Expert Systems With Applications 202 (2022) 117275

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

A multi-head attention-based transformer model for traffic flow forecasting with a comparative analysis to recurrent neural networks

Selim Reza [a], Marta Campos Ferreira [a], J.J.M. Machado [b], João Manuel R.S. Tavares [b,*]

[a] *Faculdade de Engenharia, Universidade do Porto, Rua Dr. Roberto Frias, s/n, 4200-465 Porto, Portugal*
[b] *Departamento de Engenharia Mecânica, Faculdade de Engenharia, Universidade do Porto, Rua Dr. Roberto Frias, s/n, 4200-465 Porto, Portugal*

A R T I C L E I N F O

Keywords:
Intelligent transportation system
Time-series forecasting
Deep learning
Long short-term memory
Gated recurrent unit
PeMS

A B S T R A C T

Traffic flow forecasting is an essential component of an intelligent transportation system to mitigate congestion. Recurrent neural networks, particularly gated recurrent units and long short-term memory, have been the state-of-the-art traffic flow forecasting models for the last few years. However, a more sophisticated and resilient model is necessary to effectively acquire long-range correlations in the time-series data sequence under analysis. The dominant performance of transformers by overcoming the drawbacks of recurrent neural networks in natural language processing might tackle this need and lead to successful time-series forecasting. This article presents a multi-head attention based transformer model for traffic flow forecasting with a comparative analysis between a gated recurrent unit and a long-short term memory-based model on PeMS dataset in this context. The model uses 5 heads with 5 identical layers of encoder and decoder and relies on Square Subsequent Masking techniques. The results demonstrate the promising performance of the transform-based model in predicting long-term traffic flow patterns effectively after feeding it with substantial amount of data. It also demonstrates its worthiness by increasing the mean squared errors and mean absolute percentage errors by $(1.25 - 47.8)\%$ and $(32.4 - 83.8)\%$, respectively, concerning the current baselines.

1. Introduction

Traffic flow forecasting is a critical component of an intelligent transportation system (ITS) that helps to reduce congestion cost-effectively. Efficient implementation of intelligent transportation systems will increase traffic mobility, help to save lives, and boost economies. It is a time series problem where data from one or more observation sites acquired during previous periods allows the estimation of the flow count at a future time. Statistical methodologies and data-driven approaches are commonly employed to build these models. However, statistical techniques are incapable of dealing with changing traffic circumstances and complicated route layouts (Elhenawy & Rakha, 2017). On the other hand, data-driven strategies help to overcome the limits of statistical methods, i.e., to overcome the non-linearity of traffic with promising outcomes (Yu, Yin, & Zhu, 2017).

A substantial amount of time-series data is required for models to work correctly, and the model's efficiency is primarily determined by how well it can capture the Spatio-temporal aspects of traffic conditions. Furthermore, models are hampered by distorted or missing data, reducing their ability to provide relevant and dependable forecasting results. Due to their capacity to handle vast volumes of data quickly and discover hidden or unknown traffic patterns, deep learning-based approaches, particularly gated recurrent unit (GRU) and long short-term memory (LSTM), have recently solved some of these shortcomings (Polson & Sokolov, 2017). However, to learn long-range dependencies in data sequences, transformers can work much better than LSTM and GRU because of two main reasons: Firstly, in GRU or LSTM, the path lengths grow linearly with the distance between two positions of interest; in contrast, the path lengths are constant regardless of transformers' distance. Secondly, with long sequences and because of memory limitation, parallelisation between training examples is best suited, and transformers allow more parallelisation than GRU or LSTM (Li, Wang, Tu, & Lyu, 2021). These two significant advantages of transformers might also lead to their successful adaptation in time-series forecasting, which had motivated the current study to assess their employment in traffic flow forecasting tasks.

