BI-DIRECTIONAL LSTM AND KALMAN FILTER FOR PASSENGER FLOW PREDICTION ON BUS TRANSPORTATION SYSTEMS

Hannah Wood

A thesis
in
The Department
of Concordia Institute for Information Systems Engineering

Presented in Partial Fulfillment of the Requirements for
the Degree of Master of Applied Science
(Information Systems Security)
at Concordia University
Montréal, Québec, Canada

April 2022

Concordia University

School of Graduate Studies

This is to certify that the thesis prepared:

By: Hannah Wood
Entitled: Bi-directional LSTM and Kalman Filter for Passenger Flow Prediction on Bus Transportation Systems

and submitted in partial fulfillment of the requirements for the degree of

Master of Applied Science
(Information Systems Security)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

Dr. Jamal Bentahar _____ Chair and Examiner

Dr. Amin Hammad _____ Examiner

Dr. Nizar Bouguila _____ Supervisor

Dr. Zachary Patterson _____ Co-supervisor

Approved by _____

Dr. Abdessamad Ben Hamza, Chair
Department of Concordia Institute for Information Systems Engineering

29 April 2022 _____

Dr. Mourad Debbabi, Dean
Faculty of Engineering and Computer Science

Abstract

Bi-directional LSTM and Kalman Filter for Passenger Flow Prediction in Bus Transportation Systems

Hannah Wood

Forecasting travel demand is a complex problem facing public transit operators. Passenger flow prediction is useful not only for operators, used for long-term planning and scheduling, but also for transit users. The time is quickly approaching that short-term passenger flow prediction will be expected as a matter of course by transit users. To address this expectation, a Bi-directional Long Short-Term Memory Neural Network model (BDLSTM NN) and a Bi-directional Long Short-Term Memory Neural Network Kalman Filter model (BDLSTM KF) predict short-term passenger flow and based on the dependencies between passenger count and spatial-temporal features. A comprehensive preprocessing framework is proposed leveraging historical data and extracting bi-directional features of passenger flow. The proposed model is based on [1] but adapted, applied, and analysed to produce optimal results for passenger flow forecasting on a bus route. Building on [2], a BDLSTM architecture is then combined with a Kalman filter. The Kalman filter reduces the training and testing complexity required for passenger flow forecasting. The BDLSTM-based Kalman filter produces predictions with less uncertainty than each method alone. Evaluating the BDLSTM-based Kalman filter with two months of real-world data, one year apart shows positive improvements for short-term forecasting in high complexity bus networks. It is possible to see that the BDLSTM outperforms traditional machine and deep learning techniques used in this context.

Acknowledgements

First and foremost, I'd like to express my gratitude to Concordia University and Mitacs for assisting me in completing my Master's degree. The faculty and staff at the Concordia Institute for Information Systems Engineering are wonderful people, and I would recommend them to anyone interested in pursuing a graduate degree.

None of this would be possible without my supervisors. Throughout my degree, I was fortunate to have two wonderful supervisors. Thank you, Dr. Bouguila, for taking me on as a masters student, your generosity, and experience. Thank you, Dr. Patterson, for bringing your enthusiasm and knowledge to my education.

This thesis was in part inspired by the transportation technology startup BusPas. As a team, we grateful to the *Société de Transport de Laval* (STL) for allowing us to use their bus transportation data, otherwise, this study would not be possible. Greg Vitzakis, my industry mentor, and the rest of the BusPas team gave me excellent advice and assistance. I had a terrific time working with them and look forward to continuing professional relationships. Thank you to all of my classmates and BusPas students. You've all helped to make the long days go faster and the major tasks seem less daunting.

I've met a lot of fantastic folks throughout my time in Montreal, much too numerous to name here. Outside of school, I am surrounded by a fantastic group of friends and shout out to my amazing roommates. Finally, I would like to thank whole family for their love and support; my mom and my grandmother have always cheered for me from the front row.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

A robust public transportation system is key to effectively meeting the demands of any prosperous city. With a desire to urbanize and provide accessibility to its growing population, any city would hope to have an affordable, high capacity, punctual and energy-efficient public transit system [3]. A key factor in the appreciation of public transit by users is overcrowding. User satisfaction, coupled with the health and safety riders, is compromised when overcrowding occurs, having the potential to deter citizens from choosing to take public transportation [1].

Bus services are often criticized for inconsistent arrival times, over-crowding, and ineffective routes as a city's needs change. Based on [3], there is a direct relationship between crowding and arrival time irregularity, and increasing unwanted passenger wait times. Additionally, when buses are scheduled based on headway, irregular arrival times can cause polarized passenger loads for specific lines, further increasing the irregularity of arrival times and bus bunching [3].

With more people having access to the internet through their mobile phones and the ability to transmit real-time vehicle location, real-time bus arrival information is becoming increasingly popular and almost expected from transit agencies [4]. The method by which automatic vehicle location, AVL, information is transmitted can be found in one of two forms: collected at fixed locations (i.e., sensors at each stop) or continuously (i.e., GPS device installed onboard). Providing customers with real-time arrival information

positively affects perceived waiting times, safety and security, impacts of service disruptions, and general rider satisfaction [3]. Additionally, since the emergence of COVID-19 and physical distancing, estimates of overcrowding become useful and necessary to certain passengers.

In this work, a Bi-directional Long-Short Term Memory neural network model (BDLSTM) and a Bi-directional Long-Short Term Memory neural network Kalman filter model (BDLSTM KF) are proposed and applied to short-term passenger flow prediction. Using passenger count data from the bus system of the city of Laval, in Quebec, the proposed method learns, and is reactive to, the unpredictability of the real-time system. The model is trained and calibrated to predict bus ridership, providing results at multiple time-horizons including, one-stop, half a trip and a full trip into the future. Specifically targeting volatile, high passenger flow bus routes, the chosen trip is Route 26 in the westward direction.

1.2 Related Work

Passenger flow prediction in transportation systems has been applied to many modalities with many different methods. It should be noted that passenger flow prediction is a term commonly used in the literature and will be referenced as such for the remainder of the paper [5, 6]. The passenger flow prediction in this paper describes the narrower nomenclature of busload prediction and load forecasting. Based on [1], [7], and [8] predicting passenger load can be categorized into short and long-term forecasting. Additionally, from [9] and [7], load forecasting can be computed using traditional methods or methods leveraging machine learning. Lastly, the intersection between machine learning and the Kalman filter is explored. The differences between short and long-term forecasting, a survey of machine learning methods used for passenger flow, and an introduction to the combination of a Kalman filter and machine learning are described below.

1.2.1 Short and Long Term Forecasting

Forecasting systems and models can be used to describe transportation, electrical, health care and financial systems. Load forecasting has a long history in diverse sectors; however, there are fewer direct applications to bus ridership. The advancement of accurate sensors and the ability to handle large

amounts of data have provided the ability to consider many factors related to busload [10]. The existence of numerous data sources now provides many possibilities for different modeling approaches [11]. In many simple applications of load forecasting, meaningful results are obtained with statistical methods. Applied to transportation load prediction, machine learning and data mining techniques are becoming increasingly popular.

In many different domains, load forecasting is commonly divided into the short and long term. Despite these terms being commonly used, there is no strict definition of the duration for each category [9]. While there are no strict definitions for the meaning of short- and long-term forecasting, we categorize the literature according to this convention in the rest of the review.

Short-Term Forecasting

Short-term forecasting in the transportation domain is a well-developed research area. [8] considers passenger flow forecasting in metro systems with a 30-minute window using a radial basis function neural network. [12] categorizes time prediction into eight intervals based on normal travel patterns for their dataset, revolving around peak times. They utilize a framework based on an attention sequence-to-sequence model for windows of 0-15 minutes, 15-30 minutes, 30-45 minutes, and 45-60 minutes in look-ahead time windows. [13] used peak times, only choosing 3 consecutive days for training and testing sets with a 2:1 ratio for training and testing. They chose 10-minute prediction timeslots to evaluate their busload prediction algorithm using a state-vector machine. In another study, [7] used a short-term prediction cycle of 15 minutes in advance using a look-back window of six prediction cycles, or 90 minutes. They were provided with 96 samples per day with which they use a multi-task learning temporal convolutional network to predict passenger demand. [14] produced an interactive multiple model filter algorithm to train four months worth of data in 15-minute intervals to predict bus passenger demand. In their study, they had daily and weekly models as well as a model for 15-minute time horizon. [15] defined their time interval to be 15-minutes using data from 2017 at a transfer rail station for rail. The authors defined a one-step and two-step look-ahead prediction mechanism using an SVR-LSTM model. [16] produces a short-term prediction for urban rail passenger volume using a time resolution of 15-minute intervals with a total of 67 timestamps each day. The paper outlines longer intervals of 15, 30, and 60-minute intervals using a spatial-temporal LSTM. In summary,

short-term forecasting while not strictly defined, refers to no more than two hours in advanced.

Long-Term Forecasting

The goals of Long-term forecasting are typically different from those of short-term. Long-term forecasting can help with planning and coordination of route networks, schedules, and quality of service [17]. In this vein, [17] states that their long-term predictions, one year in advance, provide trip planners an idea of travel demand at stations and stops. The authors further suggest three models: a basic calendar model based on station or stop, with specific days in the year being highlighted, i.e. school holidays, public holidays, four-day weekends; a calendar "Plus" model, highlighting 56 types of days by means of the Cartesian product of the categorical days for a station or stop; and lastly a Random Forest technique, applied to each station and stop. The Random Forest technique uses random sampling with replacing observations, called bagging, and the random selection of features, called feature bagging [17]. [18] applies long-term forecasting to passenger demand on airplanes. The paper extensively covers the previous traditional model to forecast passenger flight demand using four consecutive phases: trip generation, trip distribution, mode split, and trip assignment. The model is not considered out of date but was used as a basis for newer models such as the direct demand model, which does not need the intensive number of features that the previous model requires. The model used 18 years of training and 4 years as a test set. [19] proposes a principal component analysis and a fuzzy feed-forward neural network to handle the uncertainty of long-term power load forecasting. The paper defines short-term to be up to 1 day and long-term from 1-10 years.

1.2.2 Methods for Predicting Passenger Flow

In addition to the forecasting length, the literature on passenger flow forecasting can be divided broadly by methodological approach, a traditional statistical analysis and computational intelligence methods. Emphasis on the capabilities of machine learning-based methods will be described below. To support this work, the connection between passenger flow prediction in transportation systems and traffic flow predictions is established.

Classical Statistical (Traditional) Methods

The first methods for passenger flow transit prediction were based on traditional classical analysis [5]. These methods had difficulty handling the high dimensionality and non-linear nature of passenger flow data [5,6]. Classical methodologies for regression prediction include traditional regression-based models, the Kalman Filter model, Autoregressive Integrated Moving Average model (ARIMA) and the Least Absolute Shrinkage and Selection Operator Regression (LASSO) [2,7,20–26]. In [22], the researchers varied the composition of the Kalman filter model for rail passenger flow. The authors proposed three variations of the model for short-term prediction using one month's worth of data [22]. Their results were more robust for peak periods than non-peak periods. Time-series autoregressive integrated moving average (ARIMA) models are commonly used for time series data. ARIMA models can be used to gain information about data or forecast future points [7]. In [24], the authors predicted passenger flow for a railway application using a seasonal autoregressive integrated moving average, SARIMA model. They concluded that using four months of data, their SARIMA model was strong enough to predict monthly passenger flow on railway systems. The work by [25] proposed a model to predict passenger rail flow that combined a symbolic regression model with an ARIMA model. Finally, The Least Absolute Shrinkage, and Selection Operator Regression (LASSO), was proposed by [27]. LASSO was a baseline method in [26].

Machine Learning-Based Methods

Machine learning-based models have demonstrated their strengths in learning unpredictable, space and time-dependant features required for passenger load prediction. Applying a non-parametric regression model using previous data can easily be applied to passenger flow prediction [9]. Commonly used models that leverage machine learning are: Support Vector Regression, Multilayer Perceptron, Gated Recurrent Unit, Random Forest Regressor, and notably, the Long Short Term Memory Neural Network models [1,1,6,7,11,23,28,29]. The Multilayer Perceptron (MLP) model is a simple feed-forward neural network that can handle simple data without many complexities. By adding more layers, it is possible to make the model more complex; however, convergence speed is traded-off with more layers [6]. The Gated Recurrent Unit (GRU) is a recurrent model that is often useful in time-series data since it

can learn current data while considering previous data [1]. It is one of the simplest forms of recurrent units. In an implementation by [11], the authors predict subway passenger flow. The results are presented in terms of long-term forecasting for a 1.5 hour time horizon using the GRU network. The Long Short-Term Memory neural network (LSTM), first proposed in 1997, has many applications including short-term passenger flow forecasting. In [17], the authors use smart card data processed by a LSTM to predict origins and destinations in a subway network. The outcomes of the study provide short-term predictions of 15 minutes in the future. Additionally, in [29], researchers used a LSTM model for predicting short-term metro passenger flow. The sequence of passenger flow was altered using empirical mode decomposition and, along with the original data, was inputed to the LSTM [29]. [17] explores the prediction of passenger flow in train stations and bus and tram stops. In this study, they use short-term forecasting for a horizon time between 15-30 minutes ahead. Additionally, the long-term forecasting solution predicts passenger flows up to a year in advance.

Passenger Flow Prediction Methods Originating from Traffic Literature

Based on the work proposed by [23,30], the authors explore the link between approaches for traffic speed prediction and transportation system passenger flow. Within [23], dynamic Origin-Destination (OD) matrices processed by a LSTM are proposed to predict passenger flow on a subway system. The authors extend the study of origin destination for the flow of drivers to passenger flow, where the readings produced by speed sensors are similar to sensors producing trip counts between different stations or stops in the network. Using a stacked autoencoder model to predict traffic flow, [30], describes traffic researchers' neural network configurations applied to the transportation sector improves efficiency of the transit systems. Specifically in the works by [31], automobile traffic flow prediction defined within the subset of the Intelligent Transportation System (ITS). Researchers referencing passenger flow prediction commonly note the evolving space of ITS, describing the similarities in problem composition between traffic and transportation literature [25,32].

1.2.3 Combining Kalman Filter and Deep Learning

Kalman filters have been used extensively as a state estimation technique applied in navigation, object-tracking and time-series forecasting [33]. Considering time-varying systems, the Kalman filter (KF) is an extremely intuitive method for predicting the behaviour of a system given historical data. Using a series of noisy measurements, the Kalman filter outputs are calculated in the domain of second order statistics solving the least-squares problem. Based on its effectiveness and popularity, non-linear extensions of the Kalman filter have been proposed, such as the Extended Kalman Filter (EKF), and the Unscented Kalman Filter (UKF).

