Barrlett, Z, Han, L, Nguyen, TT and Johnson, P

A Novel Online Dynamic Temporal Context Neural Network Framework for the Prediction of Road Traffic Flow

http://researchonline.ljmu.ac.uk/id/eprint/11246/

Article

A Novel Online Dynamic Temporal Context Neural Network Framework for the Prediction of Road Traffic Flow

Zoe Bartlett and Liangxiu Han
Department of Computing and Mathematics
Manchester Metropolitan University
M1 5GD, UK
Email: z.bartlett@mmu.ac.uk and l.han@mmu.ac.uk

Trung Thanh Nguyen and Princy Johnson
Faculty of Engineering and Technology
Liverpool John Moores University
L3 5UA, UK
Email: t.t.nguyen@ljmu.ac.uk and p.johnson@ljmu.ac.uk

***Abstract*—Traffic flow exhibits different magnitudes of temporal patterns, such as short-term (daily and weekly) and long-term (monthly and yearly). Existing research into road traffic flow prediction has focused on short-term patterns; little research has been done to determine the effect of different long-term patterns on road traffic flow prediction. Providing more temporal contextual information through the use of different temporal data segments, could improve prediction results.**

In this paper, we have investigated different magnitudes of temporal patterns, such as short-term and long-term, through the use of different temporal data segments to understand how contextual temporal data can improve prediction. Furthermore, to learn temporal patterns dynamically, we have proposed a novel online dynamic temporal context neural network framework. The framework uses different temporal data segments as input features, and during online learning, the updating scheme dynamically determines how useful a temporal data segment (short and long-term temporal patterns) is for prediction, and weights it accordingly for use in the regression model. Therefore, the framework can include short-term and relevant long-term patterns in the regression model leading to improved prediction results.

We have conducted a thorough experimental evaluation with a real dataset containing daily, monthly and yearly data segments. The experiment results show that both short and long-term temporal patterns improved prediction accuracy. In addition, the proposed online dynamical framework improved predication results by 10.8% when compared with a deep gated recurrent model.

***Index Terms*—Artificial Neural Networks (ANN), Deep Learning, Deep Neural Networks (DNN), Recurrent Neural Network (RNN), Gated Recurrent Unit (GRU), Convolutional Neural Networks (CNN), Online Learning, Updating Scheme, Intelligent Transport Systems (ITS), Time-series, Prediction, Regression, Road Traffic Flow.**

I. INTRODUCTION

With rapid urbanisation of cities and towns, traffic congestion has become a critical issue for all metropolitan areas. Due to space being a scarce commodity in most urbanised areas, the only viable solution is better management of existing road infrastructures. Research in the last 20 years has concentrated on building Intelligent Transport System (ITS) through algorithm development, such as based on machine learning approaches [1] with the most recent focus on deep neural networks (DNNs). DNNs are favoured over shallow learners due to their ability to efficiently extract complex latent patterns embedded within the data [2], however, DNNs still present some challenges for time-series prediction.

Existing work into road traffic prediction has focused on using small training datasets, ranging from a few days to a few weeks [3] [4]. However, a prediction model can only be as good as its input data [5]. The temporal magnitude of the training data will determine and restrict what temporal cycles and patterns can be learnt. Despite this weakness, past research have neglected to investigate what temporal patterns are important and should be included within the training dataset. Most assume only short-term patterns, such as hourly and daily, are needed based on no prior investigations [3] [4]. Research by Williams and Hoel [6] has shown that traffic flow in urbanised areas does exhibit weekly patterns linked to the working week, however, other temporal patterns are important. Traffic flow in urbanised areas also exhibits long-term patterns, such as monthly and even yearly temporal patterns. These patterns include, but not limited to, less traffic during summer months and increased traffic in December and January. Therefore, the inclusion of short-term and long-term patterns within the training data could improve prediction results.

Furthermore, DNNs, especially in the traffic flow prediction field, are traditionally statically (not incrementally or online) trained [7] [8] [9] [10]. Therefore, the learning capacity of these models is restricted to patterns and events that occurred during the training dataset, such as recurring traffic congestion. This is impractical for real-life applications; road traffic flow traffic data is complex and stochastic [11] therefore, their prediction models must be able to adapt to previously unseen events, such as non-recurring road traffic congestion or a road traffic incident. One way to overcome this problem is to use online learning. Online learning is a training technique that uses the most recent sequential data point or points to update the model's weights and biases as soon as the data is available; this can improve the prediction accuracy of complex and stochastic sequential data, such as road traffic flow. However, online learning does have its limitations. The main disadvantage of online learning is the eventual loss of the long-term temporal patterns embedded within the training data. By continually updating the DNN's weights and biases based on the most recent data point or points, the model will eventually converge to the short-term temporal patterns, forgetting previously learnt long-term temporal patterns. This is known as *catastrophic forgetting*. Therefore, research into DNN's architectures that can learn and retain short and long-term temporal patterns during online learning need to be investigated further.

The contributions and novelty of this work include:

1) we have investigated different magnitudes of temporal patterns (long-term and short-term), through the use of different temporal data segments to understand how contextual temporal data can improve prediction; and
2) we have developed a novel online dynamic temporal context neural network framework. The framework uses different temporal data segments as input features, and during online learning, the updating scheme is able to dynamically determine how useful different temporal data segments are, and weight them accordingly for use in the regression model. Therefore, the model is able to include relevant long-term temporal patterns in the regression model leading to improved prediction results.

