Journal of Ambient Intelligence and Humanized Computing
https://doi.org/10.1007/s12652-020-02507-9

ORIGINAL RESEARCH

# Predicting irregularities in arrival times for transit buses with recurrent neural networks using GPS coordinates and weather data

**Omar Alam**[1] · **Anshuman Kush**[1] · **Ali Emami**[2] · **Parisa Pouladzadeh**[3]

Received: 26 March 2020 / Accepted: 27 August 2020


**Abstract**
Intelligent transportation systems (ITS) play an important role in the quality of life of citizens in any metropolitan city. Despite various policies and strategies incorporated to increase the reliability and quality of service, public transportation authorities continue to face criticism from commuters largely due to irregularities in bus arrival times, most notably manifested in early or late arrivals. Due to these irregularities, commuters may miss important appointments, wait for too long at the bus stop, or arrive late for work. Therefore, accurate prediction models are needed to build better customer service solutions for transit systems, e.g. building accurate mobile apps for trip planning or sending bus delay/cancel notifications. Prediction models will also help in developing better appointment scheduling systems for doctors, dentists, and other businesses to take into account transit bus delays for their clients. In this paper, we seek to predict the occurrence of arrival time irregularities by mining GPS coordinates of transit buses provided by the Toronto Transit Commission (TTC) along with hourly weather data and using this data in machine learning models that we have developed. In our study, we compared the performance of a Long Short Term Memory Recurrent Neural Network (LSTM) model against four baseline models, an Artificial Neural Network (ANN), Support Vector Regression (SVR), Autoregressive Integrated Moving Average (ARIMA) and historical averages. We found that our LSTM model demonstrates the best prediction accuracy. The improved accuracy achieved by the LSTM model may lend itself to its ability to adjust and update the weights of neurons while accounting for long-term dependencies. In addition, we found that weather conditions play a significant role in improving the accuracy of our models. Therefore, we built a prediction model that combines an LSTM model with a Recurrent Neural Network Model (RNN) that focuses on the weather condition. Our findings also reveal that in nearly 37% of scheduled arrival times, buses either arrive early or late by a margin of more than 5 min, suggesting room for improvement in the current strategies employed by transit authorities.

**Keywords** Intelligent transportation systems · ITS · Traffic flow · Neural networks · GPS locations · Weather conditions

## 1 Introduction

The importance of modeling and predicting bus arrival times for public transit has long been recognized (Kumar et al. 2014). Throughout the past decade, much work has been done to explore means of achieving faster and more reliable transit systems (Hua et al. 2018). However, public transit authorities continue to face criticisms from commuters due to discrepancies between a vehicle's scheduled and actual arrival times. These irregularities naturally have a negative impact on the commuter's daily life. Commuters may miss medical appointments, school events, or arrive late for work. With the availability of large scale pervasive data, e.g. GPS locations collected from buses, we believe that machine learning algorithms can help in predicting actual

✉ Omar Alam
omaralam@trentu.ca

Anshuman Kush
anshumankush@trentu.ca

Ali Emami
ali.emami@mail.mcgill.ca

Parisa Pouladzadeh
Parisa.Pouladzadeh@flemingcollege.ca

[1] Trent University, Peterborough, Canada

[2] Mila/McGill University, Montreal, Canada

[3] Fleming College, Peterborough, Canada

Published online: 03 September 2020

arrival times for public transit buses, and assist in strategies to overcome their discrepencies with scheduled times.

This paper aims at modelling the irregularities in arrival times for public transit buses using historical bus arrival times, stop locations, bus schedules, and weather data. Irregularities can be considered to occur in one of two ways, leads (early arrival at a stop) and delays (late arrival at a stop). We focused on predicting irregularities for transit buses for the City of Toronto, where irregularities in bus arrival times are so commonly occurring that Toronto Transit Commission (TTC) issues notes for commuters who arrive late for work due to misleading scheduling times (Star 2020).

To reduce irregularities in arrival times, transit authorities incorporate a variety of strategies to bridge the gap between actual and scheduled arrival times of buses. Among these strategies, the holding control strategy is found to be the most effective to regulate bus operations (Fu and Yang 2002). This strategy seeks to address the phenomenon called bus headway, which is a large, accumulated arrival lead or delay in a bus stop that results from a succession of leads or delays that occurred in previous stops. By holding an early-arriving bus, a bus headway can be mitigated and service reliability can be improved (Fu and Yang 2002). Another strategy is stop-skipping, which is particularly useful when buses are running late and behind their schedule (Liu et al. 2013). Despite applying these strategies, transit services continue to face delays in their daily operations, which could be due to ongoing road constructions, bus breakdown, road accidents, or other day-to-day factors. Therefore, transit authorities seek to increase the quality of service by providing passengers with predicted arrival times at a bus stop using algorithms that exploit transit data (Hua et al. 2018). With computational power becoming cheaper and easily accessible, it is increasingly feasible to use data driven models for accurately predicting arrival times by leveraging a large volume of data. These prediction models can assist in developing intelligent trip planning apps, improved scheduling systems for doctors and other businesses, and improving urban planning strategies for city authorities.

In this paper, we propose a regression task to test the ability for machine learning algorithms to predict whether a bus at a given stop and time will be early, on time, or late based on the transit and weather data for Toronto Transit Commission (TTC). The machine learning models that we experiment with include traditional feed-forward artificial neural network (ANN) and a recurrent neural network (RNN) using long short term memory (LSTM).

Our contribution can be summarized as follows:

- To our knowledge, this is the first work that investigates the impact of weather data on prediction accuracy for bus arrival times. We compare the prediction models with and without weather features. Previous work either avoided using weather data altogether, e.g. (Kumar et al. 2014) or did not find weather to be a useful feature for their prediction task (Patnaik et al. 2004).
- We used historical arrival times, weather data, and other input features for arrival time prediction for transit buses. We found that the LSTM model, a variant of Recurrent Neural Network that uses long term dependencies, yields the best predictive performance.
- We found that weather has strong relationship with arrival time prediction models. In nearly half of our data, including weather improved the prediction accuracy by 48%. We also found that including the weather data significantly improves the accuracy when predicting bus arrival times at multiple future stops in a trip.
- Because of the importance of weather, we built a separate RNN model that focuses on the weather feature and combined its result with the result of the LSTM model. This combined hybrid model improved the prediction by more than 500%.

The rest of the paper is organized as follows. The next section discusses the related work. Section 3 discusses the data collection. Section 4 discusses the machine learning models that we used. Section 5 discusses the results and Sect. 6 concludes the paper.

## 2 Related work

This section discusses related work on bus arrival prediction. In general, previous work used linear regression (LR) (Hua et al. 2018), non-parametric regression (NPR) (Chang et al. 2010; Balasubramanian and Rao 2015), or Kalmann Filters (KFT) (Shalaby and Farhan 2004).

Hua et al. (2018) use linear regression to predict bus locations. Bus location data displays non-linear relationships between its features. Therefore, data has to be converted into a linear space to be used in conventional mathematical models such as linear regression. This requires a significant amount of data pre-processing and be in turn, costly and time-consuming. Kormáksson et. al. (2014) use additive models (non linear regression models) to predict bus arrival times using General Transit Feed Specification (GTFS) data. GTFS data is standardized by Google, which is used to provide schedules and geographic information to Google Maps and other Google applications that show transit information. Regression models are easy to interpret and fast to train. Shalaby and Farhan (2004) use very limited AVL (Automatic Vehicle Location) and APC (Automatic Passenger Counter) data on Kalmann Filters (KFT) to predict the arrival time for Toronto Transit Buses. Their data size is small (only 5 days of vehicle locations). In our study, we used 3.5 million data points which were collected over a

period of four months. We use large datasets for predicting arrival times using machine learning algorithms. Wang et. al. (Wang et al. 2019) applies a multi-objective optimization technique to reduce the capacity allocation of subway systems based on different factors, e.g. number of passengers, headway, number of available trains. Their objective is to reduce the passenger wait time in the subway system. Similarly, passenger travel time was used to predict the arrival time in subway stations a in study done for the city of Beijing (Xu et al. 2020). We used several input features in our prediction models, e.g. past arrival time, day of the week, hour of the day, etc. Liu et al. (2019) studied the optimal combination of different input features in mass rapid transit (MRT) systems. However, they did not consider the weather in their study.

