TransInfo
Tier I University Transportation Center

FINAL REPORT
Developing Predictive Border Crossing Delay Models

Date of report: April 2019

Lei Lin
Research Scientist, Goergen Institute for Data Science at the University of Rochester
Andrew Bartlett
Transportation Engineer, Niagara International Transportation Technology Coalition (NITTEC), and Ph.D. Candidate, University at Buffalo
Rishabh Chauhan
Graduate Research Assistant, University at Buffalo
Yunpeng (Felix) Shi
Graduate Research Assistant, University at Buffalo
Qian Wang
Teaching Assistant Professor, University at Buffalo
Adel W. Sadek
Professor, University at Buffalo
Director, Transportation Informatics Tier I University Transportation Center
Associate Director, Stephen Still Institute for Sustainable Transportation & Logistics

Prepared by:
Department of Civil, Structural & Environmental Engineering, University at Buffalo

Prepared for:
Transportation Informatics Tier I University Transportation Center
204 Ketter Hall
University at Buffalo
Buffalo, NY 14260

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.

4. Title and Subtitle Developing Predictive Border Crossing Delay Models	5. Report Date April 2019
	6. Performing Organization Code

7. Author(s) Lei Lin, Andrew Bartlett, Rishabh Chauhan, Yunpeng Shi, Qian Wang & Adel W. Sadek	8. Performing Organization Report No.

9. Performing Organization Name and Address Department of Civil, Structural and Environmental Engineering University at Buffalo 204 Ketter Hall Buffalo, NY 14260	10. Work Unit No. (TRAIS
	11. Contract or Grant No. DTRT13-G-UTC48

12. Sponsoring Agency Name and Address US Department of Transportation Office of the UTC Program, RDT-30 1200 New Jersey Ave., SE Washington, DC 20590	13. Type of Report and Period Covered Final: January 2014 – January 2019
	14. Sponsoring Agency Code

15. Supplementary Notes

16. Abstract

In recent years, and as a result of the continued increase in travel demand across the border coupled with the need for tighter security and inspection procedures after September 11, border crossing delay has become a critical problem with tremendous economic and social costs. This project aims at taking advantage of the wealth of data, now available thanks to the recent advances in sensing and communications, to develop predictive models which can be used to predict the delay a passenger car or a truck is likely to encounter by the time the vehicle arrives at the border. Specifically, the project first developed an Android smartphone application to collect, share and predict waiting time at the three border crossings. Secondly, models, based on state-of-the-art Machine Learning (ML) techniques, were developed for interval prediction of short-term traffic volume at the border; these models were then utilized to determine optimal staffing levels at the border. Finally, by taking advantage of Bluetooth, border delay data recently collected at the three Niagara Frontier borders, the project developed deep learning models for the direct prediction of border delay. The suite of models and tools developed under this work have the potential to revolutionize border crossing management, balance traffic load at the three crossings, and help travelers avoid significant border delays.

17. Key Words: Border Crossing; Waiting Time; Crowd Sourcing; Short Term Traffic Volume Prediction Model; Android; Particle Swarm Optimization; Extreme Learning Machine; Prediction Interval; Staffing Plan; Deep Learning; Multilayer Perceptron; Convolutional Neural Networks; Long Short-Term Memory Recurrent Neural Networks; Gated Recurrent Unit Recurrent Neural Networks.	18. Distribution Statement No restrictions. This document is available from the National Technical Information Service, Springfield, VA 22161

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	80 pages	

Insert your own project cover page here

Acknowledgements

The authors would like to acknowledge the Niagara International Transportation Technology Coalition for providing data utilized in this research. They also would like to acknowledge the *Transportation Research Board*, and the *Transportation Research – Part C* journal, for publishing the material compiled in this report.

Disclaimer

The contents of this report reflect the views of the authors, who are responsible for the facts and the accuracy of the information presented herein. This document is disseminated under the sponsorship of the U.S. Department of Transportation's University Transportation Centers Program, in the interest of information exchange. The U.S. Government assumes no liability for the contents or use thereof.

TABLE OF CONTENTS

TABLE OF FIGURES

LIST OF TABLES

EXECUTIVE SUMMARY

In recent years, and as a result of the continued increase in travel demand across the border coupled with the need for tighter security and inspection procedures after September 11, border crossing delay has become a critical problem with tremendous economic and social costs. This project aims at taking advantage of the wealth of data, now available thanks to the recent advances in sensing and communications, to develop predictive models which can be used to predict the delay a passenger car or a truck is likely to encounter by the time the vehicle arrives at the border. The project is building on initial work done by UB researchers, which broke down the problem into two steps: (1) the short-term prediction of the hourly traffic volume at the border; and (2) the development of queueing models which predict delay given knowledge of the predicted volume from step 1.

In this project, UB TransInfo researchers completed the following three additional tasks: (1) the development of an Android smartphone application to collect, share and predict waiting time at the three Niagara Frontier border crossings; (2) the development of Machine Learning (ML) models for interval prediction of short-term traffic volume, which were then utilized to determine optimal staffing levels at the border; and (3) the development of deep learning models for predicting border delay directly from Bluetooth data collected at the three Niagara Frontier borders.

The Android app developed under this project is called the **T**oronto **B**uffalo **B**order **W**ait **T**ime (TBBW) app. The innovative app offers the user three types of waiting time estimates: (1) current waiting times collected at the crossings; (2) historical waiting times; and (3) future waiting time predicted for the next 15 minutes and updated every five minutes. For the current waiting time, the app can provide both the data collected by border crossing authorities as well as user-reported or "crowd-sourcing" data shared by the community of the app's users. Reporting of the data could be done either manually or automatically through a GPS tracking function provided by the smartphone. For the historical waiting time, the app provides statistical charts and tables to help users choose the crossing with the likely shortest wait time. Future waiting times are predicted by a real-time stepwise traffic delay prediction model which consists of a short-term traffic volume forecasting model and a multi-server queueing model (these were developed by UB researchers in previous research). To validate the prediction functionality of the app, its predictions were compared against real-world delay measurements for the entire month of May, 2014. The comparison showed that the model offered predictions with a mean absolute difference of 9.22 minutes. When considering only delays that are greater than 10 minutes, the model has a mean absolute difference of only 6.95 minutes.

For interval prediction, the study improved on a hybrid machine learning model based on Particle Swarm Optimization (PSO) and Extreme Learning Machine (ELM) neural network. The improved PSO-ELM models are developed for an hourly border crossing traffic dataset and compared to other state-of-the-art models. The results show that the improved PSO-ELM can always keep the mean PI length the lowest, and guarantee that the PI coverage probability is higher than the corresponding PI nominal confidence, regardless of the confidence level assumed. The study also proposes a comprehensive optimization framework to make staffing

plans for border crossing authority based on bounds of PIs and point predictions. The results show that for holidays, the staffing plans based on PI upper bounds generated much lower total system costs, and that those plans derived from PI upper bounds of the improved PSO-ELM models, are capable of producing the lowest average waiting times at the border.

Finally, the study developed deep learning models for predicting border delay directly from blue tooth delay data collected at the Niagara Frontier borders. Four deep learning techniques were utilized: Multilayer Perceptron (MLP), Convolutional Neural Networks (CNN), Long Short-Term Memory Recurrent Neural Networks (LSTM-RNN), and Gated Recurrent Unit Recurrent Neural Networks (GRU-RNN). The prediction accuracies of these models were evaluated by computing the Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and R-squared. The results suggest high-level accuracy of the deep learning techniques in predicting future traffic delays at the border crossings, with MAEs less than 3.5 minutes in predicting delays for up to 60 minutes into the future. However, no one deep learning technique emerged as a clear winner among others in predicting the delays.

Key Words: Border Crossing; Waiting Time; Crowd Sourcing; Short Term Traffic Volume Prediction Model; Android; Particle Swarm Optimization; Extreme Learning Machine; Prediction Interval; Staffing Plan; Deep Learning; Multilayer Perceptron; Convolutional Neural Networks; Long Short-Term Memory Recurrent Neural Networks; Gated Recurrent Unit Recurrent Neural Networks.

INTRODUCTION

Due to the continuous travel demand increase, coupled with tighter security and inspection procedures after September 11, border crossing delay has become a critical problem. As reported by the Ontario Chamber of Commerce, border crossing delay causes an annual loss of approximately $268.45 million for New York State. For the whole U.S., the cost is much higher (OCC, 2005). According to a press release in 2008 given by the then U.S. Transportation Secretary, Mary E. Peters, border delays cost Canadian and US businesses as many as 14 billion dollars in 2007 (USDOT, 2007). Besides their negative economic impacts, border delays, and the associated idling of traffic awaiting inspection, have a significant environmental cost. A 10-year study by Lwebuga-Mukasa *et al.* (2002) showed a positive relationship between increased commercial traffic volume at Peace Bridge border crossing between downtown Buffalo, New York and Fort Erie, Ontario, and the increased use of asthma health care.

To address these issues, transportation authorities have recently begun to provide travelers with information about current border crossing delays. This is the case for example in the Buffalo-Niagara region, for example, where the Niagara International Transportation Technology Coalition (NITTEC) has been providing such information to the public for years. In the early years, the waiting time was obtained based on very rough and approximate estimates of queue length. More recently, NITTEC is using blue-tooth identification technology to provide more accurate delay estimates to motorists, and the information is now updated every five minutes.

Regardless of the method however, there is an inherent limitation associated with providing just the *current* border delay, which is likely to be quite different from the *future* wait time that the travelers would experience by the time they arrive at the border. This is especially true if there is a significant lag between the time when travelers need to act on the information provided and the time of their arrival at the border. If the *future* waiting time can be predicted, it would be more informative for travelers and businesses to select the time to depart and the route to pursue. Moreover, with predicted border crossing delays, intelligent routing algorithms could be developed to optimally direct and route border-destined traffic in a fashion that would minimize the overall system travel time or the negative impacts on the environment.

This project aims at taking advantage of the wealth of data, now available, to develop predictive models which can be used to predict the delay a passenger car or a truck is likely to encounter by the time the vehicle arrives at the border. The project is building on previous work done by UB researchers, which broke down the problem into two steps: (1) the short-term prediction of the hourly traffic volume at the border; and (2) the development of queueing models which predict delay given knowledge of the predicted volume from step 1. For short-term prediction of hourly traffic volume, UB researchers have previously developed several short-term traffic volume forecasting models (e.g., Lin et al., 2012; Lin et al., 2013; Lin et al., 2014a). In addition, UB researchers developed transient multi-server queueing models, which take the predicted traffic volume and predicts the likely delay (Lin et al., 2014b).

In this project, we build on our previous work just described and undertake three additional major tasks to improve on border crossing delay prediction, and to provide the tools needed to

better manage the border and improve traffic operations there. These additional tasks focused on: (1) the development of an Android smartphone application to collect, share and predict waiting time at the three Niagara Frontier border crossings; (2) the development of Machine Learning (ML) models for interval prediction of short-term traffic volume (i.e., predicting an interval within which predicted delay is expected to fall with a certain probability); these intervals were then utilized to determine optimal staffing levels at the border; and (3) the development of deep learning models for predicting border delay directly from Bluetooth data collected at the three Niagara Frontier borders.

Besides the Introduction and the Conclusions section, this report is divided into three major sections, each dedicated to discussing one of the three research tasks mentioned above. It should be noted that the first two sections of the report represent a compilation of the material previously published by the authors in the following two papers, Lin et al. (2015) and Lin et al. (2018). The third section is based on material included in Chauhan's M.S. thesis submitted to the University at Buffalo in April 2019.

AN ANDROID SMARTPHONE APP FOR BORDER CROSSING WAIT TIME

The extremely data-rich environment of today provides an excellent opportunity for data mining and for extracting useful insights to help improve transportation systems' efficiency. One more factor that deserves consideration is the emergence of social media applications using smartphones which allow people to easily create, share and exchange information. For example, Waze is a community-based traffic and navigation app, acquired by Google in 2013, where drivers can share real-time traffic and road information, saving travel time, gas and money on their daily commute.

In this part of the study, an Android smartphone application (app) called the Toronto Buffalo Border Wait Time (TBBW) was developed, to allow for sharing waiting time among travelers of the three Niagara Frontier border crossings, namely the Lewiston-Queenston Bridge, the Rainbow Bridge, and the Peace Bridge. Three types of waiting times are offered based on users' preferences, including the current waiting time, the historical waiting time, and the future waiting time predicted by a real-time traffic delay prediction model.

For the current waiting time, the app can provide both the data collected by the border crossing authorities and the user-reported or "crowd-sourcing" data shared by the community of users of the app. For the historical waiting time, the app provides statistical charts and tables to help users choose the crossing with the likely shortest waiting time. Moreover, the app can also provide future border waiting time for the next 15 minutes with an updating frequency of five minutes. The future waiting times are predicted by a stepwise delay prediction model that consists of a short-term traffic volume prediction model for predicting the incoming traffic flow and a queueing model for predicting border inspection resulted delays.

PURPOSE AND SCOPE

The Niagara Frontier International Border includes three main bridges connecting Western New York, U.S. to Southern Ontario, Canada, namely the Lewiston-Queenston Bridge, the Rainbow Bridge, and the Peace. Figure 1 shows the yearly traffic volume going through Peace Bridge (one of the three crossings) from 2009 to 2013. As can be seen, for each direction to U.S. or to Canada, there are more than two million passenger vehicles and around 500,000 commercial vehicles going through Peace Bridge every year. This highlights the very large market of potential users and the great potential effect of this app. Besides that, thanks to the predictive capabilities of TBBW, it can help border crossing and customs agencies determine the optimal staffing level and the number of inspection booths needed to keep the border delay below a certain threshold.

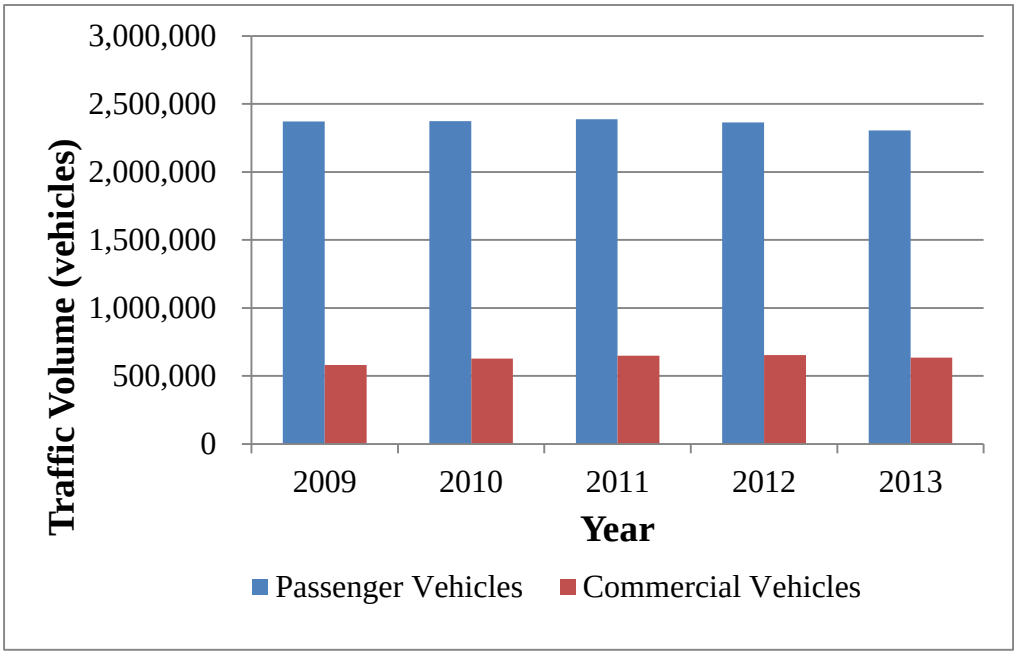


Figure 1a. Yearly traffic volume through Peace Bridge to U.S.

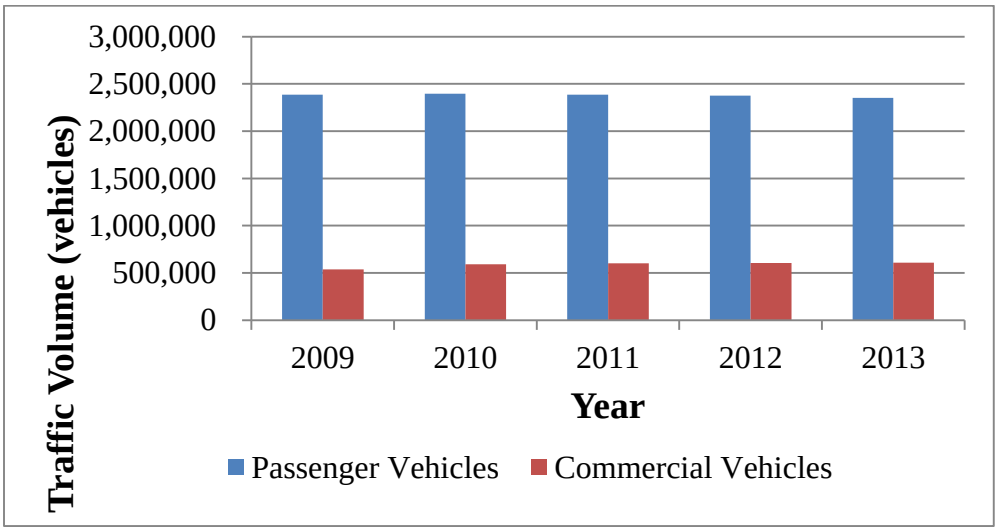


Figure 1b Yearly traffic volume through Peace Bridge to Canada

Figure 1. Yearly traffic volume of Peace Bridge from 2009 to 2013

The TBBW app was designed to collect, share and estimate border crossing waiting time by taking advantage of multiple data sources and advanced traffic prediction methods. Figure 2 summarizes the characteristics of TBBW (shown in the green color), in comparison with the existing border crossing delay dissemination method (shown in the blue color). As can be seen, TBBW provides several options for border crossing delay estimates including, user-reported or "crowd-sourcing" wait time, historical, and future wait time, in addition to the current waiting time reported by the authorities. Travelers and border management authorities can then make better decisions based on this information.

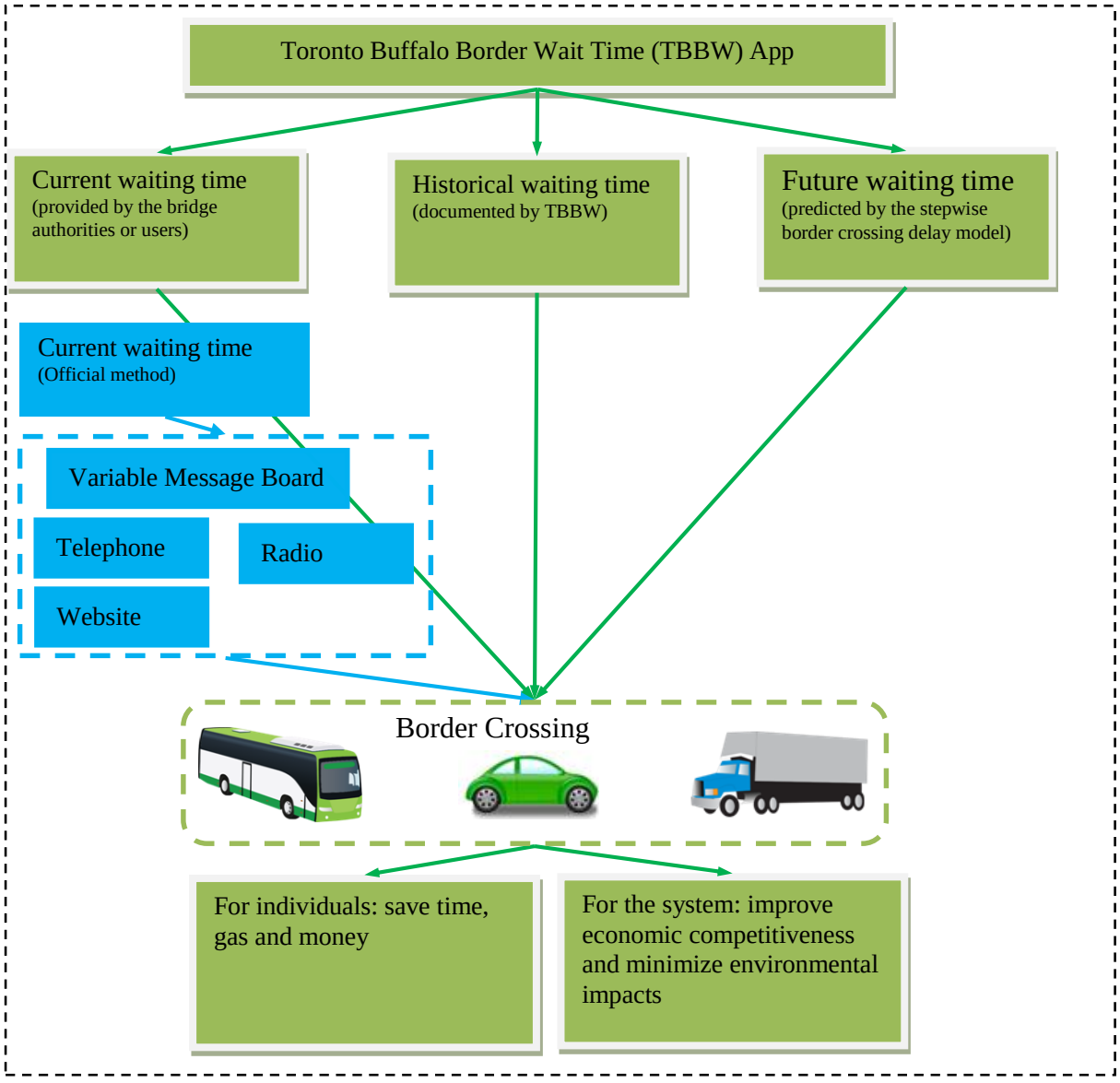


Figure 2. Comparison of TBBW with the Other Ways to Share Border Waiting Time

DATASETS

Two types of data are used to develop the TBBW app. The first dataset contains the hourly traffic volume data collected at the Peace Bridge since 2003. This is used as the input to develop the stepwise border delay prediction model and to predict the future waiting times. The second

dataset captures the current waiting times collected and maintained by the border crossing authorities. Such data are used as one source of the current waiting times provided by the app. In addition, they are stored for historical data analysis and also used as the ground truth to assess the performance of the border delay prediction model. All data are available for download from the websites maintained by the Peace Bridge authority and Niagara Falls Bridge Commission (Peace Bridge, 2014; Niagara Falls Bridge Commission, 2014).

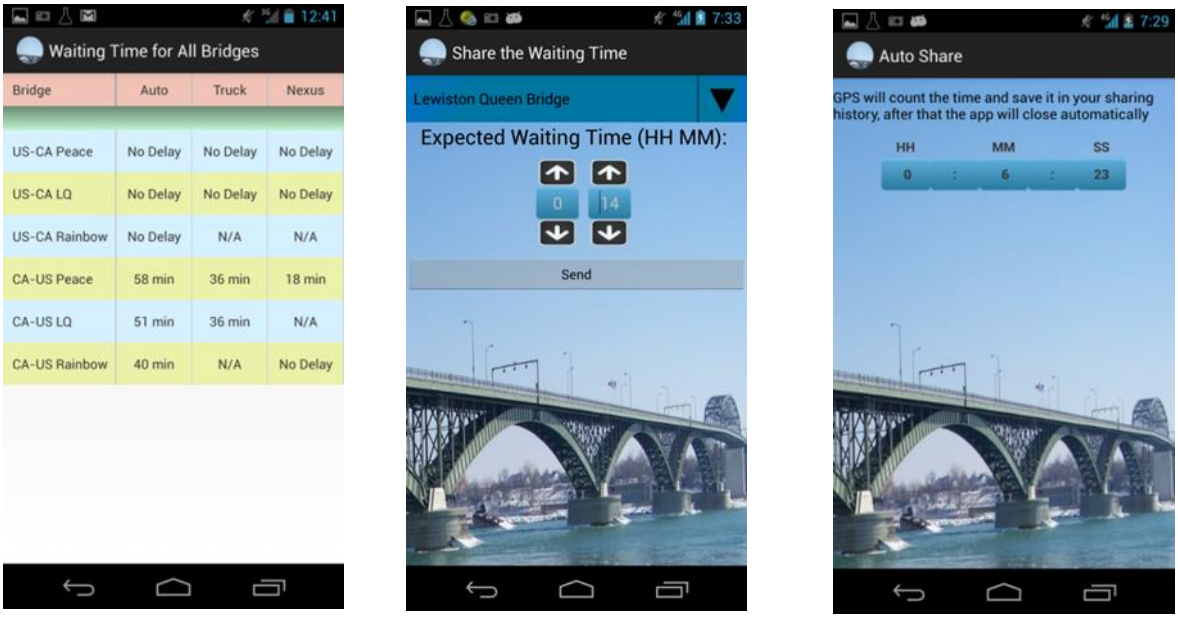
INNOVATIVE FEATURES

The TBBW app was developed on the Android platform, the most popular mobile operating system used in the U.S. TBBW is available from the Google Play store. The developed TBBW app is innovative in terms of its ability: (1) to share current waiting time; (2) to store and analyze historical waiting time; and (3) to predict future waiting time, as described below.

Sharing Current Waiting Time Function

The app employs two ways to collect current waiting time information. The first way involves downloading the waiting time data from the websites maintained by the Buffalo and Fort Erie Public Bridge Authority and the Niagara Falls Bridge Commission. At the time of the study, the current waiting time for Peace Bridge and Lewiston Queen Bridge were provided and updated every five minutes, and for Rainbow Bridge, it was updated every one hour. The information is collected and uploaded in real time to the app as shown in

Figure 3a.