The KF model is a low-complexity solution for time-series and state estimation with notable drawbacks; where a Kalman filter is modeled as a linear function influenced by Gaussian noise and it must be initialized with predefined initial states for a transition and measurement model. Therefore, a Kalman filter model may have difficulties capturing the dynamic behaviour of a non-linear, real-world systems. Additionally, the Kalman filter requires model parameters, some of which can be estimated via system modelling, the state transition matrix, A, and the measurement matrix, H. However, some parameters must be defined by the user. These parameters include process noise covariance, Q, the measurement noise covariance, R, the initial mean and covariance which vary for each application.

Many of the above drawbacks of the Kalman filter have been mediated by researchers by combining the KF architecture with deep learning. Deep learning methods make it possible to take large amounts of data to learn the system's behaviours and a Kalman filter presents reduced number training steps, quicker convergence, and a lower complexity model. The only noteworthy Kalman filter and machine learning solution in the transportation sector predicts short-term passenger flow on the Line 3 of the Light Rail Transit system in Changchun, China. The short-term passenger flow prediction method uses a Kalman filter and K-Nearest Neighbor approach. The time interval is set to 15 minutes and its motivation is to handle the variation in data at peak times. The authors' results have stronger prediction accuracy at time-series data volatility [34].

Based on an in-depth literature review, there is little work combining a Kalman filter and deep learning for passenger flow prediction in the transportation sector. Therefore, the described model applications below will vary significantly from wind speed to joint estimation.

In a paper written by [35], the authors use a feed-forward neural network and extended Kalman filter model to predict the state-of-charge of lithium-ion battery. They use RMSE as an error measurement to accurately model battery characteristics at different temperatures and charge levels. Peel uses a multivariate single-step Kalman Filter Ensemble of Neural Networks model to estimate the number of cycles remaining before failure of an unspecified complex system [36]. The regression challenge is met with a multi-layer perceptron model and radial basis function-based networks combined through a Kalman filter [36]. Aly proposes a hybrid model that includes a wavelet neural network, time series Kalman filter and recurrent Kalman filter for wind speed forecasting. The wind power model is used to for time horizons at three-hour intervals in steps of fifteen minutes using three wind power generation systems. The accuracy metrics used were R^2, MAPE, and RMSE [37]. In a paper written by [38], the authors use a two-step prediction process for air quality indicators. The authors first train a LSTM as a static prediction model from time-series data. The Kalman filter is then used as a dynamic adjustment model by adjusting the base time. The authors found that even with the two-step static and dynamic modelling, their LSTM-KF proved better results than a traditional LSTM. The authors used RMSE and R^2 as an evaluation metric.

There are many ways to classify the instances of when a Kalman filter and deep learning are combined. When looking at use cases of the model it can be easily grouped into signal processing applications and non-signal processing applications. Observing the applications for non-signal processing models, the model structures can be categorised by approaches that learn static Kalman filter parameters and those that actively regress the parameters during filtering. The active regression of the parameters during filtering are often proposed as potential online-based applications.

In summary, the primary challenge for bus passenger load prediction is related to the complexity of predicting ridership behaviour. Typical datasets for passenger flow prediction contain countless features with intricate geospatial, and temporal dependencies [2, 17, 23]. In addition to complex feature considerations, ridership behaviour, specifically bus ridership behaviour does not always follow a dependable schedule. This is especially true with surges to the system as a result of special events, the weather, or public transportation breakdowns [22]. Increasingly, LSTMs are being used in many application areas but are particularly well adapted to bus passenger flow prediction due to their ability to handle intricate geospatial and temporal

dependencies, learn unpredictable ridership behaviour and take non-uniform input data [5]. Additionally, successful passenger flow forecasting systems for transportation are heavily dependant on high computational complexity deep learning models. With continuous real-time data, transportation agencies benefit from faster output response for passenger flow prediction and especially with reduced training and re-training time. As a result, this paper proposes using bi-directional LSTM for multi-step passenger flow prediction and a bi-directional LSTM Kalman filter for single-step passenger flow prediction as these methods are particularly suited for capturing spatial and temporal, forward and backward dependencies, with reduced error and in a timely manner.

1.3 Contributions

The primary contribution of this thesis is a solution for time-series forecasting applied to short-term passenger flow prediction. Among the proposed thesis, this work has been submitted to the International Transportation Research Part C: Emerging Technologies Journal. The various components of our contribution are detailed below.

bi-directional Long-Short Term Memory neural network model In this thesis, a bi-directional long-short term memory neural network (BDLSTM) is proposed and applied to short-term passenger flow prediction. The entire machine learning pipeline is introduced from data analysis and validation to model evaluation. A comparison of univariate versus multivariate, and single-step versus multi-step models is demonstrated. Each combination is tested against multiple comparison models to prove the strength of the bi-directional long-short term memory neural network. Using a bus transportation system, the proposed method learns and is reactive to the unpredictability of the real-time system.

Bi-directional Long-short Term Memory Neural Network based Kalman filter model The second section uses an LSTM-based Kalman filter model to perform short-term passenger flow prediction. The structure of the long-short term memory neural network Kalman filter model is based on [2]. Building on the results of the first work, one of the long-short term memory neural network sections which build up the Kalman filter is updated with a bi-directional long-short term memory neural network. The

bi-directional long-short term memory neural network forms the state equation for the Kalman filter. Experiments were carried out and the effectiveness of the real-time bi-directional LSTM-based filtering algorithm is verified.

1.4 Thesis Organization

The thesis is formatted with four chapters. The remainder is formatted as follows. In Chapter 2, different machine learning techniques for passenger flow forecasting are detailed. Furthermore, Chapter 2 introduces an analysis of the data and its relevant features. Within the second chapter, the experimental framework and results, the details of the dataset, the implementation, performance metrics, comparing models, model training and results are explained. After figuring out what techniques work best for single-step passenger flow forecasting is determined, the following Chapter 3 presents the selected techniques used as a foundation to build a Kalman filter. The Kalman filter is built with a combination of long-short term memory neural networks and a bi-directional long-short term memory neural network. Comparing the work proposed by [2] and updating a unit in his model with a bi-directional long-short term memory neural network it is shown that the update to [2]'s model has higher prediction accuracy when applied to short-term passenger flow prediction. Finally, in Chapter 4, the work is concluded, some of the challenges are explained, and recommendations are provided for future works.

Chapter 2

Passenger Flow Prediction in Bus Systems Using a Bi-directional Long-Short Term Memory Neural Network

2.1 Introduction

In this chapter, a comprehensive machine learning pipeline is outlined and executed to optimize passenger flow prediction. Section 2.2 provides a complete background of the long-short term memory neural network (LSTM) and how forward and backward layers can be combined to produced a bi-directional LSTM (BDLSTM). Section 2.3, delves into background information providing an analysis of the data and its features, the model training process and the respective comparative models. This section gives an introduction to the components and experimental setup of the Bi-directional Long-Short Term Memory Neural Network model as proposed by [1]. The performance metrics that are used throughout the paper are described. Lastly, in section 2.4, the results are discussed.

2.2 Methodology

This section is dedicated to describing the architecture of the proposed model used to predict short-term passenger flow. It should be noted that "flow" and "load prediction" are used synonymously and are defined as predicting the passenger load on a bus at a given stop, provided by historical passenger load information.

2.2.1 Long-Short Term Memory Neural Network

A long-short term memory neural network is an artificial recurrent neural network. With time-series data it can benefit greatly from back propagation; it is also able to selectively remember updated information and forget historical information [33]. Through the gated activation function mechanism in the cell's architecture, it overcomes the gradient vanishing problem [39]. Pictured in Figure 2.1, the the LSTM cell can be described as a dynamic gate that decides the influence of the previous state on the current data input while learning and extracting features from the current state [40]. LSTM formulation is described using Equation 2.1 to Equation 2.6.

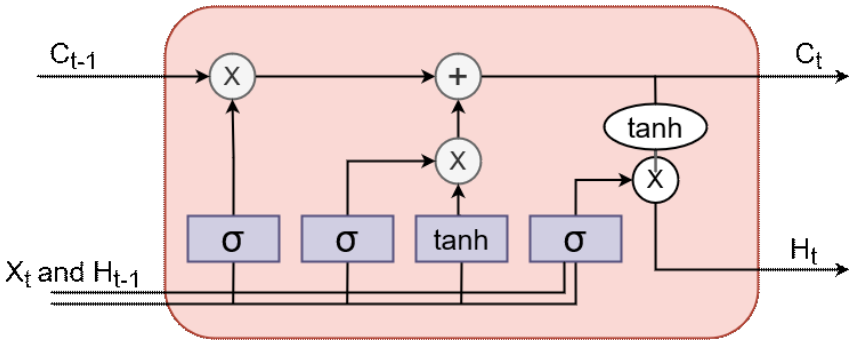

Figure 2.1: LSTM cell architecture

2.2.2 Bi-directional Long-Short Term Memory Neural Network

The Bi-directional LSTM (BDLSTM) uses two hidden layers to handle sequential data in both the forward and backward directions [1]. The bi-directional LSTM is based on the bi-directional recurrent neural network [41].

The first paper to propose such an architecture was written by [41], created a training sequence containing two hidden layers that were connected to the same output layer. Like the LSTM, the BDLSTM uses the same equations 2.1-2.6 for forward and backward layer outputs. The primary difference between the two is the sequence of the data fed into the algorithm, either forward or in reverse order. A LSTM network has an input layer, one or more hidden layers, and an output layer [33]. The hidden layer is specified in a way that the neurons update the state of the memory cells. In the input to the model, the user must define the number of neurons equal to the feature space. The output space decides the number of neurons in the output layer. A schematic of a LSTM cell is displayed in 2.1.

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

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

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

$$\bar{C}_t = \tanh \left(W_c x_t + U_c h_{t-1} + b_c\right) \tag{2.4}$$

$$C_t = f_t \times C_{t-1} + i_t \times \widetilde{C}_t \tag{2.5}$$

$$h_t = o_t \times \tanh \left(C_t\right) \tag{2.6}$$

The cell's state and previous state are represented by C_t and $C_{(t-1)}$. A cell takes as input x_t and $h_{(t-1)}$ which is the input at the current time step and the previous hidden state. These inputs are combined in a vector which is transformed by the element-by-element sigmoid function. In equations 2.1 to 2.6, W and b are the weight matrices and bias of the forget gates that influence the degree to which the current state is affected by the previous state. The flow of information is controlled by the input gate, i. The discarding of the current cell's state is decided by the output gate, o_t. The forget gate, f_t, is useful in resetting the memory [1].

2.3 Experimental Setup

To validate our proposed model, real-world data from the bus system of the city of Laval, Quebec was used as input for short-term passenger flow prediction. In this section, the process for building the model is separated into four sections: data extraction and validation, preprocessing, training, and implementation. The overview of this section can be described by Figure 2.2.

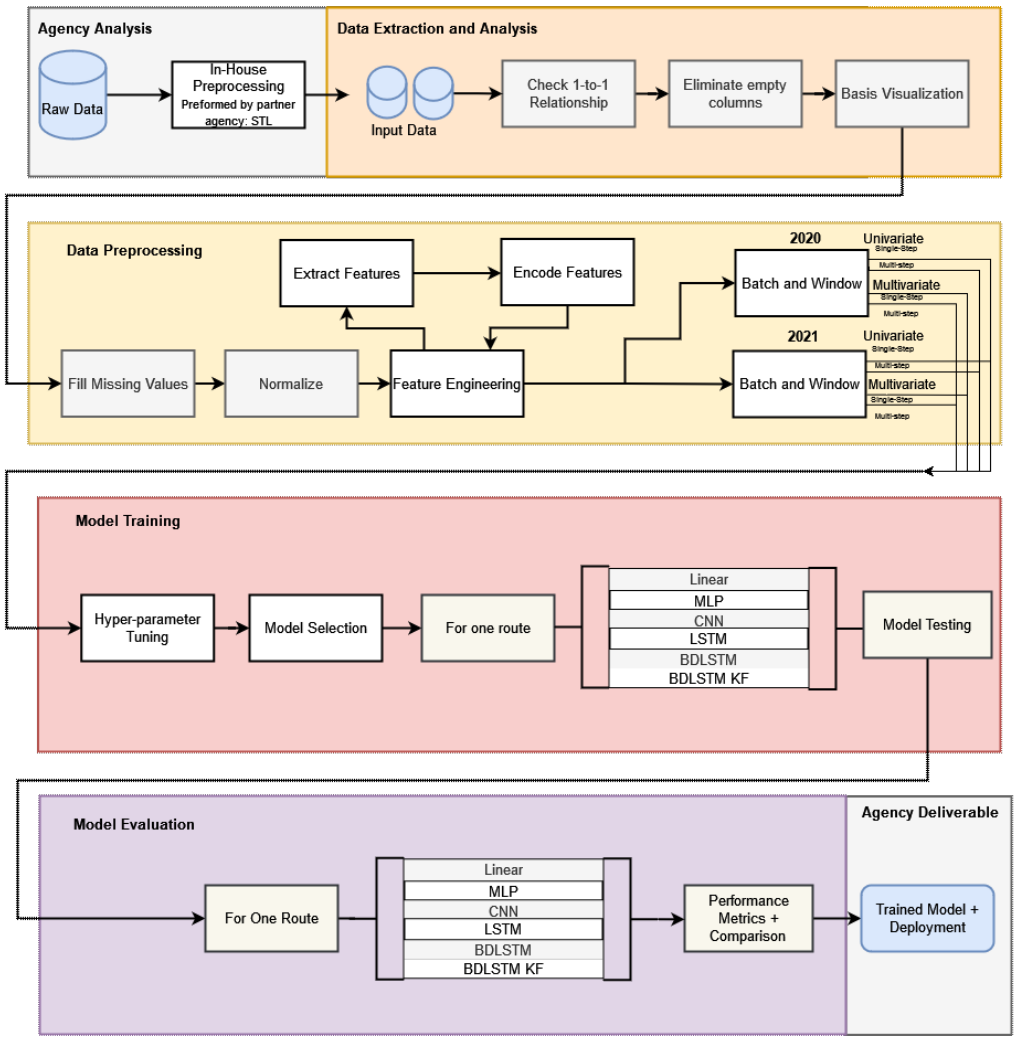

Figure 2.2: Flowchart diagram showing the process to predict passenger flow

2.3.1 Dataset Description and Feature Selection

In this study, a dataset from Laval, Quebec, Canada, was obtained. Laval is the second largest city in Quebec and a suburb of Montreal, itself the second largest city in Canada. Given its size, Laval can be characterized as having a sizable public transportation infrastructure. In one month, the Société de Transport de Laval (STL) typically produces three million observations for ∼42 routes and ∼3,000 unique stops. All available features in the dataset can be shown in Figure 2.3. In one month's worth of data for a given route, STL produces 300k observations. The dataset is comprised of automatic passenger counting (APC) and automatic vehicle location (AVL) data. Before being provided to the authors, adjustments to the data were made by STL, for example, recalculating more accurate passenger counts and readjusting the stop sequence.