Kumar et. al. (2014) compared Kalmann Filters (1960) with artificial neural network for bus arrival prediction in Chennai, India. A key finding of this experiment was that with a large volume of data, artificial neural network models give better accuracy as compared to mathematical models (linear regression and kalmann filters). Wang et. al. (2009) use a Support Vector Machine (SVM) to model traffic conditions. They used bus arrival times and bus schedules as inputs to train their model. ANN and kernalized SVM have gained popularity for predicting travel time because of their ability to solve complex and non-linear relationships among features (Chien et al. 2002; Kumar et al. 2014; Jeong and Rilett 2004). In Hua et al. (2018), the performance of linear regression, artificial neural networks and support vector machine models were compared for prediction of bus arrivals at a single stop using data from multiple routes. Linear regression's performance was poor due to non-linearity in data, however the performance of ANN and SVM were quite competitive. These approaches did not use recurrent neural networks in their predictions. Our work uses LSTM recurrent neural networks. Moreover, we use weather data in our prediction, which had not been incorporated in any way in the previous approaches.

To our knowledge, there has not been an abundance of work that uses weather data for predicting arrival times for public transit buses. Yang et. al. (2016) use a combination of genetic algorithms and support vector machines along with weather conditions to predict bus arrival time. They did not use historical arrival times and did not explore recurrent neural networks in their study. Chen et. al. ( 2004) used weather condition and automatic passenger counting data with ANN for bus arrival prediction for New Jersey county. The previous two studies only relied on weather conditions (i.e. snow, rain, fog) in their models. We consider other weather attributes, such as visibility and temperature. Patnaik et al. (2004) used weather data as features for bus arrival prediction model, however their experiment failed to show improvement with weather data.

Ke et. al. used a combination of CNN and LSTM Recurrent Neural Networks along with weather data for forecasting short-term passenger demand for ride services (Ke et al. 2017). In contrast, we use weather data for a different problem, i.e. to predict arrival times of transit buses. Rui et.al. compared the performance of a GRU Model (Gated Recurrent Neural Network) and LSTM model on yet another prediction task concerning traffic flow prediction (Fu et al. 2016).

## 3 Dataset collection

We used four datasets to build our models: (1) Live Automatic Vehicle Locations (AVL) data for Toronto Transit Comission (TTC) transit buses, collected every 20 s, (2) bus schedules (3) and bus stop locations retrieved from GTFS (General Transit Feed Specification) data, (4) hourly weather data collected from a weather station near downtown Toronto. The AVL data comprises of GPS locations for Toronto Transit Commission (TTC) buses. This data is publicly available through the NextBus API (Nextbus 2020). We collected more than 700,000 unique live GPS locations for transit buses for two routes, Route 28 and Route 8 (Fig. 1) for the City of Toronto over 3 months, from January 2018 to March 2018. Figure 2 presents

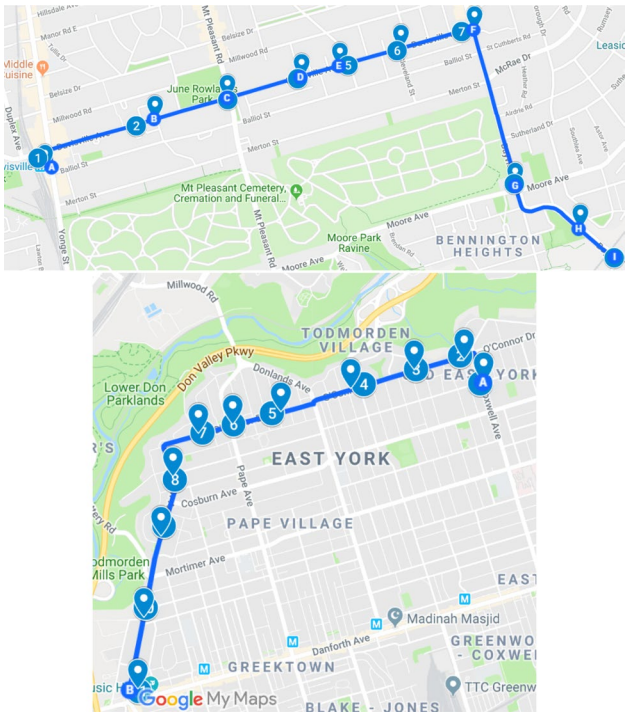


**Fig. 1** GPS locations mapped to bus stop location data for TTC routes. Markers are the GPS coordinates calculated for actual arrival time at each stop. The top map depicts Route 28, while the bottom map depicts Route 8

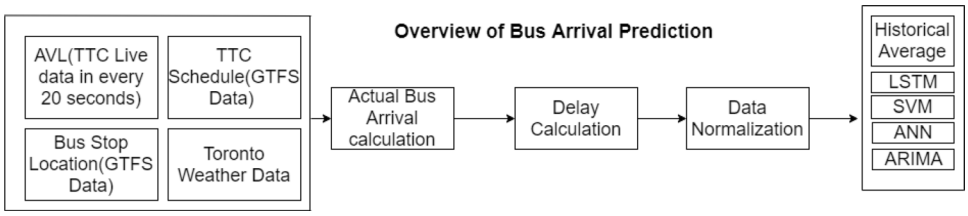


**Fig. 2** An overview of bus arrival prediction

**Table 1** Datasets used for our study

| | Data points |
|---|---|
| TTC real time data | 700,000 |
| GTFS bus stop schedule data | 18,110 |
| GTFS bus stop location data | 24 |
| Weather data | 3624 |

an overview of our study. Table 1 summarizes the datasets that we used in our study. After collecting the four datasets, we calculated the arrival time for a bus at each bus stop in the studied routes. Then, we calculated the difference between the actual arrival time of a bus at a stop and its scheduled arrival time in that stop. Based on this difference, we determined if the bus had arrived early, on time, or was delayed. Then, we normalized the data from all four datasets and used them as inputs to our models.

### 3.1 Estimating actual arrival time

The TTC data does not, in fact, specify whether a bus had arrived at a stop. The actual bus arrival time of a bus at a stop is calculated using the distance between the GPS location of the bus and the bus stop location. This distance is calculated using the *haversine* formula (Veness 2018), which is a well-known formula used to calculate the path distance between two points on the surface of the earth, and has wide range of applications, e.g. (Chopde and Nichat 2013; Basyir et al. 2017; Ingole and Nichat 2013). The formula gives the distances between two points on a sphere using their latitudes and longitudes while ignoring hills:

$$a = \sin^2(\Delta\varphi/2) + \cos\varphi_1 * \cos\varphi_2 * \sin^2(\Delta/2) \quad (1)$$

$$c = 2 * atan2(\sqrt{a}, \sqrt{1-a}) \quad (2)$$

$$d = R * c \quad (3)$$

In Equation 1, $\varphi$ is latitude, $\Lambda$ is longitude. In Eq. 2, $c$ is the angular distance in radians. In Eq. 3, $R$ is Earth's radius (mean radius = 6371 km) and $d$ is the distance between two GPS locations in kilometer. Since the real time GPS location data is collected for every 20 s, we may miss the exact time when the bus actually arrives at the bus stop. Furthermore, during a 20 s window, the bus could arrive at a bus stop and start moving again. In that case, the recorded GPS location of the bus could be further away from the bus stop.