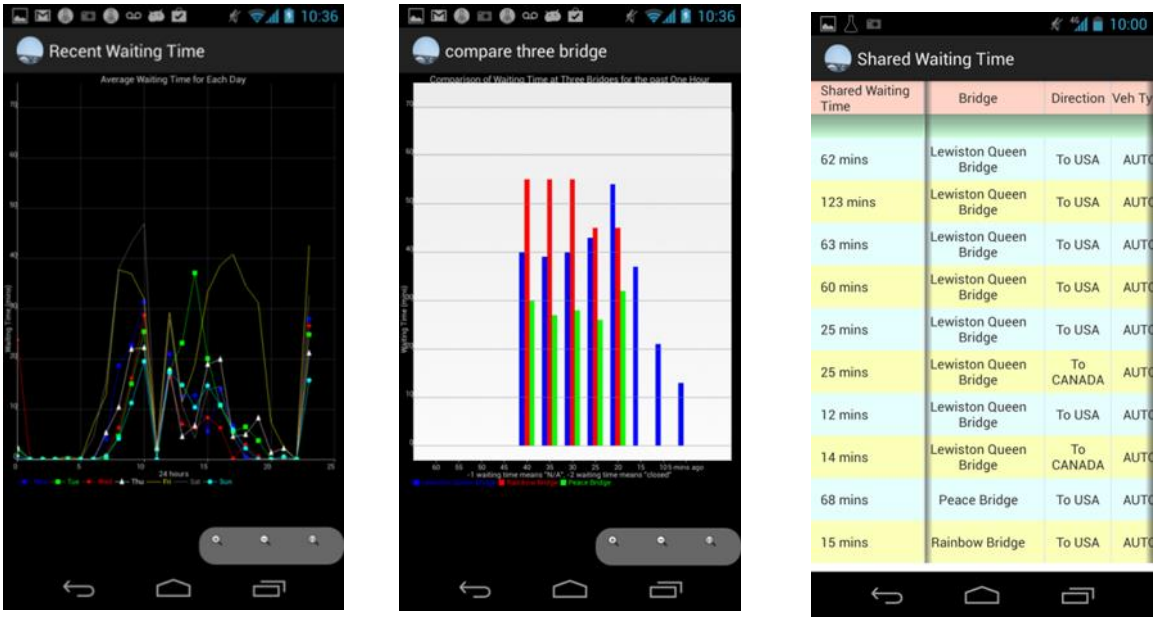
a) Official Website b) Manual Share c) Automatic Share (GPS)

Figure 3. Three ways to share current waiting time

Because the official current waiting time data is lagged (particularly for the Rainbow Bridge where it is only updated every hour), the app also provides a second way to collect the current waiting time data utilizing crowd sourcing ideas. Specifically, users are allowed to report their *experienced* border crossing delays that can be then processed and broadcasted to other users for their benefits, called crowd sourcing. The same concept has been widely applied in other traffic information sharing apps, such as the pre-mentioned Waze and the bus arrival time sharing app Tiramisu (Zimmerman et al., 2011). In TBBW, users can share their waiting times by manually inputting the data as shown in Figure 3b. They can also choose to automatically share their waiting times through their GPS-enabled smartphones as shown in
Figure 3c. This option is necessary because if users are driving, it is unsafe and illegal to manually input waiting time.

Utilizing Historical Waiting Time Function

Mining and analyzing historical border crossing waiting time data in a proper manner can provide additional insight to travelers. In TBBW, three types of graphs and charts are created based on an underlying historical waiting time database.



a) Average Waiting Times for Each Day of Week for Each Bridge

b) Comparison of Waiting Times at Three Bridges for the Past Hour

c) Waiting Times Sharing History by the registered user himself/herself

Figure 4. Three ways to utilize historical waiting time

As can be seen in Figure 4a, for each bridge, the average waiting times for each day of week are calculated and shown in one chart. This is the long term trend based on the historical data of the past month. The TBBW app also allows the users to compare the waiting times of the three bridges, based on the historical data of the past one hour, as shown in Figure 4b. Finally, because users may want to make decisions based on their own previous experiences, registered users can view their waiting times as another reference as shown in Figure 4c.

Predicting Future Waiting Time Function

Finally, in addition to current and historical analyses of wait times, the app is designed to *predict* the likely waiting time in the next 15 minutes (this estimate is also updated every 5 minutes). Predicting is based on utilizing the stepwise border crossing delay prediction model previously developed by the authors in prior research (Lin et al., 2012; Lin et al., 2013; Lin et al, 2014a). The following section will briefly describe this model and its prediction performance.

Stepwise Delay Prediction Model: The stepwise border delay prediction model is composed of two sequential modules as shown in Figure 5 below. The first module is designed to predict the *traffic volume* arriving at the border crossings for each time period (Lin et al, 2012; Lin et al, 2013; Lin et al., 2014a). Note that the economic indicators and weather and incident shown information in Figure 5 were not used in the current version of the short term traffic volume prediction model; we hope to address this in our future research). Given the predicted traffic volume as input, the second model estimates the corresponding waiting time by solving a transient multi-server queueing problem (Lin et al, 2014b).

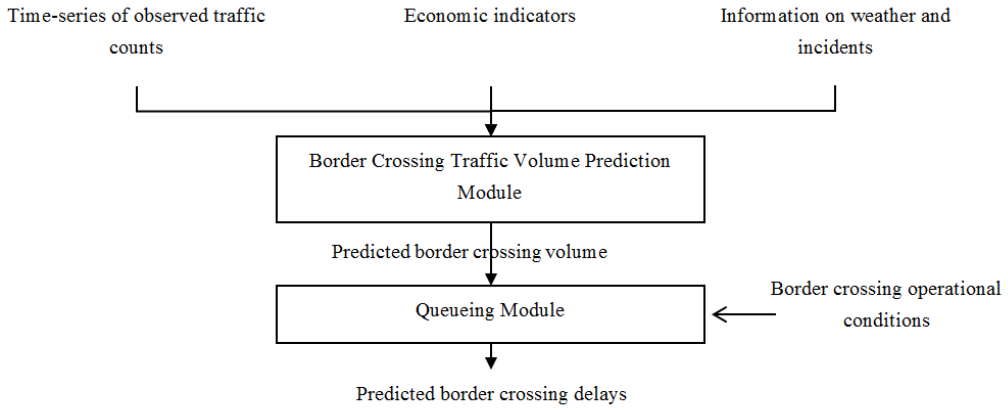


Figure 5. Framework of the stepwise delay prediction model

Border Crossing Traffic Volume Prediction Module: Three short-term *traffic volume* prediction methods have been previously tested by the authors on the border crossing traffic volume data for the Peace Bridge, namely seasonal Autoregressive Integrated Moving Average (SARIMA), support vector regression (SVR), and an enhanced spinning network (SPN) (Lin et al., 2012; Lin et al., 2013; Lin et al., 2014a). In this app, SARIMA is chosen as the prediction method because of its easiness of implementation and its moderate computational cost. As previously reported by the authors, for a testing dataset with 1,905 hourly traffic volume points, the mean absolute percentage error (MAPE) was found to be equal to 16.38% (Lin et al., 2013). It needs to be noted here that the short-term traffic volume prediction module was built using data collected from the Peace Bridge, due to the fine temporal resolution available (i.e., on the hourly basis). The traffic volumes for the other bridges were only available to the study on a daily basis at the time, and were thus deemed not sufficient for accurate waiting time prediction.

Transient Multi-server Queueing Module: In the authors' previous work, 700 observations of vehicular inter-arrival times and 571 observations of the service times (i.e. inspection time) were

collected from December 19, 2011 to January 10, 2012 at the Peace Bridge. Based on the collected observations, it was determined that the distribution of the inter-arrival times is best captured by an exponential distribution and that the service time distribution is best described as an Erlang distribution with order equal to 2 (Lin et al., 2014b). With these findings, an $M/E_{k=2}/n$ queueing model was developed to capture the queueing process at the border crossing. The transient solution of this multi-server queueing model was then derived and used to predict the border crossing waiting time.

Because the TBBW app requires that the predicted wait time be updated every five minutes, the predicted hourly traffic volume was split into a finer resolution (e.g., a five-minute resolution) before they were used for border wait time prediction by the queueing models. With the inter-arrival distribution known, this was done using the inverse cumulative function of the inter-arrival exponential distribution $F(x) = 1 - e^{-\lambda x}$, where λ is the predicted hourly volume or arrival rate.

Other input requirements of the queueing model included the number of inspection booths. However, the number of open inspection stations is typically not available ahead of time. To solve the problem, our approach at the moment involves running the queueing model for different numbers of open stations (1 to 10 in this study), and trying to estimate how many stations are actually open. Other venues to be explored in the near future are information offered by users or directly by the U.S. Customs and Border Protection. The readers can find more detailed information about these queueing models in the reference (Lin et al., 2014b).

Prediction Results: The TBBW interface of the predicted waiting time for passenger vehicles from Canada to U.S. through the Peace Bridge is shown in Figure 6.

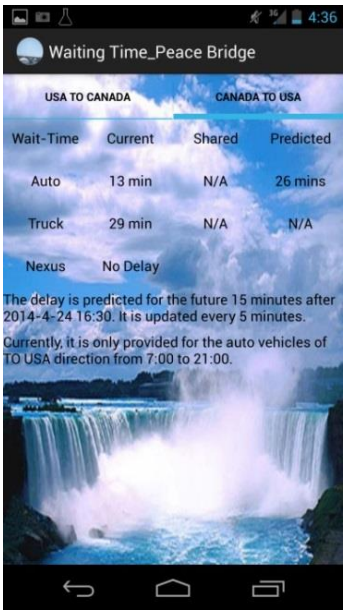


Figure 6. Predicted border crossing waiting time

In order to test the prediction performance of the stepwise delay prediction model, the research compared the predicted waiting times with the historical waiting times recorded by the border

authorities from 7:00 AM to 9:00 PM for each day of the whole month of May, 2014. Because the future waiting time is updated every five minutes, there should be a total of 5,580 predicted values for the month. However, because of several missing data points from the field observations (e.g., when the server was down and the official waiting time was recorded as "N/A"), a total of 3,103 observations were deemed valid for assessing the prediction model's performance.

The mean absolute difference (minutes) between the predicted waiting times and the officially recorded waiting times is shown in Table 1. As can be seen, the mean absolute difference for the whole dataset is 9.22 minutes. After checking the officially recorded waiting times, we find that there were a total of 2,363 data points where the wait times was recorded as being equal to 0 minutes, and the remaining 740 points had delays greater than or equal to 10 minutes. After discussions with the border crossing authorities, it was revealed that their practice was to report any wait time which was less than 10 minutes as 0 minutes delay. Given this, and in order to provide for a true evaluation of the predictive model accuracy, the testing dataset was split into two groups. The first group (2,363 data points) had an official reported delay of 0 minutes, which meant that the delay could be anywhere between 0 and 10 minutes. For that group, the mean absolute difference between the model's predictions and the officially reported delay times was as high as 9.94 minutes (it should be clear now that that absolute error is exaggerated, since the actual delay could have been anywhere between 0 and 10 minutes). The second group included points where the officially reported wait time was greater than or equal to 10 minutes. For that second group, the mean absolute difference was only 6.95 minutes.

Table 1. Prediction Performance of the Stepwise Delay Prediction Model

Data Group	**Number of data points**	**Mean Absolute Difference (minutes)**
Whole Dataset	3,103	9.22
Officially Recorded Waiting Time = 0 minutes (denoting less than 10 minute delays)	2,363	9.94
Officially Recorded Waiting Time >=10 minutes	740	6.95

For a more disaggregate view of the performance of the delay prediction model, the predicted waiting times and the historical waiting times for the peak hours 18:00-20:00 on April 22, 2014 are compared and shown on Figure 7. As can be seen in Figure 7, the mean absolute difference between the predicted waiting times and the observations is about 6.6 minutes. Most of the time, the difference is within 10 minutes, except for 19:40 for which the difference is around 20 minutes. This is most probably the result of the opening of additional inspection stations at that time without the model being aware of that (the reader may recall that there is currently no easy way for the app to discern the actual number of inspection stations open; it is hoped that in the future such information may be obtained from the Customs and Border Protection agencies). Another reason could be that the historical waiting time detected by the Bluetooth technology is

lagging in time, since the Bluetooth technology provides an estimate of the delay at the time a vehicle had joined the queue some time prior to the reporting time (that time is actually equal to the time it took the vehicle to exit the system).

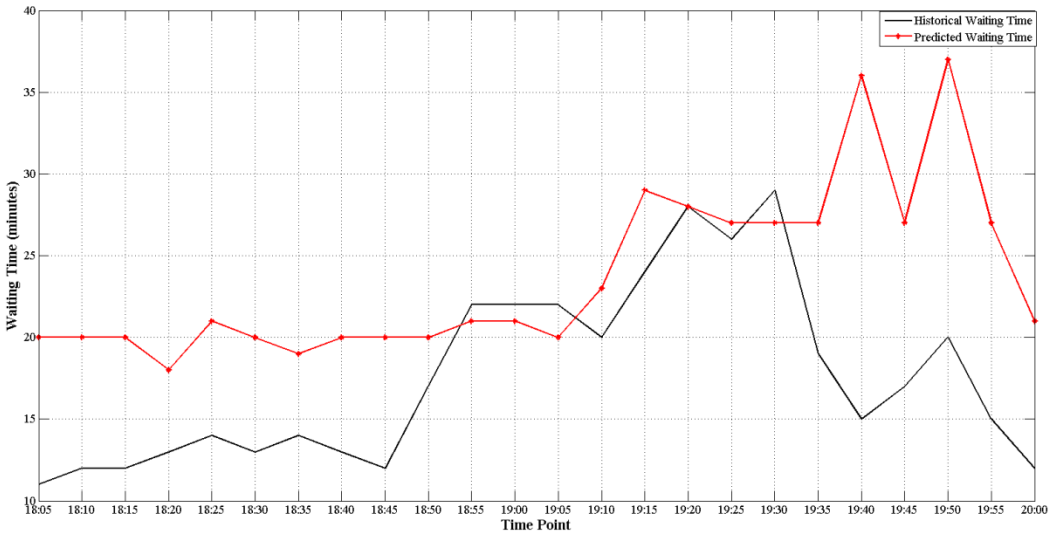


Figure 7. Prediction Performance for the peak hours of 18:00-20:00 on April 22, 2014

Front-End Service Processes of Toronto Buffalo Border Wait Time (TBBW) app

Figure 8 shows the details of the TBBW front-end service processes behind the innovative functions described above.

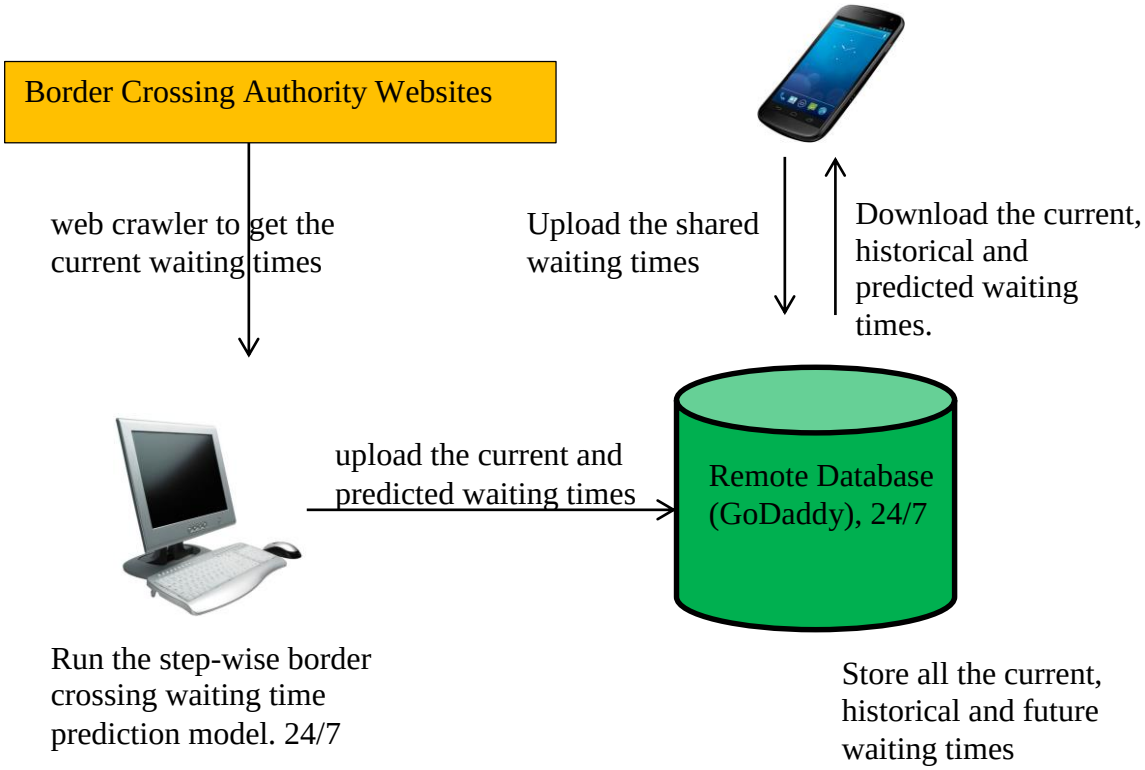


Figure 8. Flow Chart of TBBW Front-End Service Processes

As can be seen, there is a local computer which continuously runs the web crawler program to download the current waiting times from the official border crossing authority websites. That computer also continuously runs the step-wise border crossing waiting time for 24 hours per day. The current and predicted waiting times are then uploaded to the remote database which is hosted by GoDaddy (GoDaddy, 2014), an internet domain registrar and web hosting company . Unlike the local computer, this remote server can be guaranteed to be running all the time, which is important for the app users, to allow them to interact with the server at any time. The app users can upload their own experienced waiting times to the remote server, and can also download different kinds of waiting times from it. The historical graphs and charts are generated at the client side (i.e., the android smart phone).

RISKS AND CHANLLENGES

This section will summarize the risks and challenges encountered while developing the app. Some of those challenges have been addressed, while others are left for future work.

The Need for More Data

A critical piece of information for wait time prediction which is missing at this point is the number of open lanes or inspection booths. Although the delay prediction model can estimate the number of open lanes, it would be better and more accurate if the real value were to be provided by the U.S. Customs and Border Protection agencies.

Crowd Sourcing

As with any contribution-based crowd sourcing information system, a risk exists of low motivation to participate and of abuse (Steinfeld et al., 2011). To overcome this problem for TBBW, one can design a set of reward and penalty rules on the basis of the registration and login function. For example, when users share their border crossing waiting time with others, they can get some virtual points, and every period of time the user with the highest rank may be rewarded. Abuse can also be prevented through penalties. For example, users who intentionally share wrong border crossing waiting times can be identified and filtered by setting a threshold for the difference between the value provided by the user and a "best" estimate based on a combination of the officially reported waiting time and the average waiting time from other users. Users who abuse the system may also be restricted from sharing information.

GPS Location

Some privacy concerns may arise regarding the ability to share waiting time in an automatic fashion through the GPS location sharing function. To address this, the TBBW app was designed so that it does not store any of the users' GPS locations data; these data are only used to calculate the distances of the travelers from the borders and their speed, so an approximate waiting time can be estimated.

CONCLUSIONS AND FUTURE WORK

This part of the study introduced an android app TBBW which combines sophisticated transportation models with emerging mobile computing technologies to solve the wait time border crossing problem. The performance of the prediction model was assessed by comparing its predictions to those reported by the authorities for month of May, 2014. The comparison demonstrated that the predictions are quite accurate, with a mean absolute difference of only 6. 95 minutes for delays greater than or equal to 10 minutes.

Several future directions are suggested by the current work. First, at the moment, the TBBW app is only predicting the delay for the next 15 minutes, it would be better to make the prediction horizon a user-specified value. Second, although the app is currently designed for the Niagara International Frontier Borders, it can also be easily extended and applied to other US-Canadian or US-Mexico borders. The app can even be extended to predict airport delay, and delay at many other similar queueing systems, in the future.

A HYBRID MACHINE LEARNING MODEL FOR INTERVAL PREDICTION OF SHORT-TERM BORDER CROSSING TRAFFIC VOLUME

Prediction Interval versus Single-value Prediction

Most previous studies on short-term traffic volume prediction have focused on a single-value prediction of the traffic volume, and relied almost exclusively on the prediction error when assessing the effectiveness of a modeling approach (Karlaftis and Vlahogianni, 2011). Given the nonlinearity of traffic flow, traditional single-value prediction approaches are unfortunately almost guaranteed to result in high prediction errors, which could have significant negative impact on the effectiveness of traffic management schemes. In such a case, an accurate and reliable prediction interval (PI) with upper bound and lower bound would be more useful to traffic operators.

For forecasting applications in various domains, the use of PIs is quite useful because PIs try to capture the uncertainty associated with predicting the next observation, by asserting that the next observation will be contained within a given interval with a given probability. PIs are particularly useful in operational contexts where it is desired to make staffing plans. Jongbloed and Koole (2001) showed that point prediction of the call volume to a call center cannot guarantee the desired service quality at peak hours (calls need to be answered quickly on average in between 10-20 seconds). To address this, the researchers computed the PIs for the arrival rates and adapted the workforce for the call center based on the results.

Similarly, Kortbeek et al. (2015) introduced PIs to develop flexible staffing policies that would allow hospitals to dynamically respond to their fluctuating patient population by employing float nurses. PIs also have applications in the energy industry, especially in regard to wind-generated electricity. For example, owing to the variability of wind production, PIs can be used to construct contracts for supply in an auction market (Pinson et al., 2007). Within the transportation domain, PIs have been used for bus and freeway travel times prediction (Khosravi et al., 2011), who argued that PIs of travel times are more meaningful because of the underlying complex traffic processes, and given the data quality used to infer travel time. There are also a few studies that have generated PIs for short-term traffic volume forecasting (Kamarianakis et al., 2005; Guo et al., 2014; Zhang et al., 2014), and for real-time traffic speed uncertainty quantification (Guo and Williams, 2010).

From a technical standpoint, PIs can be derived in a number of ways, but with significant difference in terms of interpretation. The first approach is a frequentist approach, which assumes that the observation is itself fixed but the interval itself is random and related to the sample dataset (Cryer and Chan, 2008). A PI with a probability of 95% asserts that were the experiment to be conducted many times, about 95% of those would contain the unknown observation. Autoregressive–moving-average (ARMA) model is one of the well-known models based on this approach. The second approach is a Bayesian approach. Different from the frequentist approaches, Bayesian techniques assume the observation is random and has a probability distribution. In that case then, the PI is assumed to be fixed and is in fact derived from a posterior

distribution, estimated from a prior distribution and from previous observations. The Kalman Filter family of models is the classical example for the Bayesian approach.
Among the key challenges of generating PIs is how to quantify the variance. The assumption of constant variance, e.g. ARMA model, compromises the forecasting ability (Zhang et al., 2014). One might reasonably expect variances to vary along with a mean in a time series, especially in short-term traffic flow data. Zhang et al. (2014) pointed out that the variance of traffic flow becomes large during an accident, congestion, or other abnormal situations that last for a certain period. This is known as time-dependent conditional heteroskedasticity which means that the variance, conditional on past data, propagates according to some model. Generalized Autoregressive Conditional Heteroskedasticity (GARCH) has been proposed to capture the time-dependent variance (Bollerslev, 1986). Kamarianakis et al. (2005) applied GARCH to provide PIs for 7.5-min average traffic flow data.

Zhang et al. (2014) further pointed out that the GARCH model ignores the empirically important asymmetric effect in traffic data. Instead, they applied the Glosten-Jagannathan-Runkle GARCH (GJR-GARCH) proposed by Glosten et al. (1993), to allow the conditional variance to respond differently to the past negative and positive innovations. A hybrid model was proposed by the researchers to provide point predictions as well as PIs: spectral analysis for periodic trend, ARIMA for deterministic part and GJR-GARCH model for volatility.

In this part of the study, in this paper, we apply and improve a hybrid machine learning model called PSO-ELM for interval prediction of short-term traffic volume. Extreme learning machine (ELM) is a novel feedforward neural network with advantages such as, extremely fast learning speed and superior generalization capability (Huang et al., 2006). Furthermore, particle swarm optimization (PSO), a well-known heuristic and population based optimization method, is applied to adjust the parameters of ELM in an efficient and robust way to minimize a multi-objective function. The multi-objective function introduces two quantitative criteria called *reliability* and *sharpness* to evaluate the PIs. Simply speaking, the PSO-ELM model treat an interval as two points to be estimated. The weights of the neural network ELM are learned and optimized through PSO to contain the observations with a desired frequency (reliability) and to be as narrow as possible (sharpness). In this machine learning approach, the conditional variance is not a concern anymore.

The PSO-ELM model has been applied to wind power prediction (Wan et al., 2014). Based on the characteristics of the short-term traffic prediction problem, in this study, we improve on the PSO-ELM previously used by Wan et al. (2014), by making the parameters update in an on-line approach, and also by redefining the calculation of reliability. We then compare the improved PSO-ELM model against: (1) the original PSO-ELM of Wan et al (2014); and (2) the hybrid model by Zhang et al. (2014). The comparison is made utilizing an hourly short-term traffic volume dataset from the Peace bridge, one of the busiest US-Canadian borders. As will be shown in the paper, the results show that the improved PSO-ELM models can always keep the mean PI length the lowest, while guaranteeing that the PI coverage probability is higher than the corresponding PI nominal confidence level like 90%, 95%, or 99%. To the best of our knowledge, this is the first attempt to apply neural network based models and multi-objective optimization to interval prediction of short-term traffic volume forecasting.

Furthermore, another main contribution of our study is that we propose a comprehensive optimization framework to make staffing level plans for border crossing authorities, based on the interval predictions and point predictions of short-term traffic volume. Although there have been a few studies that looked at the optimal staffing level problem for border crossings (Yu et al., 2016; Lin et al., 2014b), none of them considered future traffic predictions in developing the border staffing level plans. Combining our previous studies of a transient multi-server queueing model for border crossings (Lin et al., 2014b), the framework we propose in this study makes optimal staffing level plans for a border crossing authority, based on the different types of short-term traffic predictions considered in this study. These are the PI upper or lower bounds from (1) the improved PSO-ELMs; (2) Zhang et al. (2014) model; and (3) point predictions from Zhang et al. (2014) model.

Experiments are then designed and repeated so that the border crossing port is operated under different optimal staffing plans with real observed traffic demand from the morning period from 7:00-12:00 from two typical days; (1) a holiday (President's Day, 02/17/2014); and (2) a normal weekday (02/10/2014). The hourly average waiting times and the total system costs (operation cost and traveler waiting cost) are recorded and compared. As we will be elaborated on later in the report, our results show that during holiday time periods, making plans based on upper bounds of PIs from the improved PSO-ELMs generated the lowest average waiting times. Moreover, applying the plans from upper bounds of PIs generally produced much lower total system costs comparing to those using PI lower bounds. For the normal Monday, the staffing plans developed based on PI upper bounds resulted in no delay what so ever, but the total system costs were slightly higher than the costs of the plans developed based on point predictions from Zhang et al. (2014). For both the holiday and normal Monday scenarios, among the staffing level plans developed based on PI lower bounds, the ones from the improved PSO-ELMs performed the best, with an acceptable level of service and system costs close to the staffing plans developed based on point predictions.

The rest of this section of the report is organized as follows. The next sub-section provides a detailed introduction of the PSO-ELM model, the multi-objective optimization function utilized and the improvements this study introduced to the original PSO-ELM. This is followed by a description of the dataset used. The results of the interval prediction using the improved PSO-ELM are then presented and compared against the original PSO-ELM and the Zhang et al. (2014) model. Following this, the PIs and point predictions are utilized to develop border crossing optimal staffing plans; the performances of the different plans developed are then compared, in terms of total system cost and average waiting times. Finally, the study's conclusions are discussed and recommendations for future research are provided.