The rest of this paper is organised as follows: Section II presents the State-of-the-art in Deep Neural Networks for Time-Series Prediction with regards to Road Traffic Flow; Section III describes the Methodology used for the experimentation; Section IV details the Experimental Evaluation; and Section V discusses the Conclusion and Future Work.

II. STATE OF THE ART IN DEEP LEARNING TRAFFIC FLOW PREDICTION MODELS

In this section, we will review and assess DNNs architectures for time-series prediction with regards to road traffic flow.

Traditionally, DNNs are used for static tasks such as image classification, however, thanks to algorithm development by Hinton et al. [12] and advances in computing power, they can now be explored further for time-series prediction. DNNs have been proven to provide better prediction results for complex noisy data; their long computational chain of layers are able to extract complex latent patterns embedded within the data [13]. The first publication, to the best of our knowledge, using DNNs for road traffic flow prediction was Lv et al. in 2014 [14]. Lv et al. stated that shallow prediction models learned an inadequate compressed representation of the relationship between the input and the output data. Therefore, a DNN is needed to ascertain the stochastic and complex non-linear properties of road traffic data. Despite this most ANNs designed for road traffic flow prediction are predominately shallow learners with only one hidden layer [14]. Therefore, one area of DNNs which has not yet been fully explored is time-series prediction for road traffic flow. More research into developing deep architectures to improve prediction accuracy for road traffic flow is now possible and needed. Research by Bartlett et al. in 2019 [13] determined that the most suitable deep regression models for road traffic flow prediction were the GRU and LSTM ANNs. Basic RNNs are unable to capture long-term dependencies within the temporal data; their learning capacity is limited to between five and ten-time lags. This severely restricts their temporal context and thus, their prediction accuracy. LSTM and GRU models, however, are able to identify latent patterns over numerous time lags, leading to improved prediction results. Therefore, the long-term temporal patterns embedded within the training data are crucial for road traffic flow prediction [13]. However, these models do have constraints, they are limited by their training data. The magnitude of the training data will determine what temporal patterns can be learnt. Therefore, deep LSTM and GRU ANNs will be the focus of this review, with attention to temporal data size and pre-processing, along with online/incremental learning.

The LSTM model [15] is an adaptation of a basic RNN model. By the addition of an internal memory (known as a *cell*) and a *constant error carousel*, the model is able to preserve the error during training and overcome the *vanishing gradient problem* suffered by basic RNN models. Zhao et al. in 2017 [3] used an LSTM model to predict road traffic flow. The input data, 500 observation points over 19 days with a time-step of five minutes, was preprocessed using an origin-destination cost (ODC) matrix to find the temporal and spatial corrections. This was done to simplify the relationships between the spatial and temporal data points to help the model find a relationship between the input and output data. The ODC matrix was then fed into an LSTM model. The prediction results were compared to five other statistical and machine learning models, including a basic RNN. Zhao et al. determined that the LSTM was the most accurate. Furthermore, preprocessing the input data in an origin-destination matrix did improved prediction accuracy. However, the predictions were not compared to a GRU model and no justification why 19 days of traffic flow data were given. Furthermore, no incremental learning was used. Shi et al. in 2018 [16] used an LSTM model to predict household energy loads. The input data, 48 hours of 929 household's energy loads (divided into pools of ten) with a time-step of 30 minutes, was preprocessed using a *pooling layer*. The pooling layer added nine other neighbouring houses' energy loads as an input feature for the LSTM model. This was done to prevent over-fitting to the training data and to compensate for the small training dataset. The predictions were compared to three other machine learning models and determined that the LSTM was the most accurate. However, a convolutional neural network (CNN) may have been more suitable for pooling neighbouring household loads which was not considered. Furthermore, no justification was given to why a small training dataset was used, nor was any incremental learning implemented.

Therefore, researchers are still using small training datasets with no justification. Small training datasets do not take advantage of the model's ability to link cause and affect over many time lags. This may be due to the big data issue. An

LSTM cell has a complex structure which results in a high computational cost. Therefore, using a large volume of training data with an LSTM model would result in lengthily, perhaps unfeasible, training times. One way to speed up training time would be to use a less computationally heavy model.

Cho et al. in 2014 [17] put forward another adaptation of an RNN to solve the vanishing gradient problem, the GRU neural network. Similar to the LSTM, the GRU can be trained to retain information over many time lags through the use of gates. GRU models are still in their infancy, therefore, there is limited research regarding them, with most papers performing comparative studies. Bartlett et al. in 2019 [13] compared different DNNs for the prediction of road traffic flow, including a deep LSTM and deep GRU model, and determined that the deep GRU model was most successful in terms of accuracy and computational speed. However, state-of-the-art research in other prediction domains, such as text and speech prediction, are using hybrid GRU models to preprocess the data before using a regression layer, such as a GRU, for prediction. The use of a preprocessing layer may improve prediction accuracy. Therefore, hybrid models which include other ANN structures, such as CNNs, should be explored further.