To mitigate these issues, we identify the GPS location where the distance between the bus and the bus stop is minimal. We do this by checking whether the bus is close to the bus stop, where closeness corresponds to the bus being within 100 m from the stop.

**Algorithm 1:** Calculating the Actual Arrival Time of a Bus

**Input:** GPSTime: Reported time for the GPS location of the bus, ScheduledTime: Scheduled arrival time of the bus at the stop
**Output**: ActualTime: Actual arrival time of a bus at a stop
Let d = Distance between the bus and the stop using the Harvsine distance equation;
Let min = $\infty$
**while** $GPSTime \leq ScheduledTime + 25\ minutes\ \ OR\ GPSTime \geq ScheduledTime - 25\ minutes$ **do**
  Calculate d;
  **if** $d \leq min$ **then**
    min = d
**if** $d\ is\ within\ 100m\ of\ the\ stop$ **then**
  ActualTime = GPSTime for the bus of d

Figure 3 illustrates how we calculate the difference between the actual and the scheduled arrival times of a bus at a particular bus stop. Let $b_t$ denote the GPS location of the bus, $t$ denote the time when the bus was at location $b_t$, and $S_t$ denote the bus stop location. Since we capture GPS locations for the bus every 20 s, we may encounter a large number of GPS locations around a particular bus stop during the scheduled arrival time for the bus.

To determine whether the bus arrives on time, we use the GPS locations of the bus that are reported within a 50 min window from its scheduled arrival at the bus stop $S_t$ (i.e. 25 min before and 25 min after the scheduled arrival time). Then, we choose the closest bus location to the bus stop within that time window, for example, $b_{t5}$ in Fig. 3. The next step is to check if that GPS location is within the vicinity of the bus stop (i.e., we check if $b_{t5}$ is within 100 m distance from the bus stop). Algorithm 1 summarizes this process.

After estimating the actual arrival time of buses at a particular bus stops, we calculate the difference between the actual and scheduled arrival times. Equation 4 calculates the actual time difference between scheduled arrival time and actual arrival time.

$$Difference_{actual} = ScheduleArrival_{time} - ActualArrival_{time} \quad (4)$$

In a similar way, Equation 5 calculates the difference between scheduled arrival time and predicted arrival time.

$$Difference_{predicted} = ScheduleArrival_{time} - PredictedArrival_{time} \quad (5)$$

If the difference is less than zero the bus arrived late and if the difference is greater than zero the bus arrived early.

After preprocessing the data, we conducted a preliminary analysis on the collected bus arrival data. We found that in more than 37% of the time the buses on these routes were either delayed more than 5 min or arrived early by more than 5 min (see Fig. 4). In some cases the delay was more than 20 min. During the period of our study, the scheduled arrival times did not change, i.e., the schedules did not get updated by TTC. Therefore, we can consider that our models predict the arrival times. However, we used the two formulas in Eqs. (4) and (5) for prediction because we were interested in the delays and early arrivals.

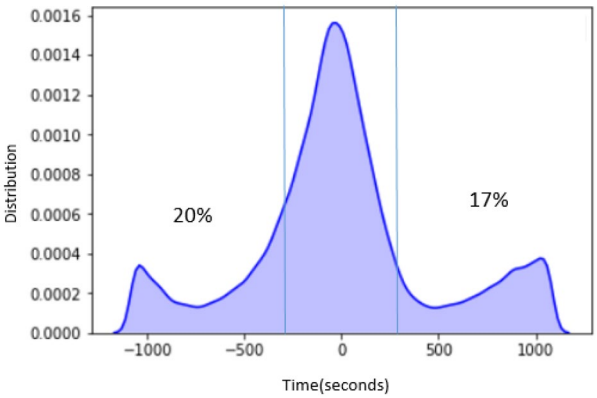


**Fig. 4** Distribution of the difference between actual arrival time and scheduled arrival time, 20% of the buses are delayed more than 5 min and 17% of the buses arrive early more than 5 min

## 4 Machine learning models

This section discusses the machine learning models that we used for predicting the arrival time of buses on selected routes. In particular, we use regression models to estimate the amount of time that a given bus deviates from its schedule. Given historical arrival times at a stop $s$, our models predict the next arrival time at stop $s + 1$.

In our study, we use four baselines to which we compare our model's results: SVR, ANN, ARIMA and Historical Average.

1. *Support Vector Regression (SVR)*: SVR (Drucker et al. 1997) is an extension of the basic support vector machines (SVM) (Boser et al. 1992). In linear regression models, the error rate is minimized, whereas in SVR models, the error is fit within a certain threshold. The model that emerges from SVR is the hyperplane that separates a maximum number of data points.
2. *Artificial Neural Network (ANN)*: (Zhang and Qi 2005): is a network of interconnected neurons, inspired by studies of biological nervous systems (Zhang and Qi 2005; Tan et al. 2005). Neurons are simple information processing units. For time-series analysis, inputs to an ANN model are observations from previous time-steps and the output corresponds to the predicted observation at the next time-step (Zhang and Qi 2005). The information

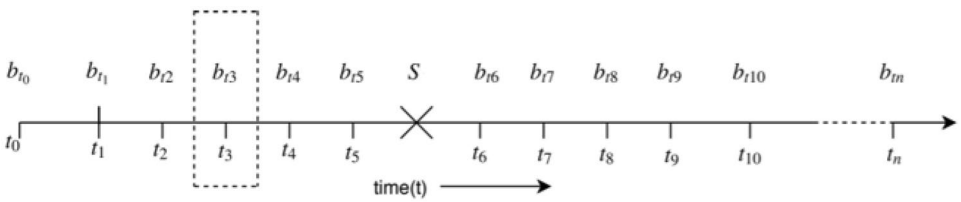


**Fig. 3** Calculating actual arrival of bus at a bus stop

received from the input nodes is processed by hidden layer units along with appropriate activation functions to determine the output.

3. *ARIMA*: ARIMA stands for Autoregressive Integrated Moving Average models. ARIMA is a mature time series prediction model based on statistics. For time series data, ARIMA predicts future values of the data entirely based on the previous data points in the series.
4. *Historical average*: Historical averages are the mean arrival time for bus trips. Historical averages are used as a common reference point to compare the performance of different machine learning models.

### 4.1 Long short-term memory (LSTM) recurrent neural networks

In Fig. 5, we show in a single LSTM cell structure how LSTM recurrent neural network maintains long term dependencies.

The LSTM architecture contains series of connected cells. Each LSTM cell consists of a special unit called a memory block in the recurrent hidden layer. The memory blocks have connections that provides necessary information to maintain temporal state of the network. LSTM cell has three gates: Input gate, Output gate and Forget gate. Input gate control the flow of input information provided to the LSTM cell. Output cell controls the output flow of cell activations into the rest of the network. Unlike conventional RNN, LSTM recurrent neural network has a separate forget gate which makes it more suitable for time-series analysis. The forget gate decides which information is relevant for the prediction task and removes irrelevant information. These gates together provides the overall memory function for LSTM recurrent neural networks.