Furthermore, each observation contains two features identifying its confidence level. The first feature represents the reliability of the bus passing by the stop. The "reliability stop" field must be "reliable," to use the data for modelling. The second reliability feature indicates the reliability of the passenger flow information. To obtain reliable load information, the two reliability fields must be "reliable". Both confidence features are represented by a binary value where "1" indicates full confidence in the sample and "0" depicts no confidence. Only data with full confidence are chosen.

Naturally, the passenger load on a bus at a particular location and time is influenced by many external features. Features can be extracted from Automatic Passenger Counting (APC), Automatic Vehicle Location (AVL), and General Transit Feed Specification (GTFS) data. Adjusting the work by [26], where the author categorizes their features to be, local basic features, GPS features, and local demand features. The local basic features are described from existing data infrastructure, such as the hour of the day. GPS features typically can be found from automatic vehicle location, AVL or general transit feed specification, GTFS data. Lastly, the local demand features are taken from historical APC data. These features are commonly seen in previous literature for bus passenger flow [7].

Local Basic Features

The local basic features taken from the existing data infrastructure are the hour of the day (HOD) and the day of the week (DOW). To give an example,

the date, 2021-03-30 13:16 will provide the HOD equal to 13; the day of the week, Tuesday, to be equal to 2; and the week of the year is equal to 12. When analyzing the historical data, it is clear there are patterns for hours in a day and differences in days of a week [1, 23, 26]. An obvious factor is the morning and afternoon peaks during the week as people are commuting to work and school [9, 10, 23].

GPS Features

Automatic vehicle location, AVL, provides an observation every 1-8 minutes. It captures an observation if the bus stops and someone embarks or alights from the bus. The STL provides AVL features using the NAD83 Canadian Spatial Reference System (CSRS98), SCoPQ zone 8. The NAD83 geometric reference system uses two axis abbreviations of "X" and "Y" respectively, using the metric system. There are three main AVL features, the coordinate in the x-direction, the coordinate in the y-direction, and a reliability measurement. The AVL reliability feature captures the relationship between the bus, the stop and the timestamp. The stop-level reliability feature supports the validity of the sample. The "reliability stop" field must be "reliable" in order to consider the data. The confidence features are represented by a binary value, where "fiable," meaning reliable, indicates full confidence in the sample, and "pas heure," meaning no hour information, depicts no confidence. Only data with full confidence are chosen.

Local Demand Features

For any observation, it is possible to directly relate the ridership of a bus on a line for a certain day over a span of multiple weeks. Mapping the same peaks over multiple days is also possible. Therefore, taking previous average demands as a feature can be critical for understanding future trends.

2.3.2 Data Extraction and Pre-Analysis

The transportation agency provided one AVL and one APC dataset for each day. The two datasets were linked by a foreign key in a one-to-one relationship. In addition to the processing done by the STL, some additional preprocessing was required before it could be used in model estimation. The steps to analyze the provided data started by visualizing the relationships

between all features, relevant features, and the trends of the busload for a given time-span, trip, route, and stop. Through visualization, it was possible to determine a typical range of busload values in 2020 and 2021 for a given route and stop as per Figure 2.4. Additionally, the trends of the busload for a given day of the week or month were realized. The required data preparation consisted of filling missing values, data normalization, feature extraction, and handling of the categorical nature of the data. The input features from the data are normalized and fed into the model. Figure 2.5 shows the number of passengers for each trip in the condensed network, on 26W, for one month, there were more than 80k riders in both 2020 and 2021. Observations of the passenger loads during the week compared to the weekends can be learned. A visualization of the differences between the days of the week can be seen in Figure 2.6. Figure 2.7 shows the peak passenger flow for a weekday, in 2020 and 2021 (using Thursday).

Filling Missing Values

Specifically, with real-world applications, it was expected to have missing values throughout the dataset. The missing data was visualized using a heat map to determine the correlation between these missing values. For example, there would also be missing busload values when there was a missing value in the busload reliability field. There was also analysis to determine if the missing values related to a certain route, bus, or date range. It was important to have a cohesive and logical method to fill or remove these values. The optimal solution was to fill missing busload values with a weighted average of that day with the previous 3 weeks. The missing values in the arrival and departure times were filled based on the arrival time for the stops surrounding target stop (using distance and average speed of the bus).

Data Encoding and Normalization

Many of the values within the features of the data fall into categories, bus stop, bus route, direction, and reliability. Many of the categorical features are nominal, with no dependence on order. There are a few ordinal, order-dependent, categorical features, such as the sequence of stops. Where features contain multiple categories, encoding the data is extremely important for computational efficiency. Initially, one-hot encoding was applied to the data; however, embedding was implemented due to the above drawbacks.

The data was normalized using a Tensorflow normalization layer between [0, 1] by subtracting the mean and dividing by the standard deviation for each feature.

Feature Extraction

The features necessary to fill the busload are the bus stop, the route and direction, and time. The time vector was split between HOD and DOW.

2.3.3 Model Training

For each day, the network produces, on average, 100k observations. Choosing one month provides more than three million samples. The data processed and evaluated are for monthly data from October 2020 and 2021, individually and together. The dataset is further reduced to analyze the agency's busiest route, 26 in the westward direction. A map produced using the data provided by the STL shows the stops to be predicted, seen in Figure 2.8.

After the dataset is preprocessed, it is split into training, validation, and test sets. The data taken to train the model is 70%, 20%, 10% for the training, validation, and test sets. The consideration of time series data was considered when slicing the data into windows, where the shuffling occurs after the model was trained. The selected data for training was taken from Route 26W over a month, which represents 100k sample data. There are $\sim$62 stops in the chosen trip. Four key features from the discussion were used, including hour of day (HOD), day of week (DOW), unique bus stop identifier, and whether the trip is in-bound or out-bound.

The model is trained and performs predictions based on a window of consecutive samples from the data. There are six different types of data windows that were created for this study. The windowed dataset was created based off a sliding windows over time-series data using Keras Tensorflow's $tf.data.Datasets$. Each window considers the following: the width of the input window (known as lookback), the width of the output window (known as time horizon), the offset between input and output windows and which features are used. There was an emphasis by STL to keep stop-level information, therefore, the width of the windows is determined by the number of stops to lookback and horizon. There are three types of windows corresponding to single-step, multi-step, and adapted multi-step prediction: each having a univariate and multivariate version. Table 2.1 describes the differ-

ence between the six different data windows. The single-step data window makes a single prediction one stop in advance, given data one previous stop. The multi-step, single shot data window utilises data from the entire previous trip to predict one entire trip in the future. The multi-step, modified sample length data window takes a trip and a half worth of data and tries to predict the remainder of the trip. All windows were used to determine the capabilities of the Bi-directional Long-Short Term Memory Neural Network model.

Window Type		**Input [In # Stop(s)]**	**Output [In # Stop(s)]**	**Offset**	**# Of Features**
Single-Step	Univariate	1	1	1	1
	Multivariate	1	1	1	4
Multi-Step, Single Shot	Univariate	# Stops In Trip	# Stops In Trip	# Stops In Trip	1
	Multivariate				4
Multi-Step, Modified Length	Univariate			(# Stops In Trip) /2	1
	Multivariate				4

Table 2.1: Description of windows generated for model training

2.3.4 Implementation

The following defines the notation used in the rest of the paper. In previous works, the most common performance metrics used to evaluate the performance of a passenger flow prediction model are the root mean square error (RMSE) and the mean absolute percentage error (MAPE) [2, 25, 26]. In the equations, y_{true} is the actual passenger flow from the agency's dataset. Predicted passenger flow is then defined as y_{pred}. The loss function used for this paper is the mean squared error. Prediction quality is measured with the root mean square error (RMSE) and the mean absolute error (MAE) as the metric functions. In the equations below, the y_{true} and y_{pred} symbolize the observed and predicted busload values, and n is the number of samples. The smaller the MAE, RMSE, and MSE values, the difference of prediction and true values resemble.

$$MSE = \frac{1}{n}\sum_{t=1}^{n}(y_{true} - y_{pred})^2 \tag{2.7}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{t=1}^{n}(y_{true} - y_{pred})^2} \tag{2.8}$$

$$MAE = \frac{1}{n}\sum_{t=1}^{n}|y_{true} - y_{pred}| \tag{2.9}$$

$$MAPE = \frac{100\%}{n}\sum_{t=1}^{n}\left|\frac{y_{true} - y_{pred}}{y_t}\right| \tag{2.10}$$

Comparison Models

Multiple models were chosen test the efficacy of the proposed approach. Each model was evaluated based on six different forecasting windows: single-step univariate data window, single-step multivariate data window, multi-step single shot univariate data window, multi-step single shot multivariate data window, multi-step univariate with modified sample length data window, and multi-step multivariate with modified sample length data window. These comparisons were chosen because they are common in the literature on passenger flow prediction [2, 7, 25].

1. A simple linear regression model is a strong baseline to compare any future approach. The linear model is comprised of the mean function and the variance function. Data to be assumed to be independent and have a Gaussian distribution. [42]

2. In comparison to the linear functions described above, a multilayer perceptron model is an interconnected network of small processing units called nodes [43]. The interconnection of simple computation can be aggregated together to solve higher complexity problems [44]. Comprised of at least three layers, the MLP model the simplest neural network yet proven effective in the fields of speech recognition and image recognition [45].

3. A convolutional neural network, CNN, gets its name from the matrix operations called convolutions. Each CNN cell is built with four layers,

which includes convolutional layer, non-linearity layer, pooling layer and fully-connected layer [46]. Convolutional neural networks are often at a disadvantage with time-series data because they do not consider temporal dependencies.

From [1], the comparison of the models does not use any masking layer or middle layers. The linear regression model uses a dense layer without an activation function, the MLP model uses two hidden layers. However, the number of nodes in the CNN and LSTMs is adapted via grid search. The total number of features in the input data equal to the number of nodes per layer to emulate the number of stops for the route specified.

	Unnamed: 0	aac_id_val	aac_id_val_voyage_journalier	aac_date28	avj_ligne	avj_dir_nseo	avj_no_voyage	avj_depart28	avj_depart28_dt	al_chronobus	...	aac_descendant	aac_charge	Line_Direction	StopsNewName	Scheduled_Arrival
201450	284512	207425057	4285965	2020-08-03	26	O	63	56340	15:39	41188	...	0.0	4.0	26-O	NS105	153.0
201451	284513	207372463	4284518	2020-08-03	26	O	23	32640	09:04	42511	...	0.0	7.0	26-O	NS106	45.0
201452	284514	207377076	4284648	2020-08-03	26	O	43	44640	12:24	41182	...	0.0	2.0	26-O	NS107	99.0
201453	284515	207398453	4285243	2020-08-03	26	O	57	53040	14:44	43421	...	5.0	8.0	26-O	NS108	-43.0
201454	284516	207409857	4285542	2020-08-03	26	O	81	62160	17:16	43020	...	0.0	1.0	26-O	NS109	187.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1014039	612575	222984979	4578178	2020-10-15	26	O	25	31440	08:44	49026	...	0.0	0.0	26-O	NS119	75.0
1014040	612641	222934774	4576658	2020-10-15	26	O	7	23220	06:27	49026	...	3.0	0.0	26-O	NS119	78.0
1014041	612728	222962879	4577518	2020-10-15	26	O	45	40440	11:14	49026	...	0.0	0.0	26-O	NS119	144.0
1014042	612840	222999435	4578624	2020-10-15	26	O	117	62400	17:20	49026	...	0.0	0.0	26-O	NS119	402.0
1014043	612971	221064284	4575208	2020-10-14	26	O	37	36840	10:14	49026	...	0.0	0.0	26-O	NS119	157.0

Figure 2.3: Sample data provided by STL

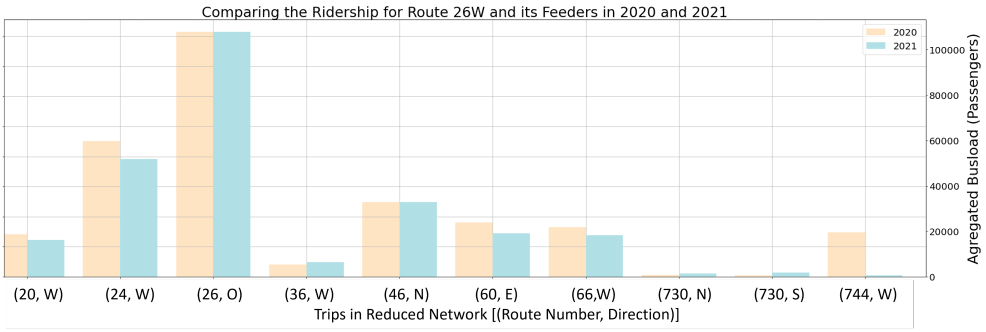

Figure 2.4: Passenger flow during October 2020 and 2021

Figure 2.5: Publicly available reference for Route 26 produced by STL

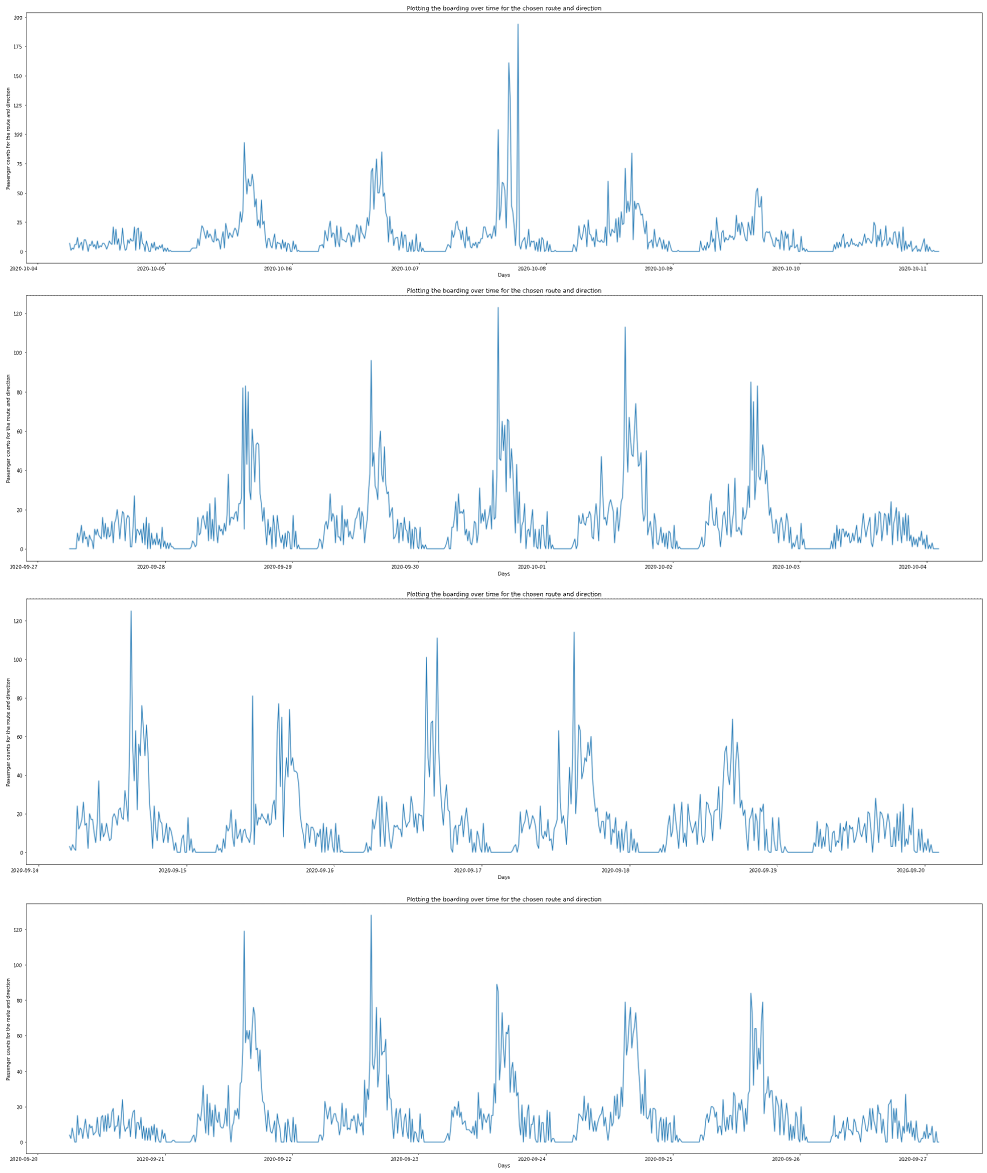

Figure 2.6: Overview of the distribution of busload sample values for a given line for each week in 2021