In terms of architecture, a transformer employs an encoder–decoder structure, with each encoder layer's primary goal being to generate information on how different elements of the input are related to one another. The decoder part does the opposite, gets all the encoding

* Corresponding author.
E-mail addresses: up202003355@fe.up.pt (S. Reza), mferreira@fe.up.pt (M.C. Ferreira), jjmm@fe.up.pt (J.J.M. Machado), tavares@fe.up.pt (J.M.R.S. Tavares).

https://doi.org/10.1016/j.eswa.2022.117275
Received 11 February 2022; Received in revised form 16 April 2022; Accepted 17 April 2022
Available online 21 April 2022

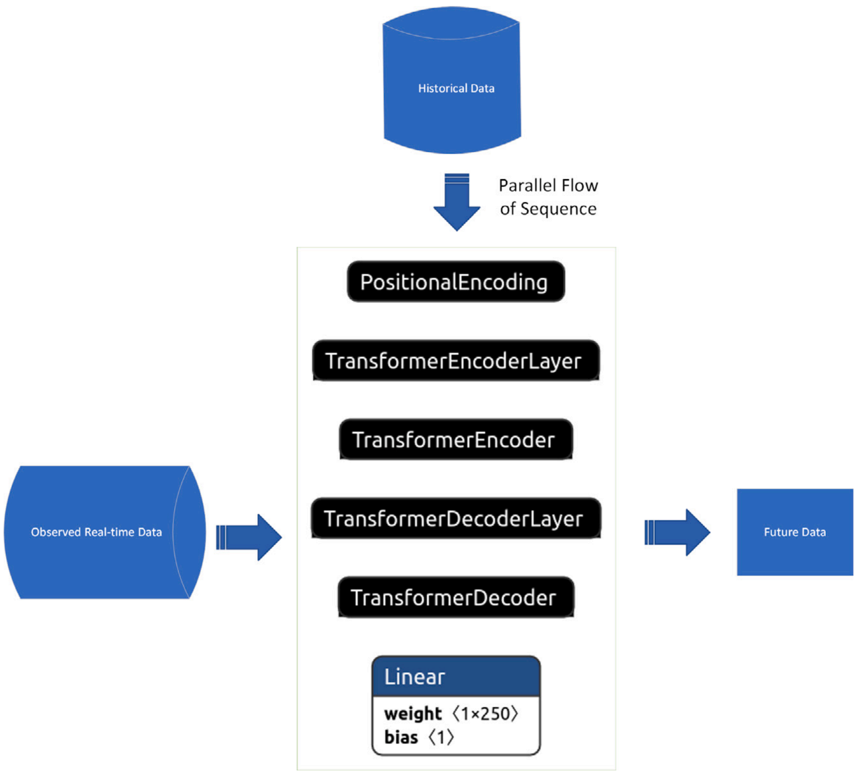


Fig. 1. Illustration of the overall idea of the proposed model.

and uses the built-in contextual information to generate the output sequence (Lim, Arık, Loeff, & Pfister, 2021). The attention mechanism depends on the association between a query and the output of key–value pairs. Instead of performing a single attention function, multi-head attention projects queries, keys and values linearly, multiple times in parallel. After that, there is a point-by-point feed-forward layer with the same parameters at each position. Therefore, it describes the problem as an individual and identical linear transformation of each element from a particular sequence (Cholakov & Kolev, 2021).

Transformer-based models proposed by Jin et al. (2021), Xu et al. (2020) and Yan, Ma, and Pu (2021) helped to solve some traffic flow forecasting problems. Xu et al. (2020) developed a spatial transformer based on the self-attention mechanism in order to capture the spatial correlation and temporal transformer, using multi-head attention concepts to handle multi-time steps temporal dependencies. Yan et al. (2021) used multi-head attention based transformer to enhance the prediction performance by modelling the road network as a complex graph, which demonstrated better latent Spatio-temporal feature identification abilities. Jin et al. (2021) pre-trained bidirectional encoder representations from Transformers (BERT) on a large dataset to predict traffic flow. The proposed model differs from these three algorithms in terms of its architectural simplicity. The first one used stacked spatial and temporal transformers, making it very complex, while the second adapted the concept of a complex graph that requires unique datasets with nodes and edges, and the third utilised pre-trained BERT. The presented model uses a conventional multi-head attention mechanism with 5 heads and stacked 5 identical encoder and decoder layers that show excellent performances in forecasting over a much longer time horizons compared to the pre-trained BERT based model.

Fig. 1 illustrates the overall structure of the proposed model. The model requires historical data to serve the efficient training purpose. Then, it depicts the future traffic flow by feeding it with real-time data from a particular road section. For the conventional models, the input time-series data needs to be serially passed, one after another, in every time step. The input of the previous state is necessary to make any operation on the current state. Such information flow is not well suited for today's Graphical Processing Units designed for parallel computation. However, the proposed model solves this problem due to its ability to pass the input sequences parallelly without the concept of time-step.

