Improving Urban Traffic Speed Prediction Using Data Source Fusion and Deep Learning

Abstract—Traffic parameter forecasting is critical to effective traffic management but is a challenging task due to the stochasticity of traffic flow characteristics, especially in urban road networks. Traffic networks can be affected by external factors, such as weather, events, accidents, and road construction networks. The impact of these factors can affect traffic flow parameters by influencing travel time, density, and operating speed. Although deep neural networks (DNNs) have recently shown promising signs in traffic prediction using big data, there still exists the issue of maximizing the use of the model capabilities by using big data sources. This paper proposes an improved urban traffic speed prediction approach involving input-level data fusion and deep learning. Motivated by deep learning prediction methods, we propose a Long Short-Term Memory Neural Network (LSTM-NN) for traffic speed prediction that combines traffic and weather datasets on an urban road network in Greater Manchester, United Kingdom. The experimental results substantiate the value of the approach when compared to the use of traffic-only data sources for traffic speed prediction.

Keywords - Long short-term neural networks, traffic data science, deep learning, data-fusion, intelligent transportation systems (ITS)

Introduction

A critical component of the success of Intelligent Transportation Systems (ITS) lies in the quality of traffic information provided to road users. Traffic status information provided to road users, businesses, and government agencies, has the potential to improve traffic by influencing travel decisions for individual travellers, fostering the implementation of better traffic management schemes to control congestion, and improving the overall efficiency of the traffic network. This forms the main intention of traffic prediction. As a result of this, a significant number of mobility services that provide traffic information at real-time have been developed in the past few years, for example, Google Maps, MapTom, Waze, etc., and this number will undoubtedly continue to grow in line with advances in information and communication technologies (ICT). All these emphasize the need for accurate and timely traffic flow prediction.

However, due to the stochastic nature of traffic, making accurate traffic predictions can be very challenging. This can be attributed to the inherent complexity of the traffic domain, which comprises individual actors (drivers or road users) that interact with other entities or variables (vehicles, road, traffic lights), all affected by external factors (weather, accident, events, etc.). Over the years, it has been identified that harsh weather can considerably affect traffic flow parameters by influencing driving behaviour [1], travel demand [2], travel mode, road safety, as well as traffic flow characteristics [3-7].

In terms of traffic operation, it is an acknowledged fact that rainfall reduces traffic capacity and operating speeds, thereby increasing congestion and road network productivity loss [8]. For instance, daily weather conditions like fog, heavy rainfall, and snow can reduce travel demand, cause trip postponement or cancellation, or affect travel mode (changing from a slow mode like walking or cycling to a faster one, such as car or train). For instance, [9] reported that rainfall can increase crash rate and injury rate by 71% and 49% respectively. Similarly, [8] shows that rainfall affected urban traffic flow characteristics by reducing traffic operating speed by 4-9%, while traffic congestion at peak periods showed a significant relationship with temperature intensity.

The majority of traffic guidance methods used in traffic estimation assume clear weather conditions, or completely ignore weather-related data sources, thereby missing out on important information that could provide more accurate traffic prediction [10]. For this reason, traffic analysts and engineers are in search of ways to incorporate non-traffic input in the form of weather data sources into traffic planning and operations, given that this has the potential to improve traffic prediction. The integration of non-traffic and big data input data sources for urban traffic prediction and management enables a clearer understanding of the dynamics of traffic flow modelling, especially in geographical locations having higher likelihoods of severe weather conditions.

This forms the motivation for this paper. We present an approach towards urban traffic speed prediction considering weather information on a deep neural network model. The goal of this paper is to investigate whether deep learning prediction models perform better with the inclusion of non-traffic input data. The objective is to measure the improvement (in terms of prediction accuracy) observed when non-traffic input data (rainfall and temperature) are included in urban traffic speed prediction.

The remainder of this paper is organized as follows. Section II presents an overview of related studies as well as the adopted technical approach towards the prediction problem, while a brief overview of data fusion techniques is presented in Section III. Data description is presented in Section IV. Section V presents the methodology and experimental setup, while Section VI presents results. We conclude the paper and discuss future work in Section VII.
I. TECHNICAL BACKGROUND AND RELATED STUDIES

Here, we distinguish between two broad categories of short-term traffic prediction: (i) parametric and (ii) non-parametric approaches. We also present details citing extant studies about prediction models related to the respective categories in the subsequent sections.