METHODOLOGY

Prediction interval

A Prediction Interval (PI) provides a lower bound and an upper bound for the future target value y_i given an input X_i. The probability that the future targets can be enclosed by the PIs is called the Prediction Interval Nominal Confidence (PINC):

$$PINC = 100(1-\alpha)\% \quad \text{Equation 1}$$

where,

the usual value of α could be 0.01, 0.05 or 0.10.

Obviously, the selection of α in PINC will impact the PIs. The PIs under different PINC levels can then be represented as follows:

$$I_i^\alpha = [L_i^\alpha,\ U_i^\alpha]$$

Equation 2

where,

L_i^α and U_i^α denote the PI lower and upper bounds of target value y_i given α .

PI evaluation criteria

The reliability and sharpness metrics are introduced in the PSO-ELM (Wan et al.,2014) to evaluate the PIs. The normalized values of these metrics are useful in the minimization of the multi-objective function, as will be discussed later.

Reliability: Reliability is regarded as a major property for validating PI models. Based on the PI definition, the future targets y_i are expected to be covered by the constructed PIs with a probability equal to the PINC $100(1-\alpha)\%$. However, the actual PI Coverage Probability (PICP) may be different from the pre-defined PINC, calculated for the dataset, as follows:

$$PICP = \frac{1}{N}\sum_{i=1}^{N} D_i^\alpha$$

Equation 3

where,

$N$is the dataset size;

D_i^α is a dummy variable equal to 1, if the real observation y_i is within the PI I_i^α, otherwise, $D_i^\alpha = 0$.

The PSO-ELM model tries to force the calculated PICP to be as close as possible to PINC. The absolute average coverage error (AACE) is applied as the reliability evaluation criterion as shown in Equation 4 .

$$R^\alpha = abs(PICP - PINC)$$

Equation 4

Naturally, the smaller the R^α, the higher the reliability.

Sharpness: Reliability considers only coverage probability. If reliability were to be utilized as the only model evaluation criterion, high reliability could be easily achieved by increasing the width of the PI, rendering the PI useless in practice (since a wide PIs may not provide accurate quantifications of uncertainties involved in the real-world processes (Wan et al., 2014; Zhang et al., 2014)). A sound PI model should be able to provide reliable, *as well as* sharp intervals. Sharpness thus should be considered as a second criterion, alongside reliability.

Suppose the width of PI I_i^α is represented by WI_i^α. The width measures the distance between the upper bound and lower bound through

$$WI_i^\alpha = U_i^\alpha - L_i^\alpha$$

Equation 5

The sharpness of PI I_i^α , denoted by S_i^α, can thus be calculated as

$$S_i^\alpha = \begin{cases} w_1 \alpha WI_i^\alpha + w_2[L_i^\alpha - t_i], & if\ y_i < L_i^\alpha \\ w_1 \alpha WI_i^\alpha, & if\ y_i\ \epsilon\ I_i^\alpha \\ w_1 \alpha WI_i^\alpha + w_2[t_i - U_i^\alpha], & if\ y_i > U_i^\alpha \end{cases}$$

Equation 6

where,

w_1 and w_2 are two user defined weights.

Equation 6 considers the width of the PI WI_i^α weighted by w_1 for all three different scenarios. Additionally, when the true value y_i is lower than the lower bound, or higher than the upper bound, an extra penalty calculated by the distance of that point to the bound and adjusted by w_2 is included. This is to prevent the possibility that the PIs become too "narrow". In practical applications, the w_1 and w_2 need to be carefully tuned.

The sharpness of PIs over the entire dataset can be calculated by taking the average of the normalized S_i^α, represented by $S_{i,norm}^\alpha$, using Equation 7and Equation 8:

$$S^\alpha = \frac{1}{N}\sum_{i=1}^{N} S_{i,norm}^\alpha$$

Equation 7

where,

$$S_{i,norm}^\alpha = \frac{S_i^\alpha - \min(S_i^\alpha)}{\max(S_i^\alpha) - \min(S_i^\alpha)}$$

Equation 8

Hybrid PSO-ELM model

Extreme Learning Machine: ELM is a single hidden-layer feedforward neural network proposed by Huang et al. (2006). It has become very popular in recent years. Previous studies have shown that ELM training is extremely fast because of the simple matrix computation, and can always guarantee optimal performance (Huang et al., 2006; Wan et al., 2014). In addition, ELM can overcome many limitations of traditional gradient based NNs training algorithms, such as finding local minima, overtraining and so on. The basic principle of ELM is as follows:

Given a short-term traffic volume dataset, suppose the traffic volume at time step i is x_i, using the traffic volumes from the previous time steps, we can construct a feature vector $X_i = [x_{i-n+1}, \dots, x_{i-1}, x_i]$ and the corresponding target value y_i, e.g. it could be the traffic volume in the next time step. Finally, suppose we have a dataset with N distinct samples $\{(X_i, y_i)\}_{i=1}^N$, where the inputs $X_i \in R^n$ and the targets $y_i \in R^m$, the following equation can be used to find the optimal structure of neural network ELM and approximate the N samples with zero error:

$$f_K(X_i) = \sum_{j=1}^{K} \beta_j \varphi\left(a_j * X_i + b_j\right) = y_i\,, i = 1, \dots, N$$

Equation 9

where,

K is the number of hidden neurons;

$\varphi(.)$ is the activation function (e.g. a sigmoid function);

$a_j = [a_{j1}, a_{j2}, \dots, a_{jn}]^T$ represents the weight vector connecting the j^{th} hidden neuron and the input neurons;
b_j denotes the bias of the j^{th} hidden neuron;
$\varphi(a_j * X_i + b_j)$ is the output of the j^{th} hidden neuron with respect to the input X_i;
$\beta_j = [\beta_{j1}, \beta_{j2}, \dots, \beta_{jm}]^T$ represents the weights at the links connecting the j^{th} hidden neuron with the m output neurons.

For simplicity, Equation 9 can be represented as:

$$H\beta = Y$$

Equation 10

where,

$$H = \begin{bmatrix} \varphi(a_1 * X_1 + b_1) & \cdots & \varphi(a_K * X_1 + b_K) \\ \vdots & \ddots & \vdots \\ \varphi(a_1 * X_N + b_1) & \cdots & \varphi(a_K * X_K + b_K) \end{bmatrix}_{N \times K}$$

Equation 11

Each row of H is the outputs at the K hidden neurons for input X_i, $i = 1, \dots, N$. β is the matrix of weights at the links connecting hidden layer and output layer and Y is the matrix of targets, respectively represented as

$$\beta = \begin{bmatrix} \beta_1 \\ \vdots \\ \beta_K \end{bmatrix}_{K \times m}$$

Equation 12

$$Y = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix}_{N \times m}$$

Equation 13

Note that in ELM, the weights a_j and biases b_j for the K hidden neurons are randomly chosen, and are not tuned during the training process. This is very different compared to the traditional gradient-based training algorithm of NNs. In this way, ELM can dramatically save the learning time. The training of ELM is simply to find β^* to minimize the objective function,

$$\|\mathbf{H}(a_1^*, \dots, a_k^*, b_1^*, \dots, b_k^*)\beta^* - T\| = \min_{\beta} \|\mathbf{H}(a_1, \dots, a_k, b_1, \dots, b_k)\beta - Y\|$$

Equation 14

where,

$\|.\|$ is the function to calculate the Euclidean distance.

Finally, a unique solution of β^* can be derived through a matrix calculation:

$$\beta^* = H^{\dagger}Y$$

Equation 15

where,

$H^{\dagger}$ is the Moore-Penrose generalized inverse of the hidden layer output matrix H, which can be derived through the singular value decomposition (SVD) method.

It is worth mentioning that to apply the ELM model for interval prediction, the target value y_i in the training dataset $\{(X_i, y_i)\}_{i=1}^{N}$ needs to be replaced with a pair of target bounds $\hat{y}_i^-$ and $\hat{y}_i^+$, which can be produced by slightly increasing or decreasing the original y_i by $\pm\rho\%, 0 < \rho < 100$. So after transformation, the training dataset for interval prediction using ELM should be $\{(X_i, \hat{y}_i^-, \hat{y}_i^+)\}_{i=1}^{N}$. Then by adjusting the number of output neurons, the ELM can directly generate the lower and upper bounds under a certain PINC level. A structure of an ELM model for interval prediction is shown in Figure 9.

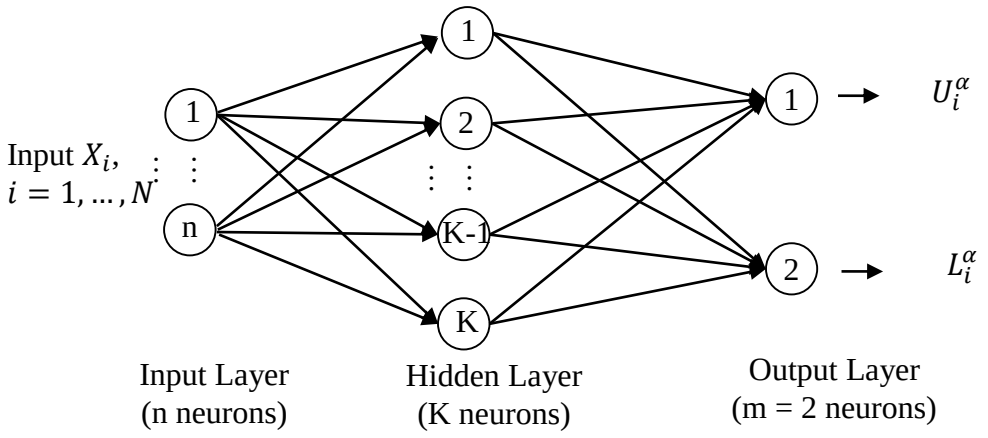


Figure 9. A structure of ELM model for interval prediction.

Multi-objective function and Particle Swarm Optimization: In this study, the Particle Swarm Optimization (PSO) algorithm was used to further adjust the parameters of ELM, by minimizing a multi-objective optimization function which considers both reliability and sharpness of PIs. Specifically, a multi-objective optimization function was constructed to achieve the trade-off between those two important criteria. Recall that in ELM, the weights β at the links connecting the hidden layer and output layer are the only parameters which need to be learned, and which can be calculated as in Equation 15 above. However, the weights β can be further tuned through PSO, in order to minimize the following multi-objective function F.

$$\min_{\beta} \; F = \gamma R^{\alpha} + \lambda S^{\alpha}$$

Equation 16

where,

R^{α} denotes the reliability as calculated by Equation 4

S^{α} denotes sharpness as calculated by Equation 7.

γ and λ are trade-off weights for the reliability and sharpness metrics defined by the user.

Some researchers have pointed out that reliability is the primary feature reflecting the correctness of the PIs, and hence should be given priority (Wan et al., 2014).

PSO is a population based heuristic optimization inspired by the social behavior of bird flocking or fish schooling (Kennedy, 2011). It is an extremely simple but efficient algorithm with fast convergence speed for optimizing a wide range of functions. In this study, it is applied to further

adjust the weights β of ELM model in order to minimize the multi-object function in Equation 16. A brief introduction of PSO is given next.

Suppose the total population of particles in the S-dimensional search space is N_P, the position of the i^{th} particle can be represented with a vector $P_i = [P_{i1}, P_{i2}, \dots, P_{iS}]^T$. Once the algorithm starts learning, each particle moves around in the space with a speed v_i. The algorithm keeps running until the user defined number of iterations N_{iter} or a sufficiently good fitness has been reached (e.g., change of object values from two continuous runs is less than a user-defined threshold). For each iteration, the velocity and position of each particle are updated as following equations:

$$v_i = wv_i + c_1 r_1 \left(P_i^b - P_i\right) + c_2 r_2 (P_g^b - P_i)$$

Equation 17

$$P_i = P_i + \phi v_i$$

Equation 18

for $i = 1,2, \dots, N_P$.

where,

w is the inertia weight;

c_1, c_2, ϕ are user-defined constants;

r_1 and r_2 are random numbers within $[0, 1]$;

P_i^b is the best position for the particle i that generated the smallest objective function value from the previous iterations;

P_g^b is the best position among particles in the global swarm that produced the smallest objective function value from the previous iterations.

Note that the velocity of the i^{th} particle for the next iteration is a function of three components: the current velocity, the distance between its own previous best position P_i^b and the current position, and the distance between the global best position P_g^b and its current position. The initialized positions of the particles are generated randomly, based on the weights β^* using Equation 15 , and the speed of the particles are randomly produced with an interval $[-v_{max}, v_{max}]$, v_{max} is a S-dimensional vector. For each iteration, the updated position of each particle will be taken as the adjusted weights β. The corresponding value from Equation 16 will be used to decide the P_i^b and P_g^b. After the algorithm stops, the P_g^b will be the finalized weights β for ELM model. The flow chart in The flow chart of PSO-ELM algorithm for interval prediction. Figure 10 shows the complete learning process of the hybrid PSO-ELM algorithm for interval prediction.

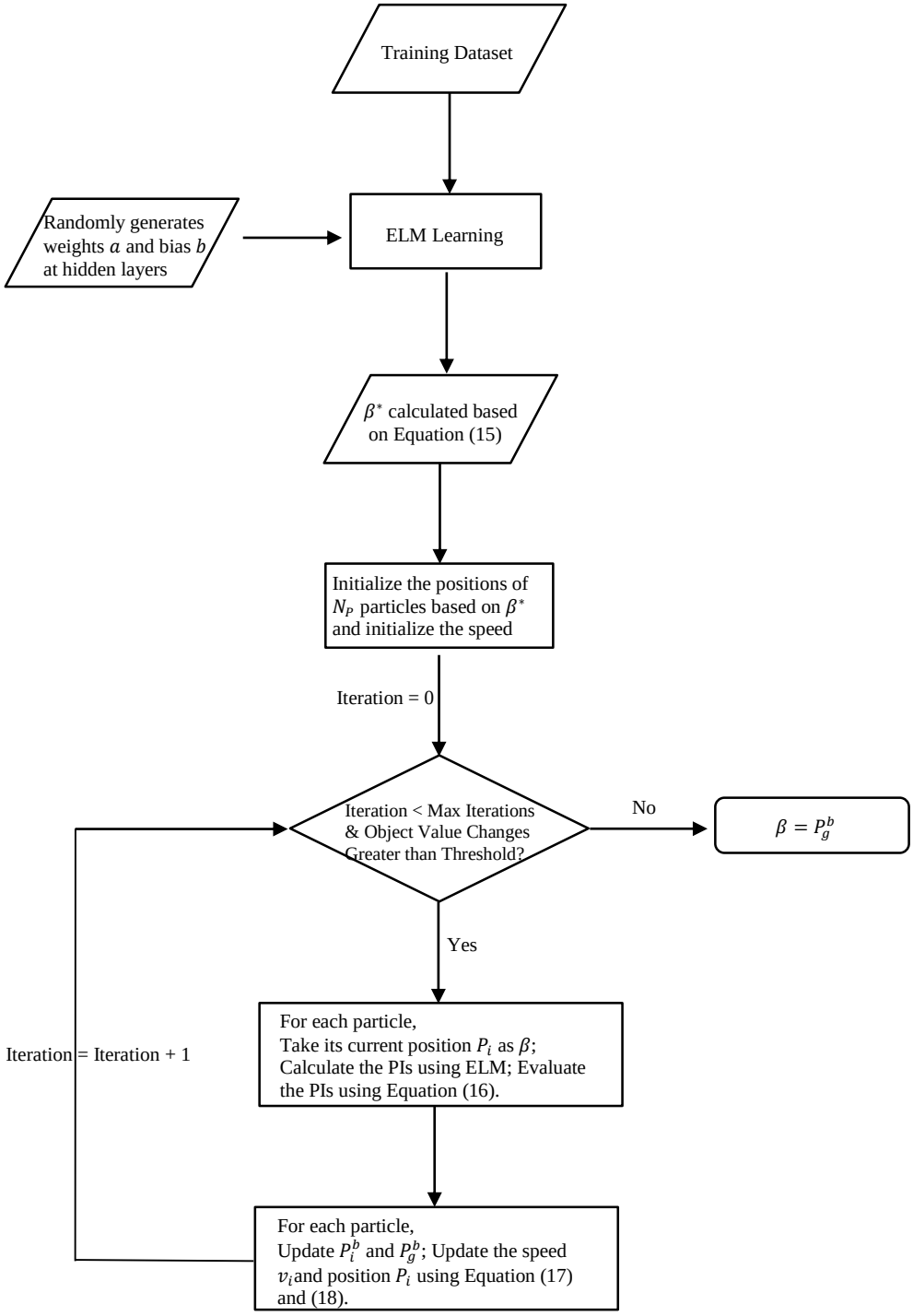


Figure 10. The flow chart of PSO-ELM algorithm for interval prediction.

As shown in Figure 10, given a dataset $\{(X_i, \hat{y}_i^-, \hat{y}_i^+)\}_{i=1}^N$, ELM algorithm can be applied first to get an optimal β^* using Equation (15). After generating the initial positions of particles on the

basis of β^*, PSO algorithm aims to find the global best position P_g^b that can minimize the multi-objective function in Equation 16. PSO algorithm continues until the maximum number of iterations is reached or until the change in the value of the multi-objective function from one iteration to the next is less than a predefined threshold. The final global best position P_g^b is then taken as the values of β to be used by ELM to make interval predictions.

Improved PSO-ELM

In this study, we improved the original PSO-ELM by making the following two refinements for short-term traffic volume prediction task. First, instead of learning the PSO-ELM model parameters based on the training dataset and then keeping them unchanged, the PSO-ELM model can be regularly updated in an on-line approach. Every period of time l, we use the newly archived traffic volume data to adjust the model parameters. For example, when the hourly traffic volumes of the next day are available, they are imported to represent a new training dataset, and the PSO-ELM model is retrained.

The second improvement is related to the PI evaluation criteria. As pointed out by Zhang et al. (2014), the lack of definite agreement on the indices of PI assessment creates a relatively new research challenge in traffic forecasting. Zhang et al. (2014) applied the PICP and the mean PI length (MPIL) which is the average distance between the upper bounds and lower bounds of the intervals to evaluate the PIs. Guo et al. (2014) proposed kickoff percentage and width to flow ratio. The kickoff percentage is the ratio of traffic flow observations lying outside of PIs, and the width to flow ratio is the average of width to flow ratios for all the PIs.

In the original PSO-ELM model, the reliability Equation 4 and the multi-objective optimization Equation 16 encourage the PICP to be as close as possible to PINC. However, it will be much better if the PSO-ELM model can generate a PICP higher than PINC, and at the same time it can also keep the PIs as narrow as possible. Therefore, we change the way of quantifying the reliability of interval prediction by simply revising Equation 4 as follows and apply it to Equation 16.

$$R^{\alpha} = PINC - PICP \qquad \text{Equation 19}$$

Therefore to minimize the objective value in Equation 16, the PSO will find a set of parameters for ELM to make PICP as high as possible and also to keep the PIs narrow.

Benchmark models

To assess the performance of the PSO-ELM model and the improvement we made, the PSO-ELM models are compared against the hybrid model by Zhang et al. (2014). This section will briefly introduce the hybrid model by Zhang et al. (2014). For further details, the reader is referred to Lin et al., 2018.

Hybrid Model by Zhang et al. (2014): Zhang et al. (2014) decomposed the traffic data into three components: a periodic trend, a deterministic component and a volatility component. In the hybrid model they proposed, spectral analysis and ARMA model were applied to capture the first

two components. What makes their model unique, however, is the volatility component where they assume that the white noise e_t is conditionally heteroscedastic instead of constant,

$$e_t = z_t\sqrt{h_t}$$

Equation 20

where,

$\{z_t\}$ is a sequence of i.i.d. random variables with zero mean and unit variance. The conditional distribution of e_t is also assumed to be i.i.d. with zero mean and a variance of h_t.

In the Generalized Autoregressive Conditional Heteroskedasticity (GARCH) model, h_t is calculated as follows:

$$h_t = a_0 + \sum_{i=1}^{p} \beta_i h_{t-i} + \sum_{j=1}^{q} \alpha_j e_{t-j}^2$$

Equation 21

which shows that the conditional variance is a linear combination of the lagged condition variance and past model sample variances.

Zhang et al. (2014) pointed out that GARCH model can capture the phenomenon observed in traffic datasets in which a large past value of sample variance tends to be followed by another large sample variance. However, it ignores the asymmetric effect in transportation system that travelers may response differently to sudden decrease or increase in travel time. To address this, the researchers applied the Glosten-Jagannathan-Runkle GARCH (GJR-GARCH) model to capture the asymmetric volatility effect:

$$h_t = a_0 + \sum_{i=1}^{p} \beta_i h_{t-i} + \sum_{j=1}^{q} (\alpha_j e_{t-j}^2 + \gamma_j e_{t-j}^2 I_{t-j})$$

Equation 22

where,

$$I_{t-j} = \begin{cases} 1 & if\ e_{t-j} < 0 \\ 0 & if\ e_{t-j} \geq 0 \end{cases}$$

Equation 23

MODELING DATASET

For this study, we considered a part of the hourly passenger car traffic volume dataset collected at the Peace Bridge, focusing on traffic entering the US from Canada. The size of the dataset is 900 observations, collected between 7:00 to 21:00 from January 1st to March 1st in 2014. The first 600 data points (01/01/2014-02/09/2014) are used to train the models (i.e., the training dataset), while the rest (02/10/2014-03/01/2014) are used to test the models.

Note that in this part of the study, our objective is to test and compare the interval prediction performances for different models, therefore a smaller dataset is much easier for use to explore the reasons behind why some of the predicted points were outside of PIs. For example, because the time period of the dataset falls within the inclement winter season in the area of the study, we could check the historical snow precipitation records for the points outside of PIs. Furthermore, the season of popular sports events (e.g., the Buffalo Sabres games) is from Oct 2013 to April

2014, which also falls within the time period considered for this period, allows us to explore other possible reasons for predictions lying outside PIs.

MODEL DEVELOPMENT AND RESULTS

Model development

First, the original, off-line PSO-ELM model was implemented in Matlab. There are quite a few hyper-parameters to tune in a PSO-ELM model. These include: (a) the multi-objective function parameters; (b) the ELM parameters; and (c) the PSO parameters. For the multi-objective function related parameters, target values in the training dataset were slightly increased and decreased by 5% in order to construct the target bounds $\{(\hat{y}_i^-, \hat{y}_i^+)\}_{i=1}^N$ (based on our experiments, this value doesn't have too much impact on the results).

As mentioned earlier, for the weights w_1 and w_2 in the sharpness calculation in Equation 6, the optimal values need to be tuned carefully for different PINC levels. When the PINC was set to 90%, the weights w_1 and w_2 were set to 6 and 0.1, respectively. When he PINC was 95%, 11 and 0.1 were used, and when PINC was 99%, 12 and 0.1 were chosen. In general, we found that larger values of w_1 generated a narrower interval, and larger w_2 made the intervals wider. Because α in Equation 6 decreased from 0.10 to 0.01 when PINC changed from 90% to 99%, we needed to increase w_1 in order to keep the predicted interval tight. Finally, for the multi-objective function Equation 16, the weights of reliability and sharpness γ and λ were both set to 1 for all three PINC levels. This means that in our study, both criteria are regarded as equally important.

For the ELM part, recall that the weights a_j and biases b_j for the K hidden neurons are randomly chosen, and are not tuned during the training process; instead, the weights β^* at the links connecting the hidden layer and output layer are calculated using Equation 15. The values of the only two parameters that could be calibrated or tuned, namely the number of neurons of the input layer and hidden layer, were determined through a grid search of possible combinations. Sets {12, 14, 16, 18} and {14, 16, 18, 20} were separately tried for the input and hidden layers, resulting in a total of 16 possible combinations. For each combination, the ELM model was run 1000 times based on the training dataset $\{(X_i, \hat{y}_i^-, \hat{y}_i^+)\}_{i=1}^N$. When the lowest multi-objective function value was found, the randomly generated weights a_j and biases b_j, and the calculated β^* were recorded. The weights a_j and biases b_j, would then fixed during the following PSO experiments, whereas the weights β^* would be used to generate the initial positions of particles in PSO algorithm. The experimental results of ELMs for three PINC levels (90%, 95% and 99%) are shown in Table 2.

As shown in Table 2, the values of the lowest multi-objective function do not appear to be very sensitive to varying the numbers of the input and hidden neurons. Nevertheless, as can be seen from the table, the optimal ELM architecture consisted of 14 neurons for the input layer, 20 neurons for the hidden layer.

Table 2. Experimental Results of ELMs for Three PINC Levels (90%, 95% and 99%)

(Input Neuron Number, Hidden Neuron Number)	Lowest Multi-objective Function Value		
	PINC (90%)	PINC (95%)	PINC (99%)
(12, 14)	0.74	0.87	0.82
(12, 16)	0.75	0.89	0.66
(12, 18)	0.78	0.84	0.80
(12, 20)	0.73	0.88	0.75
(14, 14)	0.78	0.86	0.76
(14, 16)	0.79	0.82	0.73
(14, 18)	0.77	0.81	0.66
(14, 20)	**0.73**	**0.79**	**0.66**
(16, 14)	0.75	0.85	0.71
(16, 16)	0.79	0.86	0.68
(16, 18)	0.75	0.85	0.67
(16, 20)	0.78	0.87	0.72
(18, 14)	0.84	0.87	0.73
(18, 16)	0.81	0.83	0.75
(18, 18)	0.82	0.85	0.73
(18, 20)	0.85	0.82	0.74

For the PSO part, the population number N_P was set to 50, the iteration times N_{iter} to 150, and w, c_1 and c_2 in Equation 17 were set to 0.9, 1 and 1, respectively. The optimal value for ϕ in Equation 18 was 0.5, and the maximum particle speed v_{max} was 2. Figure 11 shows the values of the objective function, and the reliability and sharpness metrics as a function of the number of iterations during the training of PSO-ELM model, with PINC equal to 95%. As can be seen, the three curves converged quite early at the 60th iteration.

The objective function value decreased from 0.79 to 0.21the absolute average coverage error (AACE), the measure of reliability dropped from 0.506 to 0.037 with a clear declining trend (recall lower values of AACE indicated higher reliability or accuracy), and the sharpness curve fluctuated up and down but stabilized at around 0.17 level finally. The changes of the curves show that PSO can improve ELM to minimize the multi-objective function value.