Following an iterative process, the LSTM model establishes a mapping between an input sequence and the irregularity in arrival time (output) from the training set. Below are the equations for the LSTM neural network:

$$Input\,Gate: i_t = \alpha(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}C_{t-1} + b_i) \quad (6)$$

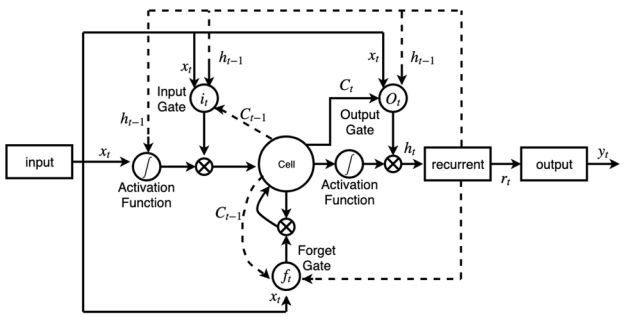


**Fig. 5** LSTM cell structure

$$Forget\,Gate: f_t = \alpha(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}C_{t-1} + b_f) \quad (7)$$

$$Cell\,Input: C_t = f_tC_{t-1} + i_t \tanh(W_{xC}x_t + W_{hC}h_{t-1} + b_C) \quad (8)$$

$$Output\,Gate: o_t = \alpha(W_{xo}x_t + W_{ho}ht - 1 + W_{co}C_t + b_o) \quad (9)$$

$$hidden\,layer\,output: h_t = o_t \tanh(C_t), \quad (10)$$

At time interval $t$, $\alpha$ is the element-wise sigmoid function $\frac{1}{1+\exp(-x)}$ and tanh represents the hyperbolic tangent function $\frac{\exp(x)-\exp(-x)}{\exp(x)+\exp)(-x)}$.

$i_t$, $f_t$ and $o_t$ are the input, forget and output gate states respectively, and $C_t$ is the cell input state.

$x_t$ is input and $b_i$, $b_f$, $b_o$ and $b_C$ are the bias terms.

$W_{xi}$, $W_{hi}$ and $W_{ci}$ are the weight matrices for the input gate. $W_{xf}$, $W_{hf}$ and $W_{cf}$ are weight matrices for forget gate.

$W_{xo}$, $W_{ho}$ and $W_{co}$ are the weight matrices corresponding to output gate. $W_{hi}$, $W_{hf}$, $W_{hc}$, $W_{ho}$ are the weight matrices connecting $h_{t_1}$ to the three gates.

The current cell state $C_t$ is generated by calculating the weighted sum of the previous cell state and the current cell state.

The LSTM Recurrent Neural Network has the ability to remove or add relevant information to the cell state, this is because cell state is adjusted by input gate and forget gate. The forget gate layer removes the irrelevant information from the cell state. It uses $h_{t-1}$ and $x_t$, and outputs a number between 0 and 1 for each input in the sequence in the previous cell state $C_{t-1}$. If the number is zero, no information passes through the gate. If the number is one, all the information passes through the forget gate.

Similarly, Input gate decides what new information will be stored to the cell state. The final output is based on the cell state of LSTM network. As explained above, the current cell state depends on the previous cell state. Therefore, the previous cell state is taken into consideration when updating the weights of the LSTM cell. This is how LSTM cell is able to maintain long term dependencies for predictions. LSTM Recurrent Neural Networks has shown promising results in solving complex machine learning tasks (Sutskever et al. 2014).

### 4.2 Recurrent neural networks for the weather feature

$$InputData = \begin{pmatrix} T^1_{t_0} & W^1_{t_0} & T^1_{t_1} & W^1_{t_1} & W^1_{t_2} \\ & & \vdots & & \\ & & \vdots & & \\ T^n_{t_0} & W^n_{t_0} & T^n_{t_1} & W^n_{t_1} & W^n_{t_2} \end{pmatrix} \quad (11)$$

$$TargetData = \begin{pmatrix} T^1_{t_0} \\ \vdots \\ \vdots \\ T^n_{t_0} \end{pmatrix} \tag{12}$$

Since weather condition has significant impact on the prediction results, we decided to create a Recurrent Neural Network (RNN) model that focuses on the weather feature. The output of this model is combined with the LSTM model discussed in the previous subsection to increase the accuracy of prediction. The RNN model takes as input the arrival times and weather readings at the current stop and at the previous stop to predict the arrival time at the next stop when the weather reading is known.

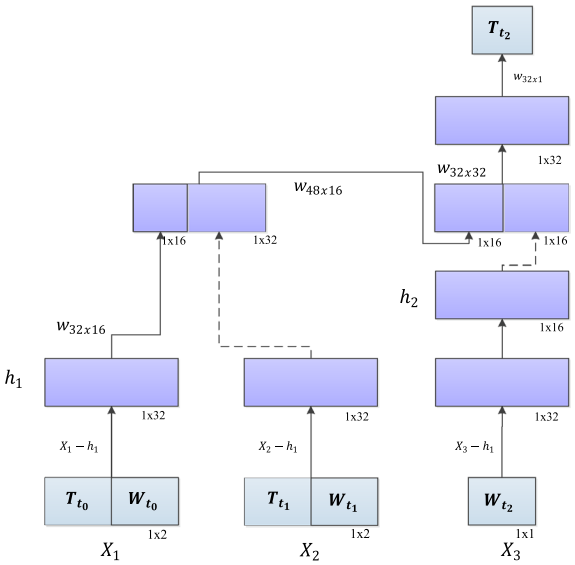


**Fig. 6** RNN architecture

In this model, a window of three is chosen for arrival times $T$ and weathers $W$. Inputs that are shown in Eq. 11, are divided to three categories $X_1$ (previous bus stop), $X_2$ (current bus stop) and $X_3$ (next bus stop). These inputs are illustrated graphically in Fig. 6. $X_1$ is comprised of $T^1_{t_0}$ and $W^1_{t_0}$ for $n$ samples. Similarly, $X_2$ is comprised of $n$ arrival times and weather readings. The third input, $X_3$ only takes weather readings. Equation 12 shows the predicted arrival times for the next bus stop. Figure 6 illustrates the architecture of our RNN model. Two hidden layers, $h_1$ and $h_2$ are set in the diagram with different matrix sizes. The inputs go to $h_1$ with the batch size of 32. After processing in $h_1$, the results will be transferred to $h_2$ in different formats. In $h_2$, the result of processing $X_1$ will be sent to a matrix of 1×16 and will be concatenated with the result of processing $X_2$, which is sent to a matrix size of 1x32. Similarly, the output of processing $X_3$ goes to a matrix of size 1×16 in $h_2$. Then all results of $h_2$ are concatenated together. Finally, the sigmoid function is applied in the last layer which provides the arrival time prediction.

### 4.3 Data preprocessing and normalization

Table 2 summarizes the list of features used in our models. We have flattened the data, i.e., we augment the bus trip travelling southbound with the next trip for the same bus travelling northbound. The first feature (time diff) is

**Table 2** Features used for model building

| Feature name | Description |
|---|---|
| *time diff* | Difference between actual arrival time and scheduled arrival time. This is variable which we are trying to predict |
| *Tag* | Specifies the direction on which bus is heading |
| *Trip.ID* | A unique number given to each trip |
| *Stop sequence* | Assigned sequence numbers starting from 1 to each bus stop in the route |
| *Distance traveled* | Cumulative distance travelled by the bus to reach the bus stop |
| *routeTag* | A unique numeric code to identify a particular route on which the bus is traveling |
| *Stop ID* | A unique numeric code to identify a particular bus stop |
| *Bus ID* | A unique numeric code to identify a particular bus |
| *Service class* | Weekday, Saturday and Sunday |
| *Day of the week* | Numerical number indicating day of the week (1-Sunday, 2-Monday..etc) |
| *Hour* | Numerical number indicating hour of the day |
| *Max temperature* | Maximum temperature in the hour |
| *Min temperature* | Minimum temperature in the hour |
| *Visibility* | Visibility in Km, i.e., how far the driver is able to see |
| *Weather condition* | Weather conditions: rain, snow, fog or haze |

calculated using live GPS locations and the TTC schedules data as discussed in the previous section. The last four features are obtained from the weather data. The rest of the features are obtained from the live bus stop locations data.

Before a machine learning model is trained, all features are converted into a vector representation (e.g., the categorical features). There are two ways to convert a categorical feature into a vector representation; one-hot Encoding and Label Encoding (Tan et al. 2005).