A. Parametric approaches to traffic prediction

According to [11], a parametric model is one that summarizes data with a set of fixed parameters by simplifying the input function to a number of assumptions. Parametric models are sometimes called model-based prediction methods, given that the model structure is predetermined using computed model parameters on empirical data. The most popular parametric prediction model is the autoregressive integrated moving average (ARIMA) model [12, 13] and [14-18].

The ARIMA \((p,d,q)\) model using lag polynomials is mathematically defined by the differencing equation:

\[
\varphi(L)(1 - L)^d y_t = \theta(L) \epsilon_t
\]  

which can be elaborated as,

\[
\left(1 - \sum_{i=1}^{p} \varphi_i L^i\right)(1 - L)^d y_t = \left(1 + \sum_{j=1}^{q} \theta_j L^j\right) \epsilon_t
\]  

where:

- \(\varphi\) represents a model constant that needs to be determined by model fitting
- \(L\) denotes the lag polynomial or backshift operator
- \(y_t\) represents the predicted outcome at time \(t\)
- \(\theta\) is a model constant to be determined by model fitting
- \(p\) number of auto-regression terms
- \(d\) number of difference terms
- \(q\) number of moving-average terms

Here, the variables \(p, d, \) and \(q\) are integers greater than or equal to zero and are used to represent the order of the autoregressive, integration, and moving average components respectively. The main assumption made by this model is that the traffic state is represented by a stationary process, implying the stationarity of the mean, variance, and autocorrelation. This constitutes a major drawback as the models tend to focus on the means, thereby missing extreme values, which are very common with traffic data. Traffic characteristics tend to display peaks especially at rush hour, as well as the swift oscillations with incidents or accidents. For this reason, ARIMA predictors display vulnerabilities when applied to traffic forecasting.

Another parametric model is the Kalman filter. A detailed description of the Kalman filter theory outside the scope of this study, but is presented in [19]. The theory comprises two sets of equations: the process and measurement equations represented by equations (2) and (3) respectively.

\[
x_t = F_{b,b-1} x_{b-1} + w_b, x_b, x_{b-1} \in \mathbb{R}^n
\]

\[
y_t = H_{b} x_b + v_b, y_b \in \mathbb{R}^m
\]

where \(\mathbb{R}^n\) and \(\mathbb{R}^m\) represent \(n\) and \(m\) dimensional real variable domains respectively, \(x_b\) and \(x_{b-1}\) are state vectors at steps \(b\) and \(b-1\) respectively. \(y_b\) is the observed measurement at time step \(b\). The transition matrix is represented as \(F_{b,b-1}\), while vectors \(w_b\) and \(v_b\) represent the process and measurement noises respectively. The Kalman filter then uses a predictor-corrector algorithm in order to estimate the \(x_b\) such that an initial tentative estimate is calculated, which is subsequently refined or filtered using the measurement or actual value \(y_b\). The method has been used within academic studies for the purpose of traffic prediction, such as in [20, 21].

B. Non-parametric approaches to traffic prediction

In this approach, the prediction models are made of algorithms that ‘learn’ from the data, thereby do not make hypotheses about the mapping function. Non-parametric models tend to select the function that best fits the training dataset, meaning they can fit many functions to a particular traffic dataset.

The \(k\)-Nearest Neighbour (\(k\)-NN) is a popular non-parametric regression model that is mostly seen as the simplest machine learning algorithm. The idea of the model is that if \(k\) most similar observations in a feature space are categorized, then the observed sample will likely belong to this category [22]. Then, the searched nearest neighbours are used for prediction. The parameters of the model are (i) state vector, (ii) distance metric, (iii) number of nearest neighbour \(k\), and (iv) prediction algorithm.

The distance metric, which measures the degree of approximation between the sample and test data, is represented by the Euclidean distance, which is calculated using the correlation coefficient weighting method and represented as:

\[
d_i = \sqrt{\sum_j \frac{w_j(V_j - \bar{v}_j)^2}{\sum_j w_j}}
\]

The prediction algorithm describes how the searched nearest neighbour groups are used in the prediction of the state vector in the next time step. This is defined as:

\[
V_m(t + 1) = \sum_{g=1}^{k} \frac{d_i^{-1}}{\sum_{g=1}^{k} d_i^{-1}} V_{g,h}(t + 1)
\]

where \(V_m(t + 1)\) represents the average speed at time \(t + 1\) on link road \(m\), \(V_{g,h}(t + 1)\) represents the searched average speed of nearest neighbor \(g\) at time \(t + 1\), and \(k\) represents the number of nearest neighbours.