The main goal of this study was to assess the applicability of transformers in traffic state forecasting and make comprehensive comparisons with the recurrent neural network (RNN) based approaches: GRU and LSTM. Hence, this article presents a multi-head attention mechanism based transformer for forecasting dynamic traffic flow information. It offers a comparative study between a multi-head attention based transformer and the GRU and LSTM RNN based models to solve traffic flow forecasting problems. The results show a much better performance of the transformer-based model relatively to the two recurrent neural network-based models in capturing long-range dependencies effectively. The main contributions of the current study are as follows:

- It presents a multi-head attention-based transformer model with 5 heads and a stack of 5 similar encoder and decoder layers, including the Square Subsequent Masking approach;
- It verifies that the number of encoder–decoder layers, in opposition to heads, bears significant consequences on the model's performances;
- It also justifies the need of a considerable volume of data samples for a transformer to achieve state-of-the-art performances.

This article is organised according to the following structures: Section 2 presents selected state-of-the-art works along with their limitations; Section 3 describes the multi-head attention mechanism based transformer model; Section 4 includes information about the experimental setups and also the obtained results; Section 5 is devoted to discuss the overall performance and feasibility of the proposed model in traffic flow forecasting, and finally, Section 6 draws the conclusions and future work.

2. Related works

In the literature, RNN based models have dominated traffic state forecasting tasks in recent years (Reza, Oliveira, Machado, & Tavares, 2021). Because of their internal memory and ability to retain their input, RNN based models are powerful and robust, allowing them to anticipate future events. Hence, they are especially good at forecasting future possibilities based on the data's sequential properties. This section summarises state-of-the-art works and their limitations and future potential. Hence, it presents previous important works using GRU, LSTM, and attention-based transformers for traffic state forecasting problems.

2.1. Long short-term memory

Traditional RNNs often suffer from vanishing gradient problems and cannot capture long-range dependencies. The Long Short-Term Memory (LSTM) models have capabilities to tackle the problem (Hochreiter & Schmidhuber, 1997) because of their unique architecture of gating mechanism, which facilitates an LSTM to control its memory state updating tasks. Ma, Tao, Wang, Yu, and Wang (2015) aimed minimising back-propagation error decay problem and applied an LSTM model to predict the future traffic speed that demonstrated relatively high errors and a lack of robustness. The authors' consideration of data's temporal relationships resulted in those shortcomings. Also, missing samples in datasets can cause significant problems in achieving the best results. A robust solution to address those problems is masking and imputation mechanisms that demonstrated a tremendous improvement in performances for traffic state forecasting problems. For example, Khan, Khan, Dey, and Chowdhury (2019a) achieved a mean absolute percentage error (MAPE) of only 2.10% in the traffic volume prediction task following this approach. However, the proposed model demonstrated inferior performance in capturing the seasonality of traffic states.

The models mentioned above suffer from a lack of robustness. The additional inclusion of weather data showed excellent improvements in mitigating the lack of robustness. For instance, Jia, Wu, Ben-Akiva, Seshadri, and Du (2017) fed the model with both speed and rainfall data that achieved a MAPE of 12.90% for forecasting the traffic speed over a 30 min horizon. Considering both the Spatio-temporal dependencies of traffic states also showed considerable robustness enhancement. For example, Zhao, Chen, Wu, Chen, and Liu (2017) proposed a greedy layer-wise unsupervised learning technique to capture both the spatial and temporal correlation in traffic states resulting in effective improvement in robustness by forecasting over a 30 min time horizon with a mean relative error (MRE) of 9.30%. Another solution to improve robustness is to eliminate the non-Gaussian disturbances in traffic states. Lu et al. (2021) introduced cascading Temporal-aware Convolutional Context (TCC) blocks and a Loss-Switch Mechanism (LSM) to implement the approach and achieved more improvements in robustness and accuracy. However, compared to normal RNN, these models are even slower to train. Furthermore, a more robust model is necessary, mainly to capture long-range dependencies of traffic states.

