretable Machine Learning for Transportation Systems : A Review. *Transportation Research Part C*, 153, 104234.

O'Flaherty, C. A. (1997). *Transport Planning and Traffic Engineering*. Elsevier.

Ounoughi, C. & Yahia, S. B. (2024). Sequence to sequence hybrid Bi-LSTM model for traffic speed prediction. *Expert Syst*, 236(2).

Pałys, Ł., Ganzha, M. & Paprzycki, M. (2022). Machine Learning for Bus Travel Prediction. *Computational Science – ICCS 2022*, 13351(Lecture Notes in Computer Science), 703–710. doi : 10.1007/978-3-031-08754-7_72.

Pasini, K. (2021). *Forecast and anomaly detection on time series with dynamic context : Application to the mining of transit ridership data*. (Thèse de doctorat, Université gustave eiffel).

Patel, A. & Rodriguez, D. (2021). Spatial Visualization of Transit Data Using KeplerGL and H3. *Journal of Transport Geography*, 88, 56–67.

Patnaik, J., Chien, S. Y. P. & Bladikas, A. K. (2004). Estimation of Bus Arrival Times Using APC Data.

Pel, A. J., Bliemer, M. C. J. & Hoogendoorn, S. P. (2012). A Review on Travel Behaviour Modelling in Dynamic Traffic Simulation Models for Evacuations. *Transportation*, 39(1), 97–123.

Pelletier, M.-P., Trépanier, M. & Morency, C. (2011). Smart Card Data Use in Public Transit : A Literature Review. *Transportation Research Part C : Emerging Technologies*, 19(4), 557–568.

Pereira, R. H., Saraiva, M., Herszenhut, D., Braga, C. K. V. & Conway, M. W. (2021). r5r : rapid realistic routing on multimodal transport networks with r 5 in r. *Findings*.

Petersen, N. C., Rodrigues, F. & Pereira, F. C. (2019a). Multi-output Bus Travel Time Prediction with Convolutional LSTM Neural Network. *Expert Systems with Applications*, 120, 426–435.

Petersen, N. C., Rodrigues, F. & Pereira, F. C. (2019b). Multi-output bus travel time prediction with convolutional LSTM neural network. *Expert Systems with Applications*, 120, 426–435. doi : 10.1016/j.eswa.2018.11.028.

Prommaharaj, P., Phithakkitnukoon, S., Demissie, M. G., Kattan, L. & Ratti, C. (2020). Visualizing public transit system operation with GTFS data : A case study of Calgary, Canada. *Heliyon*, 6(4).

Rodriguez-Deniz, H. & Villani, M. (2022). Robust Real-Time Delay Predictions in a Network of High-Frequency Urban Buses. *IEEE Transactions on Intelligent Transportation Systems*, 23(9), 16304-16317. doi : 10.1109/TITS.2022.3149656.

Rong, Y., Xu, Z., Liu, J., Liu, H., Ding, J., Liu, X., Luo, W., Zhang, C. & Gao, J. (2022). Du-Bus : A Realtime Bus Waiting Time Estimation System Based On Multi-Source Data. *IEEE Transactions on Intelligent Transportation Systems*, 23(12), 24524-24539. doi : 10.1109/TITS.2022.3210170.

Rong, Y., Yao, J., Liu, J., Fang, Y., Luo, W., Liu, H., Ma, J., Dan, Z., Lin, J., Wu, Z., Zhang, Y. & Zhang, C. (2023). GBTTE : Graph Attention Network Based Bus Travel Time Estimation. *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, (CIKM '23), 4794–4800. doi : 10.1145/3583780.3614730.

Roth, R. E. (2013). Interactive Maps : What We Know and What We Need to Know. *Journal of Spatial Information Science*, 6, 59–115.

Rumelhart, D. E., Hinton, G. E. & Williams, R. J. (1986). Learning Representations by Back-propagating Errors. *Nature*, 323(6088), 533–536.

Sabir, M. (2010). Weather and Travel Behaviour. *VU University Amsterdam*.

Sadegh, M., Karimi, A. & Hosseini, R. (2024). Systematic Feature Engineering for Transportation Prediction : PCA, Laplacian Score, and Chi-Square Methods. *Engineering Applications of Artificial Intelligence*, 130, 107890.