1. *One-Hot Encoding*: Encodes a categorical features as a one-hot numeric vector, i.e., it creates binary column for each category and returns a sparse matrix, where only the entry at the row representing the category is assigned a 1, with the remaining entries assigned 0, creating a sparse vector.
2. *Label Encoding*: Transforms categorical features to numerical features by assigning each categorical feature a unique number which can be normalized before using it as an input for machine learning model.

We have two categorical features, tag and weather conditions. Tag was converted using one-hot Encoding because it only has two categories (North and South). Weather conditions was converted using Label Encoding. Other features in our data do not require encoding because they are continuous variables.

After converting all the features into a vector representation, data was normalized using the following equation:

$$z_i = \frac{x_i - min(x)}{max(x) - min(x)} \tag{13}$$

In Eq. 13, $x_i$ is the *ith* observation of a feature and $z_i$ is the *ith* normalized data point.

### 4.4 Model training in LSTM

The input to each LSTM cells is a 3-dimensional (3D) matrix. The following discusses briefly each dimension:

1. *Sample size*: sample size refers to how many rows are given as an input to the model. In this study we used a sample size of 32.
2. *Time Steps*: time step is one point of observation in the sample. The number of steps determines steps ahead in time the model will predict. We used one, two, three, and four time steps in our model.
3. *Features*: The detailed explanation of each feature used is discussed in Table 2. Our model uses the time diff feature as a dependent feature (output of the model), which specifies the difference between scheduled arrival time and actual arrival time of bus from previous time stamp. We use 11 independent features as input to model, Trip.ID, Tag, Stop.sequence, distance travelled, maximum temperature, minimum temperature, visibility, hour, day of week, service class, weather conditions as inputs to the model. A unique property of Neural Networks is that the when the model adjusts the weights, it can reduce the effect of the irrelevant features while training by assigning them low weights. These features can still have a small negative influence on the model which can decrease its overall accuracy. Only features which gave us the highest accuracy were used in the final model. We did an ablation study, by removing one feature at a time and calculating the error rate of the model. From Table 2, we found that Stop ID and Bus ID to be insignificant to our model. Therefore, we excluded them from our model. Other features showed significant impact on the accuracy of the model.

In our LSTM model architecture, we use 12 input neurons, this represents the number of features (11 independent features and 1 dependent feature) in our dataset used for modeling. The number of neurons used in the output layer is 1 which specifies the difference between predicted arrival and scheduled arrival times (i.e., delay or early arrivals) for a bus at a stop. We tried different variations of LSTM hidden layers and tried different number of LSTM cells within each layer. For the final model selection, we choose 1 hidden layer with 100 LSTM cells with 'ReLU' (Goodfellow et al. 2016) activation function.

When the LSTM model starts training, a sequence of 3D samples (3D tuple) is given to an LSTM layer. The values of a sequence are (32, 1, 12). This means, in one iteration, the model runs 32 samples (batch size), to predict 1 time step ahead, using 12 input features (11 independent input features discussed previously and the previous reading for the dependent feature, i.e. time diff). In the next model iteration each sample will carry the cell state (weights) and a forget gate. Forget gate controls how much from the current cell state is passed to next cell, thus, ensuring that model can learn longer sequences.

When training neural networks, several decisions need to be made regarding the choice of hyperparameters used by the model. We chose the following hyperparameters for our model:

1. *Activation functions*: are non-linear mathematical functions used to combine the output of neurons at one layer for the next layer. They are important for a neural network model as they introduce non-linear properties to the model.

   We experimented with different activation functions, such as, linear, sigmoid and ReLU, for our final model we used ReLU activation function.

**Table 3** Model tuning for LSTM

| Activation | Layers | Cells | Batch size | Rote 28 | | Rote 8 | |
|---|---|---|---|---|---|---|---|
| | | | | RMSE | MAPE | RMSE | MAPE |
| ReLU | 1 | 10 | 32 | 433.15 | 0.2 | 284.77 | 0.44 |
| ReLU | 1 | 50 | 32 | 427.87 | 0.14 | 277.97 | 0.45 |
| **ReLU** | **1** | **100** | **32** | **422.22** | **0.13** | **269.49** | **0.36** |
| Linear | 1 | 50 | 32 | 426.56 | 0.17 | 283.58 | 0.55 |
| Linear | 1 | 100 | 32 | 425.52 | 0.16 | 276.74 | 0.45 |
| ReLU | 1 | 100 | 64 | 426.24 | 0.23 | 275.76 | 0.41 |
| Sigmoid | 1 | 100 | 32 | 433.58 | 0.25 | 279.62 | 0.4 |
| ReLU | 3 | 40,80,40 | 32 | 427.68 | 0.31 | 283.56 | 0.54 |
| ReLU | 3 | 40,80,40 | 64 | 427.77 | 0.28 | 283.75 | 0.54 |
| ReLU | 2 | 40,40 | 64 | 431.50 | 0.2 | 279.32 | 0.5 |

Bold values indicate that the final values used in the experiments

2. *Optimization algorithms*: help to minimize (or maximize) an error function, and they are used to compute the output in such a way that it is computationally less expensive and the model converges to the global minima rather than a local minima. We investigated RMSprop and ADAM optimizers. For the final model we used ADAM optimizer.
3. *Epochs*: specify how many full passes of the data set (epochs) should be used during training. If we use too few epochs, we may underfit the model and do not allow it to learn everything it can from the training data. If we use too many epochs, we may overfit the model, which leads to introducing noise to the model.
4. *Early stopping*: early stopping is a regularization method used to prevent the model from over-fitting. Early stopping is used to remove the need to manually adjust the value of epochs while training a model. When the error rate of the model stops decreasing it automatically stops model from training. Another method for regularization is called dropout (Srivastava et al. 2014), we found that Early Stopping works best for our model.
5. *Batch size*: batch size is the number of samples that will be propagated through the network in one iteration. A batch size can be either less than or equal to the total number of training samples. Advantages of using a small batch size is that it requires less memory for training. A small batch size also reduces the overall training time required by the model, which important when working with large datasets because it is not possible to fit all of the data into memory at once. However, if the batch size is too small, it can lead to less accurate models because we are not providing sufficient number of samples to the model, which leads to less accurate estimate of the output.

Table 3 shows different configurations of LSTM model that we tried in our experiments. For our final model, we used one hidden layer with 100 cells and one dimensional output representing the next arrival time. This configuration provides the best performance for both Route 28 and Route 8.

**Table 4** Comparison of different models for Route 28

| | Historical average | ARIMA | SVR | ANN | LSTM |
|---|---|---|---|---|---|
| MAPE | 0.91 | 0.80 | 0.68 | 0.30 | **0.13** |
| RMSE | 477.87 | 432.69 | 428.79 | 427.33 | **422.2** |

Bold values indicate that the final values used in the experiments

## 5 Results/model performance

To measure the performance of our models, we calculate its Mean Absolute Percentage Error (MAPE) and the Root Mean Square Error (RMSE) on the testing data. All models were trained ten times and the average of MAPE and RMSE error rates were considered as the final value for the models.

The equation for these performance measures are defined as follows:

$$MAPE = \sum_{t=1}^{n} \left| \frac{y_t - x_t}{y_t} \right|$$

$$RMSE = \sqrt{\frac{\sum_{t=1}^{n} (y_t - x_t)^2}{n}}$$

Where $y_t$ is the actual value and $x_t$ is the predicted value. In our case, $y_t$ is the difference of scheduled arrival time and actual arrival time, $x_t$ is the difference between scheduled arrival time and predicted arrival time, and $n$ is the number of samples.