Support vector machines (SVM) are one of the most popular machine learning prediction models. This class of models has widely been used for traffic prediction, such as in [23, 24]. The base regression model, which is used for prediction is mathematically defined as:

\[
f(X_t) = \phi(X_t)^T w + c
\]
where $\phi$ represents a user-defined function that maps the traffic speed in the memory window to the features within the higher dimension, and $c$ represents a bias. The objective function for an SVR model at time $t + l$ is defined using the structural risk minimization framework [25] represented as:

$$
\min_{w,c} \sum_{t=1}^{T} L_\theta(f(X_t - y_{t+k}) + \lambda||w||^2)
$$

where $L_\theta(f(X_t - y_{t+k})$ represents the empirical loss, which needs to be minimized via the training dataset and $\lambda||w||^2$ is a control measure used to prevent overfitting. Within the majority of the SVR applicators, the problem is solved using the sequential minimal optimization [26]. This class of non-parametric algorithms has been proven to outperform their regression-based and time series counterparts like ARIMA. For instance, [27] and [28] presented SVM prediction models and compared the results to benchmark models and both studies showed that SVMs outperformed the benchmark models.

Artificial Neural Networks (ANN) are another class of non-parametric models, inspired by the inner workings of the human neurological system, and very popular in academic studies [29]. They are considered as highly accurate prediction models due to their inherent ability to deal with multi-dimensional data, non-linearity, adept learning ability, and generalization [30].

Similarly, multilayer perceptron (MLPs) are a class of ANNs that utilize feed-forward processing [31]. This class of models are mainly used for supervised learning tasks, and are often trained using the backpropagation (BP) algorithm. A simple multi-layer network with one hidden layer is represented by equation (8) below:

$$
y = h(\varphi) = \sigma(\theta_0 + \sum_{j=1}^{N} \varphi_j g(\theta_0 + \sum_{i=1}^{M} \theta_j x_i))
$$

Where $M$ and $N$ represent the number of neurons in the input layer and the hidden layer respectively, and $g$ and $h$ are the transfer functions. $\varphi$ refers to the weight value for the neurons in the input layer, while $\theta$ is the weight or bias for the hidden layer.

However, a demerit of ANN prediction models when applied to time-series analysis is their tendency to neglect the temporal element of time-series data, which led to the development of recurrent neural networks (RNNs) [32-34]. Although RNNs perform better in sequential datasets, traditional RNNs have issues yet to be addressed [34]. Firstly, they are unable to learn from time series having long time lags, which is -- in reality -- very common in traffic datasets. Secondly, the models majorly rely on predetermined time lags prior to learning the temporal sequence, but it is a difficult process to automatically identify the optimal time window size, making them vulnerable in time-series prediction.

A solution to this was the development of Long Short-Term Memory Neural Networks (LSTM-NN), originally proposed by [35] having the objectives of modelling long-term time dependencies of time series data by determining the optimal time lag for the prediction problem. The basic architecture of the LSTM-NN having one memory block is depicted in Fig. 1. As can be seen, the model is built around the memory block (instead of tanh in traditional ANNs).

It can be observed from Fig. 1 that each memory block contains input, output, and forget gates, which respectively can be analogous to write, read, and reset functions on each cell. The multiplicative (dot product) gates allow the model to store information over long periods of time, thereby eliminating the vanishing gradient problem commonly observed in traditional neural network models [36].