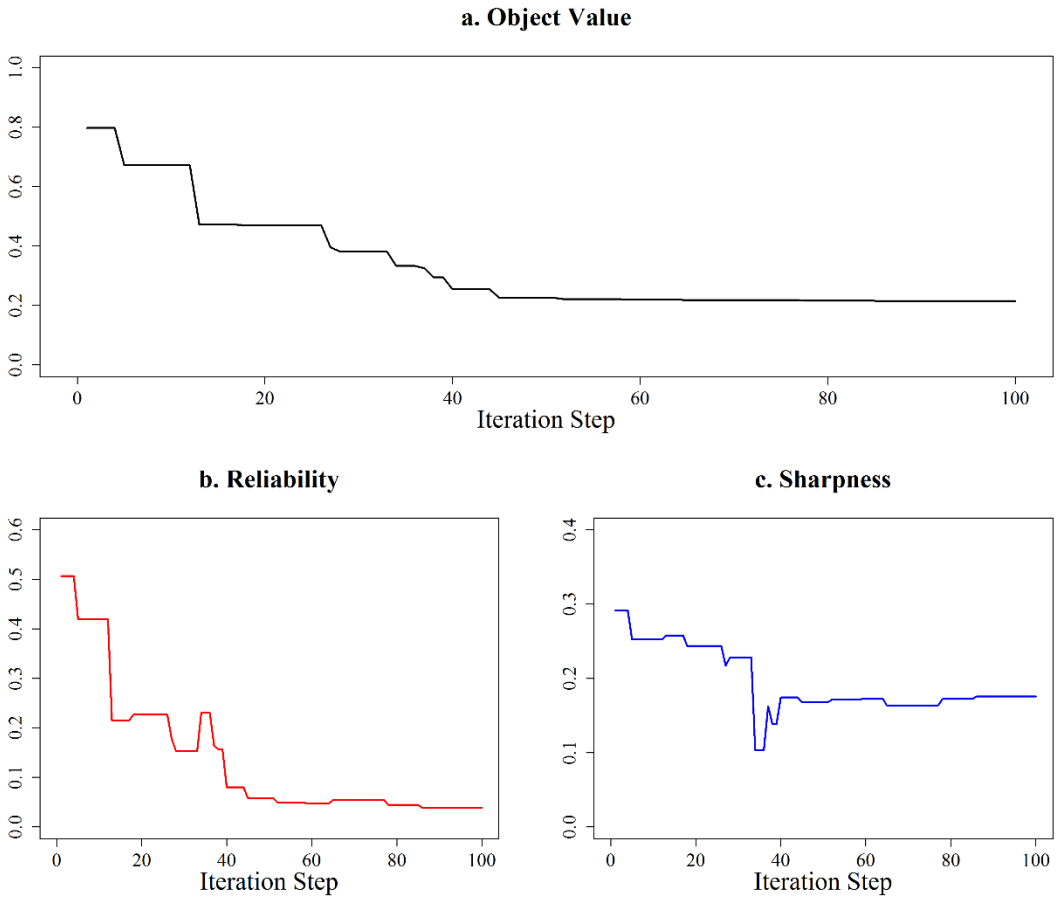


Figure 11. Optimization curves in PSO-ELM algorithm with 95% PINC (a. change of object value; b. change of reliability; c. change of sharpness).

For the improved PSO-ELM models under different PINC levels, as mentioned earlier, we replaced the calculation of reliability with Equation 19, and updated the parameters of the models every 15 points in this study. With 300 observations in testing dataset, each model was updated 20 times. The tuning of each model shared a similar process to that of the off-line PSO-ELM model.

For the hybrid model of Zhang et al. (2014), spectral analysis was conducted using R package's *TSA*, the periodogram reached local maximum at time index 3, 4, 6, 40, 80 and 120. Equation 24 lists the estimated parameters for the cyclic regression model.

$$y_t = 255.42 + 14.01\,sin\left(2 * pi * 3 * \frac{t}{600}\right) + 22.20con\left(2 * pi * 3 * \frac{t}{600}\right) + 18.59sin\left(2 * pi * 4 * \frac{t}{600}\right) + 27.01con\left(2 * pi * 4 * \frac{t}{600}\right) - 19.69sin\left(2 * pi * 6 * \frac{t}{600}\right) - 61.48con\left(2 * pi * 6 * \frac{t}{600}\right) + 56.36\,sin\left(2 * pi * 40 * \frac{t}{600}\right) - 49.80con\left(2 * pi * 40 * \frac{t}{600}\right) + 43.31\,sin\left(2 * pi * 80 *$$

$$\frac{t}{600}\Big) - 16.53con\left(2 * pi * 80 * \frac{t}{600}\right) + 18.92\, sin\left(2 * pi * 120 * \frac{t}{600}\right) + 20.93con(2 * pi * 120 * \frac{t}{600})$$

Equation 24

Figure 12 shows the original border crossing traffic flow, the estimated trend using Equation (48), and the residual part.

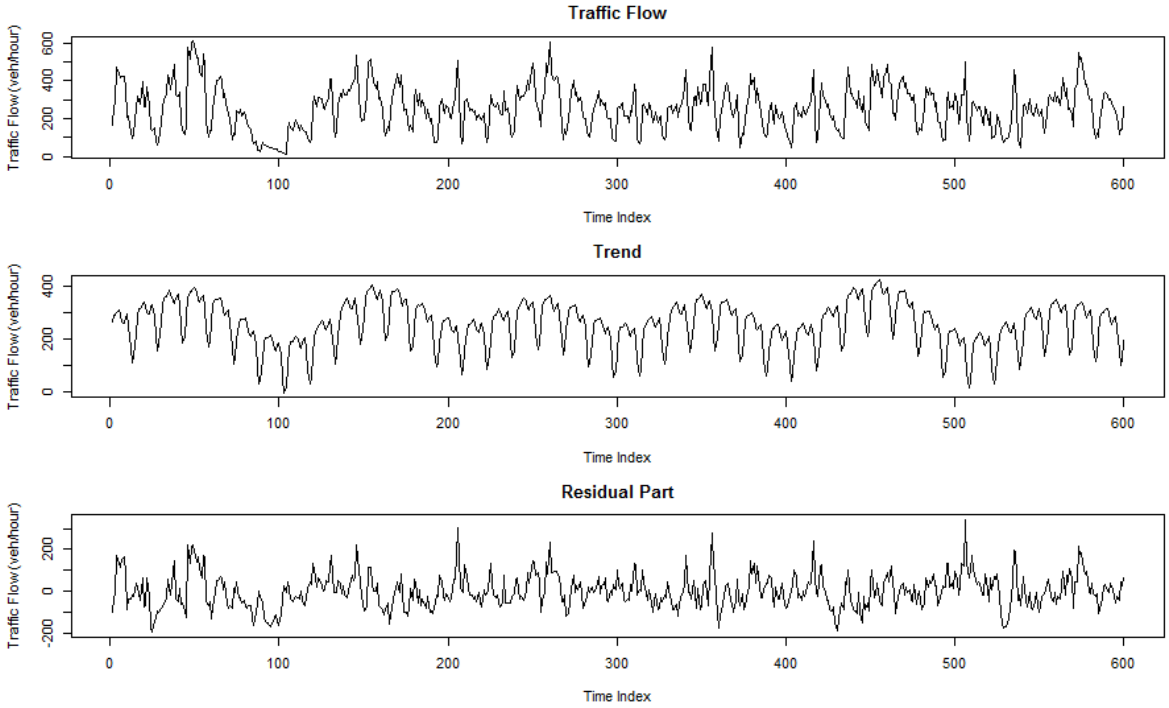


Figure 12. Decomposition of border crossing traffic flow.

Model results

In this sub-section, we compare the performance of the improved PSO-ELM, the original PSO-ELM, and the hybrid model by Zhang et al. (2014), for the three PINC levels 90%, 95% and 99%, . To compare the PIs of these models, for each model, we calculated the PICP metric introduced previously, which calculates the ratio of the 300 observations in the testing dataset falling within the PIs, and the MPIL metric which measures the average distance between the upper bounds and lower bounds of the intervals as described earlier (Zhang et al., 2014). We also calculated the reliability R^{α} and the sharpness S^{α} criteria, however, we found that the PICP and MPIL metrics can help us evaluate the models in a more straightforward way.

A number of observations could be made regarding the results of the comparison. First, because the original PSO-ELM model aims to minimize the multi-objective function by making PICP as close as PINC, model's PICPs were found to be exactly equal to the specified PINCs for all the levels (i.e., the PICP for an PINC level of 90% was found to be exactly equal to 90%). Also because the original PSO-ELM models were not updated when the new data arrives, the MPILs of the PSO-ELM models were found to be higher than those for the improved PSO-ELM and for those of the Zhang et al. model, indicating a worse performance. Moreover, the improved PSO-

ELM model provided higher PICP than the specified PINC level (for example, the PICP of the improved PSO-ELM model was 94% when the specified PINC was 90%). If we specifically focus on the performance of the improved PSO-ELM model developed in this study, one can see that the improved PSO-ELM had the smallest MPIL among all three models for the three specified PINC levels, and its PICP was higher indicating superior performance.

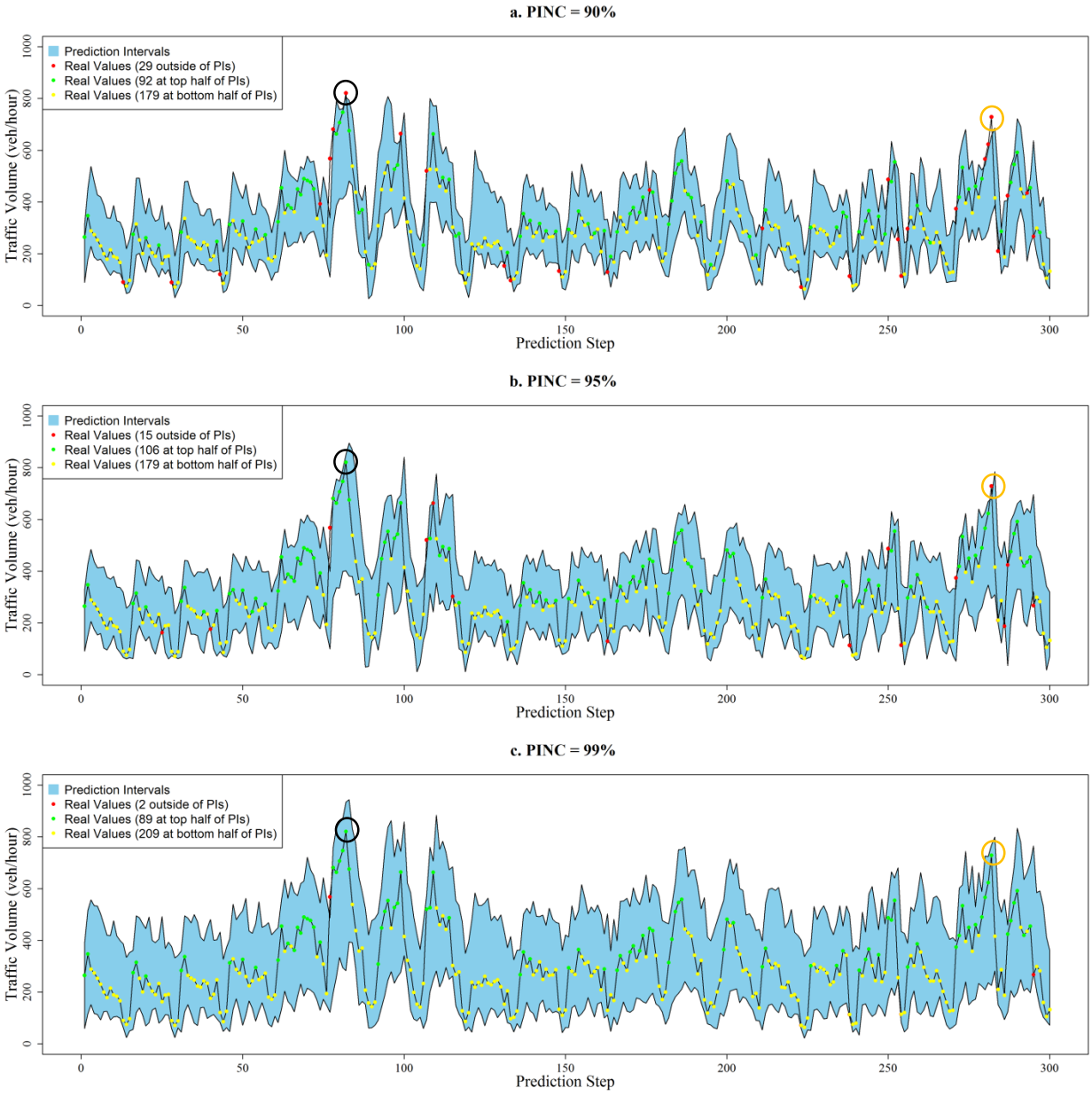


Figure 13. PIs of PSO-ELM by PINC levels (a. PINC = 90%; b. PINC = 95%; c. PINC = 99%).

Figure 13 shows the 300 real observations from the testing dataset and the prediction intervals of the original PSO-ELMs under the three PINC levels. The real observations are marked as red

when they fell outside of PIs, green when they fell within top half of PIs, and yellow within the bottom half of PIs. As can be seen, first, moving from the top figure of Figure 13, to say the ones below it, when the PINC level increases from 90% to 99%, the prediction intervals become correspondingly wider and thus naturally fewer observations fall outside the prediction intervals; specifically, there were 29, 15 and 2 points (marked in red) that fell outside PIs the PINC levels of 90%, 95% and 99%, respectively. For example, the point marked with the black circle fell outside of the prediction interval under 90% PINC, but within the PIs under the 95% and 99% levels. Similarly, the point marked with the orange circle only fell within the prediction interval when the PINC was 99%.

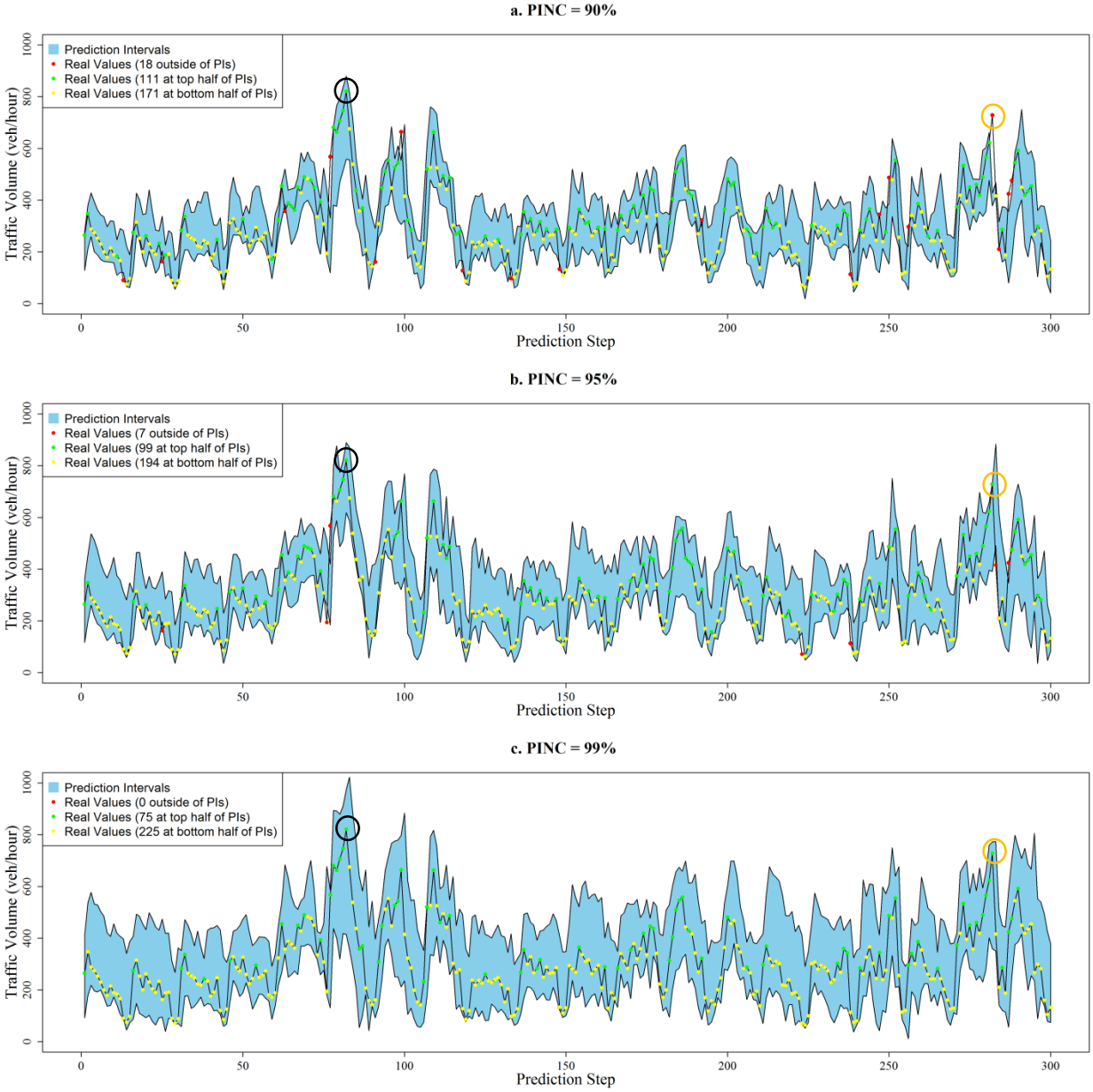


Figure 14. PIs of Improved PSO-ELM by PINC levels (a. PINC = 90%; b. PINC = 95%; c. PINC = 99%).

In the same way, the PIs of improved PSO-ELM models by PINC levels and the 300 data points are shown in Figure 14. If we compare the PIs with those under the same PINC level in Figure 13, we can notice immediately that they are much narrower. Meanwhile there are fewer points outside of PIs under the same PINC. The same point circled with black is already within the interval when it is 90% PINC, and the orange-marked point is covered by the interval when it is 95% PINC. Figure 13 and Figure 14 thus demonstrate the superior performance of the improved PSO-ELM model proposed herein.

MODEL APPLICATION FOR OPTIMAL STAFFING LEVEL PLAN DEVELOPMENT

In this section, we propose a comprehensive optimization framework to make staffing level plan recommendations for border crossing authorities, based on future traffic volume predictions from the different models described above (this includes the use of both PI bounds and point predictions from the Zhang et al. model). We then compare these different staffing levels plans in terms of average waiting times and total system cost.

Optimal staffing plan development framework

In our previous research, we proposed a generic queueing model with a Batch Markovian Arrival Process (BMAP) and Phase Types (PH) services for border crossing delay calculation (Lin et al., 2014b). The transient solution of the BMAP/PH/n queueing model was obtained using heuristic methods. We then compared the queueing models' estimates to the results from a detailed microscopic traffic simulation model of the Peace Bridge border crossing, and showed that the transient multi-server queueing model, along with the heuristic algorithm, is capable of estimating the border crossing waiting time accurately and efficiently.

In that study, we also incorporated the queueing model within an optimization framework to help inform border crossing management strategies. The optimization model is shown below.

$$\min_{B_i} C_i = C_{ope} * B_i + C_w * V_i + C_{pun}$$

Equation 25

s.t.

$$\frac{V_i * \mu}{B_i} \leq Th_w,$$

$$B_{min} \leq B_i \leq B_{max},$$

where,

C_i is the total cost of the queueing system during hour *i*;
C_{ope} is the cost per hour to operate one booth;
C_w is the hourly cost of waiting time per vehicle;
B_i is the number of open booths during hour i;
V_i is the average number of waiting vehicles during hour *i*, which can be calculated based on the transient BMAP/PH/n queueing model;
μ is the average service time (seconds);
C_{pun} is the penalty cost for changing the number of open booths from one hour to the next, calculated as follows $C_{pun} = c * |B_i - B_{i-1}|$, where c is the penalty for switching for one booth;

$\frac{V_i * \mu}{B_i} \leq Th_w$ is the constraint that ensures that the average waiting time is less than a threshold value, Th_w;
$B_{min} \leq B_i \leq B_{max}$ is the constraint for the number of available booths.

The goal of the optimization is to minimize the total system cost of the queueing system for a given hour i, including the cost for both the travelers as well as the operating agency. While doing that, the problem strives to keep the expected waiting time below a certain threshold. The total cost consists of three elements. The first element is the operating cost of opening the inspection stations, calculated by multiplying the assumed hourly cost of operating one booth by the number of booths or inspection stations open during hour i. The second element is the cost of the wait time travelers spent waiting in the queue at the border, calculated by multiplying the assumed monetary value for one hour of waiting time by the average number of vehicles in the queue during hour *i*. The third element is a penalty term designed to capture the cost of switching between an open and a closed inspection lane (or vice versa). Two constraints are included: the first constraint is added to keep the average delay per vehicle below a certain threshold while a second constraint is included to make sure the number of inspection lanes open does not exceed the physical number of lanes available at the border crossing. If the first constraint cannot be satisfied with all the available booths open, we will set B_i as the one that can minimize the total cost C_i.

Our previous study didn't try to make the optimal staffing plans based on the future traffic predictions. With the BMAP/PH/n queueing model and the optimization function, we are capable of developing optimal staffing plans for border crossing authority based on a series of different types of short-term traffic predictions, such as the PI upper or lower bounds or point predictions. We can then evaluate different optimal staffing plans in terms of waiting times and total system cost. The optimal staffing plan development framework is as follows:

Step 1: At the beginning of hour *i*, check how many vehicles are waiting in the queue V_{i-1} and record the number of open booths B_{i-1}. These are necessary inputs to the queueing model and the optimization model. Based on the next hour traffic prediction (PI upper or lower bound or point prediction), calculate the optimal number of open booths B_i as the staffing plan for hour *i* using the Equation 25

Step 2: With the optimal number of open lanes determined, use the real traffic demand for the hour *i* as the input of the BMAP/PH/n queueing model (Lin et al., 2014b), run the multi-server queueing model. This is to simulate the real world scenario if the border crossing authority followed the optimal staffing plan based on the short-term traffic prediction.

Step 3: At the end of hour *i*, record the system cost C_i and average waiting times $\frac{V_i * \mu}{B_i}$ based on the queueing model. Record the number of waiting vehicles in the queue V_i and the number of open booths B_i.

Step 4: keep running Step 1 to Step 3 until the scheduled operational period ends.

In this study, the parameters in are set as follows. For the hourly operating cost of one booth (C_{ope}), a value of \$150 was assumed. For the monetary value of one hour of wait time (C_w), it was estimated to be around \$25. The penalty for switching one booth (c) from closed to open (or vice versa) was set as \$20. Given that the maximum number of inspection stations that can be opened at the Peace Bridge is 10, which meant that $B_{min} = 1$ and $B_{max} = 10$. Average service time μ was set as 44.58 seconds (based on real-world observations from our previous research). The accepted delay threshold (Th_w) was considered as 10 minutes. More details can be found in the reference (Lin et al., 2014b).

Optimal staffing plan comparison

The optimal plan development framework allows us to calculate and compare hourly average waiting times and total system costs for the operational period for various types of predictions such as the upper bounds or lower bounds of PIs (in this part, we focus on the PIs from improved PSO-ELM and from Zhang et al. (2014) model. The experiments also tested the point predictions from Zhang et al. (2014) to verify if PIs could result in better staffing plans.

For the operational periods, we picked two representative morning periods to compare the performances of different predictions. One is 7:00-12:00 on 02/10/2014, a normal Monday, and the other one is 7:00-12:00 on 02/17/2014, President's Day. In the Tables to follow, we differentiate the predictions of diverse models by integrating the model name and the PINC level, with "U" for upper bound or "L" for lower bound. For example, "Im_PSO-ELM_90L" means the predictions are from the lower bounds of the improved PSO-ELM under 90% PINC level. For the point predictions from the model of Zhang et al. (2014), they were simply named as "Zhang_Point".

Note that as the warming-up stage of the queuing model and the staffing plan development process, the results for the first hour (i.e., 7:00-8:00) are not included in the analysis. Fig. 9. shows the average waiting times for 8:00-12:00 based on six sets of upper or lower bound predictions when PINC level is 90% and two types of point predictions. The shaded area formed by the corresponding upper and lower bounds is the average waiting times interval.

The following table, Table 3, further summarizes the total system costs for various sets of staffing level plans from the different predictions from 8:00 to 12:00 on Monday, 02/10/2014. First for the normal weekday morning hours, the staffing plans developed utilizing point predictions, perform better than the PI bound plans with the total system costs around \$3,000. Second, although implementing the staffing plans from upper bounds can keep the average waiting times all zero, the total system costs are a little higher than the plans from the point predictions because of the low real traffic demand and the extra operation costs from opening more booths. Third, considering that the PI lower bound is usually smaller than the real traffic demand, therefore it is much easier to satisfy the PI lower bound staffing plan. If the operation authority is short-staffed to implement the plans from PI upper bounds or point predictions which may require more open booths, only the plans based on the PI lower bounds from improved PSO-ELMs can still keep a reasonable level of service (less than 10 minutes from 9:00 to 12:00 in Fig. 9.), and the total system costs are only \$1,000 more comparing with the point prediction plans due to more waiting travelers.

Table 3. Total System Cost from 8:00 to 12:00 on Monday (02/10/2014)

Lower Bound Predictions	Cost ($)	Upper Bound Predictions	Cost ($)	Point Predictions	Cost ($)
Im_PSO-ELM_90L	4,092	Im_PSO-ELM_90U	3,510		
Im_PSO-ELM_95L	4,092	Im_PSO-ELM_95U	3,770		
Im_PSO-ELM_99L	5,292	Im_PSO-ELM_99U	4,370		
Zhang_90L	7,275	Zhang_90U	3,170		
Zhang_95L	12,758	Zhang_95U	3,320	Zhang_Point	3,001
Zhang_99L	14,384	Zhang_99U	3,620		

In contrast, Table 4 shows the total system costs for different plans from 8:00 to 12:00 on President's Day in 2014. We notice that the total system costs become much higher for plans based on point predictions, about $12,000 comparing with the previous $3,000 in Table 3. Again, this is mainly the result of the underestimation of high traffic demand on this holiday and the huge waiting time costs from the travelers. The poor performances of the plans from lower bounds are because of the same reason. However, note that the plans using predictions from "Im_PSO-ELM_90L" generate a cost of $13,437, which is very close to the costs from the point predictions with only $1,000 more in additional expense. Again this shows the plans from PI lower bounds of the improved PSO-ELM could be quite useful when the management authority lacks staff.