A CNN [18] is a feed-forward neural network that uses the geographical proximity of its input data points to add a geospatial dimension to the prediction function being learnt. Consequently, CNNs are traditionally used when the input data can be expressed in terms of a map, such as image analysis. Nevertheless, many other data sources possess similar characteristics. CNNs combined with RNNs have been used in image/text analysis experiments such as Peris et al. in 2016 [19], Lopez-Martin et al. in 2017 [20], and Wang et al. in 2016 [21]. This research has paved the way for CNNs to be used for road traffic flow prediction models. Road traffic flow data not only exhibits temporal patterns but also has strong spatial dependencies; it can also be influenced by the number of vehicles up and downstream from the point of prediction. Therefore, CNNs can be explored further for road traffic flow prediction. Wu et al. in 2018 [8] built upon the research by Wang et al. and developed a hybrid model to predict road traffic flow. Two GRU layers were used to detect temporal features while three CNN layers were used to detect spatial features. Their outputs were combined into a single regression layer to make a prediction. Additionally, in order to detect patterns across different time lags, three different segments of historical input data (all 105 minutes in length) were used. The segments were from: 1) immediately preceding the prediction, 2) exactly one day before the prediction, and 3) exactly one week before the prediction. The input segments were also preprocessed in an attention model before entering the RNN or CNN layers. Three months of data from 33 sensors were used to train and test the model to predict multiple time horizons of five minutes. Its results were compared to five state-of-the-art time-series prediction models, and Wu et al. determined that the GRU and CNN hybrid model was the most accurate. However, assumptions are made over the temporal segments. It has been assumed that only the daily and weekly temporal patterns are significant; no consideration was given to monthly or yearly patterns. Furthermore, the model was only trained statically, it has assumed that the relationship between the temporal data segments is constant. Once the model has learnt the temporal and spatial relationships contained within the training data it has no opportunity to update these relationship based on the current data. Therefore, it does not lend itself to real-life applications such as road traffic incidents. A model which includes online learning would be more appropriate.

In conclusion, CNNs are still in their infancy in terms of application. Many papers exploring architecture hybrids within image analysis and text/speech analysis have started to cross over into time-series prediction, however, one major hurdle that needs to be overcome for CNNs to make a significant impact on time-series prediction is its ability to detect short and long-term patterns embedded within the data. Furthermore, another issue highlighted by the literature review is the lack of consensus over what magnitude of temporal data that should be used, or, if providing historical temporal data from distant time lags ago can provide context and improve prediction accuracy. Most research fails to address the temporal element of input data. The limited research that does address the temporal element does not compare their model with and without the addition of the temporal data to assess its impact on the model's accuracy [16]. Furthermore, the additional temporal data is often chosen through expanding the current temporal dataset [22], which may be irrelevant, or with no justification[8]. Banko and Brill [5] identified that that input data used was the most important element of a successful machine learning model. Therefore, further research into input data for DNNs and temporal data is vital.

III. METHODOLOGY

A. The Proposed Framework

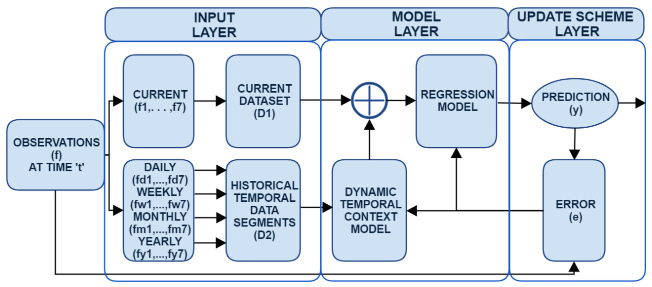

Fig. 1. The proposed framework

We have developed a novel online dynamic temporal context neural network (DTC) framework, as shown in Fig. 1. The framework uses different temporal data segments as input features, and, during online learning, the updating scheme can dynamically determine how useful different temporal data segments are for prediction accuracy. The different temporal data segments are weighted according to their usefulness for the regression model and added the current observations.

Therefore, the framework can include short and relevant long-term temporal patterns in the regression model leading to improved prediction results.

The framework can be divided into three distinct components: 1) an input layer, 2) the model layer, and 3) the update scheme layer, as seen in Fig. 1. Each layer will now be defined in more detail.

1) The Input Data Layer:
Unlike traditional regression neural networks, the proposed framework has two sources of input data. The sources of input data for the input data layer are: 1) the current observations (D_1), and 2) the corresponding different temporal data segments (D_2).

The current observations (D_1) are the traffic flow observed immediately before the prediction point ($t+1$). The current observations dataset is a 7d array, as shown in Equation 1, containing the total traffic flow and its breakdown into six different vehicle classes as shown in Table I. Vehicle classes are used as input features (f) for both the DTC model and regression model based on prior research which demonstrated that vehicle classes can improve prediction results [1].

$$D_1 = \begin{pmatrix} f_{1,t} & f_{2,t} & \cdots & f_{n,t} \\ f_{1,t-1} & f_{2,t-1} & \cdots & f_{n,t-1} \\ f_{1,t-2} & f_{2,t-2} & \cdots & f_{n,t-2} \\ \cdots & \cdots & \cdots & \cdots \\ f_{1,t-n} & f_{2,t-n} & \cdots & f_{n,t-n} \end{pmatrix} \tag{1}$$

TABLE I
AN EXTRACT FROM THE CURRENT (t) TRAFFIC FLOW OBSERVATIONS (D_1)

Total	Total	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6
t	147	12	123	2	9	2	1
t_{-1}	139	10	115	0	10	1	3
t_{-2}	142	9	117	1	9	2	3
t_{-3}	148	8	119	3	11	0	7
...	12	1	8	0	2	1	0
t_{-n}	58	3	51	0	4	0	0