Sadegh-Zadeh, S. et al. (2024). Comparative analysis of dimensionality reduction techniques : PCA, Laplacian score, and chi-square on EEG classification. *Scientific Reports*, 14, 21456. doi : 10.1038/s41598-024-71234-9.

Sahr, K., White, D. & Kimerling, A. J. (2003). Geodesic Discrete Global Grid Systems. *Cartography and Geographic Information Science*, 30(2), 121–134.

Sarhani, M., El Afia, A. & Faizi, R. (2024). Machine Learning Approaches for Transit Delay Prediction : A Comprehensive Survey. *Applied Intelligence*, 54, 4567–4592.

Schnöbauer, D. & Weibel, R. (2018). Detection of Movement Patterns in Spatio-temporal Data. *International Journal of Geographical Information Science*, 32(5), 1063–1087.

Schuster, M. & Paliwal, K. K. (1997). Bidirectional Recurrent Neural Networks. *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 253–256.

Shaheen, S. & Cohen, A. (2018). Is it time for a public transit renaissance ? *Journal of Public Transportation*, 21(1), 67–81.

Shaji, H. E., Tangirala, A. K. & Vanajakshi, L. (2018). Evaluation of Clustering Algorithms for the Prediction of Trends in Bus Travel Time. *Transportation Research Record : Journal of the Transportation Research Board*, 2672(45), 242–252. doi : 10.1177/0361198118791365.

Shaji, H. E., Tangirala, A. K. & Vanajakshi, L. (2022). Joint clustering and prediction approach for travel time prediction. *PLOS ONE*, 17(9), e0275030. doi : 10.1371/journal.pone.0275030.

Shalaby, A. & Farhan, A. (2004). Prediction Model of Bus Arrival and Departure Times Using AVL and APC Data. *Journal of Public Transportation*, 7(1), 41–61. doi : 10.5038/2375-0901.7.1.3.

Shan, S., Zhou, D. et al. (2019). Kepler.gl : Large-Scale Geospatial Data Visualization. *Proceedings of the Workshop on Data Systems for Interactive Analysis*.

Sharma, S., Mawane, N., Kuraganti, C. K., M, D. G., Taware, M., Dixit, Y. C., Mishra, S., Krishnapuram, R. & Ramesh, R. (2024). Enhanced ETA Predictions with T-GCN on Optimized Road Segments. *2024 IEEE International Smart Cities Conference (ISC2)*, pp. 1-6. doi : 10.1109/ISC260477.2024.11004294.

Shi, X., Chen, Z., Wang, H., Yeung, D.-Y., Wong, W.-k. & Woo, W.-c. (2015). Convolutional LSTM Network : A Machine Learning Approach for Precipitation Nowcasting. *Advances in Neural Information Processing Systems*, 28, 802–810.

Silva, R., Ahmed, M. & Patterson, Z. (2024). Probabilistic Graph Neural Networks for Bus Passenger Flow Prediction. *Proceedings of Concordia Transportation Research Conference*, pp. 45–56.

Singh, N. & Kumar, K. (2022a). A review of bus arrival time prediction using artificial intelligence. *Wiley Interdisciplinary Reviews : Data Mining and Knowledge Discovery*, 12(1), e1457. doi : 10.1002/widm.1457.

Singh, N. & Kumar, K. (2022b). A review of bus arrival time prediction using artificial intelligence. *Wiley Interdisciplinary Reviews : Data Mining and Knowledge Discovery*, 12(4), e1457. doi : 10.1002/widm.1457.

Sinn, M., Yoon, J. W., Calabrese, F. & Bouillet, E. (2012a). Predicting arrival times of buses using real-time GPS measurements. *2012 15th International IEEE Conference on Intelligent Transportation Systems*, pp. 1227-1232. doi : 10.1109/ITSC.2012.6338767.

Sinn, M., Yoon, J. W., Calabrese, F. & Bouillet, E. (2012b). Predicting Arrival Times of Buses Using Real-time GPS Measurements. *Proceedings of the 15th International IEEE Conference on Intelligent Transportation Systems*, 1227–1232.

Song, X., Kanasugi, H. & Shibasaki, R. (2016). Deeptransport : Prediction and Simulation of Human Mobility and Transportation Mode at a Citywide Level. *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, (IJCAI'16), 2618–2624. Repéré à http://dl.acm.org/citation.cfm?id=3060832.3060987.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. (2014). Dropout : A Simple Way to Prevent Neural Networks from Overfitting. *J. Mach. Learn. Res.*, 15(1), 1929–1958. Repéré à http://dl.acm.org/citation.cfm?id=2627435.2670313.