Table 4 shows the MAPE and RMSE values for different models for Route 28. The LSTM model substantially outperformed other models. It shows a 7 fold reduction in MAPE over historical average. A possible reason that LSTM model

**Table 5** Comparison of different models for Route 8

| | Historical average | ARIMA | SVR | ANN | LSTM |
|---|---|---|---|---|---|
| MAPE | 0.92 | 0.84 | 0.76 | 0.49 | **0.36** |
| RMSE | 292.38 | 286.64 | 279.01 | 278.69 | **269.49** |

Bold values indicate that the final values used in the experiments

**Table 6** Comparison of models with and without weather data for Route 28

| | LSTM without weather data | LSTM with weather data |
|---|---|---|
| MAPE | 0.21 | **0.13** |
| RMSE | 427.02 | **422.2** |

Bold values indicate that the final values used in the experiments

**Table 7** Comparison of models with and without weather data for Route 8

| | LSTM without weather data | LSTM with weather data |
|---|---|---|
| MAPE | 0.43 | **0.36** |
| RMSE | 279.11 | **269.49** |

Bold values indicate that the final values used in the experiments

performs better than other models is because it may account more directly to the long term dependencies between input and output features. LSTM model also was best performing for Route 8 as shown in Table 5.

We observe that the RMSE value for LSTM model is not substantially lower than the baseline models. RMSE is sensitive to large outlying errors which occurred in our data, and performs best when errors follow a normal distribution (Chai and Draxler 2014). Chai and Draxler (2014) suggest to remove the outliers that are larger than other errors by several orders of magnitude. However, we did not need to remove outliers, i.e., extreme irregularities, because MAPE clearly showed LSTM model outperforms other models, and the RMSE value for LSTM model is lower than all other baseline models. In addition, we were interested to see the impact of weather on extreme irregularities. In the next subsection, we investigate the performance of LSTM model with and without the weather data.

## 5.1 Significance of the weather data

We investigated the impact of the weather data on the accuracy of our prediction models. When we ran our models with weather data features (i.e., when we included the following features: maximum temperature, minimum temperature, visibility and weather conditions), we noticed significant improvement in the results (see Table 6 for Route 28 and Table 7 for Route 8).

Figure 7 compares the actual arrival time versus predicted arrival time with and without using weather data for Route 28. The x-axis shows the ordered observations of bus arrivals at stops. As mentioned previously, we augment the bus trip travelling on a direction with the next trip for the same bus travelling the opposite direction. This means the x-axis depicts the arrival of the bus at the first stop, followed by its arrival at the next stop. When the bus arrives at the last stop, it returns back on the same route. The next observation after the last stop would be next arrival of the same bus at the stop before the last stop. The y-axis is time in seconds. It can be observed from the plot that the model created with the weather data has better accuracy than the model that was created without the weather data. In particular, we notice that the model that was created using weather data was able to capture extreme delays and early arrivals better than the model that was created without the weather data. We notice similar trend for Route 8 (see Fig. 8).

Furthermore, we compared the results of LSTM models for different portions of the data. We observed that

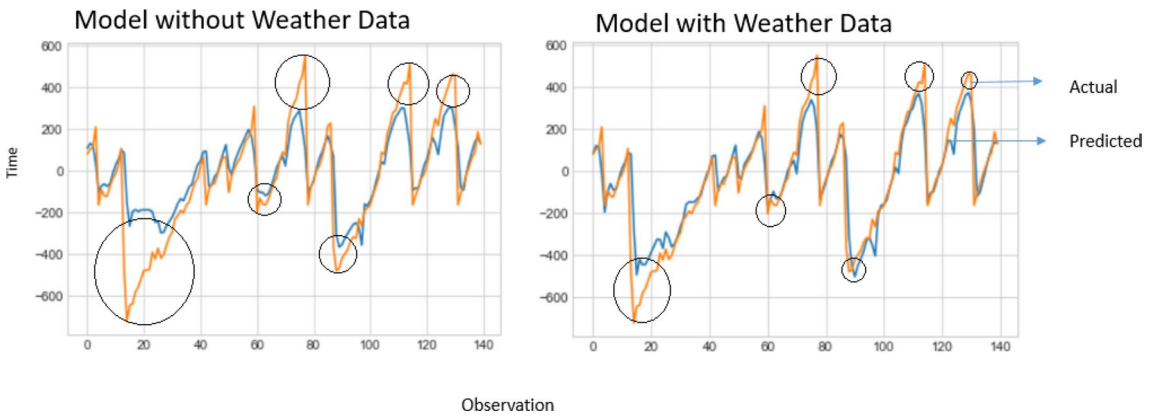


**Fig. 7** Model performance of LSTM Model with and without weather data on Route 28

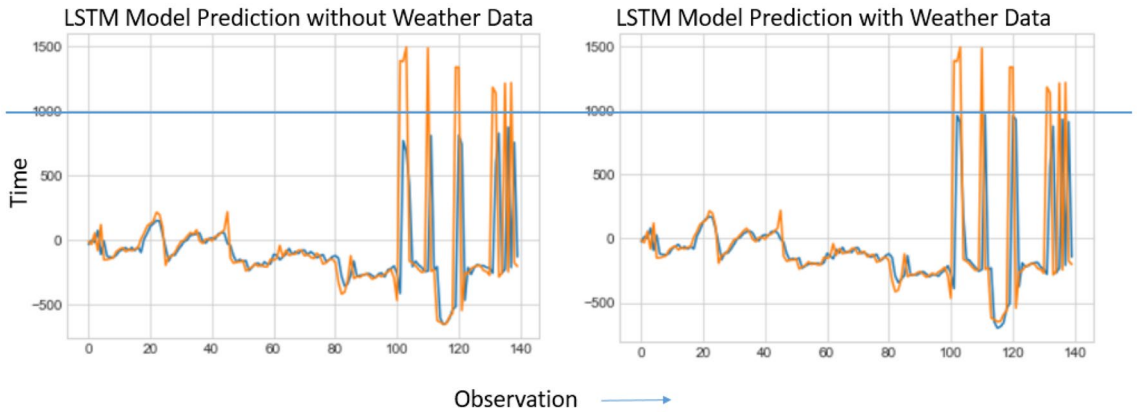


**Fig. 8** Model performance of LSTM Model with and without weather data on Route 8

**Table 8** Difference in RMSE and MAPE of with and without the weather data for Route 28

| % Data | RMSE | | | MAPE | | |
|---|---|---|---|---|---|---|
| | Weather | No Weather | % | Weather | No Weather | % |
| 16% | **10.75** | 44.12 | 310% | **9** | 34.4 | 282% |
| 33% | **20.94** | 42.74 | 104% | **20.3** | 39.5 | 95% |
| 49% | **34.27** | 47.56 | 39% | **34.1** | 50.5 | 48% |
| 66% | **55.63** | 62.12 | 11% | **47** | 63.2 | 34% |
| 82% | **116.13** | 117.72 | 1.3% | **61.58** | 72.97 | 18% |

Bold values indicate that the final values used in the experiments

**Table 9** Difference in RMSE and MAPE of with and without the weather data for Route 8

| % Data | RMSE | | | MAPE | | |
|---|---|---|---|---|---|---|
| | Weather | No Weather | %RMSE | Weather | No Weather | %MAPE |
| 16% | **4.85** | 36.91 | 661% | **4.5** | 34.2 | 660% |
| 33% | **9.25** | 35.86 | 288% | **9.7** | 37 | 281% |
| 49% | **14.32** | 36.66 | 156% | **16.36** | 41.5 | 154% |
| 66% | **22.98** | 37.49 | 63% | **21.4** | 48.2 | 100% |
| 82% | **30.41** | 41.04 | 35% | **39.47** | 62.08 | 57% |

Bold values indicate that the final values used in the experiments

for 16% of the data, the model with the weather data has much higher prediction accuracy when compared to the model created without the weather data (see Table 8). The model accuracy improves with weather by 310% when we compare RMSE and by 282% when we compare MAPE. Table 8 clearly demonstrates that weather plays a significant impact on the prediction accuracy for nearly half of the data (49%). We observed similar results for Route 8, where weather had higher impact (for nearly half of the data, the model accuracy improved by more than 150% as shown in Table 9). The impact of weather decreases as we see more data points because additional factors may also contribute to bus arrival prediction, suggesting that weather has complex non-linear relationship with bus arrival times. Examples of these factors are traffic conditions, construction zones, emergency vehicles, number of passengers which we are planning to explore in future. However, we will mitigate this issue by modelling weather and arrival times in a separate RNN model as explained by end of this section.