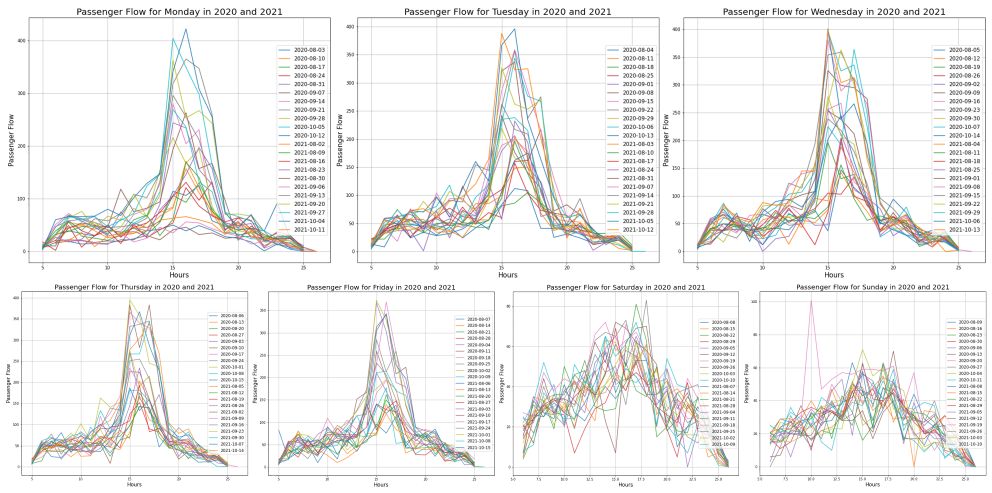

Figure 2.7: Plotting Thursday's passenger flow sample data in 2020 and 2021

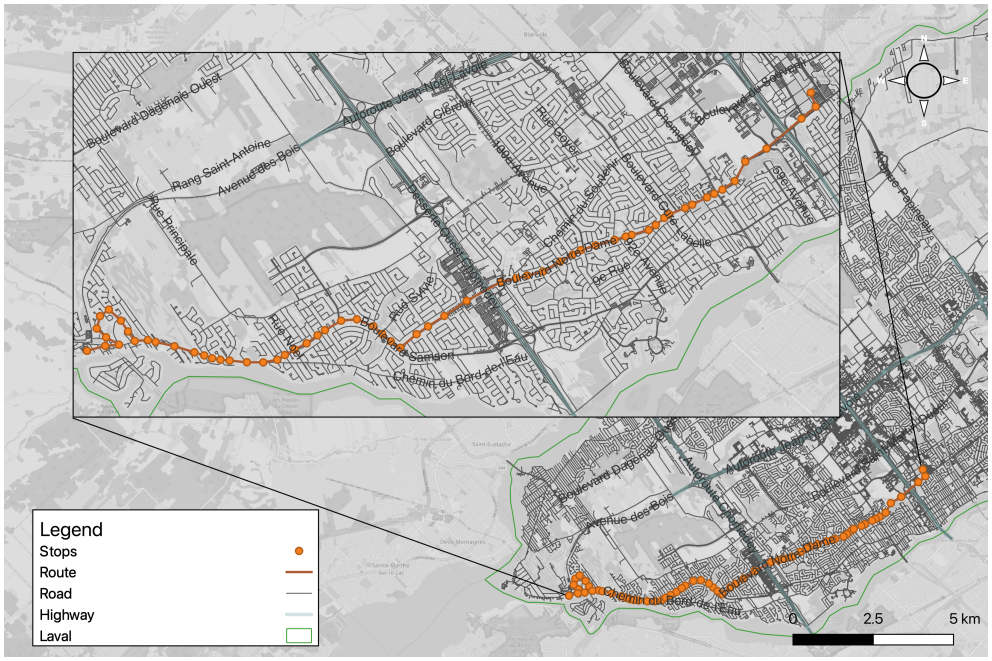

Figure 2.8: Map of stops from route 26 in Laval

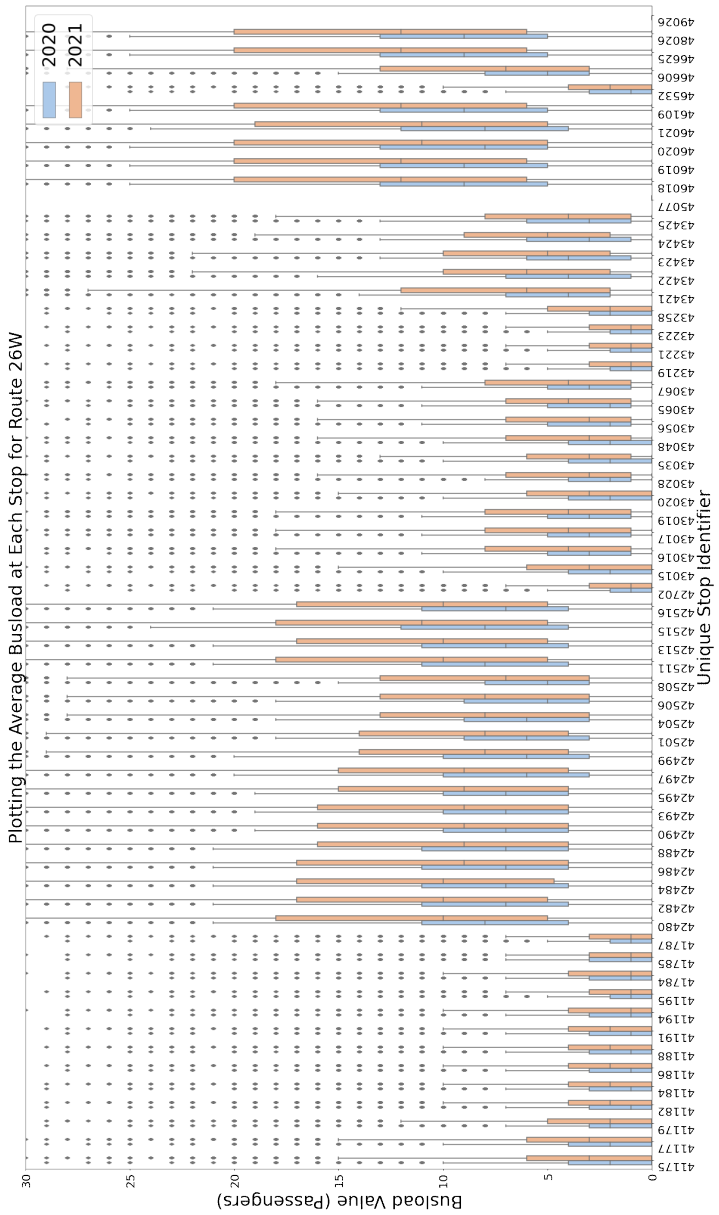

Figure 2.9: Distribution of busload sample for trip 26W in 2020 and 2021

2.4 Results

Using Route 26W, the following section provides a comparison of commonly used models for passenger flow prediction in literature. Similar to [1], the models proposed for the initial analysis are the Feed-forward Neural Network, the Long-Short Term Neural Network, and the Stacked Bi-directional/Unidirectional Long Short-term Memory Neural Network (shortened to BDLSTM). The Convolution Neural Network was added to provide additional information pertaining to the performance of the proposed model.

In this work, it should be noted that in both single-step models and multi-step models, the dataset is windowed with each timestamp representing a stop within each trip. Each window considers the number of stops of historical data to be considered in order to predict a specified number of stops into the future, commonly known as time-horizon.

The models present results trained on data from 2020 and 2021. As per the global COVID-19 pandemic, the ridership behaviour is novel in comparison to pre-COVID, historical data. Despite being trained with data during COVID, the predicted passenger flow can reflect the culture-shift and lifestyle changes associated with post-COVID living. Factors associated with a new behaviour including an increase in remote work, hybrid work, online/phone appointments and distance learning. Furthermore, the trained model is based on a route with the highest ridership.

2.4.1 Comparison Models and BDLSTM

Route 26 in the westward direction is used for model testing. It was used because not only is it the route with the most amount of trips, but it is the most frequented route by passengers in the network for the observed dates. The comparison models are to prove that the difference in performance is significant and an BDLSTM should be further investigated.

Single-Step Modelling

The analysis for the single step models can be divided into two separate categories:

1. Single-step univariate models
2. Single-step multivariate model

In both forms of single-step modelling, the pre-processing step batches the data into windows of 62 consecutive inputs and labels at a time. A single step model makes predictions one stop into the future.

It is important to note that the linear regression model and dense models are single-input single-step models. The model makes predictions one stop into the future based on a single stop in advance. The CNN, "vanilla" LSTM and BDLSTM are multi-input single step models. The model makes predictions one stop into the future based on a look-back of the number of stops in the line.

Seven models were defined for comparison: the Linear Regression model, the Support Vector Regression model (SVR), the Random Forest Regressor model (RF), the Multilayer Perceptron model (MLP), the Convolutional Neural Network (CNN), a Gated Recurrent Unit model (GRU), a simple LSTM, LSTM, against the Bi-directional LSTM, BDLSTM. The results for one route with the comparison models are shown in Table 2.2 for October 2020 and October 2021. The results of the proposed BDLSTM is emphasized. The results show that the BDLSTM outperforms the comparative models. The linear regression method has a calculated MAE of 3.119 and an RMSE value of 4.475 for the univariate case and a RMSE of 4.771 and a MAE of 3.556 for the multivariate case. The MLP model performs similarly to that of the linear regression model. With a RMSE and MAE of 4.471 and 3.200 for the univariate case and a RMSE of 4.515, and MAE of 3.226. These models are single-input single step models. They perform as expected. There is no significant difference between the two. Noteably, the Random Forest regression models perform similarly to these simpler models. The Support Vector Regression models show lower error rate than that of the RF regression models. The CNN model provides an RMSE of 4.038 but a much lower MAE term of 2.591. The CNN multivariate outperforms the univariate model. This is to be expected as there is more input data to observe where the RMSE is 4.000 and the MAE is 2.591. The CNN model outperforms the MLP and the linear regression models, this is to be expected because the CNN model has a longer look-back by the previous trip. The simple LSTM model performs better than the CNN model. This is to be expected because it is able to handle the time-series data more efficiently. In both univariate and multi-variate cases, the vanilla LSTM slightly outperforms the CNN. Comparing the simple LSTM to that of the proposed stacked bi-directional LSTM is quite significant. However, to the vanilla or simple LSTM and the BDLSTM, it is assumed that the simple gate structure and the single layer

implementation could be the cause. The simple LSTM model also does not outperform the BDLSTM with an MAE value of 3.845 and an RMSE value of 2.423. The BDSTM multivariate had the lowest MAE and MSE values of all models with 2.663 and 1.385, respectively. In summary, the BDLSTM model is the strongest method for short-term passenger flow prediction.

Comparing the results between 2020 and 2021, the prediction error in consistently higher in the 2021 dataset. Based on Figure 2.9, the busload at each stop is significantly higher in October 2021 than that of the previous year. The increase in passenger flow in 2021 from 2020 can be attributed to world events, specifically the global COVID-19 pandemic. With the added passenger count at each stop in 2021, increases the volatility and polarizes the data between stops.