The different temporal data segments (D_2) are the corresponding data that is one day, one week, one month, and one year before the prediction point ($t+1$). Each temporal data segment is a 7d array containing seven different features ($f_i \Rightarrow i \in Z : 1 \leq i \geq 7$), the total traffic flow and its breakdown into six different vehicle classes matching the current observations' shape and structure, as shown in Equation 1. In total, the different temporal data segments dataset is a 28d array, as shown in Equation 2, where d denotes daily, w denotes weekly, m denotes monthly, and y denotes yearly data segment.

$$D_2 = \begin{pmatrix} f_{[d_1,d_n],t} & f_{[w_1,w_n],t} & f_{[m_1,m_n],t} & f_{[y_1,y_n],t} \\ f_{[d_1,d_n],t-1} & f_{[w_1,w_n],t-1} & f_{[m_1,m_n],t-1} & f_{[y_1,y_n],t-1} \\ f_{[d_1,d_n],t-2} & f_{[w_1,w_n],t-2} & f_{[m_1,m_n],t-2} & f_{[y_1,y_n],t-2} \\ \cdots & \cdots & \cdots & \cdots \\ f_{[d_1,d_n],t-n} & f_{[w_1,w_n],t-n} & f_{[m_1,m_n],t-n} & f_{[y_1,y_n],t-n} \end{pmatrix} \tag{2}$$

Both sources of input data, current observations and different data segments (D_1 and D_2), are passed to the model layer for processing.

2) The Model Layer:
The model layer contains two models with different architectures: 1) the DTC model architecture, and 2) the regression (GRU) model architecture.

The proposed DTC model has a CNN structure. Traditionally, CNN structures are used for static tasks where input data can be expressed in terms of a map, such as image analysis or classification. In addition, cutting edge research into time-series prediction has used CNN to find geospatial relationships between different geographical locations to help improve prediction accuracy. Our proposed model is different from previous time-series prediction models using CNNs as we seek to find relationships between different magnitudes of temporal data segments. The model uses the different temporal data segments (D_2) to dynamically determine how useful it is for the regression model (GRU) to produce an accurate prediction. It does this by weighting the input segments. What differentiates the proposed model from traditional CNN architectures is; 1) we have used temporal data as an input features (f_i), therefore, in the proposed model the kernel scrolls 'across' the temporal data segments (D_2) and not down the temporal data like traditional arrangements of CNNs, 2) the kernel (k) used to detect temporal patterns is rectangular and not square as traditionally used in CNNs, so the kernel (k) only convolves across one line of input data at once, 3) the model uses downsampling to obtain the most relevant temporal data, therefore, no padding function is used unlike in traditional CNN structures to maintain the dimensions of the input data, and 4) the stride (s) used for the kernel (k) is equal to width of the kernel ($k = s$) to ensure that each data point is only convolved over once by the kernel (k) per layer (ℓ). This enables the DTC to reduce the dimensionality of the input data while ensuring no replications are passed on to the regression model. The DTC model will now be defined in more detail. The proposed DTC model's input is the 28d array of different temporal data segments (D_2); its structure is a CNN, as shown in Equation 2. In the convolutional layer a convolution kernel (k), also known as a filter or feature detector, convolves (slides) over the different temporal data segments (D_2) input features (f_i) until every input feature has been passed over, moving left to right. Therefore, temporal data is used as an input feature in the array columns and rows, contrary to traditional CNN structures. The convolutional operation ($k[x,y]$), where x and y define the current position of the kernel (k) in the dataset D_2, can be defined as

$$kD_2 = k \otimes f_i : f_i \in D_2[x,y] \tag{3}$$

In the proposed model the magnitude of the movement made to the right is known as a *stride* (s) and is defined the same length as the convolutional kernel (k), therefore, $s = k$, and is a rectangle, unlike traditional CNN kernels. This constraint has been set to ensure each feature (f_i) is passed over only once in each layer (ℓ) per kernel (k) to ensure that the output contains no duplication. At each stride (s) the weights (w_i) in the kernel (k) are multiplied by the corresponding indices ($d \in D_2$) position (x and y) underneath in the temporal segments data