More importantly, Table 4 shows that in this case, to keep the border crossing traffic from Canada to US moving smoothly, we'd better implement the staffing plans based on the PI upper bounds. Although the operation costs are higher, travelers spend much less time waiting at the border (Fig. 10). Therefore, the total system cost can be controlled down to as low as around $5,500. When using upper bounds, and no matter which PINC level is chosen, the total system costs based on the upper bounds of improved PSO-ELMs are always less than $6,000. Note that our analysis did not consider *indirect benefits of reduced delays* (e.g., tourists can spend more time on shopping, food and entertainment and so on in US during holidays) nor *environmental benefits* resulting from reductions of idle engines waiting at the border.

Table 4. Total System Cost from 8:00 to 12:00 on President's Day (02/17/2014)

Lower Bound Predictions	Cost ($)	Upper Bound Predictions	Cost ($)	Point Predictions	Cost ($)
Im_PSO-ELM_90L	13,437	Im_PSO-ELM_90U	5,458		
Im_PSO-ELM_95L	21,460	Im_PSO-ELM_95U	5,933		
Im_PSO-ELM_99L	25,988	Im_PSO-ELM_99U	5,783		
Zhang_90L	29,611	Zhang_90U	6,308		
Zhang_95L	35,541	Zhang_95U	6,308	Zhang_Point	12,215
Zhang_99L	39,925	Zhang_99U	5,573		

In addition to picking the two specific time periods for detailed analysis (i.e., a normal Monday and President's Day), the study also calculated the average waiting times and average costs for the entire testing data set consisting of 300 hours. The overall

performances of the staffing plans, developed based upon the different models investigated in this study, are summarized in
Table 5 and Table 6. The observations from the average waiting times shown in
Table 5 show that: (1) the staffing plans derived from using PI upper bounds result in almost zero waiting times; (2) for the staffing level plans developed based upon the PI lower bounds, the one from "Im_PSO-ELM_90L" performs the best. This may be attributed to the fact that most days in the testing dataset are normal days. For Table 6, the average costs of plans derived from PI upper bounds, are lower than the other types of plans. For the lower bound plans, the plan from "Im_PSO-ELM_90L" perform the best with average cost $1,570. Once again, the results show that when a border crossing authority is short of staff, plans derived from using the improved PSO-ELM lower bounds can result in lower average waiting times and costs.

Table 5. Average Waiting Times (mins) for 300 Hours in Testing Dataset

Lower Bound Predictions	**mins**	**Upper Bound Predictions**	**mins**	**Point Predictions**	**mins**
Im_PSO-ELM_90L	11.87	Im_PSO-ELM_90U	1.02		
Im_PSO-ELM_95L	12.52	Im_PSO-ELM_95U	0.63		
Im_PSO-ELM_99L	15.82	Im_PSO-ELM_99U	0.32		
Zhang_90L	14.77	Zhang_90U	0.74		
Zhang_95L	16.90	Zhang_95U	0.63	Zhang_Point	3.95
Zhang_99L	22.13	Zhang_99U	0.37		

Table 6. Average Costs ($) for 300 Hours in Testing Dataset

Lower Bound Predictions	**Ave-cost ($)**	**Upper Bound Predictions**	**Ave-cost ($)**	**Point Predictions**	**Ave-cost ($)**
Im_PSO-ELM_90L	1,570	Im_PSO-ELM_90U	795		
Im_PSO-ELM_95L	1,610	Im_PSO-ELM_95U	803		
Im_PSO-ELM_99L	2,030	Im_PSO-ELM_99U	846		
Zhang_90L	1,850	Zhang_90U	764		
Zhang_95L	2,110	Zhang_95U	774	Zhang_Point	871
Zhang_99L	2,980	Zhang_99U	807		

CONCLUSIONS AND FUTURE RESEARCH DIRECTIONS

This part of the study introduced and applied a hybrid machine learning model called PSO-ELM for interval prediction of short-term traffic volume. The study refined the original PSO-ELM model, to allow it to run in an on-line fashion, and redefined the reliability criterion. The paper compared the performances of the PSO-ELM models against two other models, the original PSO-ELM model and the Zhang et al. (2014) model. The models were developed utilizing an hourly traffic data set for traffic crossing the Peace Bridge International Border. The comparison results show that the PICP for the original PSO-ELM was always very close to the specified PINC, whereas the PICPs for the other models were higher than the corresponding specified levels of 90%, 95%, or 99%. Specifically, the PICP of the model by Zhang et al. (2014) was

equal to 100% for all cases, and that for the improved PSO-ELM had a PICP higher than or equal to the model. For MPIL, the improved PSO-ELM yielded the smallest value for all specified PINC it is the smallest for the improved PSO-ELM for any PINC level, followed by Zhang et al. (2014); the original PSO-ELM had a relatively high MPIL. Therefore, in general, only the PIs from the improved PSO-ELM models were found to be reliable and sharp. Furthermore, the quantitative multi-objective function allows the improved PSO-ELM models to adjust the weights of reliability and sharpness, while the statistical models don't provide such a mechanism.

The study then constructed a comprehensive staffing plan development framework to minimize total system cost for border crossing based on the upper bounds or lower bounds of the PIs, or point predictions. Experiments were conducted for the time period 7:00 to 12:00 on two typical days, one is a normal Monday, and the other one is President's Day. We found that for the holiday time period, the plan from PI upper bounds of the improved PSO-ELM reduced the hourly average waiting times the most, to be around five minutes, which also made the total system cost much lower. The plans from the lower bounds or the point predictions resulted in huge border waiting time costs because of the underestimation of the traffic demand for the President's Day holiday. On the normal Monday, the staffing level plans from point predictions performed well with reasonable average waiting times (around five minutes) for the travelers and low total system costs. In this case the plans from upper bounds produced no waiting times but caused a higher total system cost due to the extra operation cost with more booths.

In both the holiday and normal Monday scenarios, for the lower bound plans, the ones from the improved PSO-ELMs performed the best, with their average waiting times being much less than the PI lower bound plans from the Zhang et al. (2014) model, and even turning out to be less than or close to the point prediction plans. The average waiting times and costs with different staffing plans for the whole testing dataset are also calculated and compared. Similar findings are observed. In general, for the case when the border crossing authority lacks the resources to hire enough staff to implement the plans from PI upper bounds or point predictions, the plans based on the PI lower bounds from the improved PSO-ELMs appear to be still capable of maintaining a reasonable level of service with only $1,000 more in total system cost compared with the point prediction plans.

For future research, we provide the following suggestions:

1. To enhance the accuracy of the interval prediction models, future research should consider including additional variables to capture the effect of inclement weather and special events (those could be discovered from mining social media data). As a further refinement, the on-line PSO-ELM can be updated more frequently, e.g. once per hour instead of every 15 hours in this study. Future research could also explore how to adjust the weights of reliability and sharpness in the multi-objective function dynamically. For example, if the next hour is the peak-hour on a holiday, the reliability may be more important because we may want to use the upper bound of the PI to make staffing plans. On the other hand, if it is a non-peak hour, one may want to focus more on sharpness.

2. In this study, we make optimal plans purely based on either upper bounds or lower bounds of PIs. It may be of interest to interchangeably use upper and lower bounds. The simplest approach would be to use upper bounds for peak-hours, and lower bounds for non-peak hours. More sensitivity analysis need to be done in the future on parameters such as the waiting time cost per hour, the operation cost per hour, the waiting time threshold and the number of available staffs/open booths. The environmental pollution cost can also be considered in the future.

3. Third, the whole methodology can be tested on additional application scenarios such as tolling stations, subway and/or airport security checking points. It would also be interesting to test the methodology on additional datasets with finer granularity.

4. Finally, although this paper only focused on the traffic volume interval prediction at a single point, the PSO-ELM models described herein can easily be extended to the traffic state estimation problem for a whole road network with the adjustment of the number of neurons in the output layer.

DEEP-LEARNING MODELS FOR BORDER CROSSING DELAY PREDICTION

In this part of the study, we leverage a unique data set that has only been recently available which records the border crossing delay based on data collected from Bluetooth readers recently installed at the Niagara Frontier border crossings. With this unique data set, the study developed models that directly predict the future border crossing delay based on delay recorded in the previous time steps. The models are developed using deep learning methods, which have attracted a lot of attention within the research community recently and which have demonstrated significant advantages in dealing with big data problems such as computer vision and speech recognition. Deep Learning methods have also been recently applied to traffic state prediction (Lv et al., 2015; Ma et al., 2015).

DEEP LEARNING AND ITS APPLICATION IN TRANSPORTATION

A Deep learning technique is a type of Machine learning/Artificial intelligence (AI) approach (Goodfellow, Bengio, & Courville, 2016), which utilizes deep artificial neural networks for learning (Skansi, 2018). Deep learning is a method for training models through various levels of abstraction (Alpaydin, 2016). Figure 15 shows the relationship between AI, machine learning and deep learning.

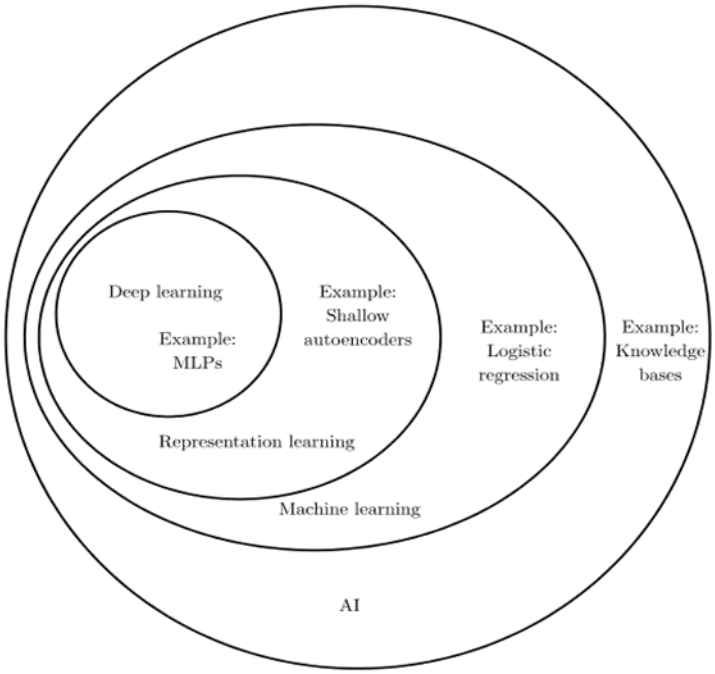


Figure 15. Relationship between AI, Machine Learning, and Deep L earning
(Goodfellow et al., 2016)

To understand deep learning, it might be necessary to understand some concepts like Neural Networks (NN) and Machine Learning (ML). A neuron is a simple processing unit and the network of these neurons and the connections between them is called an (NN) (Alpaydin, 2016). ML refers to the capability of the systems to gain their own knowledge by extracting patterns

from the raw data (Goodfellow et al., 2016). Supervised and unsupervised learning algorithms are two broad categories of machine learning algorithms (Goodfellow et al., 2016). Learning which utilizes a dataset where both the input and output are provided are supervised learning algorithms, while those which have to learn the properties of the structure from the dataset provided to them are called unsupervised learning algorithms (Goodfellow et al., 2016). Generally, a ML algorithm involves some hyper-parameters like the number of hidden units, learning rate, dropout rate, convolutional kernel width, implicit zero padding, etc. (Goodfellow et al., 2016). The learning algorithm cannot adjust the hyper-parameters by itself, however, their settings can control the behavior of the algorithm (Goodfellow et al., 2016).

Goodfellow et. al. (2016) mention that deep learning has a long history which can be traced back to the 1940s, and that it has mainly experienced three waves of development and was known by different names each time; these include the first wave from the 1940s to 1960s, the second wave from 1980s to 1990s and the third wave from 2006 (Goodfellow et al., 2016). Deep learning was known as cybernetics and connectionism during the first and second wave respectively (Goodfellow et al., 2016). Neuroscience is considered to be an inspiration for deep learning researchers, however, the modern deep learning doesn't hold neuroscience as its predominant guide (Goodfellow et al., 2016).

The usefulness of deep learning has increased with a greater amount of training data available and it has been able to solve progressively more complicated applications more accurately with time (Goodfellow et al., 2016). As per Khan et. al. (2018), the advantages of deep learning include the simplicity in generating large networks of deep learning and their easy scalability to huge datasets. Deep learning has been applied to various fields like robotics, natural language processing, search engines, online advertising, video games, and finance (Goodfellow et al., 2016). Each deep learning technique works differently. Various processes take place during the prediction of the next values by these techniques. In this section, the theory and working of the deep learning methods utilized in this research are explained in some detail.

Multilayer Perceptron (MLP)

Multilayer Perceptron (MLP) is one of most well-known of the deep learning techniques. It is made of three components, namely the input layer, hidden layer, and an output layer (Pal & Prakash, 2017). Each layer contains several numbers of neurons or nodes (Gardner & Dorling, 1998). As explained by Gardener and Dorling (Gardner & Dorling, 1998), MLPs consists of a system of neurons interconnected by weights (w). MLPs are fully connected when each neuron is connected to every other neuron in the previous and next layer. MLPs could have one or more hidden layers and their architecture is not fixed. Figure 16 shows an illustration of MLP.

Training of MLPs is a process of determining the individual weights such that the relationship that has to be modeled is accurately resolved (Gardner & Dorling, 1998). Gradient Descent is a technique that is used by the backpropagation training algorithm to train the MLPs (Gardner & Dorling, 1998)

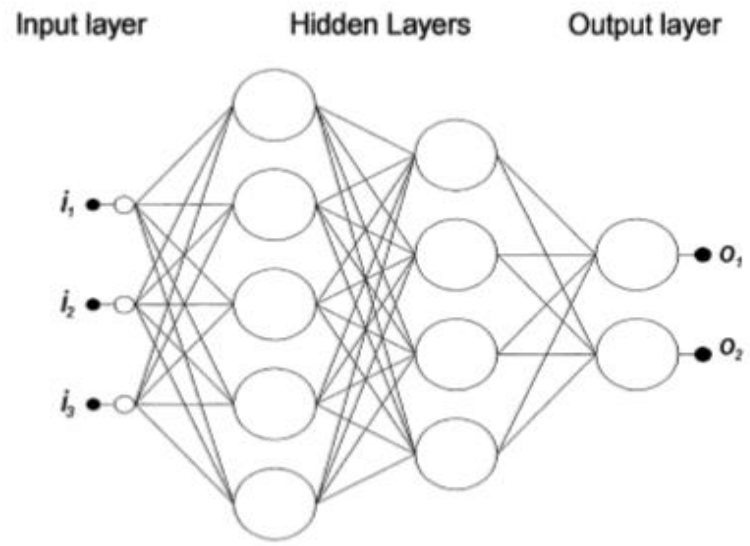


Figure 16. MLP with two hidden layers (Gardner & Dorling, 1998)

Pal and Prakash (Pal & Prakash, 2017) have described in detail the training process of MLP models. They have explained that the input features are fed from the input layer into the hidden layers, where each neuron applies a linear transformation and a non-linear activation to the input features. They demonstrated that the output (g_i) from each of these neurons is:

$$g_i = h\,(w_i x + b_i)$$

Equation 26

where wi and bi are the weights and bias of the linear transformation respectively and h is an activation function. They have pointed out that MLPs can model the non-linear relationship between the regressors and target variable with the help of non-linear activation function (Pal & Prakash, 2017). As per Skansi (Skansi, 2018), the most common activation function is sigmoid or logistic function, which outputs $\sigma(z)$ equal to $1/(1+e^{-z})$, where z (also called logit) is the sum of the product of inputs to the neuron with their respective weights plus bias. Bias is the modifiable value in each neuron (Skansi, 2018).

As explained by Pal and Prakash (Pal & Prakash, 2017), the output from the neurons of one hidden layer are fed as an input into the next hidden layer, where again transformations of the inputs take place and the outputs are fed into the next layer and this procedure goes on till the last hidden layer feeds the output layer. The process of transformation of the input layer to prediction is known as the forward pass (Pal & Prakash, 2017). They have further explained that after the forward pass is completed, loss or error (E) is computed, which is the difference between predicted value and the target value. Mean squared error (MSE) or mean absolute error (MAE) is most suitable for training the models for time series prediction (Pal & Prakash, 2017).

Next, the backpropagation algorithm is applied to compute the partial derivatives of loss with respect to the weights ($\partial E/\partial w$) in the backward direction, i.e. beginning from the output layer

and going up to the input layer, this is known as backward pass (Pal & Prakash, 2017). Finally, the weights of connections between each neuron, which were randomly initiated are now updated based on the learning rate and the results obtained from the backward pass (Pal & Prakash, 2017). As stated by Skansi (Skansi, 2018), the weights updated by the equation:

$$w_i^{new} = w_i^{old} + (-1)\, \eta\, \partial E / \partial\, w_i^{old}$$

Equation 27

where w_i^{new} is the new updated weight, w_i^{old} is the old weight and η is the learning rate

As explained by Gardener and Dorling (Gardner & Dorling, 1998), thousands of training iterations might be required for obtaining a MLP model with an acceptable level of error but the training should be stopped when the performance of the model reaches maximum on the independent test set. The weights are updated after each iteration (Pal & Prakash, 2017). The number of times the iterative weight update is repeated is called epochs (Pal & Prakash, 2017). Through the iterative process of the forward and backward pass, MLPs could understand the relationship between the dependent and independent variables and can also make predictions. As the direction of information processing in MLP is from the input layer to the output layer, they are called feed-forward neural networks (Gardner & Dorling, 1998).

Convolutional Neural Network (CNN)

Convolutional Neural Network or CNN is another deep learning method. Aghdam and Heravi (Aghdam & Heravi, 2017) have pointed out that applying a fully connected feedforward network on an image will result in a huge number of neurons, which makes them impractical for usage and therefore, the basic idea behind CNNs is to build a deep network with few numbers of parameters. The two types of convolutional layers in CNNs are 1 D convolutional layers or temporal convolutional layer, and 2 D convolutional layers or planar convolutional layers (Skansi, 2018). 2 D convolutions are generally applied to images, while 1 D convolutions are usually applied on sequential inputs (Pal & Prakash, 2017).

Aghdam and Heravi (Aghdam & Heravi, 2017) have stated that a CNN generally comprises of several convolution-pooling layers that are followed by fully connected layers. Figure 18 Figure 17 shows a diagrammatic representation of CNN. The convolutional layers usually contain multiple filters, which moves over the entire image, this movement is called convolution (Pal & Prakash, 2017). As per Khan et.al. (Khan et al., 2018), each filter is a grid of discrete numbers, which are also called the weight of the filter. They have further stated that the number of steps of the filter along the horizontal or vertical direction is called stride of the convolutional filter. They have demonstrated the convolutional operation that results in output feature maps due to the convolution between the filters and the inputs to the convolution layer.

Figure 18 shows the convolution operation of a 2 X 2 filter with a 4 X 4 input feature map to produce a 3 X 3 output feature map (Khan et al., 2018). Pal and Prakash (Pal & Prakash, 2017) have explained that the summation of the product of weights of the filter and the corresponding pixel values of the image plus a bias (optional) is the final feature from a local patch which results into a value of output feature map. Aghdam and Heravi (Aghdam & Heravi, 2017) have explained the weight sharing property of CNNs due to which the neurons in the same filter share the same set of weights, which results in a decrease in the number of parameters.

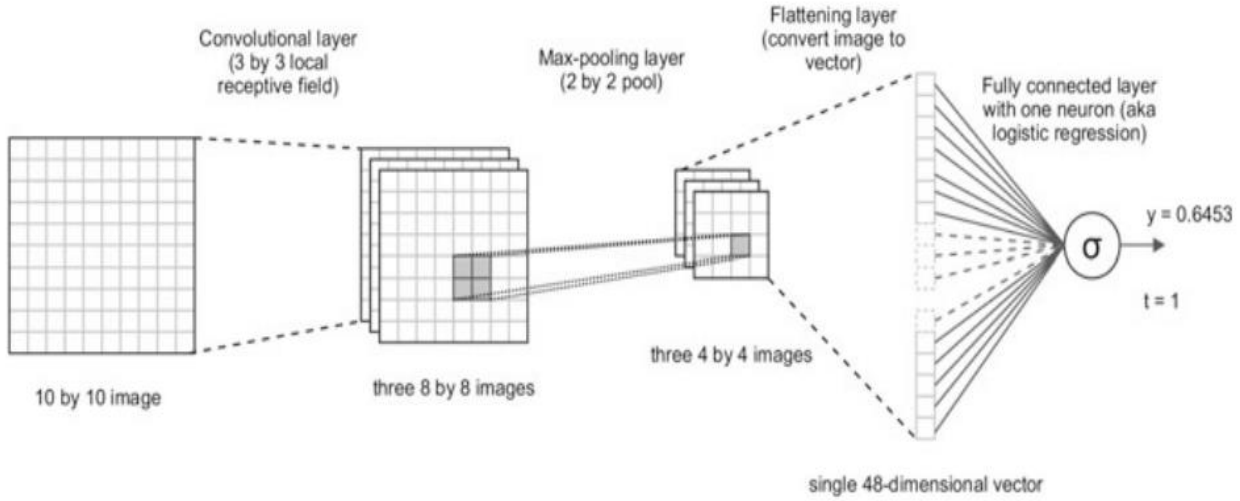


Figure 17: A typical CNN applied to a 2 D image (Skansi, 2018)

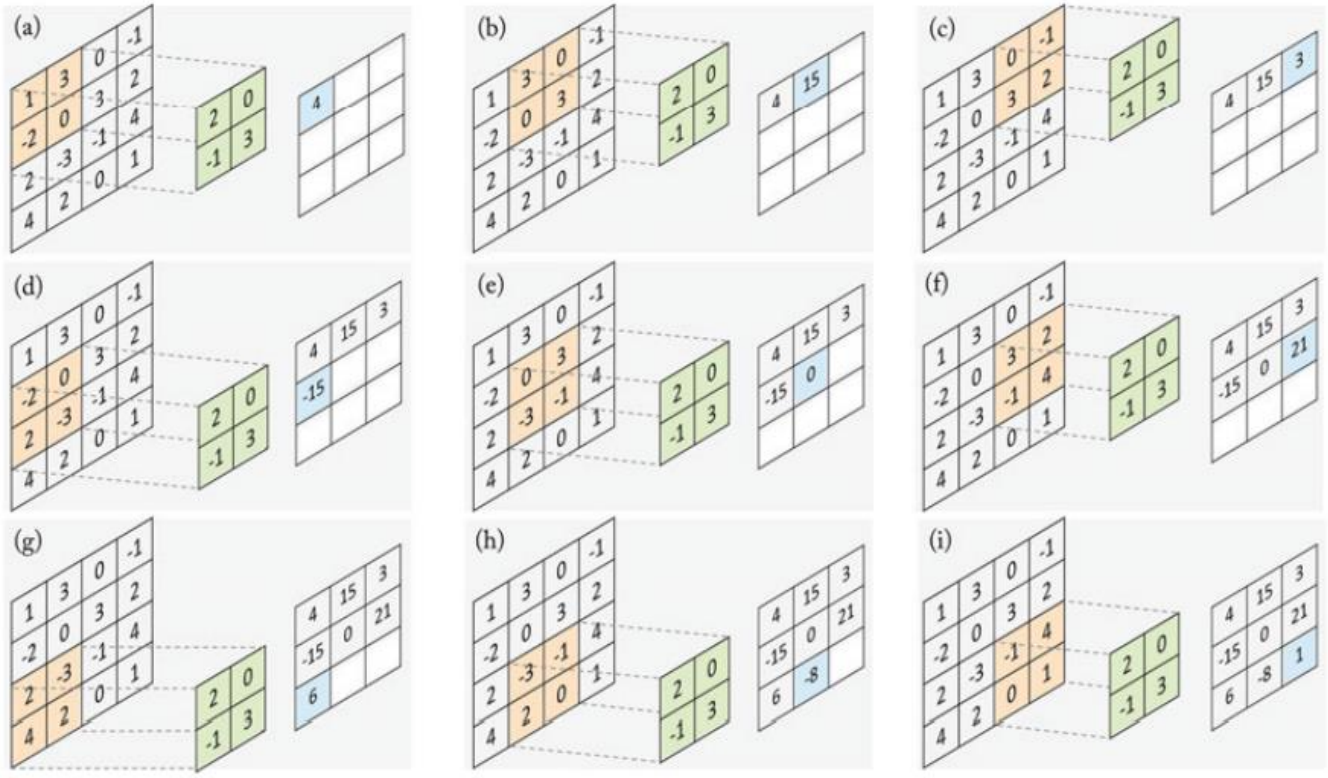


Figure 18. Stepwise operation of the convolutional layer with 2 X 2 filter and stride = 1 (Khan et al., 2018)

Khan et. al. (Khan et al., 2018) have pointed out that the spatial size of the output feature map obtained after convolution may be smaller than the input feature map. To avoid that, zero padding may be applied, which means increasing the size of input feature map each direction by padding zeroes so as to output feature map with desired size.

As per Khan et. al. (Khan et al., 2018), the dimensions of the output feature map (h' X w') from convolution operation is given by

$$h' = \{h - f - (d - 1)(f - 1) + s + 2p\}/s$$

Equation 28

$$w' = \{w - f - (d - 1)(f - 1) + s + 2p\}/s$$

Equation 29

where, h X w is the size of input feature map, filter in the convolutional layer has size f x f, d is dilation factor, s is stride and p is the increase in input feature map in each dimension due to zero padding.

The convolutional layers and fully connected layers in a CNN are generally followed by a non-linear activation function which enables the network to learn nonlinear mappings (Khan et al., 2018). As stated by Pal and Prakash (Pal & Prakash, 2017), rectified linear units (ReLu) is the popular choice for activation function, which is given by:

$$\begin{aligned} \text{ReLu}(z) &= 0, \text{ if } z \leq 0 \\ &= z, \text{ if } z > 0 \end{aligned}$$

Equation 30

Before the output from the convolutional layers is fed to dense layers, they may be passed through pooling layer (Pal & Prakash, 2017). The purpose of pooling layer is down sampling, which means to decrease the dimensionality of the feature map (Aghdam & Heravi, 2017). The pooling layers conduct combination operations on the blocks of the input feature map, which is defined by a pooling function like max or average pooling (Khan et al., 2018). A window of a pre-specified size and stride is moved across input feature map and pooling operation like max pooling takes place in which the maximum value from the selected block of input feature map is chosen (Khan et al., 2018). There are no trainable weights in the pooling layer (Pal & Prakash, 2017). As stated by Khan et. al. (Khan et al., 2018), the size (h' X w') of the output feature map from the pooling layer is given by:

$$h' = \lfloor (h - f + s)/s \rfloor$$

$$w' = \lfloor (w - f + s)/s \rfloor$$

Equation 31

where, the size of the input feature map is h X w, size of pooling region is f X f, the stride is s, and $\lfloor \cdot \rfloor$ represents floor operation.