2.2. Gated recurrent unit

Gated Recurrent Unit (GRU), a sister of the LSTM model, was introduced by Chung, Gulcehre, Cho, and Bengio (2015) that needs fewer parameters to train. Fu, Zhang, and Li (2016) was one of the pioneers to apply a GRU on the PeMS (Varaiya, 2007) datasets for traffic state forecasting, achieving similar results, but with a much faster convergence than LSTM. Bartlett, Han, Nguyen, and Johnson (2019) examined the performance of (i) standard RNN, (ii) LSTM, and (iii) GRU based approaches for traffic flow forecasting problems on the same datasets. The results demonstrated faster convergence with a root mean squared error (RMSE) of 9.26% by the GRU, which is better than the others. Modifying the architecture of the GRU by inserting an additional gate composed of an integrated decay mechanism also enhanced the accuracy and robustness (Pu, Cui, Wang, Li, & Wang, 2020). The extra gate helped the GRU handle the missing sample problem of the datasets and, hence, resulted in good performances. Also, it facilitated taking into account the temporal and local aspects of traffic flow and improved the model's transferability and reproducibility.

The literature also presented the concept of attention mechanisms incorporated with GRUs. For example, Khodabandelou, Kheriji, and Selem (2021) applied the self-attention mechanism with a GRU and exhibited an excellent performance gain by achieving a mean absolute error (MAE) of only 1.26 for a 1 h time horizon.

The input data for the above-stated models must be transmitted sequentially, one by one to the model, and past state inputs are required to conduct actions in the present state, which does not make efficient use of computation. However, the proposed architecture can efficiently tackle this problem.

2.3. Attention based transformer

The current literature provides a few references on attention-based transformer models for traffic state prediction or forecasting tasks. Self-attention based transformers, as stated by Li et al. (2019), produced excellent outcomes in forecasting tasks. However, there were three main drawbacks: (i) the self-attention matches queries against keys that are unaffected by the local context, which could lead to anomalies and underlying optimisation difficulties; (ii) the surrounding context determines whether an observed point is an aberration, a change point, or a part of the pattern; and (iii) the similarity of queries and keys is calculated based on their point-wise values, which ignores the local context entirely. A multi-head attention mechanism can solve the problems mentioned above, which is one of the objectives of the current study.

A Temporal Fusion Transformer (TFT) combining high-performance multi-horizon forecasting with interpretable insights into temporal dynamics was proposed by Lim et al. (2021). Although the main goal was to increase model's interpretability, the proposed model showed good performance by capturing the seasonality of the traffic states, but showed a non-appearance of comprehensive comparisons with the GRU or LSTM based models in terms of state-of-the-art metrics. The presented model sought to overcome this limitation based on comprehensive comparisons with the LSTM and GRU based models.

Grigsby, Wang, and Qi (2021) proposed a long-range transformer based model for dynamic Spatio-temporal forecasting, which achieved a mean absolute percentage error of 3.85 on the PeMS dataset. Although the performances were not the best compared to the graph neural network (GNN) based models, the results were encouraging enough to justify their goodness. However, the effectiveness of these models is limited to a particular road; therefore, information from the adjacent roads or weather data is essential to employ them in any region or section. Jin et al. (2021) proposed a pre-trained BERT to solve such drawbacks by achieving a MAPE of 7.72% on PeMS datasets. The model suffered from the over-fitting problem (even more than the original BERT Devlin, Chang, Lee, & Toutanova, 2018) because of the enormous parameter size. The forecasting time horizon was only 60 min, which needs improvement. The mentioned work sought to overcome the drawbacks of this model to show that forecasting over a long time horizon is very much possible with a state-of-the-art performance by training the transformer from scratch. Hence, there are many scopes to improve accuracy and robustness in using transformers for time series forecasting.

3. Methodology

This section presents the multi-head attention-based algorithm formulation, including a description of other baselines based on support vector machine (SVM), LSTM, and GRU based models, for comprehensive comparisons.

3.1. Support vector regression

A SVM looks for a line, or hyperplane in multidimensional space, that divides two or more classes. The new point is then classified based on whether it is on the positive or negative side of the hyperplane, as determined by the classes to be predicted. On the other hand, Support Vector Regression (SVR) applies the same principle as SVM to regression issues. Regression aims to identify a function approximating mapping from an input domain to real numbers based on a training sample. With SVR, the goal is to evaluate just the points within the decision boundary line. The hyperplane with the most significant number of points is the best fit line (Awad & Khanna, 2015). The construction of the SVR model utilises the radial basis function (RBF) kernel. Hence, if i_1 and i_2 are two samples represented as feature vectors in some input space, then the RBF has the following form:

$$K(\mathbf{i_1},\mathbf{i_2}) = \exp\left(-\frac{\|\mathbf{i_1}-\mathbf{i_2}\|^2}{2\sigma^2}\right), \tag{1}$$

where $\|i_1 - i_2\|^2$ represents the Squared Euclidean Distance between the two feature vectors, and γ is a free parameter (Ding, Liu, Yang, & Cao, 2021).