Consider a time-series input sequence denoted as $x = x_1 + x_2 + x_3 ... x_t$ and output sequence of $y = y_1 + y_2 + y_3 ... y_t$, where $t$ is the prediction horizon. The LSTM-NN computes the predicted output in the next time step using the historical information supplied, without being told how many backward time steps should be traced. The following set of equations are performed by the model for each prediction task:

$$
i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b)$$

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b)$$

$$c_t = f_tc_{t-1} + i_tg(W_{xc}x_t + W_{hc}h_{t-1} + b)$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b)$$

$$h_t = o_t h(c_t)$$

Where $W$ and $b$ represent the weight matrix and bias vector respectively and $\sigma(.)$ denotes a standard logistic sigmoid function defined as:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

$$g(x) = \frac{4}{1 + e^{-x}} - 2$$

$$h(x) = \frac{2}{1 + e^{-x}} - 1$$

Where $g(.)$ and $h(.)$ are the respective transformations of the sigmoid function above. The variables $i$, $f$, $o$, and $c$ are the input gate, forget gate, output gate, and cell activation vector respectively. This particular characteristic makes LSTMs reliable and accurate models for use in time series analysis and traffic prediction.

Many studies have used LSTM for traffic prediction, for instance [34], where an LSTM-NN model was used for traffic speed prediction. They compared the results to other non-parametric algorithms (SVM, Kalman Filter, and ARIMA), and concluded about the superiority in prediction accuracy of the LSTM model. Similarly, [37] presented an LSTM model
for predicting traffic flow. The comparison of benchmark models like SVM, feed-forward neural networks (FFNN), and stacked auto-encoders (SAE) revealed that the proposed model achieved greater accuracy and generalization.

In terms of the use of data fusion, [38] presented an LSTM and deep belief network (DBN) deep learning model to predict short-term traffic speed using traffic and rainfall data in Beijing, China. The results of the experiment revealed that fusing weather and traffic data sources improved the prediction performance of the models and that the LSTM model outperformed the DBN in capturing time-series characteristics of traffic speed data. Also, [39] investigated the impact of fusing weather data with traffic data for predicting traffic flow. The study incorporated a DBN deep learning model, and the results obtained showed that the combination of data sources yielded superior prediction accuracy.

Although the two related studies above incorporated rainfall data as the only non-traffic input in the training datasets, we argue that the inclusion of temperature in addition to rainfall data sources in traffic prediction can result in improved prediction accuracy. The empirical experiments recorded in this paper aimed to prove the above claim.

II. DATA FUSION

Data fusion is defined as a multifaceted process for handling the automatic detection, association, estimation, and combination of data from several sources [40]. According to [40], five main categories of data fusion techniques exist: (i) Data in-data out (DAI-DAO), (ii) Data in-feature out (DAI-FEO), (iii) Feature in-feature out (FEI-FEO), (iv) Feature in-decision out (FEI-DEO), and (v) Decision in-decision out (DEI-DEO).

For this study, we adopted the DAI-DAO data fusion technique (see Fig. 3). [40] stated in their study that fusing the data at this level can produce more reliable outputs due to the raw input of the respective datasets towards training the prediction model. Also, this technique can avoid errors that can be introduced during data pre-processing and wrangling, such as during the feature extraction or data output process. Therefore, the traffic data obtained from the inductive loop devices are immediately fused with the weather data at the data collection stage, prior to model training or development.

III. DATA DESCRIPTION

The traffic data used in this study was provided by the Transport for Greater Manchester (TfGM). The provided traffic database comprised real-time per-second observations of traffic flow characteristics (i.e. average speed, flow, density), collected using inductive loop sensors. The study area comprised 10 traffic measurement sensors, each of which was 0.3 miles apart on the arterial road. The study period spans from 1 January 2017 to 31 December 2017. The study area is an urban arterial road (Chester Road - A56) in Stretford, Greater Manchester, UK, between coordinate pairs of longitude and latitude between (53.46281, -2.28398) and (53.43822, -2.31394) as depicted by the pinpoint markers on the map in Fig. 2. This represents an ideal characteristic of serving as a conduit from a residential area to the city center. Landmark locations around are the Manchester United Football Stadium – Old Trafford – in addition to other leisure points like shopping malls, offices, restaurants, etc. Although it is an ‘A’ road, implying that it should be a motorway or freeway/highway, the section under consideration has a reduced speed limit of 30mph due to it being a busy segment, having many pedestrian crossings, business places, and stores.

The weather data obtained during the study period comprised hourly observations of temperature (Celsius) and precipitation (measured in millimeters). The rainfall was classified as Light (< 0.5mm), Moderate (0.5mm < r < 4.0mm), and Heavy (r > 4.0mm) as stated in the SYNOP FM-12 weather equipment. The weather data were obtained from the Centre for Atmospheric Studies (CAS), University of Manchester. The weather stations are located within a 3-mile radius of the study area.