As per Khan et. al. (Khan et al., 2018), fully connected layers are generally added near the end of the architecture in a typical CNN and their operation can be represented by:

$$y = f(W^T x + b)$$

Equation 32

where y is the vector of output activations and x is the vector of input activation, W is the matrix of weights of the connections between the layer units, b is bias and f (·) is a nonlinear function. Khan et. al (Khan et al., 2018) have explained that during the training process of CNN, the parameters of CNN are optimized such that the loss function is minimized. These parameters are the tunable weights in the layers of CNN (Khan et al., 2018). They have further explained that gradient based methods are used for the iterative search of the locally optimal solution at each step and to update the parameters in the direction of steepest descent (Khan et al., 2018). As the information is pushed forward, CNNs are also called feedforward neural networks (Skansi, 2018).

Skansi (Skansi, 2018) has suggested that the CNNs are easier to train because they require less number of parameters. He also pointed out that due to the shared set of weights in CNN, the problem of vanishing gradient is avoided as the same weights get updated each time even if just slightly. Additionally, the process involved in CNNs is computationally fast and can be split across many processors, which is due to the fact that training of each feature map can be done in a parallel fashion (Skansi, 2018).

Recurrent Neural Network (RNN)

Networks that have feedback loops in which some connections feed the output back into a layer as input are known as Recurrent Neural Networks or RNNs (Skansi, 2018). The diagrammatic representation of a simple RNN architecture is shown in Figure 19, where the circles with x, y, and h denotes input, output, and hidden nodes, while the squares with W^h_i, W^o_h, and W^h_h are matrices representing the input, output, and hidden weights respectively, and the polygon denotes nonlinear transformation (Bianchi et al., 2017).

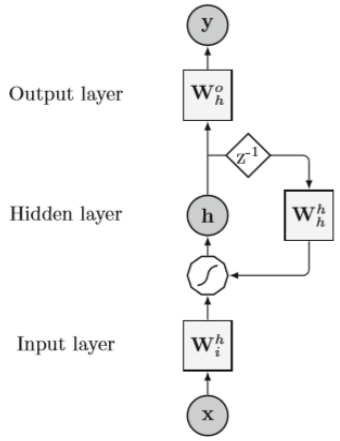


Figure 19: A simple RNN architecture (Bianchi et al., 2017)

The procedure of representing RNN as an infinite, acyclic and directed graph is known as unfolding (Bianchi et al., 2017). It comprises of replicating the hidden layer structure of the network for each time step (Bianchi et al., 2017). **Figure 20** shows the unfolded RNN. As explained by Bianchi et. al. (Bianchi et al., 2017), unlike the standard feedforward neural networks, the weight matrices of unfolded RNNs are constrained to assume the same values in all replicas of the layer. They have stated that due to this transformation, a direct relation between network weights and the loss function can be found and hence, the network can be trained with a standard learning algorithm.

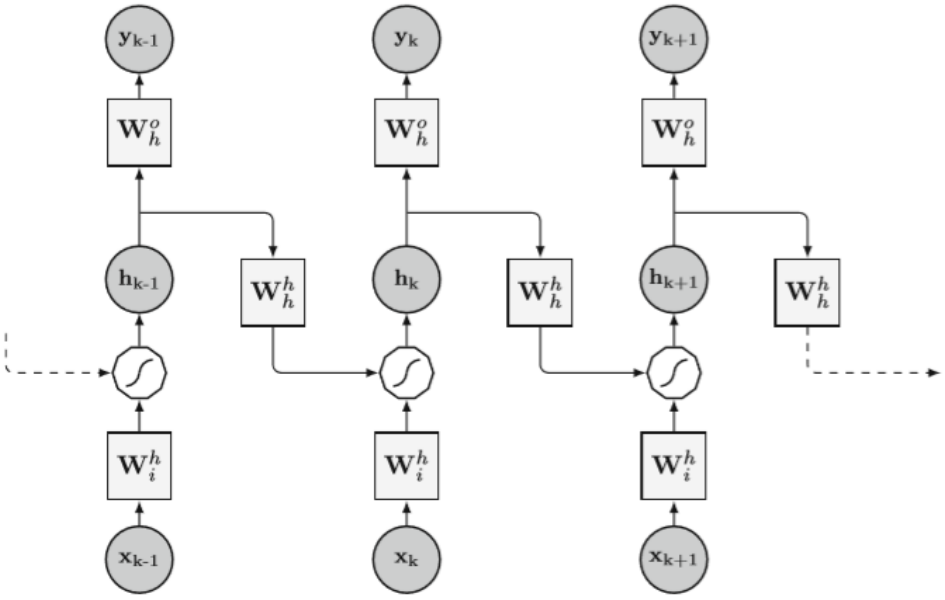


Figure 20: RNN unfolded to feedforward neural network (Bianchi et al., 2017)

As explained by Bianchi et. al. (Bianchi et al., 2017), the training procedure of RNN may be based on backpropagation through time (BTT) for propagation and distribution of prediction error to the previous states of the network. BTT is a special case of the backpropagation algorithm (Pal & Prakash, 2017). Usually, the training of neural networks involves using a gradient descent algorithm for updating its parameters so as to minimize the loss function (Bianchi et al., 2017).

Pal and Prakash (Pal & Prakash, 2017) have described the process to calculate BTT. The loss (L) or error between the predicted and target variable is found through forward pass and then the partial derivative of loss with respect to the network weights ($\partial L/\partial W$) is computed while going in the backward direction, i.e. from the loss to the weight (Pal & Prakash, 2017). However, there can be several paths connecting the loss to the weight as RNN has a sequential structure (Pal & Prakash, 2017). Therefore, the partial derivative ($\partial L/\partial W$) is computed as the summation of the partial derivatives along each path from the loss node to every time step node and this technique is called BTT (Pal & Prakash, 2017). Although, the multiplicative terms used in the computation

of gradient may be fractional and in long-range timesteps, the product of these terms might reduce the gradient to zero or to a negligibly low value, which does not allow the weights to update (Pal & Prakash, 2017). This problem is called the vanishing gradient problem (Pal & Prakash, 2017). The different types of recurrent neural network architecture - Elman Recurrent Neural Network (ELNN), Long Short-term Memory (LSTM), and Gated Recurrent Unit (GRU), are described below in detail.

Elman Recurrent Neural Network (ELNN): Elman Recurrent Neural Network is believed to be the most basic version of RNN and is also called Simple RNN or Vanilla RNN (Bianchi et al., 2017). Figure 19 shows the architecture of an ELNN. It comprises of input and output layers with feedforward connections, and hidden layers with recurrent connections (Bianchi et al., 2017). ELNN also have h which are the input for the recurrent connection (Skansi, 2018). As stated by Bianchi et. al. (Bianchi et al., 2017), at time t, the update of the internal state and the output of the network is given by:

$$h[t] = f\left(W^{h}_{i}\left(x[t] + b_i\right) + W^{h}_{h}\left(h[t-1] + b_h\right)\right)$$

$$y[t] = g\left(W^{o}_{h}\left(h[t] + b_o\right)\right)$$

Equation 33

where, W^{h}_{i}, W^{o}_{h}, and W^{h}_{h} are matrices representing the input, output, and hidden weights, respectively; x[t] is the input, y[t] is the output of the network; h[t] is the internal state, b_i, b_h, and b_o, are bias vectors and $f(\cdot)$ is the activation function of the neuron. (Bianchi et al., 2017). h[t] is generally initialized as a vector of zeroes, and it transfers the memory contents of the network at time t (Bianchi et al., 2017). Pal and Prakash (Pal & Prakash, 2017) have stated that ELNNs suffer due to vanishing and exploding gradients.

Long Short-Term Memory Recurrent Neural Networks (LSTM RNN): As ELNN faces difficulty in effectively learning the long-range dependencies because of vanishing and exploding gradients, LSTMs were developed to resolve this issue (Pal & Prakash, 2017). LSTMs can accurately model long-term and short-dependencies in the data (Bianchi et al., 2017). They do not impose any bias towards recent observations and allow the constant error to flow back through time, and by doing this, it tries to resolve the issue of vanishing gradients (Bianchi et al., 2017).

Bianchi et. al. (Bianchi et al., 2017) have explained that unlike the ERNNs, LSTMs apply a more elaborate internal processing unit, which is called the cell. They have further explained that a LSTM cell is made up of five different nonlinear components interacting in a definite manner. Additionally, a cell's internal state is modified only by linear interactions, which allows smooth backpropagation of information across time (Bianchi et al., 2017). **Figure 21** shows a cell in LSTM, where, x_t and y_t are the external input and external output of the cell respectively; h_{t-1}, h_t, y_{t-1}, and y_t are internal state variables; g_1 and g_2 are operators with nonlinear transformation, and σ_f ,σ_u, and σ_o are sigmoid in forget, update, and output gate respectively (Bianchi et al., 2017).

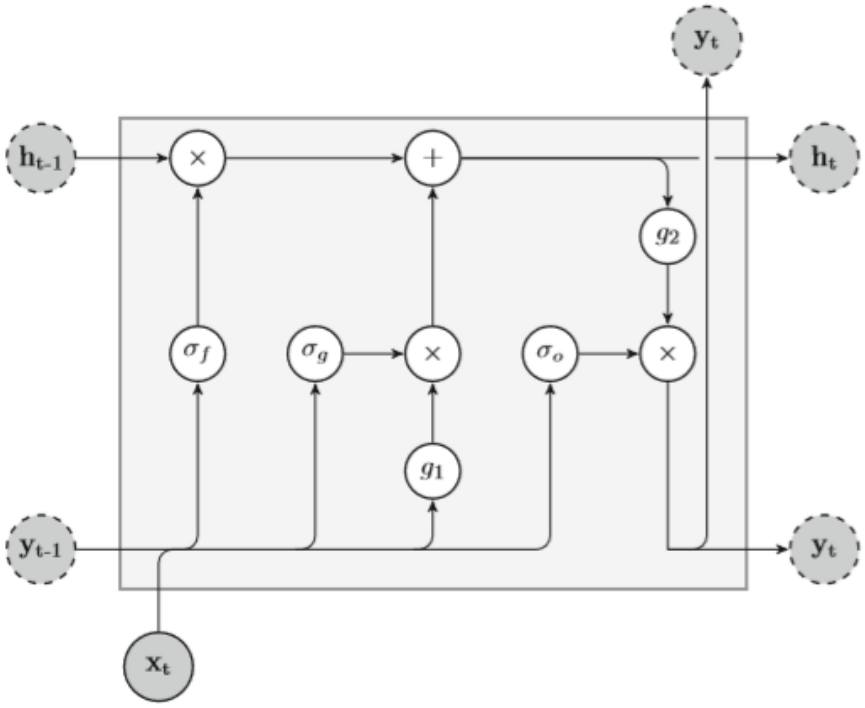


Figure 21: Cell of a LSTM (Bianchi et al., 2017)

For protecting and controlling information in the cells, LSTM uses three gates, which are forget gate, input gate and output gate (Bianchi et al., 2017). The information that must be removed from the previous cell state h[t-1] is decided by the forget gate, how much the new state h[t] must be updated by new candidate $\tilde{h}[t]$ is decided by input gate, and the part of the state to be outputted is decided by the output gate (Bianchi et al., 2017).

For updating the cell state and computing the output, the difference equations of forward pass, as given by Bianchi et. al. (Bianchi et al., 2017) are:

Forget gate:	$\sigma_f[t] = \sigma\,(W_f\, x[t] + R_f y[t-1] + b_f)$
Candidate state:	$\tilde{h}[t] = g_1(W_h x[t] + R_h y[t-1] + b_h)$
Input gate:	$\sigma_u[t] = \sigma\,(W_u x[t] + R_u y[t-1] + b_u)$
Cell state:	$h[t] = \sigma_u[t]\,(\cdot)\,\tilde{h}[t] + \sigma_f[t]\,(\cdot)\,h[t-1]$
Output gate:	$\sigma_o[t] = \sigma(W_o x[t] + R_o y[t-1] + b_o)$,
Output:	$y[t] = \sigma_o[t]\,(\cdot)\,g_2(h[t])$

Equation 34

where, x[t] is the input vector to the cell at time t; $\sigma(\cdot)$ denotes a sigmoid function; $g_1(\cdot)$ and $g_2(\cdot)$ are point wise nonlinear activation function, and $(\cdot)$ is the entry wise multiplication between two vectors; W_o, W_u, W_h, and W_f are weight matrices applied to input of the cell; R_o, R_u, R_h, and R_f are weights matrices of the recurrent connections; b_o, b_u, b_f, and b_h are bias vectors (Bianchi et al., 2017).

However, as per Skansi (Skansi, 2018), the interpretations of LSTM are just metamorphic and it very rarely works like a human brain. Bianchi et. al (Bianchi et al., 2017), have also pointed out

that practically, the forget and update gate of LSTM never opens or closes totally and contents of the cell may change over time.

Gated Recurrent Unit Recurrent Neural Network (GRU RNN): Introduced in 2014, GRUs are a simpler version of LSTM and can resolve the issue of long-term dependencies in ELNNs (Pal & Prakash, 2017). GRUs are capable of adaptably capturing the dependencies at different time scales (Bianchi et al., 2017). As stated by Pal and Prakash (Pal & Prakash, 2017), GRU has fewer trainable weights than LSTM. Figure 22 shows the architecture of a GRU cell.

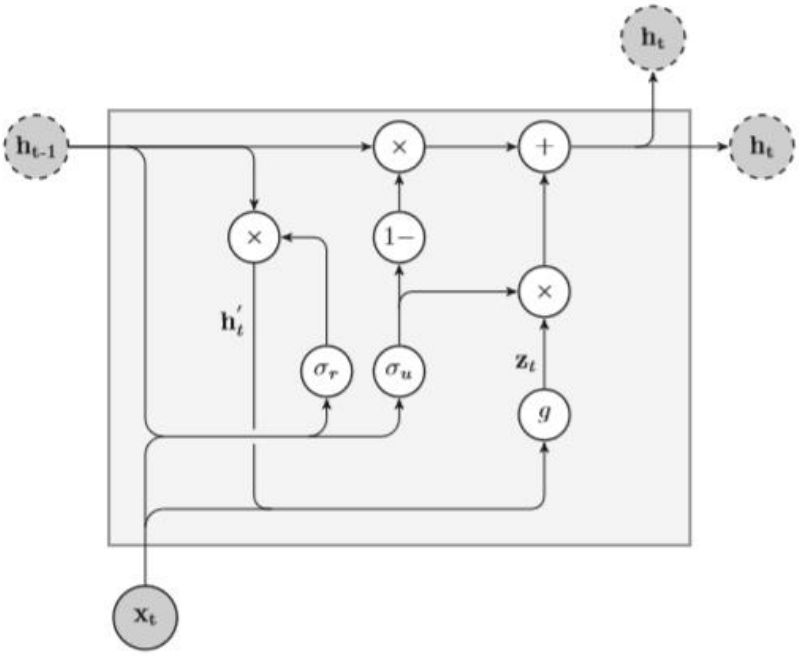


Figure 22: Cell of a GRU (Bianchi et al., 2017)

GRU has two gates, which are update gate, and reset gate (Pal & Prakash, 2017). The forget and input gates in LSTM are combined to form a single update gate in GRU (Bianchi et al., 2017). The function of update gate is to adaptively decide how much must be remembered or forgotten by the hidden units, while the memory of a cell can be reset by the reset gate (Bianchi et al., 2017).

The state equations of GRU given by Bianchi et. al. (Bianchi et al., 2017) are:

Reset gate:	$r[t] = \sigma(W_r h[t-1] + R_r x[t] + b_r)$
Current state:	$h'[t] = h[t-1] \,(\cdot)\, r[t]$
Candidate state:	$z[t] = g(W_z h'[t-1] + R_z x[t] + b_z)$
Update gate:	$u[t] = \sigma(W_u h[t-1] + R_u x[t] + b_u)$
New state:	$h[t] = (1-u[t]) \,(\cdot)\, h[t-1] + u[t] \,(\cdot)\, z[t]$

Equation 35

where, $\sigma(\cdot)$ denotes a sigmoid function; $g_1(\cdot)$ and $g_2(\cdot)$ are point wise nonlinear activation function, and $(\cdot)$ is the entry wise multiplication between two vectors; W_u, W_z, W_r, R_u, R_z, and R_r

are weights matrices of the recurrent connections; b_u, b_z, and b_r are bias vectors (Bianchi et al., 2017).

Optimization

Some kind of optimization is generally involved in the deep learning algorithms. However, the optimization algorithm used in deep models is quite different from the traditional optimization algorithms (Goodfellow et al., 2016). Unlike in pure optimization, in machine learning, a cost function $J(\theta)$ is reduced in order to improve some performance measure P, which is defined based on the test set (Goodfellow et al., 2016). An optimization algorithm is known as batch or deterministic gradient method if it uses the whole training set, while it is called stochastic or online method if only one example is used at a time (Goodfellow et al., 2016). Additionally, the optimization algorithm is called minibatch or minibatch stochastic or simply stochastic method if the training examples used are more than one but not all (Goodfellow et al., 2016). The optimization algorithms that were tried in the deep learning models developed for this study are briefly discussed below.

Stochastic Gradient Descent (SGD): In contrast to gradient descent which follows downhill the gradient of the whole training set, SGD just uses randomly selected minibatches instead of the whole training set and hence, significantly accelerates the process (Goodfellow et al., 2016). Learning rate is the step size of following downhill and may be selected by trial and error or by observing the learning curve which plots the objective function as a function of time (Goodfellow et al., 2016). As per Goodfellow et. al. (Goodfellow et al., 2016), in SGD, the initial parameter θ is updated in iteration k as $\theta - \epsilon\hat{g}$, where ϵ is the learning rate in iteration k, $\hat{g}$ is gradient estimate which is equal to 1/m times the gradient of loss function L with respect to θ, which can be mathematically represented as:

$$\hat{g} \leftarrow +(1/m)\nabla_\theta \sum_i L(f(x^{(i)};\theta),y^{(i)})$$

$$\theta \leftarrow \theta - \epsilon\hat{g}$$

Equation 36

where $y^{(i)}$ is the target corresponding to a minibatch $\{x^{(1)}, ..., x^{(m)}\}$ of m examples taken from the training set (Goodfellow et al., 2016).

Regularization

For machine learning models, the error measured on the training set is called training error, while that measured on the unobserved or new inputs is called generalization error or test error (Goodfellow et al., 2016). As per Goodfellow et. al. (Goodfellow et al., 2016), underfitting and overfitting are two challenges in machine learning. Underfitting is said to occur when the model is unable to get sufficiently low training error while overfitting occurs when there is a big gap between the training error and test error (Goodfellow et al., 2016).

As described by Goodfellow et. al. (Goodfellow et al., 2016), regularization refers to any change made in the learning algorithm to decrease its generalization error, but not the training error. There are various techniques described in the literature for applying regularization. Some of the regularization strategies for deep learning models discussed by Goodfellow et. al. (Goodfellow et al., 2016) includes parameter norm penalties, norm penalties as constrained optimization, data augmentation, noise robustness, early stopping, parameter tying and parameter sharing, dropout,

and adversarial training. Dropout technique was used in the deep learning models that were developed in this study for predicting short-term delay at border crossings. Hence, in this section, only the regularization by dropout is discussed.

Dropout is the technique used for improving the learning of neural networks and reduce overfitting (Skansi, 2018). Dropout can be applied by adding the dropout parameter π having a value between 0 and 1, and by doing so every weight will be set to zero with a probability of π in each epoch (Skansi, 2018). As per Goodfellow et. al. (Goodfellow et al., 2016), the advantages of dropout are that it is very computationally inexpensive and it does not restrict much the choice of model or training procedure that can be used.

MODELING DATASET

The dataset used in this part of the research was obtained from the Niagara International Transportation Technology Coalition (NITTEC). It contains the traffic delay data for the U.S. bound car traffic moving over each of the three Niagara frontier border crossing bridges – the Peace Bridge, the Lewiston-Queenston Bridge, and the Rainbow Bridge. Delay data were recorded via BLUFAX Bluetooth readers at a frequency of one minute. For this study, the data used was collected from March 1, 2018, to December 31, 2018, which is a total of ten-months data.

The data had very few missing points, only 24, out of a total of 13,21,920 data points from all the three bridges combined, were missing. The missing values were filled by taking the average of the previous value and the next available value. By observing the data, it was found that traffic delay did not fluctuate much every minute and hence, it seemed rational to aggregate the data to five-minute intervals. This U.S. bound traffic delay data of ten months on the three bridges aggregated to five-minute intervals is referred to herein as the 'complete set'.

Further, to evaluate the effect of days of the week on traffic delay and also, to assess the ability of deep learning techniques to model categorized data, the complete set was split into 'weekday set' and 'weekend set'. The weekday set consisted of the delay data only during weekdays, whereas the weekend set comprised of delay data only during weekends. Therefore, three set of data were available for the study – complete set, weekday set, and weekend set. The complete set, weekday set, and weekend set comprise of 88128, 62784, and 25344 data points respectively.

Data Collection

The data used for this study was collected by the Bluetooth readers installed in recent years at the border crossings: The Peace Bridge, the Lewiston-Queenston Bridge, and Rainbow Bridge. This data had the information about the U.S. bound passenger cars' traffic delay collected at an interval one minute. The Bluetooth wait time measurement system installed at the Peace Bridge and the Lewiston-Queenston Bridge crossings uses Traffax readers and FastLane BluFaxWeb software for computing the average wait time by vehicle time and direction (Roelofs et al., 2016).

Bluetooth is a telecommunications industry specification that defines the way in which the digital devices can interconnect easily using short-range wireless communications ("tpa-na

Traffic Monitoring," n.d.). The BluFax monitoring system can collect the travel times by sampling a portion of actual travel times from the traffic stream ("tpa-na Traffic Monitoring," n.d.). It measures the travel times by matching the MAC addresses of Bluetooth devices at two different locations ("tpa-na Traffic Monitoring," n.d.).

As stated at tpa-na.com ("tpa-na Traffic Monitoring," n.d.), Bluetooth devices have the following advantages over other existing methods:

- Unlike existing point detection technology, which includes inductive loops, radar detectors, image processors, etc., Bluetooth technology measures travel time directly by the equipment, and as a result, has greater accuracy ("tpa-na Traffic Monitoring," n.d.).
- It can be applied globally because of the proliferation of the Bluetooth standard protocol ("tpa-na Traffic Monitoring," n.d.).
- It can measure the travel times for different modes like highway vehicles, rail, and pedestrians because the Bluetooth devices are associated with people, not the vehicle ("tpa-na Traffic Monitoring," n.d.).
- As there are no databases of Bluetooth addresses, Bluetooth technology offers more privacy than toll tag tracking, cellular telephone geolocation, or license plate surveys ("tpa-na Traffic Monitoring," n.d.).
- The field installation procedure is simple ("tpa-na Traffic Monitoring," n.d.).

The red pointers in Figure 23, Figure 24, and Figure 25, shows the location of Bluetooth reader installation at the different bridges. The information about this location was provided by NITTEC.

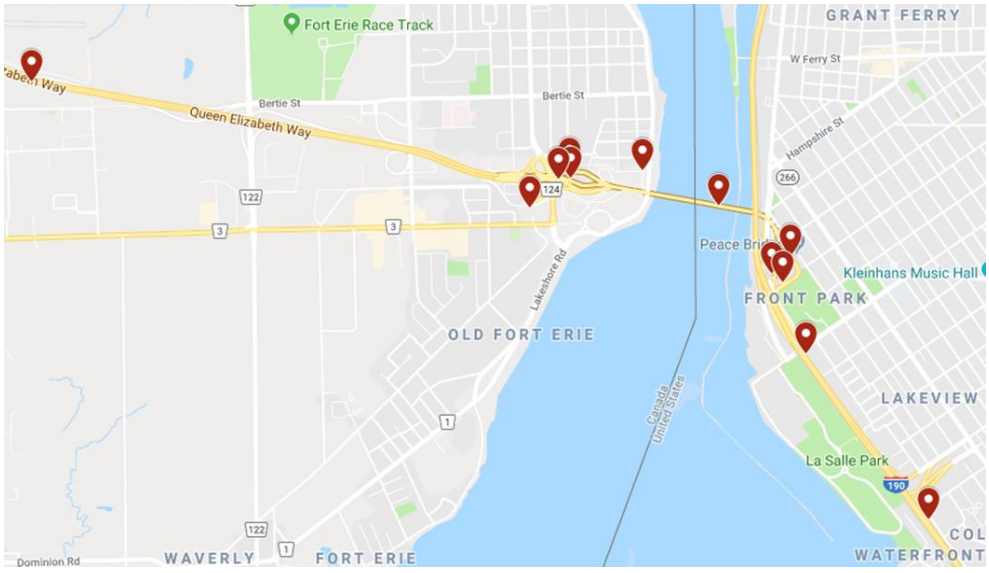


Figure 23: Location of Bluetooth readers at Peace Bridge

Figure 24. Location of Bluetooth readers at Queenston Lewiston Bridge

Figure 25. Location of Bluetooth readers at Rainbow Bridge

Model Development

In this study, the short-term car traffic delay was predicted by forecasting time series using the deep learning methods, namely Multilayer Perceptron (MLP), Convolutional Neural Networks (CNN), Long Short-Term Memory Recurrent Neural Networks (LSTM RNN), and Gated Recurrent Unit Recurrent Neural Networks (GRU RNN).

Each modeling dataset (complete set, weekday set, and weekend set) was split into train set, validation set, and test set. The train set was used for training the models. The validation set was used for selecting the hyperparameters by comparing the performance of various models to select the best one. Lastly, the test set was used to report the results from the selected model. The first 60% of the dataset formed the train set, the next 20% was used as a validation set and the last 20% as a test set.

The dataset that was fed into the model was a time series of traffic delay (in minutes) at every 5 minute interval. The model takes traffic delay at prior time steps as its input and outputs the traffic delay predicted for the future. The model was developed such that it outputs the predicted delay for the next 12 time steps. Therefore, the model output is the predicted traffic delay for the next 5 minutes, 10 minutes, 15 minutes, and so on till the next 60 minutes in the future. The number of prior time steps that the model takes as input can be varied. Mathematically, if the time is t and the model is set to take n number of prior time steps then the model inputs will be the delay at time t, t-5, t-10, and so on up to n time steps. Whereas, the model outputs will be the delay at time t+5, t+10, t+15, and so on till t+60.

The models developed trained using the whole data or complete data are called ‘complete set’ models. The models trained only on weekday data are called ‘weekdays’ model and those trained only on weekend data are called ‘weekends’model. The predicted delays obtained from the models were compared with the actual delay to evaluate the predictive accuracy of deep learning models, compare the performance of the four deep learning techniques and to find the effect of data classification on model’s prediction accuracy.