3.2. Long short-term memory and gated recurrent unit

The introduction of long short-term memory and gated recurrent units solved the short-term memory and vanishing gradient problem of RNNs. There are internal gates that can regulate information flow, as shown in Fig. 2. These gates can determine which data should be kept and discarded in a sequence. It can then send essential data down long sequences to generate predictions.

LSTMs work with the help of the cell state and numerous gates. The cell state works as a transportation highway as relative information is transmitted through the sequence series. In principle, the cell state, like a network's memory, can carry relevant information throughout the sequence processing. As an outcome, the short-term memory effects get reduced due to the flow of information from previous time steps to later phases. As the cell state travels, information gets added or withdrawn through the gates. The gates consist of several neural networks that determine which information is to allow passing. During training, the gates learn what information is essential to keep or forget (Sherstinsky, 2020).

On the other hand, GRU is a newer recurrent neural network similar to an LSTM. It can abandon the cell state to use the hidden state to transfer data sequence. Compared to the complex architecture of LSTM, it consists of only two gates: a reset gate and an update gate that generally helps GRU to train faster than LSTM (Dey & Salem, 2017).

Here, it is used a $sigmoid$ function with $sigmoid$ curve characteristics as an activation function within LSTM and GRU models, which for a z variable can be defined as:

$$S(z) = \frac{1}{1+e^{-z}} = \frac{e^z}{e^z+1} = 1 - S(-z). \tag{2}$$

3.3. Transformer

A transformer using an encoder–decoder with stacked self-attention and point-wise, fully connected layers, maps an input sequence of $(s_i1,\ldots,s_in)$ representations to a sequence of continuous $c = (s_c1,\ldots, s_cn)$ representations. Then, for a given c, the decoder generates an output $(s_o1,\ldots,s_on)$ sequence of symbols, one element at a time, as can be seen on Fig. 3. The raw input data is first normalised using a Min–Max Normalisation mechanism and then converted to sequences. Each sequence element is mapped to an integer token and, then, passed through an input embedding layer, which is a simple lookup table that pairs each integer with a continuous learned vector. Once the transformer encoder does not have any embedded recurrent neural network layer, this study uses positional encoding to inject positional information into the embedding. $cosine$ and $sine$ functions created two separate vectors for every odd and even time step. Adding each of them to their corresponding embedding vector would inject the positional information.

For a X input, if D is the dimension of the embedding vector and $p_e(m,2n)$ the elements of the positional encoding vector (P), then, every even time step ($p_e(m,2n)$) is defined as:

$$p_e(m,2n) = \sin(m\exp[-2n\log(1000)/D]). \tag{3}$$

Again, if $p_e(m,2n+1)$ refers to the elements of the positional encoding P vector, then, every odd time step ($p_e(m,2n+1)$) is defined as:

$$p_e(m,2n+1) = \cos(m\exp[-2n\log(1000)/D]). \tag{4}$$

The output of the positional encoding will give another vector with the same dimension by $X+P$.

3.3.1. Encoder

Following multiple trial–error testing, the proposed model provides the best results for a stack of $N = 5$ identical layers to make up the encoder. Each layer constitutes a multi-head self-attention mechanism and a position-wise, fully connected feed-forward network. Each sub-layer used a residual connection, followed by layer normalisation. That is, the output of every sub-layer is:

$$Output = LayerNorm(y + Sublayer(y)), \tag{5}$$

where $Sublayer(y)$ is the function that the sub-layer implements. All sub-layers inside the model, including the embedding layers, produce dimension outputs: $D = d_{model} = inputfeature\_size = 250$, to support these residual connections.

3.3.2. Decoder

Similarly, following several trial–error testing, a stack of $N = 5$ identical layers also makes up the transformer decoder. Generally, the number of layers of the decoder is two times that of the encoder. However, during this study, an identical number of decoder layers provided the best results. The encoder also adds a third sub-layer to each layer to perform multi-head attention over the encoder stack's output and the two sub-layers. Each sub-layers utilises residual connections, similar to the encoder, followed by layer normalisation. The stacked decoder self-attention sub-layers needed modifications to prevent their current positions from mixing with the following positions. Because the masking (Square Subsequent Masking) and the output embeddings get compensated by one place, only the known outputs at positions less than i resulted in the prediction of position i.