Star, T. T. [https ://www.thestar.com/news/gta/2017/12/01/late-for-work-the-ttc-can-give-you-a-note-for-that.html]. (Last accessed : 2020). TTC Gives Notes For Affected Customers Arriving Late For Work.

Sterman, B. P. & Schofer, J. L. (2016). Factors Affecting Reliability of Urban Bus Services. *Journal of Transportation Engineering*, 142(5).

Strathman, J. G. & Hopper, J. R. (2000). Empirical Analysis of Bus Transit On-time Performance. *Transportation Research Part A : Policy and Practice*, 27(2), 93–100.

Subramaniyan, A. B., Wang, C., Shao, Y., Li, W., Wang, H., Zhang, G. & Ma, T. (2023). Hybrid Recurrent Neural Network Modeling for Traffic Delay Prediction at Signalized Intersections Along an Urban Arterial. *IEEE Transactions on Intelligent Transportation Systems*, 24(1), 1384-1394. doi : 10.1109/TITS.2022.3201880.

Sun, Y., Spall, J., Wong, W. & Zhao, X. (2025). Real-time Bus Travel Time Prediction and Reliability Quantification : A Hybrid Markov Model. Repéré à https://arxiv.org/abs/2503.05907.

Supercomputing Conference Investigators. (2024). Neural network optimization and performance analysis for real-time embedded inference. *Proceedings of SC24 : International Conference for High-Performance Computing, Networking, Storage and Analysis*. doi : 10.1145/3688046.

Sutskever, I., Vinyals, O. & Le, Q. V. (2014a). Sequence to Sequence Learning with Neural Networks. Dans Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D. & Weinberger, K. Q. (Éds.), *Advances in Neural Information Processing Systems 27* (pp. 3104–3112). Curran Associates, Inc. Repéré à http://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.pdf.

Sutskever, I., Vinyals, O. & Le, Q. V. (2014b). Sequence to Sequence Learning with Neural Networks. *Advances in Neural Information Processing Systems (NeurIPS)*.

Suwardo, W., Napiah, M. & Kamaruddin, I. (2010). ARIMA models for bus travel time prediction. *Journal of The Institution of Engineers, Malaysia*, 71(2), 49–58.

Swamidass, P. M. (Éd.). (2000). MAPE (mean absolute percentage error)MEAN ABSOLUTE PERCENTAGE ERROR (MAPE). *Encyclopedia of Production and Manufacturing Management* (pp. 462–462). Boston, MA : Springer US. doi : 10.1007/1-4020-0612-8_580.

Tan, H. & Liu, Y. (2022). Spatial Analysis of Urban Transit Systems with H3 : Identifying Delay Hotspots. *Transportation Research Part C : Emerging Technologies*, 128, 89–101.

Tan, P.-N., Steinbach, M. & Kumar, V. (2005). *Introduction to Data Mining, (First Edition)*. Boston, MA, USA : Addison-Wesley Longman Publishing Co., Inc.

Tang, W., Chen, Q. & Liu, M. (2025). Attention-Based CNN-LSTM Hybrid Model for Traffic Flow Prediction. *Proceedings of the 2025 IEEE ITSC Conference*, pp. 234–241.

Technologies, U. [Consulté le 2024-01-15]. (2018). H3 : A Hexagonal Hierarchical Geospatial Indexing System. Repéré à https://uber.github.io/h3.

Technologies, U. [Consulté le 2024-01-15]. (2023a). H3 Documentation. Repéré à https://h3geo.org/docs.

Technologies, U. [Consulté le 2024-01-15]. (2023b). Kepler.gl : A Powerful Open Source Geospatial Analysis Tool for Large-scale Data Sets. Repéré à https://kepler.gl.

Tétreault, P. R. & El-Geneidy, A. M. (2010). Estimating bus run times for new limited-stop service using archived AVL and APC data. *Transportation Research Part A : Policy and Practice*, 44(6), 390–402. doi : 10.1016/j.tra.2010.03.009.

Tipping, M. E. & Bishop, C. M. (1999). Probabilistic Principal Component Analysis. *Journal of the Royal Statistical Society : Series B*, 61(3), 611–622. doi : 10.1111/1467-9868.00196.