Model Specifications		**2020**		**2021**	
Model	**Data Type**	**RMSE**	**MSE**	**RMSE**	**MSE**
Linear	Univariate	4.475	3.189	8.202	5.758
	Multivariate	4.771	3.557	8.493	6.091
Dense	Univariate	4.472	3.201	8.117	5.675
	Multivariate	4.516	3.226	7.798	5.505
SVR	Univariate	5.470	3.547	8.278	5.207
	Multivariate	4.891	3.701	7.098	5.461
RF	Univariate	4.603	3.225	7.460	4.998
	Multivariate	4.083	2.797	7.294	4.903
CNN	Univariate	4.038	2.648	7.149	4.845
	Multivariate	4.000	2.591	7.044	4.812
RNN	Univariate	4.009	2.546	7.041	4.661
	Multivariate	4.000	2.591	7.037	4.656
Vanilla LSTM	Univariate	3.846	2.423	6.984	4.585
	Multivariate	3.296	2.279	6.843	4.410
BDLSTM	Univariate	2.751	1.479	2.279	1.183
	Multivariate	**2.663**	**1.385**	**2.161**	**1.141**

Table 2.2: Performance comparison of the proposed model with other comparative models for single-step passenger flow prediction in 2020 and 2021

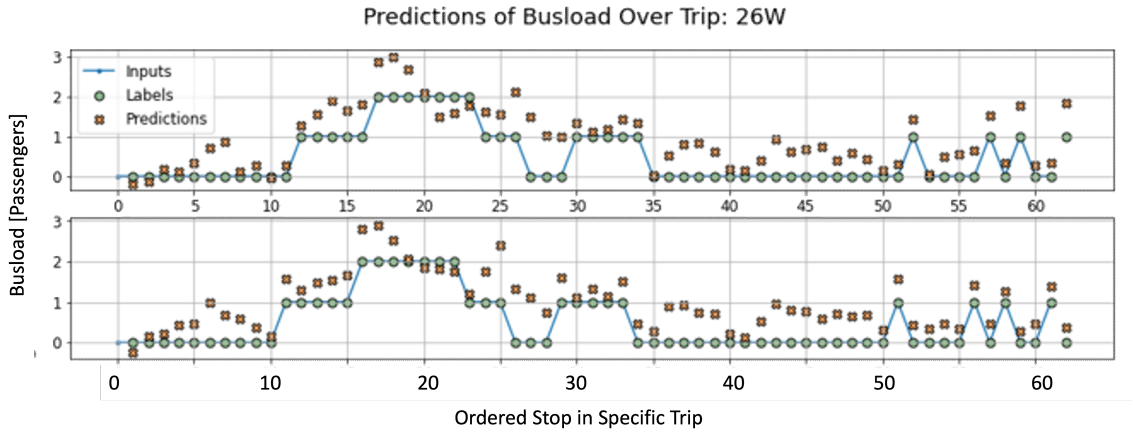

Figure 2.10: Visualization of single-step BDLSTM trip predictions

Idx	Model Description
0	Linear_SingleStep_Univariate
1	Linear_SingleStep_Multivariate
2	Dense_SingleStep_Univariate
3	Dense_SingleStep_Multivariate
4	CNN_SingleStep_Univariate
5	CNN_SingleStep_Multivariate
6	GatedRecurrentUnit_SingleStep_Univariate
7	GatedRecurrentUnit_SingleStep_Multivariate
8	LSTM_SingleStep_Univariate
9	LSTMv_SingleStep_Multivariate
10	LSTM_SBU_SingleStep_Univariate
11	LSTM_SBU_SingleStep_Multivariate
12	StateVectorRegression_SingleStep_Univariate
13	StateVectorRegression_SingleStep_Multivariate
14	RandomForest_Regression_SingleStep_Univariate
15	RandomForest_Regression_SingleStep_Multivariate

Figure 2.11: Performance comparison of the proposed model with comparative models for route 26W, single-step passenger flow prediction in 2020

Multi-Step Modelling, Single shot

The analysis for the multi-step models can be divided into two separate categories:

1. Multi-step univariate models

2. Multi-step multivariate model

The model must learn to predict a range of future values in a multi-step prediction. In both forms of multi-step modelling, the pre-processing step batches the data into windows of 62 consecutive inputs and labels at a time. As a result, unlike a single step model, which predicts only a single future busload value, a multi-step model predicts a sequence of busload values one full trip in advance. This multi-step model performs single shot predictions where an entire trip worth of time series data is predicted at once.

The results of one route with the comparison models are shown in Table 2.3 for October 2020 and 2021.

As expected the performance of the models in multi-step prediction are worse than those of the single-step prediction. This is due to the nature of the single shot prediction where we are predicting all the features across all output time steps given a look-back of a previous trip. Using the comparative models it is possible to see in Figure 2.17. Even with the advanced proposed models, the multi-step models have higher MAE. It can be hypothesized that the multi-step predictor inevitably introduces errors, with multi-step modelling, each subsequent prediction causes compounding errors change the input distribution for future prediction steps.

Multi-step Modelling, Modified Sample Length

The analysis for the multi-step models can be divided into two separate categories:

1. Multi-step univariate models

2. Multi-step multivariate model

As presented, based on Table 2.4, it is possible to see that there is no significant improvement between the multi-step linear model and the proposed multi-step BDLSTM model for passenger flow forecasting. This is to be expected because the nature of a single shot multi-step model. Expecting to predict the passenger flow for an entire trip provided data from the previous trip, is unrealistic because each trip throughout a day, month and year varies. A slight modification of the multi-step prediction method is predicting the 3/4 of the next trip given the entire previous trip and the start of

Model Specifications		2020		2021	
Model	**Data Type**	**RMSE**	**MSE**	**RMSE**	**MSE**
Linear	Univariate	4.597	3.386	8.202	5.758
	Multivariate	4.711	3.525	8.290	5.960
Dense	Univariate	4.607	3.424	8.069	5.796
	Multivariate	4.528	3.336	7.824	5.644
CNN	Univariate	4.422	3.209	7.660	5.567
	Multivariate	4.417	3.197	7.414	5.387
Vanilla LSTM	Univariate	4.354	3.143	7.497	5.449
	Multivariate	4.318	3.1300	7.414	5.387
BDLSTM	Univariate	7.657	5.576	7.657	5.576
	Multivariate	**4.379**	**3.163**	**7.407**	**5.371**

Table 2.3: Performance comparison of the proposed model with other comparative models for single-shot, multi-step passenger flow prediction in 2020 and 2021

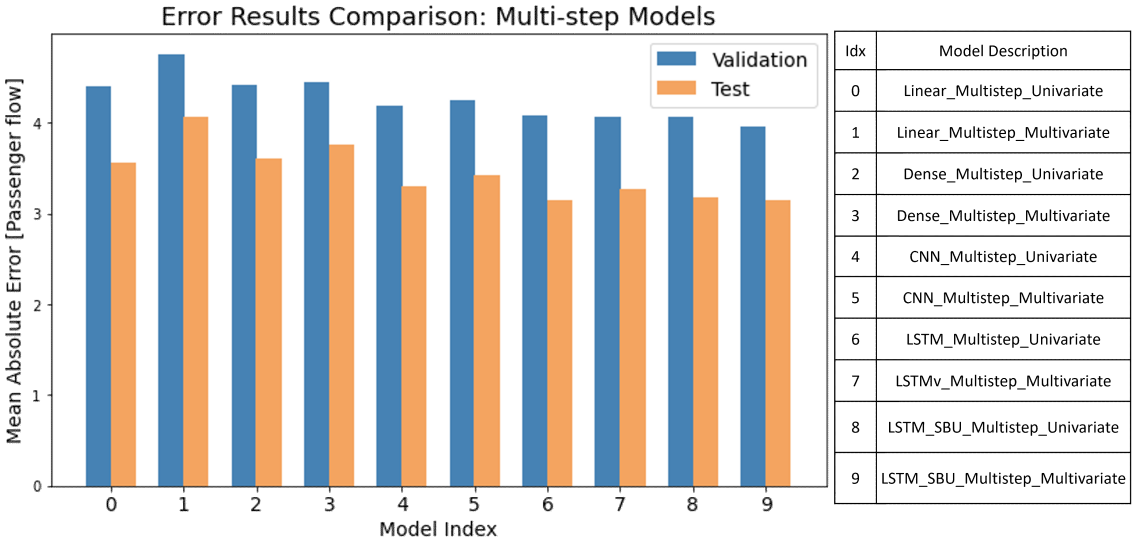

Figure 2.12: Performance comparison of the proposed model and comparative models for route 26W, multi-step passenger flow prediction in 2020

the current trip. Based on the results in Table 2.4 it is possible to see the significant improvement, 3 passenger closer to the actual values using the

modified multi-step prediction method.

Model Specifications		2020		2021	
Model	**Data Type**	**RMSE**	**MSE**	**RMSE**	**MSE**
Linear	Univariate	4.452	3.165	7.921	5.569
	Multivariate	4.630	3.375	8.167	5.776
Dense	Univariate	4.462	3.222	7.865	5.521
	Multivariate	4.444	3.200	7.750	5.494
CNN	Univariate	4.144	2.838	7.228	5.002
	Multivariate	4.239	2.929	7.366	5.126
Vanilla LSTM	Univariate	3.834	2.505	7.256	4.988
	Multivariate	3.895	2.562	6.857	4.716
BDLSTM	Univariate	3.673	2.301	5.643	3.456
	Multivariate	**3.665**	**2.264**	**5.065**	**3.278**

Table 2.4: Performance comparison of the proposed model with other comparative models for a modified sample length, multi-step passenger flow prediction in 2020 and 2021

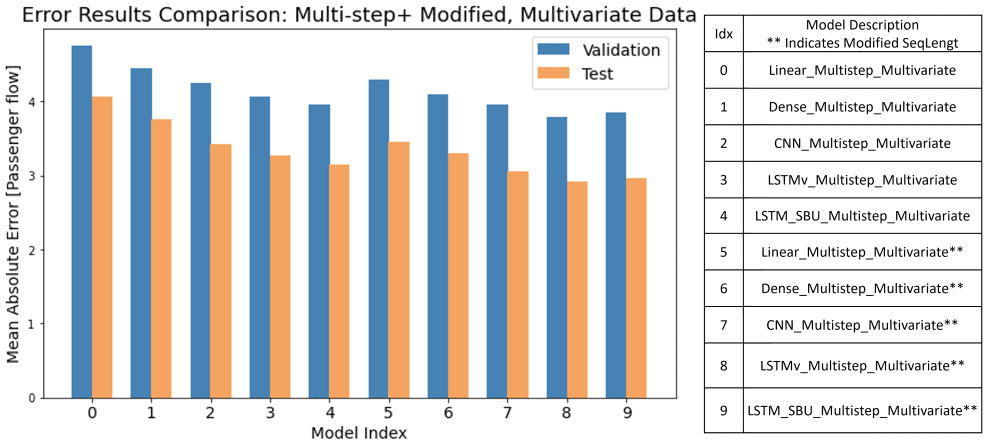

Figure 2.13: Performance comparison of the proposed model and comparative models for route 26W, modified sample length, multi-step passenger flow prediction in 2020

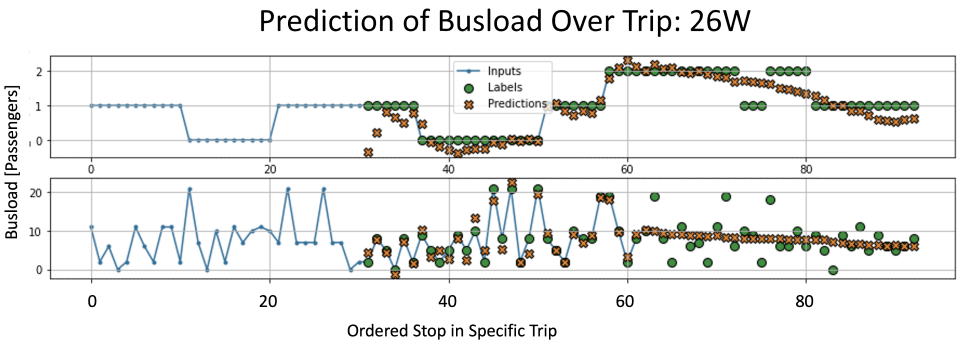

Figure 2.14: Forecast visualization of the univariate, multi-step, BDLSTM for route 26W, using modified sample length in 2020

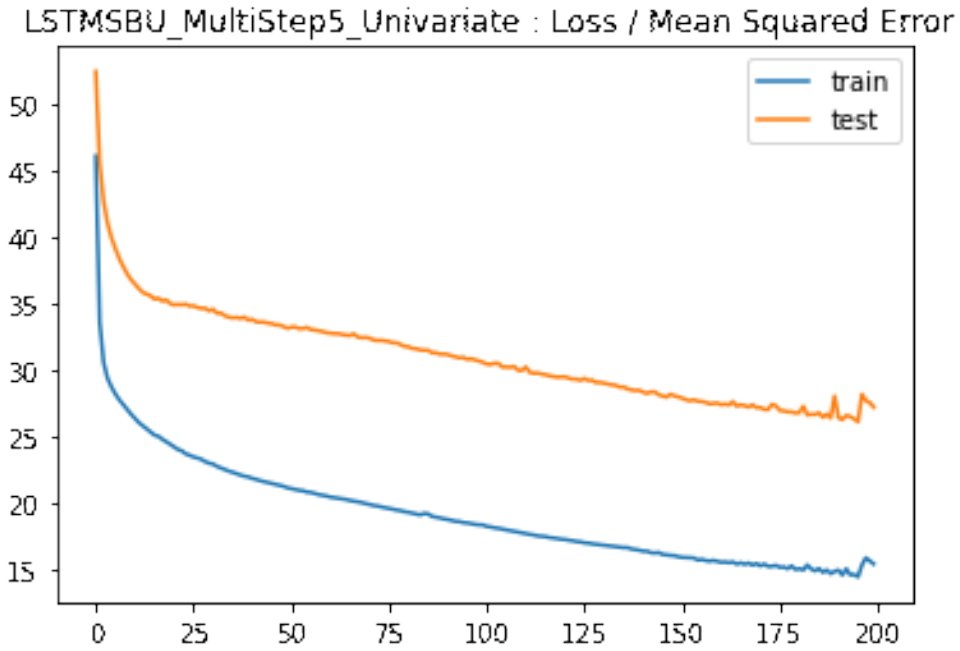

Figure 2.15: Learning curve of the univariate, multi-step, BDLSTM for route 26W, using modified sample length in 2020

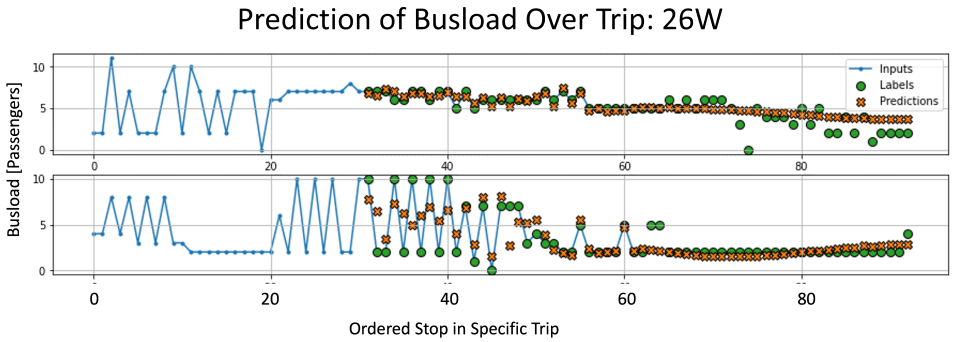

Figure 2.16: Forecast visualization of the multivariate, multi-step, BDLSTM for route 26W, using modified sample length in 2020

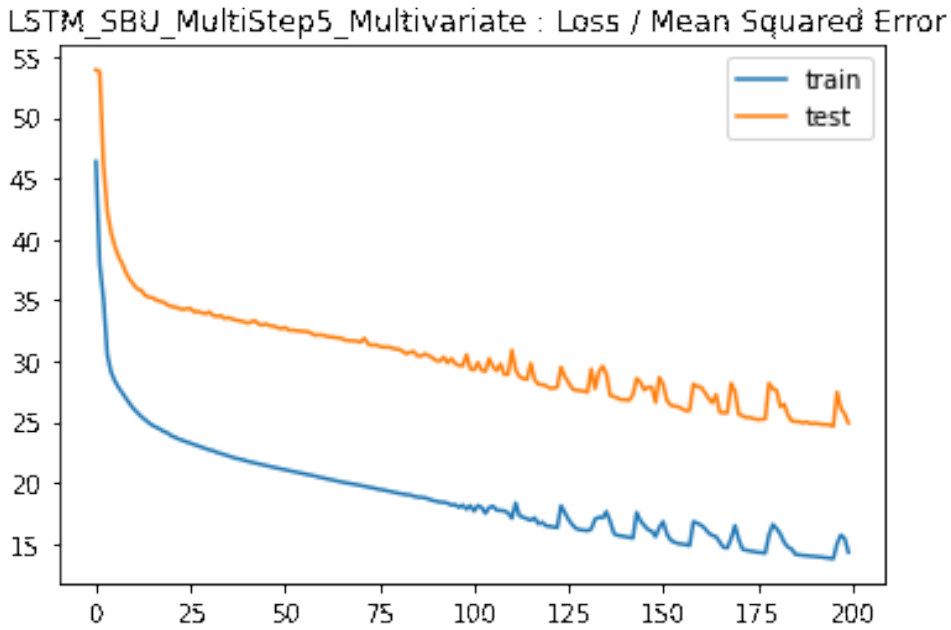

Figure 2.17: Learning curve of the multivariate, multi-step, BDLSTM for route 26W, using modified sample length in 2020

Chapter 3

Passenger Flow Prediction on Bus Systems Using a Bi-directional Long-Short Term Memory Neural Network Kalman Filter

3.1 Introduction

Passenger flow prediction is a widely researched topic within the public transportation sector. Applying passenger flow forecasting for bus systems is based on a multitude of factors including a particular trip, time of day and day of the week. Increasing the reliability of passenger flow prediction on buses could not only support capacity and planning for agencies but could provide valuable information for passengers. In this chapter, the objective is to optimize passenger flow prediction by designing a comprehensive evaluation setup and propose a novel Bi-directional Long-Short Term Memory Neural Network Kalman filter model (BDLSTM KF) based on the implementation of a Long-Short Term Memory Neural Network Kalman filter by [2]. The remainder of the chapter is organized as follows, Section 3.2 provides a complete background of the Kalman filter (KF), its tuning parameters and how deep learning can be combined with a Kalman filter. Section 3.3, describes the details of the dataset, the respective comparison models and the proposed model and its training process. The performance metrics that are

used throughout the paper are described. Lastly, in section 3.4, the results are discussed.