**Table 10** Comparison of models with different features for Route 28

| | Visibility | Weather conditions | Temperature | All weather features |
|---|---|---|---|---|
| MAPE | 0.17 | 0.15 | 0.18 | **0.13** |
| RMSE | 424.76 | 423.65 | 425.08 | **422.2** |

Bold values indicate that the final values used in the experiments

To investigate further how much impact an individual weather feature has on the model, we created three LSTM models by just removing one feature and keep other features.

**Table 11** Comparison of models with different weather features for Route 8

| | Visibility | Weather conditions | Temperature | All weather features |
|---|---|---|---|---|
| MAPE | 0.42 | 0.40 | 0.42 | 0.36 |
| RMSE | 278.30 | 278.43 | 281.51 | 269.49 |

The first model removes visibility, the second model removes weather conditions (rain, snow, haze, fog), and the third model removes temperature. Table 10 shows the comparison of different LSTM models as we remove different features from the model for Route 28. The MAPE value increases from 0.13 to 0.17 when we remove visibility feature from the model. Similarly, when we keep all the other features except the weather conditions the MAPE value increases to 0.15. Removing temperature increases the MAPE value to 0.18. Similar observations were found for Route 8 (see Table 11). These results suggest that all weather features that we use in our models are important to achieve better prediction accuracy.

## 5.2 Multi-stop forecasting models

Apart from comparing different machine learning models, we also compared the accuracy of the LSTM model in predicting irregularities for multiple future stops in a trip (i.e., predicting the delay/early arrivals for the future arrivals of the bus after its immediate next scheduled arrival).

We created 4 different models: $(s+1, s+2, s+3, s+4)$. The first model was discussed througout the paper and predicts one stop ahead in time (i.e., given the historical arrival times and weather data for stop $s$, it predicts the irregularities for the next scheduled bus arrival at the next stop $s+1$). The second model predicts the irregularities for the bus arrival at stop $s+2$. Similarly the third and fourth models predict irregularities for the bus arrival at stop $s+3$ and $s+4$, respectively. Figures 9 and 10 show the comparison between the MAPE% errors when predicting irregularities for multiple stops with and without the weather data.

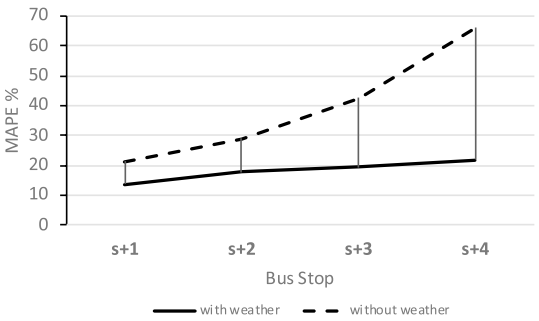


**Fig. 9** Prediction accuracy with and without weather features for multiple stops for Route 28

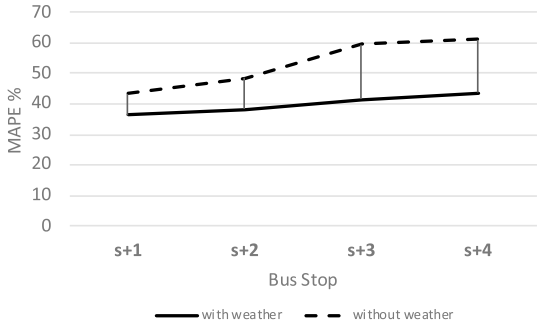


**Fig. 10** Prediction accuracy with and without weather features for multiple stops for Route 8

It is clear from Figs. 9 and 10 that the model performance decreases as we predict for multiple future stops ahead in time. This is similar to the findings by (Duan et al. 2016), (Hua et al. 2018) and (Kormáksson et al. 2014)). However, we found that when weather data was excluded (the dotted lines), the rate of decrease in prediction accuracy increases as we predict for more future stops. This suggests that weather plays a significant role when predicting arrival times or their irregularities for multiple future stops.

## 5.3 Modelling weather feature with RNN model

Since the previous experiments clearly established that weather has a significance influence on the prediction results, we decided to use this feature in a separate RNN model and combine the result with the LSTM model (which also included the weather features as discussed previously). The final prediction is the average of the two models. The architecture of the RNN model was discussed in Sect. 4. Our motivation was to investigate whether we can improve the prediction accuracy if we create a model dedicated to the weather. We tested and trained the RNN model with different hyper parameters and finally we have tuned the hyper parameters as follow:

- learning rate = 0.001
- training epochs = 300
- batch size = 32
- display step = 1

Table 12 compares the performance of this model with the LSTM model for route 28. The RMSE of our new hybrid model showed improvement of 562.38% over the LSTM model for route 28 for 82% of the data. For route 8, the improvement was 873.85% as shown in Table 13. We also noticed that the accuracy does not decrease when we add more data to the model, contrast to the findings in Sect. 5.1. This could be because the RNN model focuses on the weather features, while the LSTM model includes other

**Table 12** Difference in RMSE for the LSTM model and our LSTM+RNN (weather) model for Route 28

| | RMSE | | |
|---|---|---|---|
| % Data | LSTM+RNN | LSTM | % |
| 49% | **12.97** | 34.27 | 264.23% |
| 66% | **18.55** | 55.63 | 299.90% |
| 82% | **20.65** | 116.13 | 562.38% |

Bold values indicate that the final values used in the experiments

**Table 13** Difference in RMSE for the LSTM model and our LSTM+RNN (weather) model for Route 8

| | RMSE | | |
|---|---|---|---|
| % Data | LSTM+RNN | LSTM | % |
| 49% | **7.28** | 14.32 | 196.71% |
| 66% | **5.84** | 22.98 | 393.50% |
| 82% | **3.48** | 30.41 | 873.85% |

Bold values indicate that the final values used in the experiments

features along with the weather. In other words, in small portion of the data, weather condition played a significant role in improving the prediction results in the LSTM model. However, when a separate RNN model is used for weather, its role to improve accuracy included larger segments of the data.

## 6 Conclusion

Nowadays, complex machine learning algorithms can be applied quickly over large datasets, thanks to the advances in the area of big data analytics. This paper investigates different prediction model for irregularities in bus arrival times, using machine learning algorithms. In particular, we built Long Short-Term Memory Recurrent Neural Network models to predict the next arrival time for a bus at a particular stop. Our prediction models use historical bus arrival data, i.e. real time GPS locations for Toronto Transit buses, bus schedules obtained from a Google API, and weather condition data obtained from a weather station in Toronto. Our analysis show that Toronto transit buses experience significant irregularities in arrival times. In nearly 37% of times, transit buses are either delayed or arrive early by more than 5 min, showing great room for improvement. To our knowledge, this is the first work to investigate the impact of weather on bus arrival prediction. We found that weather plays a significant role improving prediction accuracy. Therefore, we built a prediction model that combines two machine learning models: an LSTM model that focuses on a range of input features, e.g. arrival times and hour of the day, and an RNN model which focuses on the weather features. We also investigated prediction accuracy for multiple scheduled arrival of buses ahead in time using weather data. In future, we plan collect more data in order to run our experiments over the entire year. Our current study covers the Winter season and the beginning of the Spring season in Toronto. We plan to extend our study to cover all weather seasons. In addition, we plan to extend our work on bus arrival prediction by using machine learning algorithms with additional datasets, such as passenger count and traffic condition. Furthermore, we plan to use different RNN extensions, such as the Gated Recurrent Unit (GRU) (Cho et al. 2014; Che et al. 2016).