Tirachini, A. (2013). Bus Dwell Time : The Effect of Different Fare Collection Systems, Bus Floor Level and Age Profile of Passengers. *Transportmetrica A : Transport Science*, 9(1), 28–49.

Trebaticky, P. (2005). Recurrent Neural Network Training with the Extended Kalman Filter. *[Recurrent Neural Network Training with the Extended Kalman Filter*, 1, 57–64.

Trompet, M., Liu, X. & Graham, D. J. (2011). Development of Key Performance Indicators to Compare Regularity of Service between Urban Bus Operators. *Transportation Research Record*, 2216(1), 33–41.

Uber Technologies. [https://eng.uber.com/h3/]. (2018). H3 : A hexagonal hierarchical spatial index. Repéré à https://eng.uber.com/h3-a-hexagonal-hierarchical-spatial-index/.

Uber Technologies Inc. [Présentation du système d'indexation géospatiale hiérarchique H3]. (2018). H3 : Uber's Hexagonal Hierarchical Geospatial Indexing System. Repéré à Notetechniqueinterne,UberTechnologiesInc.

United Nations, Department of Economic and Social Affairs, Population Division. [Rapport des Nations Unies sur les perspectives de l'urbanisation mondiale]. (2023). World Urbanization Prospects : The 2023 Revision. Repéré à UnitedNations,NewYork.

van Lieshout, R. N., Bouman, P. C. & Huisman, D. (2020). Determining and Evaluating Alternative Line Plans in Out-of-Control Situations. *Transportation Science*, 54(3), 740–761.

van Lint, J. W. C., Hoogendoorn, S. P. & van Zuylen, H. J. (2005). Accurate Freeway Travel Time Prediction with State-Space Neural Networks Under Missing Data. *Transportation Research Part C : Emerging Technologies*, 13(5-6), 347–369.

van Oort, N. & Cats, O. (2016). Improving Public Transport Decision Making, Planning and Operations by Using Big Data : Cases from Sweden and the Netherlands. *IEEE Conference on Models and Technologies for Intelligent Transportation Systems*, 399–404.

van Oort, N. & van Nes, R. (2011). Regularity Analysis for Optimizing Urban Transit Network Design. *Public Transport*, 3(2), 85–103.

Vapnik, V. N. (1999). An Overview of Statistical Learning Theory. *Trans. Neur. Netw.*, 10(5), 988–999. doi : 10.1109/72.788640.

Vapnik, V. N. (1995). *The Nature of Statistical Learning Theory*. Springer.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł. & Polosukhin, I. (2017). Attention is All You Need. *Advances in Neural Information Processing Systems*, 30.

Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Li
'o, P. & Bengio, Y. (2018). Graph Attention Networks. *ICLR*. Repéré à https://arxiv.org/abs/1710.10903.

Veness, C. (Last Accessed : 2018). Movable Type Scripts : Calculate distance, bearing and more between Latitude/Longitude points. Repéré à URL:https://www.movable-type.co.uk/scripts/latlong.html.

Vijaya Kumar, B., Reddy, S. & Narayanan, K. (2024). Explainable AI for Transit Operations : Methods and Applications. *AI Society*, 39, 1234–1250.

Vlahogianni, E. I., Karlaftis, M. G. & Golias, J. C. (2014). Short-term traffic forecasting : Overview of objectives and methods. *Transport Reviews*, 34(1), 4–24. doi : 10.1080/01441647.2013.951992.

Wang, B., Huang, J. & Xu, J. (2019a). Capacity optimization and allocation of an urban rail transit network based on multi-source data. *Journal of Ambient Intelligence and Humanized Computing*, 10(1), 373–383.

Wang, C., Zhao, F., Luo, H., Fang, Y., Zhang, H. & Xiong, H. (2025). Towards Effective Transportation Mode-Aware Trajectory Recovery : Heterogeneity, Personalization and Efficiency. *IEEE Transactions on Mobile Computing*, 24(4), 2832-2846. doi : 10.1109/TMC.2024.3501280.

Wang, D., Zhang, J., Cao, W., Li, J. & Zheng, Y. (2018). When Will You Arrive ? Estimating Travel Time Based on Deep Neural Networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1), 2500–2507. doi : 10.1609/aaai.v32i1.11877.