3.2 Methodology

This section provides pertinent background information on the Kalman filter and combining deep learning techniques with a Kalman filter. The Kalman filter is an extremely popular state estimation technique used in navigation, object-tracking and time-series forecasting [33]. The novel use cases of combining deep learning and a Kalman filter overall produce models with less uncertainty than each method alone [22, 47].

3.2.1 Kalman filter

One of the most well-known Bayesian filter theories is the Kalman filter, which is a linear optimal status estimate approach. The Kalman filter was created by RE Kalman in 1960. When it was created its applications were primarily for signal processing for communications [48]. A Kalman filter can be used in any situation when you have ambiguous information about a dynamic system and want to make an educated guess about what it will do next. Even in the presence of noise, the Kalman filter is usually extremely good at figuring out what actually happened. It is a versatile and effective tool for merging data in the face of uncertainty [49].

Since the publication of [48], there have been many applications from navigation, orbit calculations, integrated dynamic positioning, microeconomics and image processing [50]. With its vast applications, the Kalman filter's standard approach is known to be a fast, efficient and strong anti-interference method for dynamic systems. With the many different applications, the Kalman filter has been adapted through different configurations and assumptions. The well-known configurations beyond the discrete Kalman filter (KF), are the Extended Kalman filter (EKF) and the Unscented Kalman filter (UKF) [50].

The Kalman filter algorithm uses time-series noisy measurements, to produce estimates of future states of a system. A series of measurements provides a more accurate forecast than that of a singular measurement. The Kalman filter equations can be categorized into two groups: time update equations and measurement update equations. The time update equations describe the

Represents	Symbol	Description	Represents	Symbol	Description
State	$\vec{x}_t$	State vector	Measurement	$\vec{z}_t$	Measurement vector
	$\hat{x}_{t-1}$	Previous state		—	NA
	$\hat{x}'_t$	Predicted current state		—	NA
	$\hat{x}$	Current state		$\hat{z}_t$	Current measurement
	A	State transition matrix		H	Measurement matrix
	w_k	Process noise vector		v_k	Measurement noise vector

Table 3.1: Notation of state and measurement for the discrete Kalman filter (KF)

prediction phase, the Kalman filter produces estimates of the current state variables and error covariance to obtain *a priori* estimates for the next step.

A dynamic system can be represented by a Kalman filter via a state and measurement equation. The state and measurement notation varies from implementation to implementation. State and measurement are represented by x_t and z_t, respectively. Each of the state variables are defined by the Kalman filter state equation 3.1. The estimate of the initial state is commonly called the measurement or observation, defined by a similar equation 3.2. Table 3.1, shows the definitions of each of the components in the system state equation and the measurement equation. It should be noted that based on the linear and Gaussian assumptions of the system, w_k and v_k are assumed to be independent, positive definite, zero-mean Gaussian white noise vectors. Furthermore, Table 3.1 details the notation of state and measurement throughout the Kalman filter model. The state transition matrix, A, relates the previous state and the current step state. The measurement matrix, H, relates the state, x_k to the measurement, z_k.

$$x_k = Ax_{k-1} + w_k, p(w)\ N(0, Q) \tag{3.1}$$

$$z_k = Hx_k + v_k, p(v)\ N(0, R) \tag{3.2}$$

In every time step t, the KF produces a new estimate x_t using only

the previous estimate x_{t-1} and the new measurement y_t. As a result, the computational complexity of the KF does not grow in time. The prediction stage is structured with the following equations:

$$x_k^- = Ax_{k-1} \tag{3.3}$$

$$P_k^- = AP_{k-1}A^T + Q \tag{3.4}$$

where A is the state transition matrix, P is the error covariance matrix and Q is the process noise covariance. The prime superscript on the the state or covariance variables defines a *priori* estimate given the prior time step. Similarly as mentioned in Table3.1, the equivalent variables without superscripts refer to *posteriori* estimates given the observations at that time step. The Kalman filter's initial step allows for the recursive inclusion of all points in the series, resulting in each state estimate having some dependence on all previous state estimates. The smoothness of the resultant series is affected by the process noise covariance matrix in 3.1, with low covariance values increasing the degree of smoothing.

Second, the update step updates the state estimate with the measurements. The update stage is given by:

$$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1} \tag{3.5}$$

$$x_k = x_k^- + K_k(z_k - Hx_k) \tag{3.6}$$

$$P_k^- = (I - K_k H)P_k^- \tag{3.7}$$

where K_k is the Kalman gain, H is the measurement matrix that maps from state to measurement space, R is the measurement noise covariance and z_k is an measurement at time k.

The Kalman gain calculated in 3.5 can be considered as providing a weighting between the measurements and the previous state estimates. Thinking about the Kalman gain, K, in practice, the actual measurement, z_k, is more "trusted," as the measurement error covariance, R, approaches zero, while the predicted measurement, $H\hat{x}_k$, is less trusted. Meanwhile, as the error covariance, P_k, approaches zero, the actual measurement, z_k, becomes less and less trusted, whereas the predicted measurement, $H\hat{x}_k$, becomes more and more trusted.

Filter Parameters and Tuning

Beyond the theoretical applications of the filter, it is common practice to take sample data to determine a baseline for the measurement covariance, R, and the process covariance, Q before applying the filter [49]. The process to determine these parameters is often done offline and is commonly known as system identification.

Using sample measurement data, calculating the measurement covariance, R for the sample, provides some initial values for the variance of the measurement error. However, it is shown by [49], that choosing poor initialization parameters for the filter can cause the filter to diverge or underperform.

The determination of the process covariance, Q, is generally more challenging because it is not possible to directly observe the system that is to be estimated. In the literature [2] it's described how it's possible to "inject," enough uncertainty into the process by choosing a proper value for Q.

In the case of non-linear models, often the choice is less deterministic [51]. For example, this noise source is often used to represent the uncertainty in the process model. Sometimes a very poor model can be used simply by "injecting" enough uncertainty via the selection of Q. Certainly in this case one would hope that the measurements of the process would be reliable. In either case, whether or not we have a rational basis for choosing the parameters, superior filter performance (statistically speaking) can often be obtained by "tuning" the filter parameters and state prediction parameters. Tuning is usually performed off-line, frequently with the help of another (distinct) Kalman filter [52].

3.2.2 Combining Deep Learning models with a Kalman filter

Estimating state variables of a dynamic systems, the Kalman filter is commonly used. An extremely attractive attribute of the Kalman filter is that it is a lightweight algorithm that requires only previous state information to make an intelligent prediction about the system's current state. The primary disadvantages of the Kalman filter model are its model parameter initialization and state estimation assumptions. These two disadvantages are addressed through an LSTM-based Kalman filter model.

Addressing the estimation of Kalman filter parameters requires a strong understanding of the system to be modelled, which is not always possible [52].

There are diverse application scenarios for the Kalman filter, and there has been significant work done to optimize state parameter estimation. Recently, deep learning-based approaches have been applied to the Kalman filter to perform originally ambiguous parameter selection. There are two main lines of research that combine deep learning and Kalman filter models for temporal regularisation. It is common to categorise the approaches as those that learn static Kalman filter parameters and those that actively regress the parameters during filtering. In both cases, choosing parameters based on learnt data greatly improves the likelihood of the filter converging [2].

State estimation, the goal of the Kalman filter, can easily be extended to time-series forecasting where sequential data can be represented by a series of states. Recurrent neural networks were specifically created for analyzing sequential data often applied to time-series forecasting [15]. Introducing a gate mechanism to the traditional recurrent neural network, the LSTM model is capable of learning long-term dependencies within the data. Therefore, using an LSTM model as a component to perform state estimation is a logical application.

3.3 Proposed Model and Framework

The proposed model could be considered as a hybrid of both categories. The Long-Short Term Memory Neural Network-based Kalman filter, LSTM KF, actively regresses the Kalman parameters using historical data and non-linear methods. In [2], the author proposes three distinct Kalman filters, one to predict process covariance, Q, one to predict the measurement covariance, R and one to predict the transition function, F. The goal of the proposed Kalman filter and Neural Network hybrid is to use the simplicity of a Kalman filter without having to specify a transition function, F or a fixed process or measurement covariance matrices Q or R.

3.3.1 Model

Taking on the assumptions that the new measurements are noisy estimate of the underlying LSTM modules to predict the internals of the Kalman filter cell structure, where H is assumed to be the identity matrix, leaving equations 3.8-3.9 to define the LSTM KF model. An architecture diagram in Figure 3.1, provides insight on the internals of the cell of the LSTM KF.

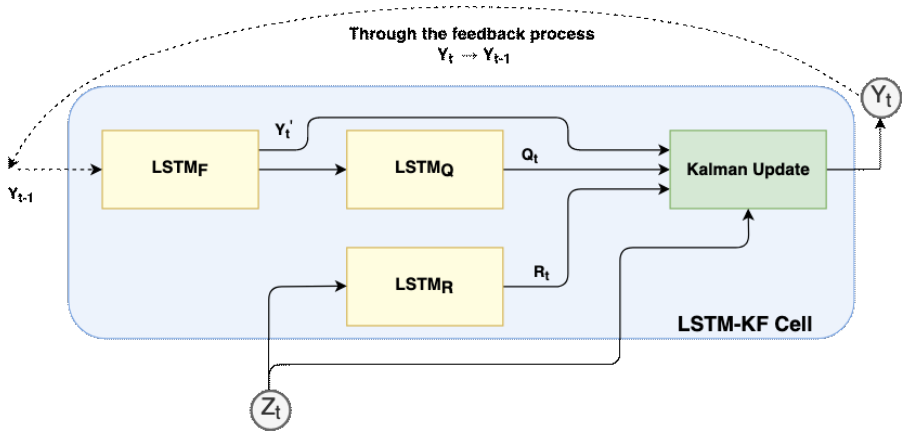

Figure 3.1: Operation of a LSTM Kalman filter cell

$$y_t = f(y_{t-1}) + w_t, w_t \ N(0, Q) \tag{3.8}$$

$$z_t = y_t + v_t, v_t \ N(0, R) \tag{3.9}$$

Before using the LSTM-based Kalman filter to predict passenger flow, the appropriate model should be built first. To form the nonlinear transfer function of the state, x_k, the error covariance matrix, Q, and the observation noise covariance matrix, R, need to be chosen. The preparation procedures for the LSTM-based KF are discussed at a given time step:

- The LSTM$_f$ takes the previous busload prediction and outputs an updated prediction value.
- The LSTM$_Q$ takes the current predicted value of the busload from the LSTM$_f$ and predicts a value for the process covariance.
- The LSTM$_R$ is unique as it takes single-step multivariate values as an input and outputs a predicted value for the measurement covariance.

The element-wise exponential is taken from the outputs of the LSTM cells and they are translated into a batched diagonal tensor prior to future calculations.

Prediction

The prediction step is slightly modified from the original Kalman filter to

include the outputs of the LSTM layers. In equation 3.10, the f is modeled by the $LSTM_f$ module. Prior to the matrix calculations in equation 3.12, the Jacobian of f with respect to the newly predicted state, $\hat{y}_k^-$ is taken. In equation 3.12, the capital letters represent matrices, where Q_t is the output of the $LSTM_Q$. The initial value of P_{k-1} is set as a trainable matrix with normal distribution.

$$\hat{y}_t^- = f(\hat{y}_{t-1}) = LSTM_F(\hat{y}_{t-1}) \tag{3.10}$$

$$F = J_f(\hat{y}_{t-1}) \tag{3.11}$$

$$P_t^- = FP_{t-1}F^T + Q_t \tag{3.12}$$

Update

Provided the assumption that the incoming measurements are noisy estimates of the underlying state, equation 3.5 defined by the Kalman filter turns into 3.13, setting H equal to the identity matrix. The value R_t represents the output of $LSTM_R$. To calculate the Kalman gain, in the equation 3.13, K, the value P_t^- is calculated in the prediction step. The value z_t in equation 3.14 is the observed measurement at time, t. In the last equation of the update step, equation 3.15, P_t^- is calculated in the prediction step, I is the identity matrix and K_t is calculated earlier.

$$K_t = P_t^-(P_t^- + R_t)^{-1} \tag{3.13}$$

$$y_t = y_t^- + K_k(z_t - y_t^-) \tag{3.14}$$

$$P_t^- = (I - K_tH)P_t^- \tag{3.15}$$

Throughout the prediction and update process, the end result is to produce a new prediction of the busload one time horizon ahead, as denoted as $\hat{y}_t$.

3.3.2 Experiments

In this section, extensive testing is described and the performance of the proposed model by [2] is presented against that of a combined BDLSTM KF. Real-world data from the Société de Transport de Laval's, STL, bus system is used for short-term passenger load forecast to validate the proposed model.