For this study, the data collected from the database was aggregated into hourly intervals in order to match the obtained
weather data aggregation level. The total dataset comprised hourly observations of traffic flow characteristics (speed, flow, density) and weather data (rainfall and temperature) for the year 2017, resulting in 8,760 observations of seven (7) variables. The data for the first 6,000 observations (i.e. 250 days from 01-January-2017 to 07-September-2017) were used as the training set, while the remaining observations were used for testing and validation. This accounted for a training-testing ratio of 70:30.

IV. METHODOLOGY AND EXPERIMENTAL SETUP

LSTM-NNs have the ability to self-learn from nonlinear, complex datasets, which makes them sometimes described as ‘black box’ models. Fig. 4 presents a time expanded structure of an LSTM-NN. For this study, we adopt a time-window prediction approach, as presented in the figure. This approach makes use of multi-step lag observations as the input for the training data. For instance, the current time $t$, and the set time window set as $2$ enables the use of the prior time steps $t - 1$ and $t - 2$ as input variables for the input $x_t$. This way, the value of the target variable at the next time horizon $t + 1$ will be $y_t$.

The models are to learn from the provided historic traffic and weather datasets. The first scenario involves the use of a traffic-only dataset to predict hourly traffic speed up to 24 time steps ahead (i.e. 24 hours). Therefore, the first model is trained and tested using a dataset comprising traffic only input data. Secondly, a combination of historic traffic and rainfall datasets are unified and used for the second model. The overall prediction algorithm is presented in Algorithm 1.

In the third and final scenario, we combine historic traffic data with the respective rainfall and temperature observations corresponding to the traffic dataset time intervals and use this unified dataset for training the model.

**Algorithm 1 Speed Prediction using LSTM and Data Fusion**

**Input:** observed sequences $X = x_1, x_2, x_3, ..., x_T$  
**Output:** predicted traffic speed sequence $Y = y_1, y_2, y_3, ..., y_T$  
1. Split $X$ into training and testing data with 70:30 ratio  
2. In training data, select batch size $b$, and at time $t$, generate lookback observations $x_t, x_{t-1}, ..., x_1$ as input, and single-step ahead $x_{t+1}$ as predicted value $y_t$.  
3. Set random initialization for model parameters weight $w$ and bias $c$.  
4. Train model using feedforward greedy layer-wise algorithm using bi-directional processing, and update model parameters  
5. Update model learning by backpropagation algorithm using optimizer (Adam in this case), minimizing loss function $L = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|^p$, where $\hat{y}_i$ is the predicted sequence.  
6. Use testing data for model testing and validation, while another batch of training data is used for another retraining process  
7. Reiterate until training set is exhausted  
8. Return prediction output sequence $Y$.

The proposed model is bi-directional (the hidden layer of the previous layer serving as the visible/input layer of the next layer), in order to accommodate the temporal dependency. The generic architecture of the model used in this study is presented in Fig. 5. For all the inter-connected layers (except the output layer), the activation function utilized is the Rectified Linear Unit (ReLU), which introduces non-linearity to the learning process, defined according to equation (17).

$$g(z) = \max(0, z)$$  

(17)

To have a baseline for comparison, we apply ARIMA to predict traffic speed 24 time steps ahead. The adopted ARIMA $(p, d, q)$ model after optimizing and tuning parameters was ARIMA $(2,0,3)$. Details about the tuning and parameterization of this model are out of the scope of this study and are thus intentionally left out. Therefore, the adopted methodology for this study is as listed in the stages below.

- Stage 1. Gather and clean traffic and weather data and aggregate into 60-min observations, separating into train and test datasets respectively.
- Stage 2. Fuse datasets using DAI-DAO approach.
- Stage 3. Apply traffic-only training dataset to train LSTM-NN model (model ID – LSTM_traffic)  
- Stage 4. Apply fused traffic and weather dataset to train LSTM-NN model (model ID – LSTM_rainfall)  
- Stage 4. Apply fused traffic and weather dataset to train LSTM-NN model (model ID – LSTM_rainfall_temperature)  
- Stage 5. Use test dataset to evaluate prediction performance, comparing results with benchmark (ARIMA(2,0,3) model), LSTM_traffic, and LSTM_rainfall, and LSTM_rainfall_temperature models respectively.