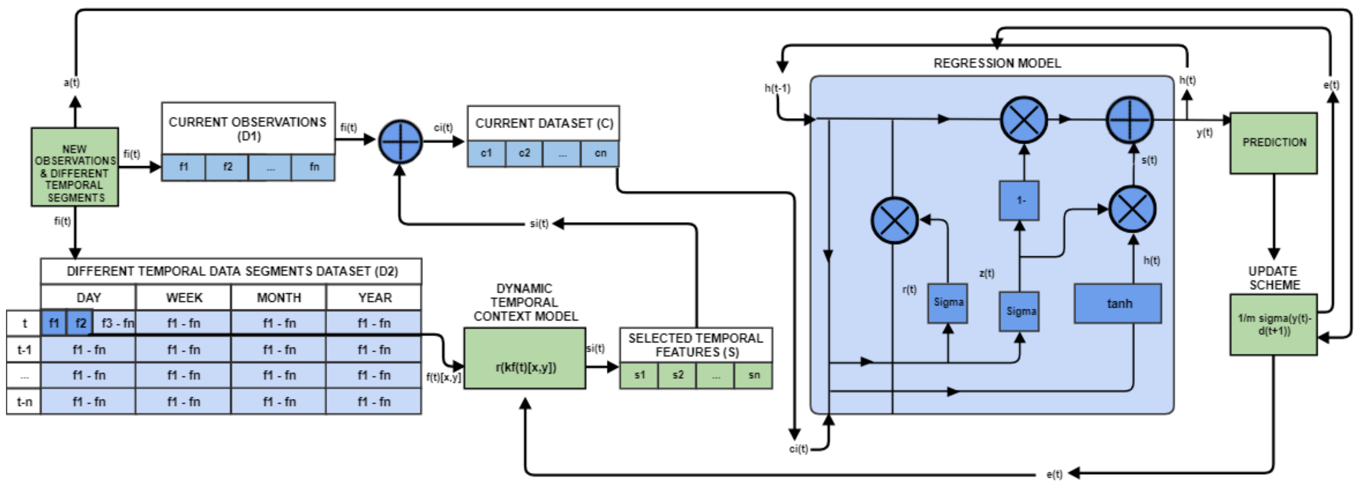

Fig. 2. The proposed Dynamic Temporal Context Framework

($k \otimes d$) to create the *convolution*. The calculated values are used to create one output value, as shown in Equation 3, and used to construct the *feature map* (M), as shown in Fig. 2. What is considered an important temporal pattern by the proposed model is learned during the training process. Multiple kernels (k) can be used to detect multiple important temporal patterns in the temporal data segments. Every hidden layer (ℓ_h) has at least one kernel (k), and the depth of the feature map (M) is determined by the number of kernels in the hidden layer (ℓ_h). The number of kernels (k) and hidden layers (ℓ_h) the DTC contained was optimised through grid search.

It should be noted that although the literature refers to the above process as a convolution, technically the implementation in the proposed model, and most other implementations of CNNs, used a *correlation operation*. Both operations are closely related, with both being a neighbourhood operation. The only significant difference between the two operations is during the calculation of a convolution the kernel (k) is rotated 180°; the kernel (k) does not rotate during the correlation calculation. Therefore, for clarification, in the paper when referring to the convolution operation of our proposed model, we are referring to a correlation operation.

The convolutional operation is linear, therefore, an *activation layer* (ℓ_a) follows the convolutional layer to account for the non-linear relationship between the data points. In the proposed model a rectified linear unit (ReLU), as seen in Equation 4, activation function was used.

$$r(m) = MAX(0, m) : m \in M \tag{4}$$

An ReLU was used to normalise the output of the DTC between the range of $0-x$, to ensure the none of the temporal data segments would be negatively weighted. The feature map (M) is then fed the activation layer (ℓ_a); a ReLU function (r) was applied to each data point (m) in the feature map (M) matrix to transform the data into the set range. The output of the activation layer (ℓ_a), the *activation map* (A), contains the same dimensions as its input, the feature map (M). The activation map (A) is then fed into the pooling layer (ℓ_p). The pooling layer (ℓ_p) is used to condense the temporal data segments while preserving the important temporal patterns (features (f). A sliding window is used to move across the activation map (A), and one value is chosen per stride (s), as shown in Fig. 2. Again, the stride is equal to the size of the window ($s = k$) to ensure no duplication in the output. Therefore, the activation map (A) is downsampled and reduced in width, to a width of q_p, as shown in Equation 5, where q_a is the width of the activation layers (ℓ_a) input. The value chosen in the sliding window is the largest value (*max pooling*).

$$q_p = \frac{q_a - k}{s} + 1 \tag{5}$$

Traditionally, the output of the pooling layer (ℓ_p) is calculated as

$$o = \frac{q - K + 2P}{s} + 1 \tag{6}$$

where p represents a padding function added to increase the dimensions of the output data back to its original magnitude. However, as downsampling was the aim of the proposed model, no padding function (p) was used in the proposed model.

Different from the existing time-series models using CNN where the prediction models are based on static data, our proposed DTC model is dynamic and seeks to find a relationship between different magnitudes of temporal data segments promptly. In the proposed DTC model, the output is the most relevant temporal features (S) for prediction. The selected temporal features (S) are then added to the current observations (D_1) to create the current dataset (C) and passed through to the regression model (GRU), as shown in Fig. 2. Based on previous research [13] the regression layer used was a deep a GRU model. A GRU model works through the use of gates; each gate is a neural network. The gates included in a standard GRU cell are an *update gate* and a *forget gate*, as shown in Fig. 2 in the regression model cell. The current input ($c_t \in C$) and the previous hidden state (h_{t-1}) is added together and passes through the update gate, as shown in Equation 7. The update gate decides what data should be forgotten and what should be added. A Sigmoid activation function is used to squash the values of the input between zero and one, where b is the bias.

$$u = \sigma(w_{cu}c_t + w_{hu}h_{t-1} + b_u) \tag{7}$$

Next, the same input (c_t and h_{t-1}) is passed through the reset gate with a Sigmoid activation function (as shown in Equation 8). The reset gate is used to decides how much of the past information should be forgotten, as shown in Fig. 2.

$$r = \sigma(w_{cr}c_t + w_{hr}h_{t-1} + b_r) \tag{8}$$

The hidden state (h) is then updated using the reset gate and the current input (c_t) (as shown in Equation 9), where the product of the reset gate (r_t) and the weighted previous hidden state ($w_{hh}h_{t-1}$) is the *Hadamard* product.

$$h_t = tanh(w_{ch}c_t + (1 - r_t) \circ w_{hh}h_{t-1} + b_h) \tag{9}$$

Finally, the hidden state is updated using the update gate to determine what information from the current memory should be stored, as shown in Equation 10.

$$h_t = z_t \circ h_{t-1} + (1 - z_t) \circ h_t \tag{10}$$

The output then predicts the number of vehicle (y_t) at the next time point ($t + 1$), as shown in Fig. 2. Once the regression model, GRU, has made its first prediction (y_t) using the test data, the prediction (y_t) and the actual value (a_{t+1}) are then passed to the Update Scheme layer, as shown in Fig. 1.