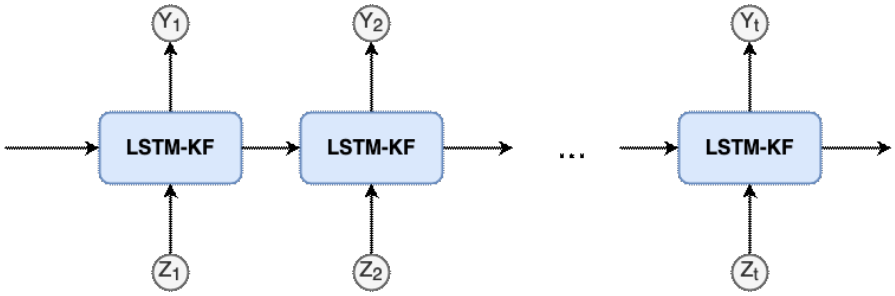

Figure 3.2: Unfolded architecture of a LSTM Kalman filter

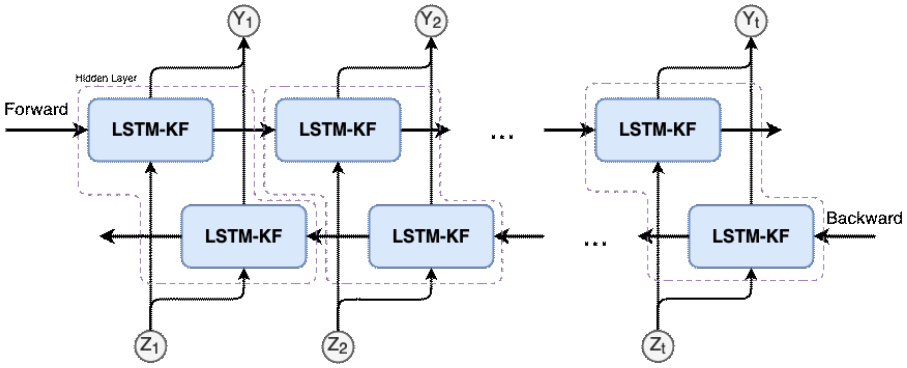

Figure 3.3: Unfolded architecture of a Bi-directional LSTM Kalman filter

Datasets and Input Features

The dataset used in the LSTM-based Kalman filter is a single step dataset. The single-step dataset is identical to that used in Chapter 2. The chosen single-step dataset is multivariate. The multivariate dataset uses 4 key features, including hour of day (HOD), day of week (DOW), unique bus stop identifier, and whether the trip is in-bound or out-bound. The data was windowed based on the width, time offset and input/label features. The dataset was created based off a sliding windows over time-series data using Keras Tensorflow's $tf.data.Datasets$. The width of the input window is equivalent to the number of stops in a particular trip. The output window or time horizon, is equal to one, in this single step model. The windows are split into feature and output pairs. The data represents Route 26W, for a month, which represents 100k data samples. There are 62 stops in the chosen trip. The data taken to train the model is 70%,20%, 10% for the training, validation, and test sets. The considerations of time series data was taken into

account when slicing the data into windows, where the shuffling occurs after the model was trained.

Machine Configurations

In all experiments, a Processor Intel(R) Core(TM) i9-10850K CPU 3.60GHz machine was used.

Hyperparameter Tuning and Model Training Configurations

In this section, the hyper-parameters/implementation used for each method compared are described. Based on the Kalman filter model, it is mandatory to initialize model parameters prior to training.

Hyperparameter Tuning The proposed LSTM-based Kalman filter model has many trainable and tunable parameters. Experimentation is required to identify the optimal weights which is known as hyperparameter tuning. The learning rate and the dropout rates for each layer were tuned based on model performance metrics, mean absolute error. Leveraging Keras Tensorflow's TensorBoard provides a thorough visualization of the tunable parameters, known a the HParams dashboard. Based on the grid search method, the number of neurons in the LSTM cell are chosen to be 256.

Model Training Configurations Before the model is trained, it is necessary to initialize all parameters that are initially independent of data. The first are the instances of the LSTM cells involved. The initialization not only includes creating the instances of the cells but also getting the initial state provided a batch size. Additionally, as a default value, the bias of the forget gate at initialization is set to 1.0. Based on Keras Tensorflow, each cell state would be zero-filled; this includes $LSTM_F$, the state transition matrix, $LSTM_Q$, the transition noise covariance matrix, and the measurement covariance matrix, $LSTM_R$. For each of the LSTM cells, there are the corresponding weight and bias vectors that are initialized to a normal distribution. These weights and bias vectors are trainable matrices to be tuned in testing. The initialization of these vectors follow [2] recommendations. The purpose of the weight and bias vectors are to properly shape the output matrix acting as a dense layer. Finally, an error covariance P had to be initialized to perform the first prediction step. This matrix was initialized with a normal

distribution based on the output feature space. The regression loss function applied to the data is the mean squared error, based on multiple authors of deep learning-based Kalman filters, the mean square error is commonly used [2, 33, 38]. As mentioned above, in the modified LSTM Kalman filter, there are trainable matrices, weights and biases declared in the initialization. These trainable parameters correspond to the output of the LSTMs as well as the error covariance matrix. The model is trained based on its mean squared error. To iteratively update the trainable network weights the Adam optimizer in Keras Tensorflow was used.The Keras implementation of LSTM was used with batch length equal to that of the trip size for 150 epochs.

Accuracy Score and Metrics

In this section, a detailed description for the accuracy metrics used throughout the experiments is left the same from the above chapter which is the Root Mean Squared Error (RMSE) and the mean absolute error (MAE). Throughout all experiments, MAE is used as an evaluation metric.

$$RMSE = \sqrt{\frac{1}{n}\sum_{t=1}^{n}(y_{true} - y_{pred})^2} \tag{3.16}$$

$$MAE = \frac{1}{n}\sum_{t=1}^{n}|y_{true} - y_{pred}| \tag{3.17}$$

3.4 Results

In this section, the results of the proposed Bi-directional Long Short-Term Memory Neural Network Kalman filter are analyzed and compared with the original proposed model by [2]. To observe the properties of the proposed model, extensive study of temporal feature learning, spatial feature learning, and model robustness is carried out.

3.4.1 Comparison of a Single-Step Model using a LSTM Kalman filter and BDLSTM Kalman filter

The primary benefit of the proposed LSTM based Kalman filter is its abilities to choose Kalman filter parameter from historical data. Once the Kalman

filter parameters are chosen, the complexity of the Kalman filter model is extremely low at O(n3) referencing the naive/brute force computation scheme. As the dataset extends beyond theoretical pre-cleaned datasets, the actual application of passenger flow can be considered. The transportation agency, STL, is a forward-thinking agency who hopes to support the municipal government towards the "Smart City," title. The computational complexity becomes a topic of discussion when considering the emergence of public-facing IoT devices. These in-field devices could communicate its predictions in real-time based on its environment. Having a highly accurate, low complexity algorithm would be beneficial to IoT computational capacities often having limited resources and low-power. The re-training of the bias and weights can be updated via LTE connection instead of performing the training on-board the IoT device.

The proposed model boasts accessibility to the model's internal parameters, the cells' weights and biases. Access to lower level components of the LSTM cells provides a higher level of control. Weights are the true values associated with each feature, and they indicate how important that feature is in forecasting the final result. Bias, also known as a y-intercept in a regression equation, is used to shift the activation function to the left or right. Unlike the models used in the previous chapter, which are trained on series of batched data, the LSTM cell is the logic to process single step data. Additionally, the layer configuration of multiple cascaded LSTM cells benefits from having ensemble neural network configuration. Ensemble neural networks are multiple distinct models combined together to obtain better generalization [53].

The model proposed by [2], takes very little time to converge. As oppose to training of a traditional neural network, during training, the model is learning Kalman filter parameters. The Kalman filter parameters which are based on historical data and outputted by the LSTM cells. The output parameter values are expoentialized and fixed to the diagonal, the overall variance of these values is limited. Furthermore, the covariance matrices produced by the LSTM cells are responsible for deciding the noise of the system; where the LSTM cell could decide that the new measurement is completely disregarded. The likelihood of that occurring is unlikely based on the size and validation process of the historical dataset. The convergence speed of the model is an asset for the real-world application to be provided to the transportation agency.

An end-to-end bi-directional-LSTM based model is proposed and trained

to exploit temporal information prior to and following the current timestep. The bi-directional aspect of training is to form a more complete representation of the sequential data. Based on many proposed bi-directional model publications, the bi-directional feature applied recurrent neural network allows the prediction mechanism to see the future and be trained as such. The connections between a bi-directional recurrent unit allows for prior and later hidden states enable the model to learn temporal information before and after the current frame.

Bi-directional Advantage Convergence Speed Trade Off

Despite the improvement to the performance metrics between the unidirectional LSTM-KF and the bi-directional LSTM-KF, a significant difference can be noted in convergence speed while training. To further understand this observation, two learning curves for the LSTM KF are shown in 3.6. Drawing a comparison to the the previous chapter, learning curves of the BDLSTM and the LSTM KF are pictured in Figure 3.5 The number of epochs is a hyperparameter that specifies how many times the learning algorithm will run through the entire training dataset. The unidirectional model has the ability to converge near 35 epochs, whereas, the bi-directional model doubles the training time nearing 70 epochs.

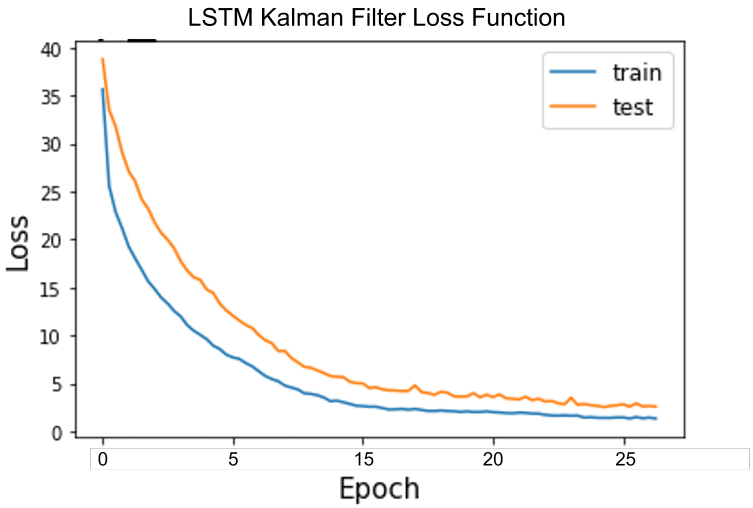

Figure 3.4: Unidirectional LSTM Kalman filter learning curve

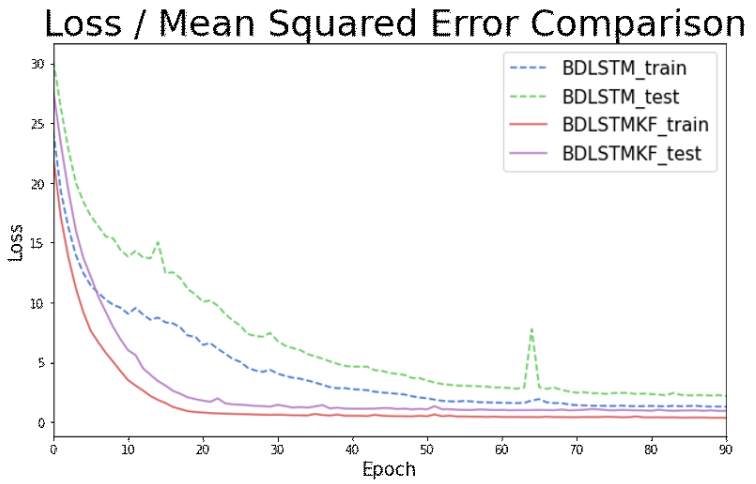

Figure 3.5: Unidirectional LSTM Kalman filter learning curve

Model	**Dataset Type**	**RMSE**	**MAE**	**Time To Converge**
Unidirectional	Validation	9.921	5.072	4.5h
	Test	9.554	4.797	
Bi-directional	Validation	9.482	5.750	5.5h
	Test	**9.192**	**4.564**	

Table 3.2: Comparison table of a LSTM KF and a BDLSTM KF

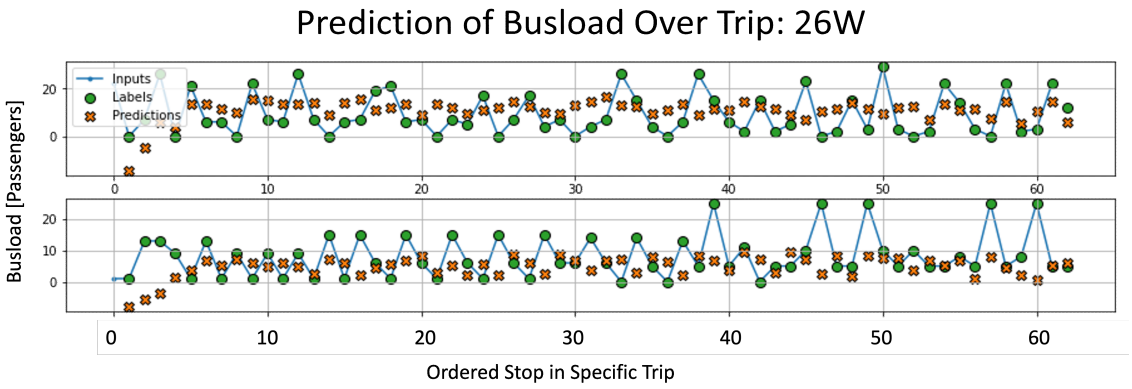

Figure 3.6: LSTM Kalman filter single-step forecast

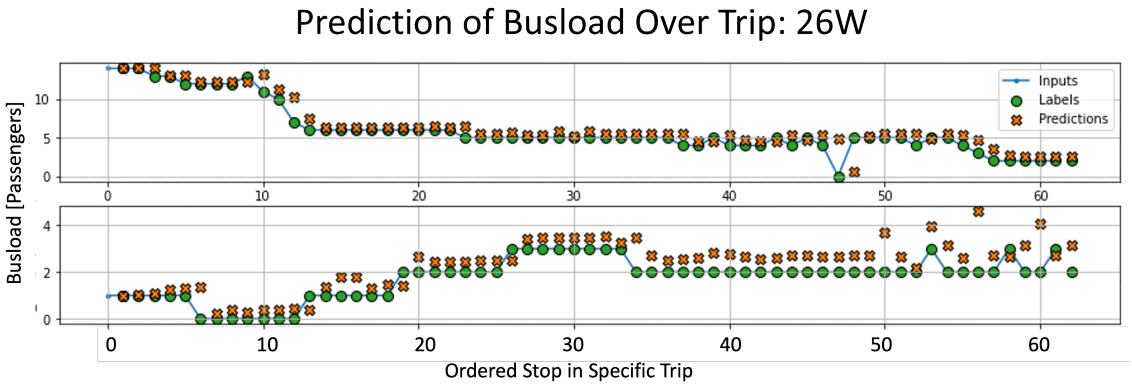

Figure 3.7: BDLSTM Kalman filter single-step forecast

Chapter 4

Conclusion

To incentivize a city to improve the efficiency of their transportation systems is not challenging when it reduces carbon emissions, removes congestion from the roads and improves the public's commuting experience. Passenger flow forecasting is a heavily studied topic by transportation researchers [17]. Predicting the passenger load on a given system provides necessary insight for transportation agencies in the areas of route planning, route optimization and overall understanding of ridership behaviours. Optimizing a given transportation system improves a city, making it more desirable to live and work in. While budgeting and streamlining a city's infrastructure is a continuous challenge, improving bus systems, removes cars from the road, reducing carbon emissions and congestion [9]. In summary, the typical application for passenger flow forecasting is to understand bus ridership behaviour, for route planning, and improvements to scheduling [17]. The short-term forecasting supports passengers through trip planning and daily operation [9].