3.3.3. Multi-head attention

The training helped to learn the attention mechanism composed of queries (Q), keys (K) and values (V). The idea is that a query vector will be compared to a set of critical vectors to determine how compatible they are. Each key vector comes paired with a value vector, and the greater the compatibility of a given key with the query, the more significant influence the corresponding value will have on the output of the attention mechanism. From the initial embedding layer of the beginning of the encoder, a linear transformation of the embeddings extracted key and value vectors. Each element of the sequence also provided the query vector. Computing a dot product between the query and each key, including the key from the same element, called dot product attention, has been calculated, resulting in a set of unnormalised weights (alpha vectors): $(\alpha_0,\alpha_1,\ldots.,\alpha_n)$. To normalise them, they are divided by their $\sqrt{d_k}$ dimensionality and get passed to a softmax layer:

$$(\alpha_0,\alpha_1,\ldots.,\alpha_n) = \left(\frac{softmax(\alpha_0,\alpha_1,\ldots.,\alpha_n)}{\sqrt{(d_k)}}\right). \tag{6}$$

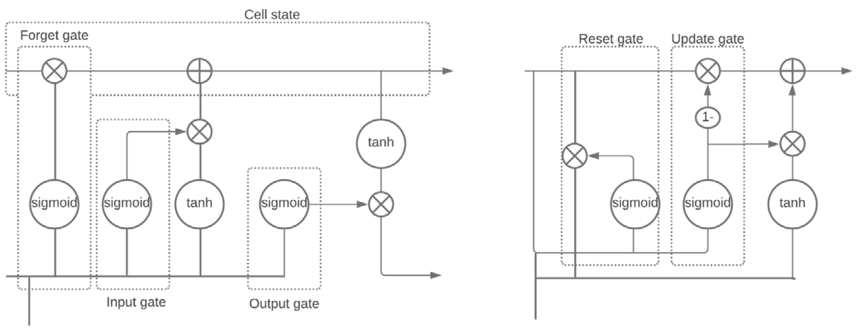


Fig. 2. Architectures of LSTM (on the left) and GRU (on the right) models.
Source: Adapted from Khan, Khan, Dey, and Chowdhury (2019b).