Wang, H., Liu, Y. & Chen, X. (2024a). XGBoost-Based Traffic Flow Prediction with Multi-Source Data Fusion. *IEEE Access*, 12, 34567–34580.

Wang, J., Chen, X. & Guo, S. (2009, Oct). Bus travel time prediction model with v - support vector regression. *2009 12th International IEEE Conference on Intelligent Transportation Systems*, pp. 1–6.

Wang, S., Li, J. & Zhang, H. (2024b). Spatial Clustering Analysis of Transit Delay Hotspots Using Machine Learning. *Computers, Environment and Urban Systems*, 108, 102056.

Wang, X. & Zhang, H. (2020). Dynamic Visualization for Real-Time Transit Analysis Using KeplerGL. *Transportation Research Part B : Methodological*, 134, 234–245.

Wang, X., Yao, L., Liu, W., Li, C., Bai, L. & Waller, S. T. (2020). Mobility irregularity detection with smart transit card data. *Advances in Knowledge Discovery and Data Mining : 24th Pacific-Asia Conference, PAKDD 2020, Singapore, May 11–14, 2020, Proceedings, Part I 24*, pp. 541–552.

Wang, Y., Zheng, Y. & Xue, Y. (2019b). Travel Time Estimation of a Path Using Sparse Trajectories. *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 25–34.

Wang, Z., Liu, H. & Zhang, Y. (2024c). Graph Neural Networks for Traffic Forecasting : A Survey. *ACM Computing Surveys*, 56(9), 1–38.

Warnakulasuriya, A., Weerasinghe, C., Wickramarathna, H., Ratneswaran, S. & Thayasivam, U. (2024). Explainable Bus Arrival Time Prediction Model with Improved Features Related to Topography and Points of Interest. *2024 IEEE 27th International Conference on Intelligent Transportation Systems (ITSC)*, pp. 2131-2136. doi : 10.1109/ITSC58415.2024.10920146.

Webb, A. et al. (2025). Evaluating Real-Time and Scheduled Public Transport Data. *ISPRS International Journal of Geo-Information*, 14(7), 243.

Wei, L., Yu, Z. & Jin, Z. (2024). Applications of Graph Neural Networks in Transportation : A Comprehensive Review. *IEEE Transactions on Knowledge and Data Engineering*, 36(8), 3456–3478.

Welch, G. & Bishop, G. (2006). An Introduction to the Kalman Filter. *UNC Chapel Hill, Department of Computer Science*. Repéré à https://www.cs.unc.edu/~welch/kalman/. Technical Report TR 95-041.

Welding, P. I. (1957). The Instability of a Close-interval Service. *Journal of the Operational Research Society*, 8(3), 133–142.

Wessel, N. & Widener, M. G. S. (2017a). Discovering the space–time dimensions of schedule padding and delay from GTFS and real-time transit data. *Journal of Geographical Systems*, 19(1), 93–107. doi : 10.1007/s10109-016-0244-8.

Wessel, N. & Widener, M. J. (2017b). Discovering the Space–Time Dimensions of Schedule Padding and Delay from GTFS and Real-Time Transit Data. *Journal of Geographical Systems*, 19(1), 93–107. doi : 10.1007/s10109-016-0244-8.

Wessel, N., Allen, J. & Farber, S. (2017a). Constructing a Routable Retrospective Transit Timetable from a Real-time Vehicle Location Feed and GTFS. *Journal of Transport Geography*, 62, 92–97.

Wessel, N., Allen, J. & Farber, S. (2017b). Constructing a routable retrospective transit timetable from a real-time vehicle location feed and GTFS. *Journal of Transport Geography*, 62, 92–97. doi : 10.1016/j.jtrangeo.2017.04.012.

Wessel, R. & Widener, M. (2018). Schedule Padding in Transit Systems : The Case of Toronto's Buses. *Transportation Research Part A : Policy and Practice*, 113, 57–71.

Williams, B. M. & Hoel, L. A. (2003). Modeling and Forecasting Vehicular Traffic Flow as a Seasonal ARIMA Process : Theoretical Basis and Empirical Results. *Journal of Transportation Engineering*, 129(6), 664–672.

Williams, R. J. (1992, Jun). Training recurrent networks using the extended Kalmann filter. *[Proceedings 1992] IJCNN International Joint Conference on Neural Networks*, 4, 241–246 vol.4. doi : 10.1109/IJCNN.1992.227335.