3) The Update Scheme Layer:
The primary objectives of the Update Scheme layer are: 1) to update the weights and biases in the DTC model to dynamically and timely adjust the most relevant temporal features from the temporal data segments dataset (D_2) for use in the regression model, and 2) to update the weights and biases in the GRU model to allow the model to adjust and adapt to changing temporal trends within the time-series data. This was done through online learning. Once a prediction (y_t) has been made, the actual value (a_t) is added as a new line of observations to the current observations dataset (D_1) and its corresponding temporal data segments are added to D_2, as shown in Fig. 2. The prediction (y_t) and actual observation (a_t) are then compared, and its error, ϵ ($y_t - a_t$), is computed and passed back to the DTC model. This is done to update the model's weight (w_i) and biases (b_i) contained within the kernels (k_i) to allow the model to dynamically adjust the most relevant temporal data segments for regression based on the most recent time-series data. This is achieved through the use of a stochastic gradient descent method [23] and a small window of the most recent data segments in dataset D_2. During backpropagation, using a small window of the most recent data in D_2, the gradient of the error (ϵ) is found with respect to the DTC model's weights (w_i) and biases (b_i) using differentiation, as seen in Equation 11.

$$\frac{\delta\epsilon}{\delta w_i} \quad \text{and} \quad \frac{\delta\epsilon}{\delta b_i} \tag{11}$$

The error's (ϵ) gradient is then backpropagated through the model, from the output layer (ℓ_o) to the input layer (ℓ_i), to find the global minima. In each layer (ℓ) the gradient is scaled by a learning rate (l) as shown in Equation 12.

$$w_{i,t} = w_{i,t-1} - l\frac{\delta\epsilon}{\delta w_i} \quad \text{and} \quad b_{i,t} = b_{i,t-1} - l\frac{\delta\epsilon}{\delta b_i} \tag{12}$$

The weights (w_i) and biases (b_i) in the kernel (k_i) within the DTC model are then updated accordingly to minimise the error (ϵ). Once the DTC model is updated, the new temporal features are selected ($s_{1,t+1} - s_{n,t+1}$) and added to new current observations (D_1) to create an updated current dataset (C), as shown in Fig. 2. A window of the new current dataset (C), is then fed to the regression model (GRU) to update the weights (w_i) and biases (b_i) in the GRU layers. The regression model is updated to improve the prediction accuracy of the overall model by adapting to temporal trends within the time-series data.

The regression model is also updated using stochastic gradient descent method [23]. The current input ($c_{t+1} \in C$) and the previous hidden state (h_t) is added together and passed through the update gate, as shown in Equation 7. The GRU cell processes the input as described in Equation 7 to 10, and the gradient of the error (ϵ) is found with respect to the regression model's weights (w_i) and biases (b_i) using differentiation, as seen in Equation 11. The error's (ϵ) gradient is, again, backpropagated through the regression model, from the output layer (ℓ_o) to the input layer (ℓ_i), to find the global minima. In each layer (ℓ) the gradient is scaled by a learning rate (l) as shown in Equation 12. The weights (w_i) and biases (b_i) within the regression model are then updated accordingly to minimise the error (ϵ). Once the Updating Scheme has updated the regression model, a new prediction is made (y_{t+1}) and the cycle continues.

IV. EXPERIMENTAL EVALUATION

In this section, we have focused on two research questions: 1) how do different temporal data segments affect prediction accuracy? 2) can a dynamic temporal context framework that is able to include both short-term and relevant long-term temporal patterns improve prediction accuracy?

A. Data Description

Both the proposed dynamic temporal context and deep gated recurrent unit model were applied to an existing real-life dataset collected from a typical busy urbanised arterial road between Manchester and Liverpool, UK. The dataset consisted of three months of data collected between 1st January to 31st March 2016, with a time horizon of five minutes (26,195 data point). Historic datasets, referred to as temporal data segments, were added as input features to to to give the data temporal context. The temporal data segments added to the original dataset were the previous day, week, month, and year, as shown in Table II.

All temporal data segments were three months in length, with a time horizon of five minutes, and 26,195 data points, to

TABLE II
TEMPORAL DATASETS

Dataset	Description
1	Current dataset with no temporal data segments
2	Current dataset with previous day temporal data segment
3	Current dataset with previous week temporal data segment
4	Current dataset with previous month temporal data segment
5	Current dataset with previous year temporal data segment
6	Current dataset with all temporal data segments

correspond with the original dataset. The input data had input also had input features of different vehicle classes, as shown in Table III, as different vehicle classes have been shown to improve prediction accuracy [1].

TABLE III
CLASSES OF VEHICLE TYPE

Class No.	Vehicle Type
1	Motorcycles
2	Car or Van
3	Car or Van with Trailer
4	Rigid Goods
5	Articulated HGV
6	Bus or Coach

Therefore, the total dataset contains 26,195 data points 35 different input features. Two months of the dataset was used to train and validate the framework and one month was used for testing. No data points were missing, therefore, no pre-cleaning of the data was necessary.