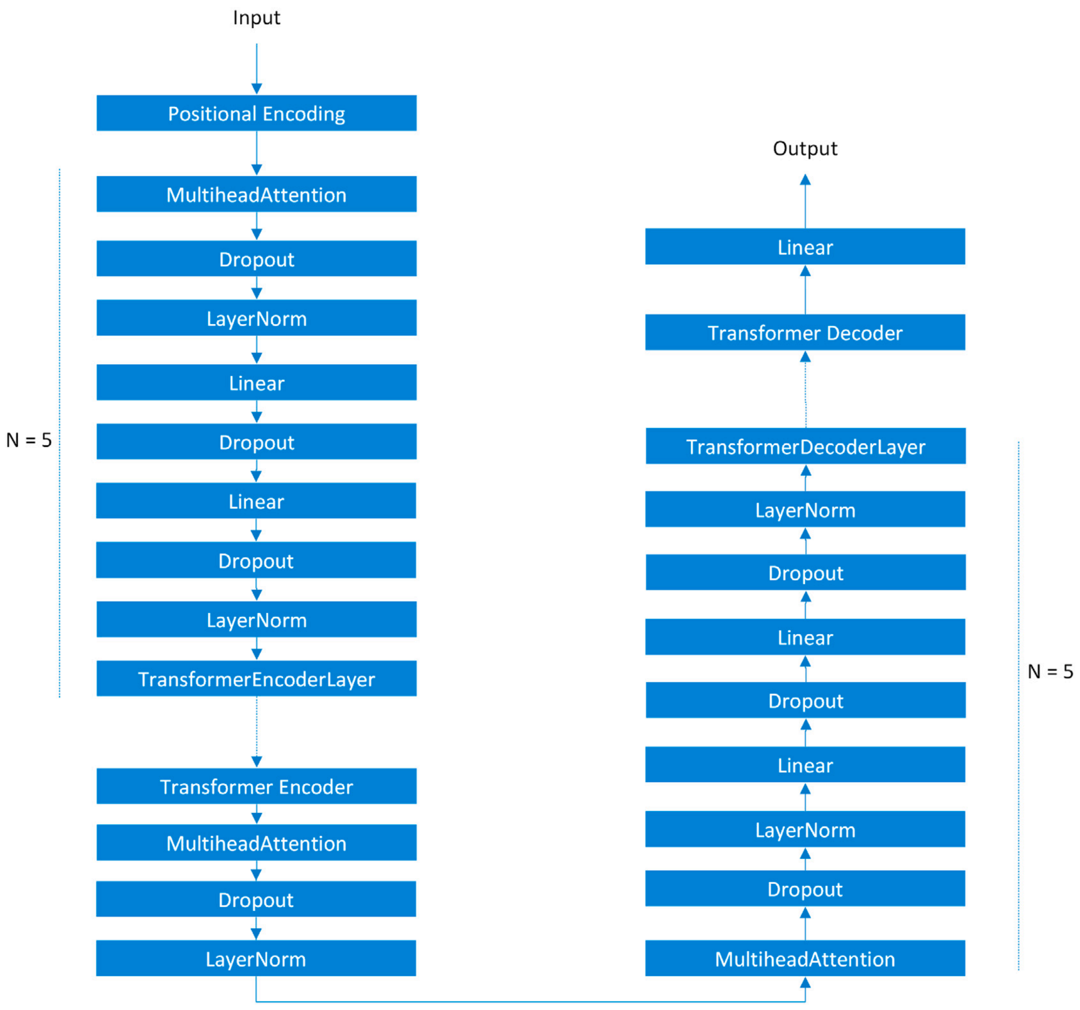


Fig. 3. Proposed multi-head attention based transformer model architecture.

Once the weights are normalised, they can be used to take a linear combination of the value vectors, which is the output of the attention mechanism. Here, rather than executing a single attention function with keys, values, and queries, the queries, keys, and values h times are linearly projected to dk, dq, and dv dimensions using separate learned linear projections. Simultaneous application of the attention function to these projected versions of queries, keys, and values, resulted in dv-dimensional output values. One more concatenation and projection (Eq. (7)) of them yielded the final values, as depicted in Fig. 4. Multi-head attention allows the model to simultaneously attend to information from various representation sub-spaces at multiple locations. On the other hand, a single attention head leads to a kind of averaging effect that limits the resolution of the learned representations:

$$MultiHead(Q, K, V) = Concat(head_1, \dots, head_h)W^O. \tag{7}$$

With h attention heads, h sets of learned projection metrics can be found, where $head_i = Attention(QW_i{}^Q, KW_i{}^K, V_i{}^W)$ with the queries and keys having the same dimensionality due to the dot product, and can be represented by $W_i{}^Q \epsilon R^{D\times d_k}$, $W_i{}^K \epsilon R^{D\times d_k}$, and $W_i{}^V \epsilon R^{D\times d_v}$. The h output metrics are concatenated and then multiplied by another weight matrix W^O, which is basically a squared matrix, and is represented by $W^O \epsilon R^{hd_v\times D}$ (Vaswani et al., 2017).

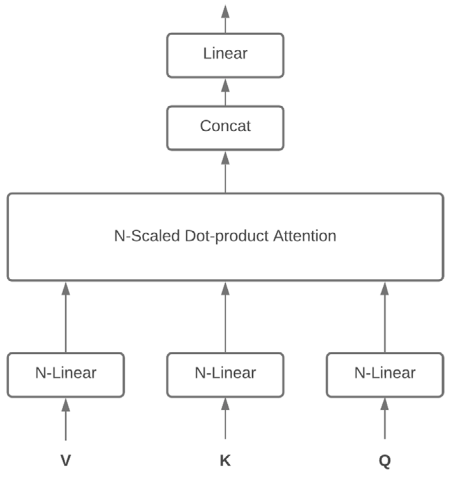


Fig. 4. Used multi-head attention architecture.
Source: Adopted from Vaswani et al. (2017).