Wilson, N. H. M. et al. (2008). Impact of Real-time Transit Information on Transit Performance. *Transportation Research Board*.

Wu, C.-H., Ho, J.-M. & Lee, D. T. (2004a). Travel-Time Prediction with Support Vector Regression. *IEEE Transactions on Intelligent Transportation Systems*, 5(4), 276–281.

Wu, C.-H., Ho, J.-M. & Lee, D. T. (2004b). Travel-time prediction with support vector regression. *IEEE Transactions on Intelligent Transportation Systems*, 5(4), 276–281. doi : 10.1109/TITS.2004.837813.

Wu, H. & et al., J. X. (2021). Autoformer : Decomposition Transformers with Auto-Correlation for Long-Term Series Forecasting. *Advances in Neural Information Processing Systems (NeurIPS)*.

Wu, H., Xu, J., Wang, J. & Shang, J. (2021). Autoformer : Decomposition Transformers with Auto-Correlation for Long-Term Series Forecasting. *arXiv preprint arXiv :2106.13008*. Repéré à https://arxiv.org/abs/2106.13008.

Wu, Z., Pan, S., Long, G., Jiang, J. & Zhang, C. (2020). Graph WaveNet for Deep Spatial-Temporal Graph Modeling. *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*.

Xu, J., Wu, Y., Jia, L. & Qin, Y. (2020). A reckoning algorithm for the prediction of arriving passengers for subway station networks. *Journal of Ambient Intelligence and Humanized Computing*, 11(2), 845–864.

Xuan, Y., Argote, J. & Daganzo, C. F. (2011). Dynamic Bus Holding Strategies for Schedule Reliability : Optimal Linear Control and Performance Analysis. *Transportation Research Part B : Methodological*, 45(10), 1831–1845.

Xue, F. & Collaborators. (2025). Hybrid Spatio-Temporal Attention Networks for Large-Scale Traffic Prediction. *arXiv preprint arXiv :2504.01234*. Repéré à https://arxiv.org/abs/2504.01234. À compléter avec les informations réelles si disponible.

Xue, J., Tan, R., Ma, J. & Ukkusuri, S. V. (2025a). Data Mining in Transportation Networks with Graph Neural Networks : A Review and Outlook. *arXiv preprint arXiv :2501.16656*. Repéré à https://arxiv.org/abs/2501.16656. 41 pages, 6 figures.

Xue, J., Tan, R., Ma, J. & Ukkusuri, S. V. (2025b). Data Mining in Transportation Networks with Graph Neural Networks : A Review and Outlook. *CoRR*, abs/2501.16656. doi : 10.48550/arXiv.2501.16656.

Yang, M., Chen, C., Wang, L., Yan, X. & Zhou, L. (2016). Bus Arrival Time Prediction using Support Vector Machine with Genetic Algorithm. *Neural Network World*, 26, 205–217. doi : 10.14311/NNW.2016.26.011.

Yao, D., Zhang, C., Zhu, Z., Huang, J. & Bi, J. (2018a). Trajectory Clustering via Deep Representation Learning. *Neural Networks*, 97, 177–186.

Yao, H., Wu, F., Ke, J., Tang, X., Jia, Y., Lu, S., Gong, P., Ye, J. & Li, Z. (2018b). Deep Multi-view Spatial-Temporal Network for Taxi Demand Prediction. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1).

Yao, H., Wu, F., Ke, J., Tang, X., Jia, Y., Lu, S., Gong, P., Ye, J. & Li, Z. (2018c). Deep Multi-View Spatial-Temporal Network for Taxi Demand Prediction. *AAAI*.

Yu, B., Yang, Z. & Yao, B. (2011). An Improved Ant Colony Optimization for Vehicle Routing Problem. *European Journal of Operational Research*, 196(1), 171–176.

Yu, B., Yin, H. & Zhu, Z. (2018a). Spatio-Temporal Graph Convolutional Networks : A Deep Learning Framework for Traffic Forecasting. *Proceedings of IJCAI*. Repéré à https://arxiv.org/abs/1709.04875.

Yu, B., Yin, H. & Zhu, Z. (2018b). Spatio-temporal Graph Convolutional Networks : A Deep Learning Framework for Traffic Forecasting. *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, pp. 3634–3640.

Zaharia, M., Chen, A. & Davidson, A. (2024). MLOps for Production-Ready Machine Learning Systems. *ACM Queue*, 22(2), 34–56.