The models developed for this study were build using Keras (v2.2.2) deep learning library, backend by TensorFlow (v1.9.0) in the programming language Python (v3.5.6) on a computer with 8.00 GB RAM and Intel®Core™i56200U CPU. The various libraries used in the python codes includes scikit-learn, numpy, matplotlib, pandas, and math. The python codes used in this study for developing and comparing the deep learning models are inspired by the codes presented by (Brownlee, 2016, 2018; Pal & Prakash, 2017). Additionally, (Ardit, 2017; “Customizing Ticks | Python Data Science Handbook,” n.d.; “Home - Keras Documentation,” n.d.; “Matplotlib: Python plotting — Matplotlib 3.0.3 documentation,” n.d.; “scikit-learn: machine learning in Python — scikit-learn 0.20.3 documentation,” n.d.; “The Python Tutorial — Python 3.7.2 documentation,” n.d.) were a source of help in the writing of the Python codes.

MODEL RESULTS

This section provides the results of the deep learning models developed to predict the U.S. bound car traffic delay at the three border crossings between the U.S. and Canada – Peace Bridge (PB), Lewiston-Queenston Bridge (QL), and Rainbow Bridge (RB). The prediction performance of models was measured by the Mean Absolute Error (MAE), Root Mean Square Error (RMSE), and R squared (R^2). The results obtained from each of the deep learning models are presented in detail in the subsequent sub-sections.

Multilayer Perceptron (MLP)

The MLP model was developed to predict the U.S. bound car traffic delay for the next 5 minutes, 10 minutes, 15 minutes, … up to 60 minutes. However, for simplicity, the model results of only next 5, 15, 30, 45, and 60 are shown in

.

Table 7 : MLP model result

Prediction Horizon	Measure	PB			QL			RB		
		Whole set	Week days	Week ends	Whole set	Week days	Week ends	Whole Set	Week days	Week ends
	MAE	1.14	1.38	0.92	1.02	1.03	1.08	1.09	1.27	1.62
	RMSE	2.94	3.49	3.28	2.12	2.17	1.96	1.7	1.6	2.34
5 minutes	R^2	0.95	0.89	0.96	0.95	0.94	0.96	0.93	0.81	0.83
	MAE	2.1	2.16	2.01	1.93	1.87	2.19	2.02	2.15	2.57
	RMSE	7.02	6.6	8.5	7.65	7.01	9.21	3.37	3.04	4.31
15 minutes	R^2	0.79	0.7	0.86	0.83	0.8	0.86	0.79	0.66	0.73
	MAE	2.77	2.71	2.72	2.64	2.44	3.06	2.68	2.76	3.35
	RMSE	4	3.96	4.29	4.76	4.65	5.24	5.22	5.77	6.54
30 minutes	R^2	0.64	0.52	0.75	0.71	0.68	0.74	0.65	0.5	0.6
	MAE	3.08	2.98	3.15	3	2.75	3.59	3.06	3.03	3.83
	RMSE	4.52	4.4	5.11	5.48	5.34	6.16	6	6.17	7.25
45 minutes	R^2	0.54	0.41	0.64	0.62	0.57	0.64	0.54	0.43	0.5
	MAE	3.11	3.08	3.49	3.25	2.96	3.99	3.34	3.24	4.24
	RMSE	4.85	4.68	5.75	6.1	5.82	6.96	6.61	6.64	8.01
60 minutes	R^2	0.48	0.33	0.54	0.53	0.49	0.54	0.44	0.34	0.39
Computation time (s)		165.93	92.74	90.34	114.15	57.28	29.74	62.63	65.73	29.57

From

, and considering all time horizons, it can be seen that MAEs of weekdays are mostly less than those of their corresponding complete set (for 5 minute prediction in particular, MAEs of the complete set were actually less than those of their corresponding weekdays). Additionally, it can be noted that except for a few predictions, the MAEs of weekends are the highest among the three datasets of the same bridge and prediction horizon.

It can also be observed that with the increase of prediction horizon, MAE has always increased while R squared has always decreased for all datasets and all bridges. This, to be expected, decrease in the prediction accuracy with the increase in prediction horizon is reflected in Figure 26 and Figure 27. Figure 26 and Figure 27 compare the actual delay with the predicted delay for the next 5 and 30 minutes respectively. It can be seen that the model is able to very well follow the actual delay in the case of the next 5 minute delay prediction, while its performance has decreased when predicting 30 minutes into the future, especially for the high peaks arriving abruptly.

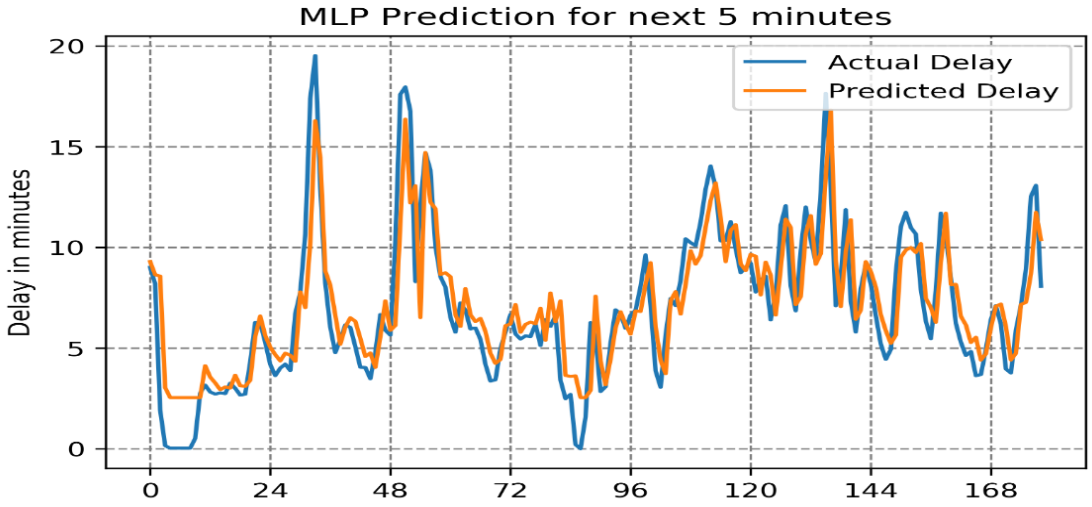


Figure 26. Comparing the actual U.S. bound traffic delay with 5 minutes ahead prediction of delay by the MLP model at Peace Bridge for a sample of 180 data points

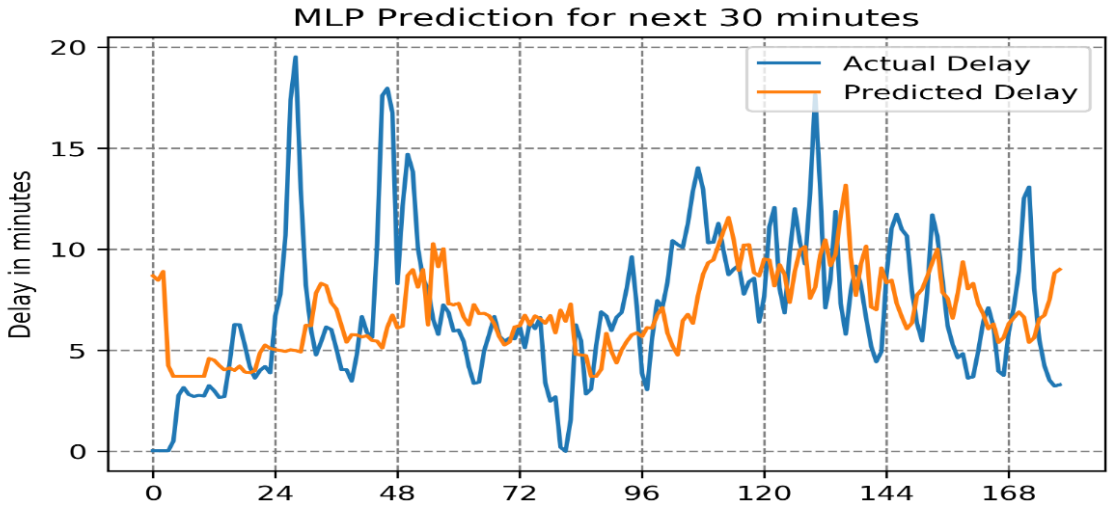


Figure 27: Comparing the actual U.S. bound traffic delay with 30 minutes ahead prediction of delay by the MLP model at Peace Bridge for a sample of 180 data points

Convolutional Neural Networks (CNN)

shows the model results of the CNN model in predicting delay for the next 5, 15, 30. 45, and 60 minutes. From Table 8, it is clear that the MAEs of weekdays are mostly lesser their corresponding complete set, whereas, weekends always had the highest MAE among the three datasets of the same bridge and same prediction horizon.

Table 8: CNN model results

Prediction Horizon	Measure	PB			QL			RB		
		Whole Set	Week days	Week ends	Whole set	Week days	Week ends	Whole Set	Week days	Week ends
5 minutes	MAE	0.91	0.98	1.09	0.93	1.01	1.38	1.14	0.98	1.22
	RMSE	2.86	2.79	3.85	1.96	2.15	2.55	1.8	1.58	2.02
	R^2	0.95	0.94	0.96	0.96	0.94	0.94	0.93	0.94	0.95
15 minutes	MAE	1.97	2.1	2.11	1.93	1.85	2.29	2.07	1.93	2.19
	RMSE	6.82	6.22	8.09	7.59	6.98	9.46	3.35	2.96	4.11
	R^2	0.8	0.73	0.85	0.83	0.8	0.84	0.79	0.78	0.85
30 minutes	MAE	2.57	2.64	2.72	2.58	2.42	3.04	2.7	2.51	3.03
	RMSE	3.9	3.79	4.3	4.79	4.67	5.42	5.21	4.85	5.43
	R^2	0.66	0.56	0.75	0.71	0.67	0.72	0.65	0.65	0.72
45 minutes	MAE	2.9	2.97	3.19	2.91	2.72	3.52	3.02	2.85	3.51
	RMSE	4.45	4.21	5.03	5.51	5.36	6.37	5.84	5.6	6.39
	R^2	0.56	0.46	0.65	0.61	0.57	0.62	0.57	0.53	0.61
60 minutes	MAE	3.12	3.03	3.5	3.16	2.94	3.89	3.27	3.04	3.89
	RMSE	4.77	4.42	5.53	6.09	5.85	7.09	6.34	6.05	7.17
	R^2	0.49	0.4	0.58	0.53	0.49	0.52	0.49	0.45	0.51
Computation time (s)		266.41	268.21	124.1	479.51	264.37	382.8	358.26	484.07	332.62

Similar to MLPs, the MAEs have always increased and R squared has always decreased with the increase in prediction horizon. The longest computation time for any CNN model developed in this study was 484.07 seconds, while the minimum was 124.1 seconds.

Figure 28 compares the actual and predicted delay for the next 30 minutes by the CNN model. It can be seen from **Figure 28**, that the prediction made by this model is able to follow the trend of the time series.

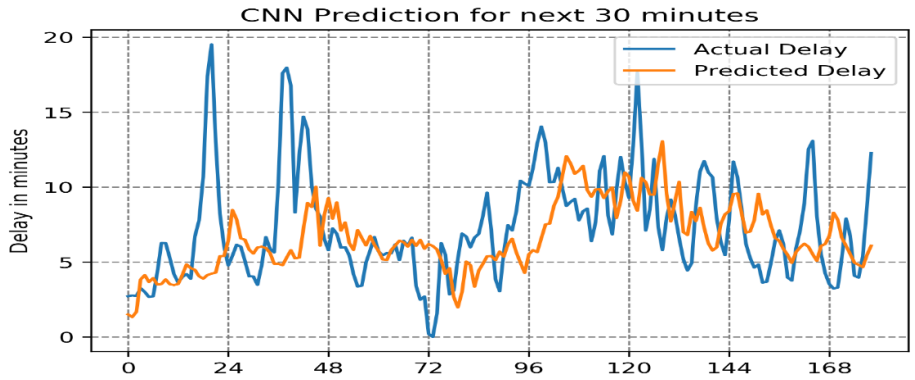


Figure 28. Comparing the actual U.S. bound traffic delay with 30 minutes ahead prediction of delay by CNN model at Peace Bridge for a sample of 180 data points

Long Short-Term Memory Recurrent Neural Networks (LSTM RNN)

The model results of LSTM RNN model are tabulated in Table 9. From Table 9 it can be noted that the MAE of weekdays are always the least among the whole set or the weekends, while the weekends are often the highest. With the increase in the prediction horizon, the MAEs have increased, while R squared has reduced. The computation time of LSTM RNN models developed in this study varied from 50.41 seconds to 184.07 seconds.

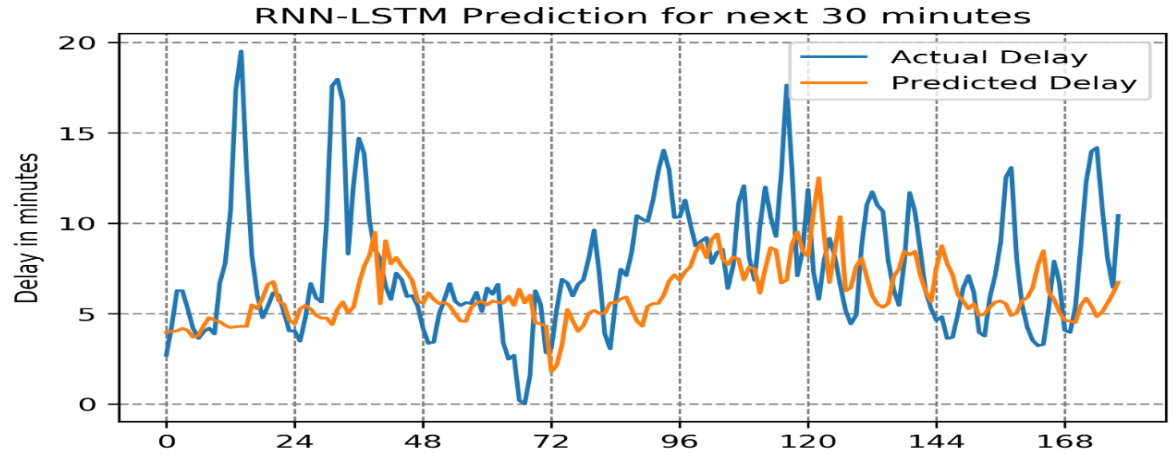


Figure 29 shows the graphical representation of actual delay and delay predicted by LSTM RNN model. The graph shows the ability of the model to tune its prediction with the changes in the prior time steps of the time series.

Table 9. LSTM RNN model results

Prediction Horizon	Measure	PB			QL			RB		
		Whole set	Week days	Week ends	Whole Set	Week days	Week ends	Whole Set	Week days	Week ends
5 minutes	MAE	1.02	0.91	0.95	1.14	0.86	1.34	1.27	0.92	1.51
	RMSE	3.33	3.16	3.2	2.45	1.94	2.58	1.9	1.5	2.37
	R^2	0.93	0.93	0.97	0.94	0.95	0.95	0.86	0.94	0.87
15 minutes	MAE	2.01	1.93	1.98	1.99	1.84	2.27	2.14	1.87	2.43
	RMSE	6.95	6.28	8.06	7.37	6.86	8.85	3.31	3	4.06
	R^2	0.78	0.72	0.86	0.82	0.81	0.85	0.74	0.79	0.76

30 minutes	MAE	2.51	2.44	2.66	2.6	2.43	3.04	2.73	2.47	3.16
	RMSE	3.96	3.73	4.35	4.92	4.58	5.3	5.35	4.75	6.17
	R^2	0.65	0.57	0.74	0.69	0.69	0.74	0.64	0.66	0.64
45 minutes	MAE	2.88	2.74	3.07	2.94	2.72	3.53	3.09	2.78	3.57
	RMSE	4.46	4.17	5.02	5.51	5.26	6.21	6.01	5.4	6.85
	R^2	0.56	0.47	0.65	0.61	0.58	0.64	0.54	0.56	0.56
60 minutes	MAE	3.12	3.01	3.39	3.14	2.93	3.92	3.34	2.99	3.94
	RMSE	4.85	4.46	5.51	6.07	5.73	6.98	6.5	5.85	7.52
	R^2	0.47	0.39	0.58	0.53	0.51	0.54	0.46	0.49	0.46
Computation time (s)		184.07	136.06	67.39	167.89	171.03	62.85	166.93	150.95	50.41

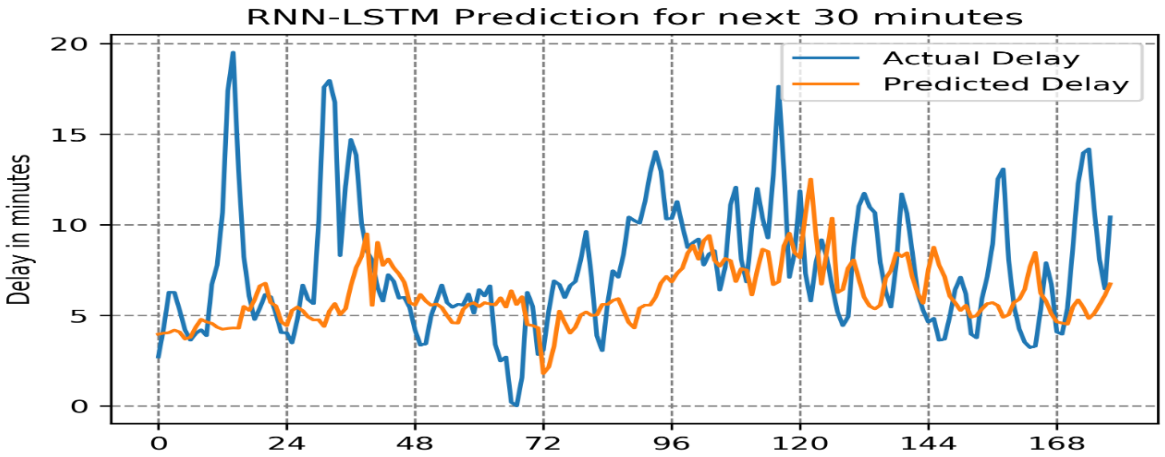


Figure 29 : Comparing the actual U.S. bound traffic delay with 30 minutes ahead prediction of delay by LSTM RNN model at Peace Bridge for a sample of 180 data points

Gated Recurrent Unit Recurrent Neural Networks (GRU RNN)

Table 10 shows the prediction results for the next 5, 15, 30, and 45 minutes, where it can be seen that the MAEs of weekdays are usually less than that of their corresponding complete set. Similar to the MLP, CNN, and LSTM RNN, the MAE and R squared of GRU RNN increases and reduces respectively with the increase in prediction horizon. The computation time of GRU RNNs developed for this study ranges between 30.09 seconds and 173.5 seconds.

Table 10: GRU RNN model results

Prediction Horizon	Measure	PB			QL			RB		
		Whole Set	Week days	Week ends	Whole set	Week days	Week ends	Whole set	Week days	Week ends
5 minutes	MAE	1.21	0.99	1.13	0.96	1.05	1.23	1.37	1.27	1.16
	RMSE	2.88	2.8	5.22	2.04	2.48	2.36	2.01	1.66	1.93
	R^2	0.94	0.94	0.95	0.96	0.93	0.96	0.89	0.84	0.96
15 minutes	MAE	2.11	2.08	2.02	1.93	1.92	2.17	2.26	2.13	2.19
	RMSE	7.01	6.26	7.97	7.48	6.68	8.93	3.52	2.96	4.13

	R^2	0.78	0.73	0.86	0.83	0.79	0.86	0.74	0.68	0.85
30 minutes	MAE	2.6	2.58	2.68	2.58	2.45	2.92	2.8	2.8	3.01
	RMSE	4.03	3.78	4.34	4.7	4.76	5.08	5.38	5.91	5.37
	R^2	0.64	0.56	0.74	0.72	0.66	0.76	0.63	0.48	0.73
45 minutes	MAE	2.96	2.79	3.17	2.93	2.74	3.41	3.18	3.13	3.47
	RMSE	4.53	4.15	5.05	5.42	5.44	5.95	6.19	6.58	6.31
	R^2	0.54	0.47	0.65	0.63	0.56	0.67	0.51	0.35	0.62
60 minutes	MAE	3.15	3.12	3.49	3.17	2.92	3.82	3.44	3.31	3.86
	RMSE	4.9	4.47	5.57	5.97	5.89	6.74	6.68	6.87	7.12
	R^2	0.46	0.39	0.57	0.55	0.48	0.57	0.43	0.29	0.52
Computation time (s)		173.5	111.96	41.47	207.15	111.63	30.75	165.34	114.08	30.09

Figure 30 shows the graphical representation of actual delay and delay predicted by GRU RNN model for the next 30 minutes, where it can be seen that the predicted values mostly follow the actual delay but misses out abrupt peaks.

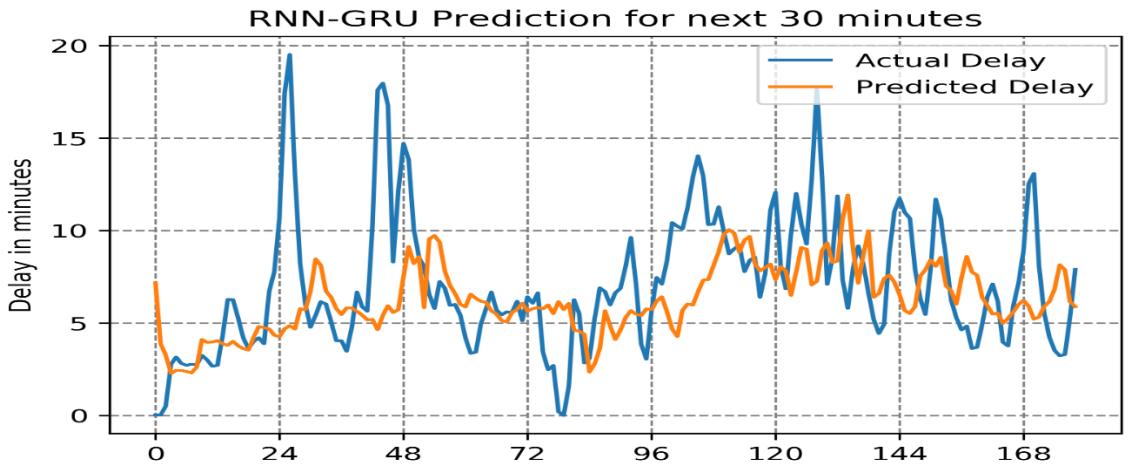


Figure 30: Comparing the actual U.S. bound traffic delay with 30 minutes ahead prediction of delay by GRU RNN model at Peace Bridge for a sample of 180 data points

Model Comparison

To find the most suitable deep learning technique in predicting short term delay, the model results of the four techniques used in this study were compared for all three bridges and for different prediction horizons. Figure 31and Figure 32 compare the four techniques based on the MAEs of the predicted delay at each bridge for 30 minutes ahead and 60 minutes ahead prediction respectively.

From Figure 31, it can be noticed that the lowest MAE for the next 30 minute prediction at PB and RB were obtained from LSTM RNN and MLP models, respectively. Whereas, both CNN and GRU RNN models gave the lowest MAE for 30-minute prediction at QL. Further, from Figure 32 it can be seen that the lowest MAE for the next 60-minute prediction at PB, QL, and

RB were given by MLP, LSTM RNN, and CNN model, respectively. Clearly, there does not seem to be an absolute winner among these techniques.

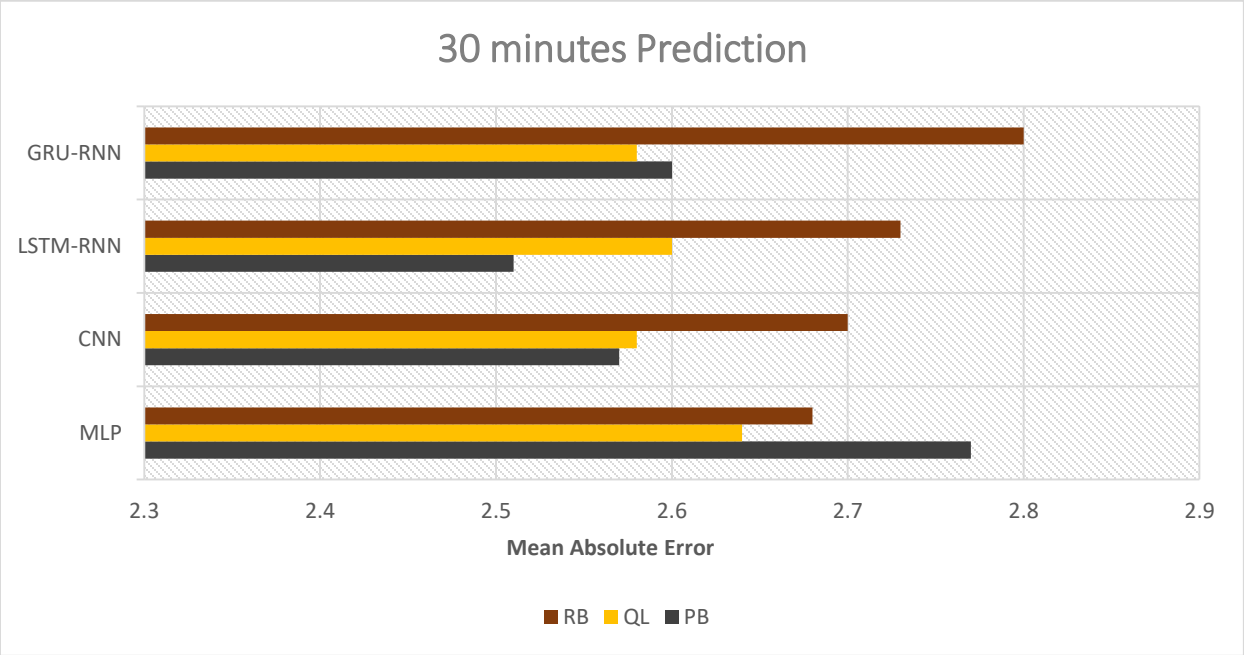


Figure 31: Comparing MAE of delay prediction for next 30 minutes by MLP, CNN, LSTM-RNN, and GRU-RNN at PB, QL, and RB

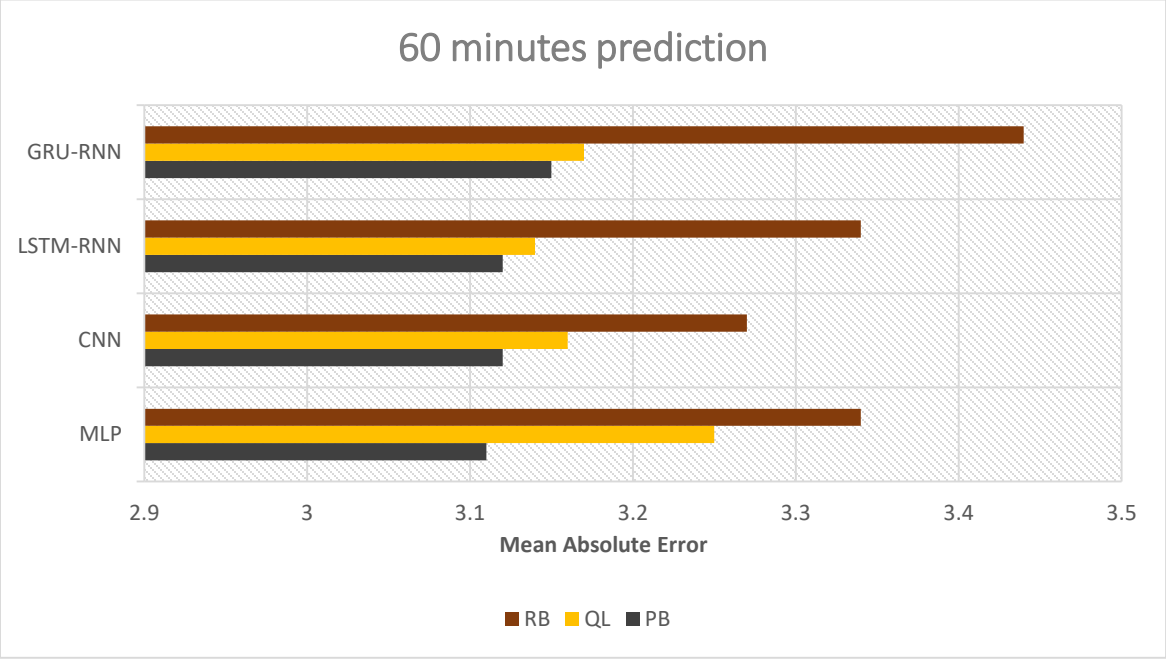


Figure 32: Comparing MAE of delay prediction for the next 60 minutes by MLP, CNN, LSTM-RNN, and GRU-RNN at PB, QL, and RB

As mentioned earlier, the delays at prior time steps were fed into the model as input. The number of prior time steps that were fed into the models was considered as one of the hyper-parameters of the model, which were selected using manual hyper-parameter tuning. In other words, the study tried using different numbers of prior time steps as input, and evaluated the results in order to determine the best number of prior time steps that should be used.

shows the number of prior time steps that gave the time steps for each model. It must be noted here that each time step represents a 5-minute interval. Hence, for instance, 12-time steps would refer to 12 X 5 = 60 minutes, i.e., delays for the last 60 minutes.