## References


Balasubramanian P, Rao KR (2015) An adaptive long-term bus arrival time prediction model with cyclic variations. J Public Transport 18:1–18. https://doi.org/10.5038/2375-0901.18.1.6

Basyir M, Nasir M, Suryati S, Mellyssa W (2017) Determination of nearest emergency service office using haversine formula based on android platform. EMITTER Int J Eng Technol 5(2):270–278

Boser BE, Guyon IM, Vapnik VN (1992) A training algorithm for optimal margin classifiers. In: Proceedings of the fifth annual workshop on computational learning theory, ACM, New York, NY, USA, COLT '92, pp 144–152. https://doi.org/10.1145/130385.130401

Chai T, Draxler RR (2014) Root mean square error (rmse) or mean absolute error (mae)? arguments against avoiding rmse in the literature. Geosci Model Dev 7(3):1247–1250. https://doi.org/10.5194/gmd-7-1247-2014

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

Che Z, Purushotham S, Cho K, Sontag D, Liu Y (2016) Recurrent neural networks for multivariate time series with missing values. Sci Rep. https://doi.org/10.1038/s41598-018-24271-9

Chen M, Liu X, Xia J, Chien SIJ (2004) A dynamic bus-arrival time prediction model based on apc data. Comput Aided Civ Infrastruct Eng 19:364–376. https://doi.org/10.1111/j.1467-8667.2004.00363.x

Chien SIJ, Ding Y, Wei C (2002) Dynamic bus arrival time prediction with artificial neural networks. J Transport Eng 128(5):429–438. https://doi.org/10.1061/(ASCE)0733-947X(2002)128:5(429)

Cho K, van Merriënboer B, Gülçehre Ç, Bahdanau D, Bougares F, Schwenk H, Bengio Y (2014) Learning phrase representations using rnn encoder–decoder for statistical machine translation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Association for Computational Linguistics, Doha, Qatar, pp 1724–1734. http://www.aclweb.org/anthology/D14-1179

Chopde NR, Nichat MK (2013) Landmark based shortest path detection by using a* and haversine formula. Int J Innov Res Comput Commun Eng 1(2):298–302

Drucker H, Burges CJC, Kaufman L, Smola AJ, Vapnik V (1997) Support vector regression machines. In: Mozer MC, Jordan MI, Petsche T (eds) Advances in neural information processing

systems 9, MIT Press, Cambridge, pp 155–161, http://papers.nips.cc/paper/1238-support-vector-regression-machines.pdf
Duan Y, Lv Y, Wang FY (2016) Travel time prediction with lstm neural network. In: 2016 IEEE 19th international conference on intelligent transportation systems (ITSC), pp 1053–1058
Fu L, Yang X (2002) Design and implementation of bus–holding control strategies with real-time information. Transp Res Rec 1791(1):6–12
Fu R, Zhang Z, Li L (2016) Using lstm and gru neural network methods for traffic flow prediction. pp 324–328. https://doi.org/10.1109/YAC.2016.7804912
Goodfellow I, Bengio Y, Courville A (2016) Deep learning. The MIT Press, Cambridge
Hua X, Wang W, Wang Y, Ren M (2018) Bus arrival time prediction using mixed multi-route arrival time data at previous stop. Transport 33(2):543–554
Ingole P, Nichat MMK (2013) Landmark based shortest path detection by using dijkestra algorithm and haversine formula. Int J Eng Res Appl (IJERA) 3(3):162–165
Jeong R, Rilett R (2004) Bus arrival time prediction using artificial neural network model. In: Proceedings. The 7th international IEEE conference on intelligent transportation systems (IEEE Cat. No.04TH8749), pp 988–993. https://doi.org/10.1109/ITSC.2004.1399041
Kalman RE (1960) A new approach to linear filtering and prediction problems. Trans ASME J Basic Eng 82(Series D):35–45
Ke J, Zheng H, Yang HXC (2017) Short-term forecasting of passenger demand under on-demand ride services: a spatio-temporal deep learning approach. Transport Res Part C Emerg Technol. https://doi.org/10.1016/j.trc.2017.10.016
Kormáksson M, Barbosa L, Vieira MR, Zadrozny B (2014) Bus travel time predictions using additive models. In: 2014 IEEE international conference on data mining, pp 875–880. https://doi.org/10.1109/ICDM.2014.107
Kumar V, Kumar BA, Vanajakshi L, Subramanian SC (2014) Comparison of model based and machine learning approaches for 1 bus arrival time prediction. Transportation Research Board 93rd Annual Meeting. http://docs.trb.org/prp/14-2518.pdf
Liu L, Chen RC, Zhao Q, Zhu S (2019) Applying a multistage of input feature combination to random forest for improving mrt passenger flow prediction. J Ambient Intell Hum Comput 10(11):4515–4532
Liu Z, Yan Y, Qu X, Zhang Y (2013) Bus stop-skipping scheme with random travel time. Transport Res Part C Emerg Technol 35:46–56. https://doi.org/10.1016/j.trc.2013.06.004
Nextbus Nexbus public feed. https://www.nextbus.com/xmlFeedDocs/NextBusXMLFeed.pdf. Accessed 2020
Patnaik J, Chien S, Bladikas A (2004) Estimation of bus arrival times using APC data. J Public Transp 7(1):1
Shalaby A, Farhan A (2004) Prediction model of bus arrival and departure times using avl and apc data. J Public Transport 7(1):41–61. https://doi.org/10.5038/2375-0901.7.1.3
Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R (2014) Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res 15(1):1929–1958. http://dl.acm.org/citation.cfm?id=2627435.2670313
Star TT (2020) Ttc gives notes for affected customers arriving late for work. https://www.thestar.com/news/gta/2017/12/01/late-for-work-the-ttc-can-give-you-a-note-for-that.html. Accessed 2020
Sutskever I, Vinyals O, Le QV (2014) Sequence to sequence learning with neural networks. In: Ghahramani Z, Welling M, Cortes C, Lawrence ND, Weinberger KQ (eds) Advances in neural information processing systems 27, Curran Associates, Inc., pp 3104–3112. http://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.pdf
Tan PN, Steinbach M, Kumar V (2005) Introduction to data mining, 1st edn. Addison-Wesley Longman Publishing Co., Inc., Boston
Veness C (2018) Movable type scripts: calculate distance, bearing and more between latitude/longitude points. URL:https://www.movable-type.co.uk/scripts/latlong.html
Wang B, Huang J, Xu J (2019) Capacity optimization and allocation of an urban rail transit network based on multi-source data. J Ambient Intell Hum Comput 10(1):373–383
Wang J, Chen X, Guo S (2009) Bus travel time prediction model with v-support vector regression. In: 2009 12th International IEEE conference on intelligent transportation systems, pp 1–6
Xu J, Wu Y, Jia L, Qin Y (2020) A reckoning algorithm for the prediction of arriving passengers for subway station networks. J Ambient Intell Hum Comput 11(2):845–864
Yang M, Chen C, Wang L, Yan X, Zhou L (2016) Bus arrival time prediction using support vector machine with genetic algorithm. Neural Netw World 26:205–217. https://doi.org/10.14311/NNW.2016.26.011
Zhang P, Qi M (2005) Neural network forecasting for seasonal and trend time series. Eur J Oper Res 160:501–514. https://doi.org/10.1016/j.ejor.2003.08.037