Zaki, M., Ashour, I., Zorkany, M. & Hesham, B. (2014). Online Bus Arrival Time Prediction Using Hybrid Neural Network and Kalman filter Techniques.

Zhang, C., James, J. Q. & Liu, Y. (2020). Spatial-Temporal Graph Attention Networks : A Deep Learning Approach for Traffic Forecasting. *IEEE Access*, 7, 166246–166256.

Zhang, J., Zheng, Y. & Qi, D. (2017). Deep Spatio-Temporal Residual Networks for Citywide Crowd Flows Prediction. *Proceedings of the AAAI Conference on Artificial Intelligence*, 31(1).

Zhang, J. & Haghani, A. (2015). A Gradient Boosting Method to Improve Travel Time Prediction. *Transportation Research Part C : Emerging Technologies*, 58, 308–324.

Zhang, L., Wang, Q. & Liu, Y. (2024). Evaluating Lightweight Deep Learning Architectures for Real-Time Traffic and Transit Prediction. *Transportation Research Part C : Emerging Technologies*, 153, 104237. Analyse des compromis entre précision, latence d'inférence et complexité de modèles LSTM compacts et Transformers pour des applications temps réel.

Zhang, P. & Qi, M. (2005). Neural network forecasting for seasonal and trend time series. *European Journal of Operational Research*, 160, 501–514. doi : 10.1016/j.ejor.2003.08.037.

Zhang, Y., Wang, S. & Li, Z. (2019). Urban traffic flow prediction with advanced PCA-based spatio-temporal analysis. *Transportation Research Part C*, 105, 29–43. doi : 10.1016/j.trc.2019.05.021.

Zhao, J., Gao, Y., Tang, J., Zhu, L. & Sun, L. (2019a). Highway Travel Time Prediction Using Sparse Tensor Completion Tactics and K-Nearest Neighbor Pattern Matching Method. *Journal of Advanced Transportation*.

Zhao, J., Wang, L. & Li, C. (2019b). Transfer Learning for Deep Traffic Prediction : Adapting Pre-Trained Models to New Cities. *IEEE Transactions on Intelligent Transportation Systems*, 20(12), 4473–4485. Étude sur l'adaptation de modèles profonds pré-entraînés pour la prédiction du trafic dans de nouvelles localités.

Zhao, L., Wu, J. & Chen, M. (2024). Comparative Analysis of LSTM, Random Forest and XGBoost for Road Traffic Prediction. *Proceedings of the 14th International Conference on Vehicle Technology and Intelligent Transport Systems*, pp. 156–165.

Zhao, L., Song, Y., Zhang, C., Liu, Y., Wang, P., Lin, T., Deng, M. & Li, H. (2020). T-GCN : A Temporal Graph Convolutional Network for Traffic Prediction. *IEEE Transactions on Intelligent Transportation Systems*, 21(9), 3848–3858.

Zhao, X., Li, Y. & Wang, Z. (2019c). Systematic and stochastic delays in public transit systems : A case study. *Transportation Research Record*, 2673(12), 85–93. doi : 10.1177/0361198119853573.

Zhao, Z., Chen, W., Wu, X., Chen, P. C. Y. & Liu, J. (2017). LSTM network : a deep learning approach for short-term traffic forecast. *IET Intelligent Transport Systems*, 11(2), 68–75. doi : 10.1049/iet-its.2016.0208.

Zheng, H., Yuan, J. & Chen, L. (2017). Short-Term Load Forecasting Using EMD-LSTM Neural Networks with a Xgboost Algorithm for Feature Importance Evaluation. *Energies*, 10(8). doi : 10.3390/en10081168.

Zhou, H., Zhang, S., Peng, J. & autres. (2021a). Informer : Beyond Efficient Transformer for Long Sequence Time-Series Forecasting. *Proceedings of AAAI*.

Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H. & Sun, W. (2021b). Informer : Beyond Efficient Transformer for Long Sequence Time-Series Forecasting. *Proceedings of AAAI Conference on Artificial Intelligence*. Repéré à https://arxiv.org/abs/2012.07436.

Zhou, J., Wang, L. & Chen, Y. (2024). Scalability of H3 Indexing for Real-Time Transit Data Processing. *Proceedings of the 2024 ACM SIGSPATIAL Conference*, pp. 89–98.