The model architecture of the LSTM-NN is dependent on key parameters, which need to be determined prior to model training. These parameters are hidden layer size, number of units within each hidden layer, number of epochs/iterations for training, batch size, dropout rate, and optimizer (see Table 1).

**TABLE I. SUMMARY OF LSTM MODEL HYPER-PARAMETERS**

<table>
<thead>
<tr>
<th>S/No</th>
<th>Hyper-Parameter</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Hidden Layer Size</td>
<td>1, 2, 3, 4, 5, 6</td>
</tr>
<tr>
<td>2</td>
<td>Hidden Layer Units</td>
<td>128, 256, 512, 1024</td>
</tr>
<tr>
<td>3</td>
<td>Number of Epochs</td>
<td>100, 200, 300, 400, 500, 600, 700</td>
</tr>
<tr>
<td>4</td>
<td>Batch Size</td>
<td>8, 16, 32, 64, 128, 256, 512</td>
</tr>
<tr>
<td>5</td>
<td>Dropout Rate</td>
<td>0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7</td>
</tr>
<tr>
<td>6</td>
<td>Optimizer</td>
<td>Adam, SGD, Adadelta, rmsprop, Nadelta, Nadam</td>
</tr>
</tbody>
</table>

The process of selecting the combination of these parameters is more of an art than a science, which means that the process is performed in many iterations before arriving at an optimum choice. The hidden layer size represents the model depth, while the number of hidden units represents the model width. The batch size represents the optimization applied to the training model, specifying the number of samples to be read at a time during the training instance. The
number of epochs represents the number of times that the model will be retrained, while the dropout rate is used to prevent model over-fitting and improve generalizability. Finally, the optimizer is the function that is used to update or train the model.

In order to identify the optimal set of hyper-parameters to apply, we applied sensitivity analysis to the validation/testing dataset. This would enable us to have a better understanding of the sensitivity of each hyper-parameter. The result of these experiments is presented in Fig. 6. Fig. 6(a) shows the impact of model depth on prediction error (Mean Absolute Error MAE). As can be seen, the best accuracy is obtained with a model depth of 6 layers. Similarly, Fig. 6(b) shows the impact of dropout rate on model prediction error. As can be seen, the dropout rate improves prediction by reducing model overfitting, until an optimum is reached, where the error begins to increase. Fig. 6(c) plots the effect of model width on MAE. For this model, it can be seen that the optimal model width was that having 128 units. Fig. 6(d) and Fig. 6(e) plot batch size and epoch count respectively. As Fig. 6(d) shows,
smaller batch sizes improve model accuracy. Fig. 6(e) shows that the model improves in accuracy at 300 epochs or iterations of model training and retraining. Finally, Fig. 6(f) plotted the sensitivity analysis of optimizer and reveals that the best optimizer was the Adam, while the stochastic gradient descent (SGD) optimizer performed worst. The results from the analysis above enabled the choice of hyper-parameters used for the purpose of this prediction problem.

For model evaluation, we apply three prediction accuracy evaluation metrics – Mean Absolute Error (MAE), Root Mean Square Error (RMSE), and Mean Square Error (MSE), which are all defined by the respective equations below.

\[
MAE = \frac{1}{n} \sum_{i=1}^{n} |e_i|
\]

\[
RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^{n} e_i^2}
\]

\[
MSE(x,y) = \frac{1}{n} \sum_{i=1}^{n} (x_i - y_i)^2
\]

Where \(e_i, i = 1, 2, ... n\) represents \(n\) samples of modal errors, \(x_i\) and \(y_i\) respectively represent the input and output values.

The experiment environment used for this study was a single machine running Windows 10 Operating System with Intel® Core™ i7-6800K CPU @3.40 GHz, 32-GB Memory, and NVIDIA Quadro K420 GPU. The GPU is used for model training due to large computation demand needed for model training in order to accelerate computation and the development environment was performed in R version 3.5.1.

V. RESULTS

The test dataset used for this study spans from 8-September-2017 to 31-December-2017. As stated in the previous section, we benchmarked the predictions from the LSTM models with an ARIMA(2,0,3) model. The observed results for the three models are presented in Table 2. As can be seen from the table, the performance results show that the deep learning models significantly outperform the benchmark model, which is consistent in extant studies.