B. Model Architectures and Hyperparameters

There is currently no standard procedure or analytical calculation to determine the optimal structure or setup for any ANN, therefore, the architecture and hyperparameters of all neural networks used during experimentation were optimised using prior knowledge from the literature review or heuristics through grid search.

The setup of all weights and biases were randomly initialised based on work by Zhao et al. in 2017 [3]. The dropout rates were optimised at 50% fitting based on work by [24] in 2014 [24]. The optimiser used during training and online learning was a stochastic gradient descent method, AdaMax, designed by Kingma and Ba in 2014 [23]; this optimiser was chosen as it is an adaptive gradient method which keeps an exponentially decaying average of the past gradients, therefore, suitable for online learning.

All other hyperparameters and architectural structures, such as the number of layers, nodes, learning rate, update window size, were found using a random grid-search. The grid-search searched through different architectural structures ranging from two to six layers (excluding any input and output layers) with different hyperparameters to find the optimal setup for all models.

C. Performance Metrics

In order to evaluate and compare the accuracy of all the models, a performance metric was used. The Root Mean Squared Error (RMSE), as shown in Equation 13, was used to measure the average deviation between the predicted value and the actual value of the road traffic flow.

$$RMSE = \sqrt{\frac{\sum_{t=1}^{n}(y_t - a_t)^2}{n}} \tag{13}$$

y_t is the predicted value at time t, a is the actual value at time t, and n is the number of time steps predicted.

D. The Evaluation of Different Temporal Data Segments and the Proposed Dynamic Temporal Context Framework

To examine how different temporal data segments affect prediction accuracy, we have applied a deep gated recurrent unit model to six different datasets, as shown in Table II). For each dataset, the model was run multiple times to optimise the parameters and to ensure significance. In total 3,600 models were trained.

TABLE IV
THE PREDICTION ACCURACY OF DIFFERENT TEMPORAL DATASETS USING A DEEP GATED RECURRENT UNIT MODEL FOR ROAD TRAFFIC FLOW

Model	Temporal Dataset	RMSE (%)
Deep Gated Recurrent Unit	**1**	14.644
Deep Gated Recurrent Unit	**2**	13.950
Deep Gated Recurrent Unit	**3**	13.575
Deep Gated Recurrent Unit	**4**	14.010
Deep Gated Recurrent Unit	**5**	14.570
Deep Gated Recurrent Unit	**6**	13.574
The Proposed DTC Framework	**6**	12.244

Table IV shows that the inclusion of the weekly temporal data segment provided the most improvement to the prediction accuracy, with an RMSE of 13.575%, more than the daily temporal data segment, which had an RMSE of 13.95%. This will be due to the weekday and weekend split linked to the working week, which traffic flow in most urbanised areas exhibits.

Interestingly, Table IV also shows that the addition of any temporal data segment, even long-term, improved the prediction accuracy of the model. Therefore, long-term temporal patterns, such as monthly and yearly patterns, embedded within the data, have aided the prediction model. Furthermore, including all temporal data segments improved the prediction accuracy further, with an RMSE of 13.574%. This shows that both short and long-term temporal patterns embedded within the traffic flow data are important for the prediction of road traffic flow and can improve prediction results.

To evaluate the effectiveness of the proposed dynamic temporal context framework, we have used the sixth dataset, as shown in Table II, and compared its prediction results with a deep gated recurrent unit model. The results are shown in Table IV.

The proposed framework was more successful than the deep gated recurrent unit model at predicting road traffic flow using the same existing real input data (dataset six from Table II), with an RMSE of 12.244% and 13.574% respectively. This not only demonstrates the importance of temporal context for accurate road traffic flow prediction but also shows that the temporal context must also be relevant. Using the proposed dynamic temporal context layer has enabled the framework to provide only relevant temporal data segments to the regression model (deep gated recurrent unit model) dynamically in real-time. This had lead to a 10.8% improvement in the prediction accuracy.

V. Conclusion

Accurate prediction of road traffic flow is crucial for intelligent transport system management. Previous research into road traffic flow prediction has focused on short-term patterns, such as hourly, daily, and weekly. Little research has investigated the effect of different long-term patterns, such as monthly and yearly on traffic flow prediction accuracy. In this work, we have investigated different magnitudes of temporal patterns (short and long-term) by using different temporal data segments to assess how contextual temporal data effects prediction accuracy. Also, we have proposed a dynamic temporal contextual framework, which, unlike other prediction models, can dynamically incorporate both short and relevant long-term temporal patterns. This is achieved by using different temporal data segments as input features and, through online learning, the model can dynamically determine which is relevant for regression to provide an accurate prediction in real-time. The different temporal data segments and proposed framework were evaluated using an existing real dataset and compared against a comparable prediction model (a deep gated recurrent unit model). The experimental results show that the inclusion of any short or long-term temporal pattern does improve prediction accuracy. Furthermore, the proposed framework improved prediction accuracy by 10.8% when compared to the deep gated recurrent unit model, with an RMSE of 12.244% and 13.574% respectively.

For future research, the CNN structure of the DTC model should be explored further to provide more contextual information for the regression model to improve prediction accuracy further. In this paper we have restricted the input data to one geographical point, however, it would be interesting to explore on a network level and analysis what, where, and when temporal patterns are more relevant. This analysis could help construct future prediction models and aid in long-term planning of incidents such as roadworks and sporting events.