Table 11: Number of prior time steps used as input for different models

Border Crossing	Model	MLP	CNN	LSTM-RNN	GRU-RNN
Peace Bridge	Complete set	16	24	30	18
	Weekdays	16	18	24	18
	Weekends	24	16	18	48
Lewiston-Queenston Bridge	Complete set	24	18	60	36
	Weekdays	16	18	18	48
	Weekends	36	30	48	48
Rainbow Bridge	Complete set	16	30	48	36
	Weekdays	16	18	18	24
	Weekends	36	18	18	24

EFFECT OF DATA CLASSIFICATION ON MODEL RESULTS

As previously mentioned, the traffic delay time series dataset was classified into weekdays and weekends and thus, three different types of models were developed using the three datasets, which were: the complete or the whole set, the weekdays dataset, and the weekends dataset. This section presents the results of the analysis done to investigate the effect of data classification on model results.

The model performance of the complete set models may be compared with weekdays and weekends models by simply comparing the MAEs in their prediction. However, this might not give the most accurate results as it involves oversampling. This is because the MAEs obtained from the complete set models take into account both weekdays and weekends data. To evaluate the effect of data classification on the performance of deep learning models in a way that avoids the problem of oversampling, the models were run again and the MAEs in the prediction of a specific interval of weekday data points and weekend data points by the complete set model were compared with the MAEs in the prediction of the same data points by the weekdays and weekends model respectively. The length of this interval of data points used for the comparison was selected as 288 as it represents the observations of 24 hours ((24*60)/5 =288).

Table 12 shows the comparison of the MAEs in the prediction of weekday data points and weekend data points of the complete set models with the MAEs in the prediction of the same data points by the weekday and weekend model respectively for 30 minutes ahead U.S. bound traffic delay prediction at PB. From
Table 12, it can be seen that the MAEs of the weekday data points for the complete data set were consistently either less than (or equal to) those obtained for the weekday-specific model. On the other hand, the MAE for the weekend data-points for the complete set did not show a consistent trend, with the values for the complete set less than the weekend-specific model for MLP and LSTM-RNN, and greater than the weekend-specific model for CNN and GRU-RNN.

Models	Complete set		Weekday Models	Weekend Models
	Weekday data points	Weekend data points		
MLP	2.25	3.25	2.25	3.44
CNN	2.27	3.38	2.36	3.28
LSTM-RNN	2.15	3.29	2.19	3.39
GRU-RNN	2.13	3.39	2.2	3.21

Table 12: Comparison of MAEs of specific data points in the prediction of delay 30 minutes in future at PB by different models

To further analyze the effect of data classification, the same comparison was made for predictions 60 minutes in the future, as shown in Table 13.

Models	Complete set		Weekdays	Weekends
	Weekday data points	Weekend data points		
MLP	2.33	3.96	2.38	4.13
CNN	2.5	4.1	2.56	4.05
LSTM-RNN	2.45	3.91	2.48	4
GRU-RNN	2.3	4.01	2.52	4.06

Table 13: Comparison of MAEs of specific data points in the prediction of delay 60 minutes in the future at PB by different models

From Table 13, it was observed that mostly the MAEs of the weekday data points and weekend data points of complete set were lesser than the MAEs of weekdays model and weekends model respectively. However, there are exceptions to this. In all, it doesn't seem that the data classification increases the predictive accuracy of the models.

DISCUSSION

To the best of our knowledge, our study is the first time that (1) Bluetooth Data from the three Niagara Frontier Border Crossings are used for predicting the border crossing delay, and (2) Deep Learning techniques like CNN, LSTM-RNN, and GRU-RNN are used for predicting delay at the border crossings. The models developed in this study predict delay for next 5, 10, 15, …

and so on up to 60 minutes into the future. This information can guide the travelers in selecting the border crossing based on the delay situation. Travelers might have the tendency to choose the border crossing with the least delay and this would help in the uniform distribution of traffic across the crossings. Lin et al. (Lin, Wang, Sadek, et al., 2014) also emphasized on the necessity of predicting the future border crossing delay, which can be helpful for the border crossing authorities in determining the needed staffing level, and also in routing the border-destined traffic intelligently.

The delay prediction process used in this study did not include any data cleaning. The delays were just aggregated to 5 minutes and were fed to the models as an input. This makes the model development and prediction process very straight forward and easy. Unlike some of the previous studies (A. M. Khan, 2010; Lin F. B. & Lin M. W., 2001), the delay is predicted by models that were developed (trained, validated, and tested) using field data collected by the Bluetooth readers. Also for the delay analysis part, some of the previous studies (Zhang & Lin, 2017; Zhang et al., 2017) which analyzed the delay at Niagara Frontier Border Crossings lacked the availability of delay data collected by the Bluetooth readers from the Rainbow Bridge. However, the current study analyses the delay using the data collected from all three border crossings. This might have helped in making this research more realistic and appropriate for real-world applications.

The current study differs in many ways from the previous studies which aimed to predict the delays at the border crossings. Lin and Lin (Lin F. B. & Lin M. W., 2001) proposed a delay model which was based on various factors like vehicle processing capacity of a toll gate or inspection gate, volume/capacity ratio, number of available gates, etc. However, the models developed in the current study can predict delay by just using previous delays. Khan (A. M. Khan, 2010) and Moniruzzaman et. al. (Moniruzzaman et al., 2016) developed ANN for predicting the crossing time, whereas the current study extended these works by using deep learning techniques like CNN, LSTM-RNN, and GRU-RNN which have not been used for this purpose earlier.

Moniruzzaman et. al. (Moniruzzaman et al., 2016) forecasted the crossing times for trucks at the Ambassador Bridge border crossing through the ANN model and the training of the model relied on lags of crossing time, truck volume on the bridge, hours of day, and day of week. In contrast, the current study predicted the delays for the passenger cars on the Niagara Frontier Border Crossings. Another difference is that the training of models in the current study was based purely on previous delays at the crossings. Having just one variable might limit the prediction accuracy of the models but makes the model easy to develop and makes it easier to obtain the input data. Additionally, it was felt that the information of time of day is already taken into account by the model as the previous delays are fed as the input. Lastly, rather than just focusing on one border crossing as in the case of Moniruzzaman et. al. (Moniruzzaman et al., 2016), the current study focuses on predicting delay at three border crossings which are located in the same region. This clearly broadens the scope of the current study as the idea is not just to develop models which can predict delay at any border crossing but also to guide the travelers in making decision about selecting the crossing that is most appropriate to them, which in turn should help in increasing the efficiency of the borders.

In this study, the delays are predicted directly using the delay data collected from the Bluetooth readers. This can be seen as an improvement over the stepwise border crossing delay prediction model suggested in some of the past studies (Lin, Wang, & Sadek, 2014; Lin, Wang, Sadek, et al., 2014), in which first the future volumes were predicted using past volumes and then using the predicted future volumes, future delay were forecasted. This simplicity in the procedure is also reflected in the prediction accuracy of the current study. The mean absolute difference for 15 minutes ahead forecast of delay at Peace Bridge by this study was just 1.97 minutes (by complete set CNN model), while that obtained by Lin et. al. (Lin, Wang, Sadek, et al., 2014) was about 6.6 minutes. This suggests a higher prediction accuracy of the current study. Not just the improvement in prediction accuracy, the computation time of the models developed in this study were drastically shorter than that required by Lin et. al. (Lin, Wang, & Sadek, 2014).

The results from this study suggest high-level accuracy of the deep learning techniques in predicting future traffic delays at the border crossings, with MAEs less than 3.5 minutes in predicting delays for up to 60 minutes into the future by complete set models. Previous studies (Dalto et al., 2015; Fu et al., 2016; Koprinska et al., 2018; Ma et al., 2015) have also supported the superior prediction performance of some of the techniques used in this study. However, prediction-wise, no one deep learning technique emerged as a clear winner among others in predicting the delays. The best deep learning technique that gives the most accurate results changed with the border crossing and the prediction horizon. Although, mostly the prediction accuracies obtained from each of the technique were pretty close to each other. The hyper-parameters of the deep learning models were selected manually by observing the change in the models' performance on the validation set. The set of hyper-parameters that gave the least mean absolute error (MAE) was finalized. However, it was found that the MAEs did not change significantly by tuning the hyper-parameters, suggesting that the models are robust.

CONCLUSIONS AND FUTURE WORK

Traffic delays at the United States-Canada border crossings have adverse effects on the economy as well as the environment. There have been some studies in the past which aimed to analyze the delay at the Niagara Frontier Border Crossings (Zhang & Lin, 2017; Zhang et al., 2017) and others which predicted the future delay/crossing time at the border crossings (A. M. Khan, 2010; Lin F. B. & Lin M. W., 2001; Lin, Wang, & Sadek, 2014; Lin, Wang, Sadek, et al., 2014; Moniruzzaman et al., 2016). This study was intended to build upon these previous efforts.

This study predicted passenger cars' traffic delays at the three Niagara Frontier Border Crossings, namely the Peace Bridge, the Lewiston-Queenston Bridge, and the Rainbow Bridge for the next 60 minutes into the future. This was based upon border wait time data collected by Bluetooth readers recently installed at the crossings. Border crossing traffic delays were predicted using four deep learning techniques, namely Multilayer Perceptron (MLP), Convolutional Neural Networks (CNN), Long Short-Term Memory Recurrent Neural Networks (LSTM-RNN), and Gated Recurrent Unit Recurrent Neural Networks (GRU-RNN). The hyper-parameters of the models were selected through manual hyper-parametric tuning.

Limitations and Future Work

The current study focused on predicting future delay using four deep learning techniques, namely MLP, CNN, LSTM-RNN, and GRU-RNN. This task can also be performed by using some other

deep learning techniques so as to explore the prediction performance of those techniques. This research can also be extended by making predictions using statistical models and comparing their performance with the techniques employed in the current study. This can help in conducting comparative analysis among the deep learning techniques and statistical models.

In the current study, predictions were made using just previous delay as input to the deep learning models. The prediction performance can be improved by adding more input variables like car volume, number of open lanes, weather information, holidays, traffic accidents, etc. Lastly, similar future delay predicting models can also be developed for Canada bound traffic over the Niagara Frontier Border Crossings. Additionally, this work can be extended to other border crossings as well.

STUDY'S CONCLUSIONS

This project was intended to take advantage of the wealth of data, now available thanks to the recent advances in sensing and communications, to develop predictive models for predicting border crossing delays at the Niagara Frontier border crossing. Specifically, the project first developed an Android smartphone application to collect, share and predict waiting time at the three border crossings. Secondly, models, based on state-of-the-art Machine Learning (ML) techniques, were developed for interval prediction of short-term traffic volume at the border; these models were then utilized to determine optimal staffing levels at the border. Finally, by taking advantage of Bluetooth, border delay data recently collected at the three Niagara Frontier borders, the project developed deep learning models for the direct prediction of border delay. The suite of models and tools developed under this work have the potential to revolutionize border crossing management, balance traffic load at the three crossings, and help travelers avoid significant border delays.

REFERENCES

Aghdam, H. H., & Heravi, E. J. (2017). *Guide to convolutional neural networks: a practical application to traffic-sign detection and classification*. Springer International Publishing AG.

Alpaydin, E. (2016). *Machine learning: the new AI.* MIT Press, Cambridge, MA.

Ardit. (2017). *How to measure the execution time of a Python script.* Retrieved March 13, 2019, from PythonHow website: https://pythonhow.com/measure-execution-time-python-code/

Bianchi, F. M., Maiorino, E., Kampffmeyer, M. C., Rizzi, A., & Jenssen, R. (2017). *Recurrent neural networks for short-term load forecasting: an overview and comparative analysis.* Springer International Publishing AG. https://doi.org/10.1007/978-3-319-70338-1

Bollerslev, T. (1986). Generalized autoregressive conditional heteroskedasticity. *Journal of econometrics*, 31(3), 307-327.

Border delays costing billions: Ontario Chamber of Commerce laments crossing impact, says jobs and trade lost at "choke point." (2004). *Traffic World*, 268(28), 24. Retrieved from General OneFile.

Brownlee, J. (2016). *Time Series Prediction with LSTM Recurrent Neural Networks in Python with Keras*. Retrieved March 13, 2019, from Machine Learning Mastery website: https://machinelearningmastery.com/time-series-prediction-lstm-recurrent-neural-networks-python-keras/

Brownlee, J. (2018). *How to Develop Deep Learning Models for Univariate Time Series Forecasting*. Retrieved March 13, 2019, from Machine Learning Mastery website: https://machinelearningmastery.com/how-to-develop-deep-learning-models-for-univariate-time-series-forecasting/

Buffalo Sabres Schedule 2013-2014. http://sabres.nhl.com/club/schedule.htm?season=20132014. Accessed on 07/14/2016. Cryer, J.D., & Chan, K.S., 2008. Time Series Analysis: With Applications in R. Springer.

Dalto, M., Matuško, J., & Vašak, M. (2015). Deep neural networks for ultra-short-term wind forecasting. *2015 IEEE International Conference on Industrial Technology (ICIT)*, 1657–1663. https://doi.org/10.1109/ICIT.2015.7125335

Fu, R., Zhang, Z., & Li, L. (2016). Using LSTM and GRU neural network methods for traffic flow prediction. *31st Youth Academic Annual Conference of Chinese Association of Automation (YAC)*, 324–328. https://doi.org/10.1109/YAC.2016.7804912

Gabler, N., & Moens, A. (2012). *Measuring the Costs of the Canada-US Border*. Retrieved from The Fraser Institute website: https://www.fraserinstitute.org/

Gardner, M. W., & Dorling, S. R. (1998). Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. *Atmospheric Environment*, 32(14–15), 2627–2636. https://doi.org/10.1016/S1352-2310(97)00447-0

Ghalanos, A. (2013). Rugarch: Univariate GARCH Models, R package version 1.2-7.

Ghosh, B., Basu, B., & O'Mahony, M. (2009). Multivariate short-term traffic flow forecasting using time-series analysis. IEEE Transactions on Intelligent Transportation Systems 10(2), 246-254.

GoDaddy. (2014). Available at https://www.godaddy.com. Accessed on 11/05/2014.

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

Guo, J., & Williams, B. (2010). Real-time short-term traffic speed level forecasting and uncertainty quantification using layered Kalman filters. *Transportation Research Record: Journal of the Transportation Research Board,* (2175), 28-37.

Guo, J., Huang, W., & Williams, B. M. (2014). Adaptive Kalman filter approach for stochastic short-term traffic flow rate prediction and uncertainty quantification. *Transportation Research Part C: Emerging Technologies*, 43, 50-64.

Huang, G. B., Zhu, Q. Y., & Siew, C. K. (2006). Extreme learning machine: theory and applications. *Neuro-computing*, 70(1), 489-501.

Hyndman, R. J., & Khandakar, Y. (2007). *Automatic time series for forecasting: the forecast package for R (No. 6/07).* Monash University, Department of Econometrics and Business Statistics.

Jongbloed, G., & Koole, G. (2001). Managing uncertainty in call centres using Poisson mixtures. *Applied Stochastic Models in Business and Industry*, 17(4), 307-318.

Kamarianakis, Y., Kanas, A., & Prastacos, P., 2005. Modeling traffic volatility dynamics in an urban network. *Transportation Research Record: Journal of the Transportation Research Board*, (1923), 18-27.

Karlaftis, M. G., & Vlahogianni, E. I. (2011). Statistical methods versus neural networks in transportation research: Differences, similarities and some insights. *Transportation Research Part C: Emerging Technologies*, 19(3), 387-399.

Kennedy, J. (2011). *Particle swarm optimization*. In Encyclopedia of machine learning (pp. 760-766). Springer US.

Keras Documentation. (n.d.). Retrieved March 13, 2019, from https://keras.io/

Khan, A. M. (2010). Prediction and Display of Delay at Road Border Crossings. *The Open Transportation Journal*, 4(1). Retrieved from https://benthamopen.com/ABSTRACT/TOTJ-4-9

Khan, S., Rahmani, H., Shah, S. A. A., & Bennamoun, M. (2018). *A guide to convolutional neural networks for computer vision*. https://doi.org/10.2200/S00822ED1V01Y201712COV015

Khosravi, A., Mazloumi, E., Nahavandi, S., Creighton, D., & Van Lint, J. W. C. (2011). Prediction intervals to account for uncertainties in travel time prediction. *IEEE Transactions on Intelligent Transportation Syste*ms, 12(2), 537-547.

Koprinska, I., Wu, D., & Wang, Z. (2018). Convolutional Neural Networks for Energy Time Series Forecasting. *2018 International Joint Conference on Neural Networks (IJCNN)*, 1–8. https://doi.org/10.1109/IJCNN.2018.8489399

Kortbeek, N., Braaksma, A., Burger, C. A., Bakker, P. J., & Boucherie, R. J. (2015). Flexible nurse staffing based on hourly bed census predictions. *International journal of production economics*, 161, 167-180.

Lin F. B., & Lin M. W. (2001). Modeling Traffic Delays at Northern New York Border Crossings. *Journal of Transportation Engineering*, 127(6), 540–545.

Lin, L., Wang, Q., & Sadek, A. W. (2012). Multiple-Model Combined Forecasting Method for Online Prediction of Border Crossing Traffic at Peace Bridge. In *Transportation Research Board 91st Annual Meeting*, No. 12-3398.

Lin, L., Wang, Q., & Sadek, A.W. (2013). Short-term forecasting of traffic volume: evaluating models based on multiple data sets and data diagnosis measures. *Transportation Research Record: Journal of the Transportation Research Board*, (2392), 40-47.

Lin, L., Wang, Q., Huang, S., & Sadek, A. W. (2014a). On-line prediction of border crossing traffic using an enhanced Spinning Network method. *Transportation Research Part C: Emerging Technologies*, 43, 158-173.

Lin, L., Wang, Q., & Sadek, A. W. (2014b). Border crossing delay prediction using transient multi-server queueing models. *Transportation Research Part A: Policy and Practice*, 64, 65-91.

Lin, L., Wang, Q., Sadek, A. W., & Kott, G. (2015). An Android Smartphone Application for Collecting, Sharing, and Predicting Border Crossing Wait Time. In *Transportation Research Board 94th Annual Meeting* (No. 15-1971).

Lin, L., J.C. Handley, X. Wen, and A.W. Sadek. A (2018). Hybrid Machine Learning Model for Interval Prediction of Short-term Traffic Volume and its Application to Optimal Staffing Level Plan Development. *Transportation Research Part C 92, pp. 323–348.*

Lv, Y., Duan, Y., Kang, W., Li, Z., & Wang, F. Y. (2015). Traffic flow prediction with big data: a deep learning approach. *IEEE Transactions on Intelligent Transportation Systems*, 16(2), 865-873.

Lwebuga-Mukasa, J.S., S.J. Ayirookuzhi and A.Hyland. (2002). Traffic Volumes and Respiratory Health Care Utilization among Residents in Close Proximity to the Peace Bridge before and after September 11, 2001. *Journal of Asthma*, Vol. 40, No. 8, pp. 855-864.

Ma, X., Tao, Z., Wang, Y., Yu, H., & Wang, Y. (2015). Long short-term memory neural network for traffic speed prediction using remote microwave sensor data. *Transportation Research Part C: Emerging Technologies*, 54, 187-197.

Masum, S., Liu, Y., & Chiverton, J. (2018). Multi-step Time Series Forecasting of Electric Load Using Machine Learning Models. In *L. Rutkowski, R. Scherer, M. Korytkowski, W. Pedrycz, R. Tadeusiewicz, & J. M. Zurada (Eds.), Artificial Intelligence and Soft Computing* (pp. 148–159). Springer International Publishing.

Matplotlib: Python plotting — Matplotlib 3.0.3 documentation. (n.d.). Retrieved March 13, 2019, from https://matplotlib.org/

Moniruzzaman, M., Maoh, H., & Anderson, W. (2016). Short-term prediction of border crossing time and traffic volume for commercial trucks: A case study for the Ambassador Bridge. *Transportation Research Part C: Emerging Technologies*, 63, 182–194. https://doi.org/10.1016/j.trc.2015.12.004

Niagara Falls Bridge Commission. (2014). Available at http://niagarafallsbridges.com. Accessed on 04/12/2014.

Ontario Chamber of Commerce (OCC). (2005). *Cost of Border Delays to the United States Economy*, 2005. Available online at: http://www.thetbwg.org/downloads/Cost%20of%20Border%20Delays%20to%20the%20United%20States%20Economy%20 - %20April%202005.pdf.

Pal, A., & Prakash, P. (2017). Deep Learning for Time Series Forecasting. In *Practical Time Series Analysis*. Retrieved from https://app.knovel.com/hotlink/pdf/id:kt011LNBAP/practical-time-series/deep-learning-time-series

Peace Bridge. (2014). Available at www.peacebridge.com. Accessed on 04/12/2014.

Petris, G., Petrone, S., & Campagnoli, P. (2009). *Dynamic Linear Models with* R (pp. 31-84). Springer, New York.

Pinson, P., Chevallier, C., & Kariniotakis, G. N. (2007). Trading wind generation from short-term probabilistic forecasts of wind power. *IEEE Transactions on Power Systems*, 22(3), 1148-1156.

Polson, N. G., & Sokolov, V. O. (2017). Deep learning for short-term traffic flow prediction. *Transportation Research Part C: Emerging Technologies*, 79, 1–17.

Roelofs, T., Preisen, L., & Helgeson, C. (2016). Performance measures and reporting for international border crossings. *ENTERPRISE Transportation Pooled Fund Study TPF-5 (231)*. Retrieved from https://rosap.ntl.bts.gov/view/dot/30813

scikit-learn: machine learning in Python — scikit-learn 0.20.3 documentation. (n.d.). Retrieved March 13, 2019, from https://scikit-learn.org/stable/

Skansi, S. (2018). *Introduction to Deep Learning: From Logical Calculus to Artificial Intelligence*. Springer International Publishing, AG. https://doi.org/10.1007/978-3-319-73004-2

Stephenson, P. (2016). Waiting on the Canada-U.S. Border. *Mobility in History*, 7(1). https://doi.org/10.3167/mih.2016.070118

Steinfeld, A., J. Zimmerman, A. Tomasic, D. Yoo, and R. D. Aziz. (2011). Mobile Transit Information from Universal Design and Crowdsourcing. *Transportation Research Record: Journal of the Transportation Research Board,* Vol. 2217, No. 1, pp. 95-102.

U.S. Department of Transportation (USDOT), Office of Public Affairs. (2008). *US Department of Transportation Unveils New Program to Fight Border Congestion*. Available at: http://www.dot.gov/affairs/fhwa1208.htm.

VanderPlas, J. (2016). *Python Data Science Handbook*. Retrieved March 13, 2019 from https://jakevdp.github.io/PythonDataScienceHandbook/04.10-customizing-ticks.html

Wan, C., Xu, Z., Pinson, P., Dong, Z. Y., & Wong, K. P., 2014. Optimal prediction intervals of wind power generation. *IEEE Transactions on Power Systems*, 29(3), 1166-1174.
Weather Underground, Accessed on 2016. https://www.wunderground.com/history/

Yu, M., Ding, Y., Lindsey, R., & Shi, C., 2016. A data-driven approach to manpower planning at US–Canada border crossings. *Transportation Research Part A: Policy and Practice*, 91, 34-47.

Zhang, Z., & Lin, L. (2017). Abnormal Spatial-Temporal Pattern Analysis for Niagara Frontier Border Wait Times. *ITS World Congress 2017 Montreal* https://arxiv.org/abs/1711.00054

Zhang, Z., Lin, L., Zhu, L., & Sharma, A. (2017). Bi-National Delay Pattern Analysis For Commercial and Passenger Vehicles at Niagara Frontier Border. *ArXiv:1711.09723 [Cs]*. Retrieved from http://arxiv.org/abs/1711.09723

Zhang, Y., Zhang, Y., & Haghani, A., 2014. A hybrid short-term traffic flow forecasting method based on spectral analysis and statistical volatility model. *Transportation Research Part C: Emerging Technologies*, 43, 65-78.

Zimmerman, J., A. Tomasic, C. Garrod, D. Yoo, C. Hiruncharoenvate, R. Aziz, and A. Steinfeld. Field Trial of Tiramisu: Crowd-Sourcing Bus Arrival Times to Spur Co-design, 2011. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 1677-1686.