In terms of the effect of the data fusion, Table 2 shows an improvement in terms of the prediction accuracy when a combined temperature and rainfall dataset is utilized (shaded row). Also noticeable from the table is the improvement in prediction accuracy when rainfall data-sources were integrated. This is consistent with studies that carried out similar research [38, 39], emphasizing the relevance of infusing weather-related datasets in traffic prediction. From the empirical results of this study, it can be gathered that the introduction of weather-related data to the prediction model enables effective learning and produces more accurate modelling of the given dataset, resulting in a significantly more accurate prediction.

Fig. 7 presents a plot of performance comparison of the respective LSTM models. As can be seen from Fig. 7(a), the predictions from the LSTM models are satisfactory, with the most accurate being the model with the combined rainfall and temperature data sources. Similarly, Fig. 7(b) shows the error for the individual predicted values compared to the actual observations. Again, it is clear from the figure that the model with combined data sources out-perform the model that used traffic-only input (green line). The results from Table II and Figure 7 enables us to reach a conclusion that the use of non-traffic input can significantly improve urban traffic speed prediction using LSTM-NN prediction models.

VI. CONCLUSION

Weather is known to affect traffic flow and therefore is an issue for traffic management authorities. We argue that recent studies on traffic prediction focus mainly on traffic data sources alone, disregarding weather-related data sources. In this paper, we investigated the impact of integrating traffic and weather data sources for traffic prediction. To achieve this, we used a 5-layer bi-directional LSTM-NN architecture and showed that data fusion of traffic, rainfall, and temperature datasets can provide more accurate traffic speed prediction in urban traffic. We did this using historic traffic and weather datasets on an urban arterial road (Chester Road – A56) in Greater Manchester, the UK as a case study.

Summarily, the results highlight the benefit of integrating weather-related data sources (i.e. non-traffic input) in traffic parameter prediction. Empirical results showed that the inclusion of temperature data resulted in a marginal increase in prediction accuracy. This can be explained by one or more reasons. Firstly, it could be the case that the increase in temperature actually impacted the traffic status, thereby making it an important variable in the model. Alternatively, it could be the case that the months where higher temperatures were experienced coincided with the summer months in the UK, where most road users embark on vacations, and schools are out of session.

Future work will consider the inclusion of big data sources that may have the potential to impact traffic status such as road construction works, events, social media, and accidents to improve traffic parameter prediction. Traffic analysts can leverage the availability of data in the big data era to get more information about the traffic situation, which would improve traffic congestion management. Advanced Traveller Information Systems (ATIS) like Waze and TomTom already capitalize on crowd-led data to improve their traffic navigation and route guidance system.

<table>
<thead>
<tr>
<th>Model ID</th>
<th>MAE</th>
<th>RMSE</th>
<th>MSE</th>
<th>MAE (mph)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARIMA</td>
<td>2.4640</td>
<td>3.1419</td>
<td>0.9871</td>
<td>-11.1</td>
</tr>
<tr>
<td>LSTM_rainfall_temperature</td>
<td>0.049</td>
<td>0.0892</td>
<td>0.008</td>
<td>-0.2</td>
</tr>
<tr>
<td>LSTM_rainfall</td>
<td>0.071</td>
<td>0.1095</td>
<td>0.012</td>
<td>-0.3</td>
</tr>
<tr>
<td>LSTM_traffic</td>
<td>0.1338</td>
<td>0.3658</td>
<td>0.039</td>
<td>-0.6</td>
</tr>
</tbody>
</table>

Secondly, the results obtained from this study can serve as a call for deeper research into the investigation of the impact of temperature on traffic prediction. For instance, the incorporation of a larger dataset including warmer months of the year, investigating the impact in other geographical locations that have more sunlight or higher temperatures or considering the warmer months in the UK over a long period of time (say 5 or 10 years). This has the potential to present
interesting research on the impact of temperature on traffic prediction.

Finally, the use of more complex deep learning methods for traffic prediction will be investigated, using big data sources. For instance, ensemble deep learning models can be utilized to improve traffic prediction, such as combining image recognition Convolution Neural Network (CNN) with LSTM or RNN for improved traffic prediction performance using fused data sources.

VII. REFERENCES

7. Hogema, J., Effects of rain on daily traffic volume and on driving behaviour. 1996.