Acknowledgements

This research is jointly funded by Manchester Metropolitan University and Liverpool John Moores University. We would also like to thank Transport for Greater Manchester for providing the traffic flow data.

References

[1] Z. Bartlett, L. Han, T.T. Nguyen, and P. Johnson. A Machine Learning Based Approach for the Prediction of Road Traffic Flow on Urbanised Arterial Roads. In *16th International Conference on Smart City 2018*, 2019.

[2] Wenhao Huang, Guojie Song, Haikun Hong, and Kunqing Xie. Deep Architecture for Traffic Flow Prediction: Deep Belief Networks With Multitask Learning. *IEEE Transactions on Intelligent Transportation Systems*, 15 (5):2191–2201, 10 2014.

[3] Zheng Zhao, Weihai Chen, Xingming Wu, Peter C. Y. Chen, and Jingmeng Liu. LSTM network: a deep learning approach for short-term traffic forecast. *IET Intelligent Transport Systems*, 11(2):68–75, 3 2017.

[4] Rui Fu, Zuo Zhang, and Li Li. Using LSTM and GRU neural network methods for traffic flow prediction. In *2016 31st Youth Academic Annual Conference of Chinese Association of Automation (YAC)*, pages 324–328. IEEE, 11 2016.

[5] Michele Banko and Eric Brill. Mitigating the paucity-of-data problem. In *Proceedings of the first international conference on Human language technology research - HLT '01*, pages 1–5, Morristown, NJ, USA, 2001. Association for Computational Linguistics.

[6] Billy M. Williams and Lester A. Hoel. Modeling and Forecasting Vehicular Traffic Flow as a Seasonal ARIMA Process: Theoretical Basis and Empirical Results. *Journal of Transportation Engineering*, 129(6):664–672, 11 2003.

[7] Zhiyong Cui, Ruimin Ke, and Yinhai Wang. Deep Bidirectional and Unidirectional LSTM Recurrent Neural Network for Network-wide Traffic Speed Prediction. 1 2018.

[8] Yuankai Wu, Huachun Tan, Lingqiao Qin, Bin Ran, and Zhuxi Jiang. A hybrid deep learning based traffic flow prediction method and its understanding. *Transportation Research Part C: Emerging Technologies*, 90:166–180, 5 2018.

[9] Hao-Fan Yang, Tharam S. Dillon, and Yi-Ping Phoebe Chen. Optimized Structure of the Traffic Flow Forecasting Model With a Deep Learning Approach. *IEEE Transactions on Neural Networks and Learning Systems*, 28(10):2371–2381, 10 2017.

[10] Nicholas G. Polson and Vadim O. Sokolov. Deep learning for short-term traffic flow prediction. *Transportation Research Part C: Emerging Technologies*, 79:1–17, 6 2017.

[11] Arief Koesdwiady, Ridha Soua, and Fakhreddine Karray. Improving Traffic Flow Prediction With Weather Information in Connected Cars: A Deep Learning Approach. *IEEE Transactions on Vehicular Technology*, 65(12): 9508–9517, 12 2016.

[12] Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. A Fast Learning Algorithm for Deep Belief Nets. *Neural Computation*, 18(7):1527–1554, 7 2006.

[13] Zoe Bartlett, Liangxiu Han, Trung Thanh Nguyen, and Princy Johnson. Prediction of Road Traffic Flow Based on Deep Recurrent Neural Networks. In *The 5th IEEE Smart World Congress*, Leicester, UK, 2019. IEEE.

[14] Yisheng Lv, Yanjie Duan, Wenwen Kang, Zhengxi Li, and Fei-Yue Wang. Traffic Flow Prediction With Big Data: A Deep Learning Approach. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–9, 2014.

[15] Sepp Hochreiter and Jrgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 11 1997.

[16] Heng Shi, Minghao Xu, and Ran Li. Deep Learning for Household Load Forecasting A Novel Pooling Deep RNN. *IEEE Transactions on Smart Grid*, 9(5):5271–5280, 9 2018.

[17] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. 6 2014.

[18] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[19] lvaro Peris, Marc Bolaños, Petia Radeva, and Francisco Casacuberta. *Video Description Using Bidirectional Recurrent Neural Networks*. Springer, Cham, 2016.

[20] Manuel Lopez-Martin, Belen Carro, Antonio Sanchez-Esguevillas, and Jaime Lloret. Network Traffic Classifier With Convolutional and Recurrent Neural Networks for Internet of Things. *IEEE Access*, 5:18042–18050, 2017.

[21] Jin Wang, Liang-Chih Yu, K Robert Lai, and Xuejie Zhang. Dimensional Sentiment Analysis Using a Regional CNN-LSTM Model. In *54th Annual Meeting of the Association for Computational Linguistics*, pages 225–230, Berlin, Germany, 2016. Association for Computational Linguistics.

[22] Teng Zhou, Guoqiang Han, Xuemiao Xu, Zhizhe Lin, Chu Han, Yuchang Huang, and Jing Qin. δ-agree AdaBoost stacked autoencoder for short-term traffic flow forecasting. *Neurocomputing*, 247:31–38, 7 2017.

[23] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *3rd International Conference for Learning Representations*, San Diego, 12 2014.

[24] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, and Ruslan Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.