This thesis investigates the use of a BDLSTM and BDLSTM Kalman filter to predict the busload at a given stop on a route in Laval, Canada. Large bus system datasets produced by the Société de Transport de Laval (STL), describes the behaviour of the citizens of Laval and the requirements on the bus system. Verified and validated data by the agency, informative features, and multi-year data, the proposed model was able to learn the behaviour of any proposed route in the bus network. An analysis of the data is presented, delving into the available features of the dataset and their relevance. To handle real-world data, filling missing values, the procedure and thought process was described. Presenting a comparison of the 2020 and 2021 data,

noting the significant difference between lockdown pandemic data and non-lockdown pandemic data. Specifically, the passenger flow increase from 2020 to 2021 and the polarization of data surround the daily peaks. Furthermore, an analysis of the proposed route, Route 26W, its feeders and how the route fluctuates over various time intervals: a day, week and month. Resources from STL are also presented, providing a visualization of the chosen route on a map.

Beyond data analysis, multiple models were used as a baseline to compare the effectiveness of preprocessing steps, pertinent features, and potential time horizons. A comparison of single-step and multi-step models was demonstrated for the prediction of passenger flow. The multi-step prediction included a simple one-shot multi-step prediction and a multi-step model with a time horizon of five stops in advance. As expected, single-step models significantly outperform multi-step models, however, multi-step models that predict less than one whole trip in advanced have improved performance. Comparing a vanilla LSTM to that of the proposed bi-directional method highlights the relationship between passenger flow and time. Shown in Table 2.3 using recurrent unit models can successfully learn complex spatial and temporal features. Using direct analysis, it is confirmed that using input data from the previous trip is sufficient to predict for the following trip.

In the first chapter, a lower complexity, single-step model initially proposed by [2] is adapted and tested. The paper by [2], introduces a novel configuration of a LSTM-based Kalman filter. The model boasts learning Kalman filter parameters based on historical data using LSTM models. It was observed that once trained, the Kalman filter has a lower computational burden than that of modern neural networks.Provided the intentions of the transportation agency partner, a solution requiring less computational power is appealing and beneficial. Based on the above work, the efficacy of bi-directional LSTMs, when applied to passenger flow prediction of bus systems, it is only reasonable to apply the improvements of the bi-directional to the LSTM-based Kalman filter. Proposing the bi-directional Long-Short Term Memory Neural Network Kalman filter not only improves the performance metrics, but it also improves convergence speed. On average, the LSTM KF converges 3x faster than any of the proposed models of the first chapter. This reduces the computational complexity required when updating and training the model with new data.

With ride sharing and mobility for hire applications, learning short-term dependencies and behaviours of transit users is a pressing tactic to maintain

and grow bus ridership. Adding value to the above work for the Société de Transport de Laval (STL), future work would include designing and developing a means to access real-time data from the agency and applying the proposed models to an online learning application.

Appendix A

Abbreviation Table

Abbreviation	Description
APC	Automatic passenger count
AVL	Automatic vehicle location
BDLSTM NN	Bi-directional Long Short-Term Memory Neural Network model
CNN	Convolutional Neural Network
DOW	Day of the week
GRU	Gated Recurrent Unit
GTFS	General Transit Feed Specification
HOD	Hour of the day
ITS	Intelligent Transportation Systems
KF	Kalman filter
LSTM NN	Long Short-Term Memory Neural Network model
MAE	Mean absolute error
MLP	Multilayer Perceptron model
MSE	Root mean squared error
OD	Origin Destination Matrix
RF	Random Forest Regression model
RMSE	Mean squared error
STL	Société de transport de Laval
SVR	Support Vector Regression model

List of References

[1] Zhiyong Cui, Ruimin Ke, Ziyuan Pu, and Yinhai Wang. Deep bidirectional and unidirectional lstm recurrent neural network for network-wide traffic speed prediction. *arXiv preprint arXiv:1801.02143*, 2018.

[2] Huseyin Coskun, Felix Achilles, Robert DiPietro, Nassir Navab, and Federico Tombari. Long short-term memory kalman filters: Recurrent neural estimators for pose regularization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5524–5532, 2017.

[3] Erik Jenelius. Data-driven metro train crowding prediction based on real-time load data. *IEEE Transactions on Intelligent Transportation Systems*, 21(6):2254–2265, 2019.

[4] Transportation Research Board. Information for Authors: a guide for preparing and submitting manuscripts for presentation at the TRB Annual Meeting and for Publication in TRB's Journal. Technical Report 2, Washington, D.C., April 2012.

[5] Lijuan Liu and Rung-Ching Chen. A novel passenger flow prediction model using deep learning methods. *Transportation Research Part C: Emerging Technologies*, 84:74–91, 2017.

[6] Zhe Zhang, Cheng Wang, Yueer Gao, Jianwei Chen, and Yiwen Zhang. Short-term passenger flow forecast of rail transit station based on mic feature selection and st-lightgbm considering transfer passenger flow. *Scientific Programming*, 2020, 2020.

[7] Kunpeng Zhang, Zijian Liu, and Liang Zheng. Short-term prediction of passenger demand in multi-zone level: Temporal convolutional neural network with multi-task learning. *IEEE Transactions on Intelligent Transportation Systems*, 21(4):1480–1490, 2020.

[8] Haiying Li, Yitang Wang, Xinyue Xu, Lingqiao Qin, and Hanyu Zhang. Short-term passenger flow prediction under passenger flow control using a dynamic radial basis function network. *Applied Soft Computing*, 83:105620, 2019.

[9] Ljubisa Sehovac and Katarina Grolinger. Deep learning for load forecasting: Sequence to sequence recurrent neural networks with attention. *IEEE Access*, 8:36411–36426, 2020.

[10] Catherine Lawson, Alex Muro, and Eric Krans. Forecasting bus ridership using a "blended approach". *Transportation*, 48, 04 2021.

[11] Shouwei Sha, Jing Li, Ke Zhang, Zifan Yang, Zijian Wei, Xueyan Li, and Xin Zhu. Rnn-based subway passenger flow rolling prediction. *IEEE Access*, 8:15232–15240, 2020.

[12] Siyu Hao, Der-Horng Lee, and De Zhao. Sequence to sequence learning with attention mechanism for short-term passenger flow prediction in large-scale metro system. *Transportation Research Part C: Emerging Technologies*, 107:287–300, 2019.

[13] Qian Chen, Wenquan Li, and Jinhuan Zhao. The use of ls-svm for short-term passenger flow prediction. *Transport*, 26:5–10, 03 2011.

[14] Rui Xue, Daniel Jian Sun, and Shukai Chen. Short-term bus passenger demand prediction based on time series model and interactive multiple model approach. *Discrete Dynamics in Nature and Society*, 2015, 04 2015.

[15] Jianyuan Guo, Zhen Xie, Yong Qin, Limin Jia, and Yaguan Wang. Short-term abnormal passenger flow prediction based on the fusion of svr and lstm. *IEEE Access*, 7:42946–42955, 2019.

[16] Xin Yang, Qiuchi Xue, Meiling Ding, Jianjun Wu, and Ziyou Gao. Short-term prediction of passenger volume for urban rail systems: A deep

learning approach based on smart-card data. *International Journal of Production Economics*, 231:107920, 2021.

[17] Florian Toqué, Mostepha Khouadjia, Etienne Come, Martin Trepanier, and Latifa Oukhellou. Short amp; long term forecasting of multimodal transport passenger flows with machine learning methods. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pages 560–566, 2017.

[18] Marc C Gelhausen, Peter Berster, and Dieter Wilken. A new direct demand model of long-term forecasting air passengers and air transport movements at german airports. *Journal of Air Transport Management*, 71:140–152, 2018.

[19] Toly Chen and Yu-Cheng Wang. Long-term load forecasting by a collaborative fuzzy-neural approach. *International Journal of Electrical Power Energy Systems*, 43(1):454–464, 2012.

[20] YORGOS J STEPHAN EDES, Panos G Michalopoulos, and Roger A Plum. Improved estimation of traffic flow for real-time control. *and Characteristics*, 7(9):28, 1980.

[21] Ma Guo-feng. Forecast of railway passenger traffic based on a grey linear regression combined model. *Computer Simulation*, 2011.

[22] Pengpeng Jiao, Ruimin Li, Tuo Sun, Zenghao Hou, and Amir Ibrahim. Three revised kalman filtering models for short-term rail transit passenger flow prediction. *Mathematical Problems in Engineering*, 2016, 2016.

[23] Florian Toqué, Etienne Côme, Mohamed Khalil El Mahrsi, and Latifa Oukhellou. Forecasting dynamic public transport origin-destination matrices with long-short term memory recurrent neural networks. In *2016 IEEE 19th international conference on intelligent transportation systems (ITSC)*, pages 1071–1076. IEEE, 2016.

[24] Miloš Milenković, Libor Švadlenka, Vlastimil Melichar, Nebojša Bojović, and Zoran Avramović. Sarima modelling approach for railway passenger flow forecasting. *Transport*, 33(5):1113–1120, 2018.

[25] Linchao Li, Yonggang Wang, Gang Zhong, Jian Zhang, and Bin Ran. Short-to-medium term passenger flow forecasting for metro stations using a hybrid model. *KSCE Journal of Civil Engineering*, 22(5):1937–1945, 2018.

[26] Yuandong Wang, Xuelian Lin, Hua Wei, Tianyu Wo, Zhou Huang, Yong Zhang, and Jie Xu. A unified framework with multi-source data for predicting passenger demands of ride services. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 13(6):1–24, 2019.

[27] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

[28] Yuxing Sun, Biao Leng, and Wei Guan. A novel wavelet-svm short-time passenger flow prediction in beijing subway system. *Neurocomputing*, 166:109–121, 2015.

[29] Quanchao Chen, Di Wen, Xuqiang Li, Dingjun Chen, Hongxia Lv, Jie Zhang, and Peng Gao. Empirical mode decomposition based long short-term memory neural network forecasting model for the short-term metro passenger flow. *PloS one*, 14(9):e0222365, 2019.

[30] Hao-Fan Yang, Tharam S. Dillon, and Yi-Ping Phoebe Chen. Optimized structure of the traffic flow forecasting model with a deep learning approach. *IEEE Transactions on Neural Networks and Learning Systems*, 28(10):2371–2381, 2017.

[31] Shengnan Guo, Youfang Lin, Ning Feng, Chao Song, and Huaiyu Wan. Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 922–929, 2019.

[32] Hao LIN, Leixiao LI, and Hui WANG. Survey on research and application of support vector machines in intelligent transportation system. *Journal of Frontiers of Computer Science and Technology*, 14(6):901–917, 2020.

[33] Zhuangwei Shi. Incorporating transformer and lstm to kalman filter with em algorithm for state estimation. *arXiv preprint arXiv:2105.00250*, 2021.

[34] Shidong Liang, Minghui Ma, Shengxue He, and Hu Zhang. Short-term passenger flow prediction in urban public transport: Kalman filtering combined k-nearest neighbor approach. *Ieee Access*, 7:120937–120949, 2019.

[35] Cheng Chen, Rui Xiong, Ruixin Yang, Weixiang Shen, and Fengchun Sun. State-of-charge estimation of lithium-ion battery using an improved neural network model and extended kalman filter. *Journal of Cleaner Production*, 234:1153–1164, 2019.

[36] Leto Peel. Data driven prognostics using a kalman filter ensemble of neural network models. In *2008 international conference on prognostics and health management*, pages 1–6. IEEE, 2008.

[37] Hamed HH Aly. A proposed intelligent short-term load forecasting hybrid models of ann, wnn and kf based on clustering techniques for smart grid. *Electric Power Systems Research*, 182:106191, 2020.

[38] Xijuan Song, Jijiang Huang, and Dawei Song. Air quality prediction based on lstm-kalman model. In *2019 IEEE 8th Joint International Information Technology and Artificial Intelligence Conference (ITAIC)*, pages 695–699. IEEE, 2019.

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

[40] Thomas Fischer and Christopher Krauss. Deep learning with long short-term memory networks for financial market predictions. *European Journal of Operational Research*, 270(2):654–669, 2018.

[41] Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681, 1997.

[42] Lihui Zhao, Yuhuan Chen, and Donald W Schaffner. Comparison of logistic regression and linear regression in modeling percentage data. *Applied and environmental microbiology*, 67(5):2129–2135, 2001.

[43] M.W Gardner and S.R Dorling. Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. *Atmospheric Environment*, 32(14):2627–2636, 1998.

[44] R. Lippmann. An introduction to computing with neural nets. *IEEE ASSP Magazine*, 4(2):4–22, 1987.

[45] Philip D Wasserman and Tom Schwartz. Neural networks. ii. what are they and why is everybody so interested in them now? *IEEE expert*, 3(1):10–15, 1988.

[46] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi. Understanding of a convolutional neural network. In *2017 international conference on engineering and technology (ICET)*, pages 1–6. Ieee, 2017.

[47] Shaoqiang Yu, Caiyun Shang, Yang Yu, Shuyuan Zhang, and Wenlong Yu. Prediction of bus passenger trip flow based on artificial neural network. *Advances in Mechanical Engineering*, 8, 10 2016.

[48] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. 1960.

[49] Greg Welch, Gary Bishop, et al. An introduction to the kalman filter. 1995.

[50] Qiang Li, Ranyang Li, Kaifan Ji, and Wei Dai. Kalman filter and its application. In *2015 8th International Conference on Intelligent Networks and Intelligent Systems (ICINIS)*, pages 74–77. IEEE, 2015.

[51] Simone Togni, Theoklis Nikolaidis, and Suresh Sampath. A combined technique of kalman filter, artificial neural network and fuzzy logic for gas turbines and signal fault isolation. *Chinese Journal of Aeronautics*, 34(2):124–135, 2021.

[52] Guy Revach, Nir Shlezinger, Xiaoyong Ni, Adria Lopez Escoriza, Ruud JG van Sloun, and Yonina C Eldar. Kalmannet: Neural network aided kalman filtering for partially known dynamics. *arXiv preprint arXiv:2107.10043*, 2021.

[53] MA Ganaie, Minghui Hu, et al. Ensemble deep learning: A review. *arXiv preprint arXiv:2104.02395*, 2021.