4. Experiments

An NVIDIA DGX Station using CUDA 10.1 version facilitates the training process. The used computer station consists of 4 NVIDIA Tesla V100 tensor core GPUs covering a 128 GB GPU memory. The open-source PyTorch machine learning library served as the platform for coding using Python. The first step for code implementation consisted of defining a class for Positional Encoding. The mathematical formulation of Eqs. (3) and (4) helped to acquire the positions of the elements of the positional encoding vector in even and odd time steps. Then, adding each element to its corresponding embedding vector provided the positional information. Secondly, for model construction, another class was built following the descriptions of Sections 3.3.1–3.3.3. The build-in 'torch.nn' package served as the building platform of the encoder and decoder layers. A Square Subsequent Mask function was built within the class to generate a square mask for the sequence by filling the masked and unmasked positions with $-infinite$ and 0.0, respectively. Thirdly, for data preprocessing, a separate function was defined based on the size of the input window to generate the sequences. The 'sklearn.preprocessing' module helped to import the Mini–Max Scalar function. Fourthly, another function of model training was build-up. The pre-built mean squared error function and build-in Stochastic Gradient Descent (SGD) from the 'torch.optim' package demonstrated their advantage in calculating the loss and algorithm optimisation. The computational costs increase with the increase in the number of encoder–decoder layers. It was observed that the insertion of more layers within the transformer encoder and decoder eventually reduces the model's performance. A stacked 5 layers of encoder–decoder architecture with 5 heads provided the best results with a total training time of 19.4 h for 100 epochs. A support vector regression, long short-term memory, and gated recurrent unit-based models were also trained in the same environment to achieve comprehensive comparisons. A 64 unit input layer, two hidden layers with a dropout of 0.2, and a dense output layer constituted the LSTM and GRU studied architectures. The mean absolute percentage error (MAPE) and separate mean squared error (MSE) metrics were used for evaluation purposes.

4.1. Mean absolute percentage error and mean squared error

The proposed model's accuracy was assessed based on the mean absolute percentage error and mean squared error metrics. MAPE for each period was calculated as the average absolute per cent error between the actual and predicted values. If x, y are the actual and predicted values, N the number of samples, and x_i and y_i the values of the ith samples in x and y, respectively, then MAPE has the following form:

$$\mathrm{MAPE}(\mathbf{x},\mathbf{y}) = \frac{100\%}{N}\sum_{i=1}^{N}\left|\frac{x_i - y_i}{x_i}\right|. \tag{8}$$

On the other hand, MSE is the average of the squares of the errors, i.e., the average squared difference between the actual and estimated values. If x, y are the actual and predicted values, N the number of samples, x_i and y_i the values of the ith samples in x and y, respectively, then MSE between them is:

$$\mathrm{MSE}(\mathbf{x},\mathbf{y}) = \frac{1}{N}\sum_{i=1}^{N}(x_i - y_i)^2. \tag{9}$$

4.2. Dataset

The Caltrans' Performance Measurement System (PeMS) dataset (Chen, 2002), which includes real-time traffic data from over 39,000 individual detectors covering the interstate system in all of California's main urban areas over a 10-year period for historical study, was used. It contains 6 months aggregated traffic flow per 5 min of 2028 lane points in the southern California region dated from $1-6-2021$ to $15-12-2021$ with a 99 359 number of samples.

4.2.1. Data analysis

To perform meaningful and accurate forecasts, time-series data analysis involves an inherent understanding of the data, such as seasonality and trends. Generally, line charts are used as a visualisation tool for this kind of analysis. Over an extended period, the trend, which can be upward, downward, or stationary (as analysed in this study), shows the general direction of the time series data. On the other hand, seasonality demonstrates the trends in timing, direction, and magnitude observed between regular intervals due to seasonal factors. It is known that the raw data is additive as the flow amplitude is always proportional to its mean distribution. Hence, the data represents an additive time series where the sum of the components expresses each observation. After that, the application of the decomposition mechanism identified valuable insights, as shown in Fig. 5. The x-axis and y-axis represent the number of samples and traffic flow per 5 min, respectively. The top-most and bottom-most figures represent the line plots of the original data and residual analysis of the data, respectively, while the two figures in between present the found trends and seasonality of the original data, respectively. Only a chunk of datasets between the 1000 to 2000 samples was used for building Fig. 5 so it can be easily understandable. The KPSS test where a *p-value* (Baum, 2018) above 0.05 confirms the stationary properties of the raw data.

A final inspection of the seasonality includes utilising the autocorrelation function (ACF) and lag plot analysis. If autocorrelation is small, it indicates that the data follows small patterns. The ACF plot usually reveals traditional repeated spikes at the multiples of the seasonal window if there is a strong seasonal pattern. The above analysis reveals that the used dataset follows small patterns, Fig. 6.

4.3. Data pre-processing

By comparing the properties of data points, the proposed algorithm tries to detect trends in the dataset. Problems emerge when the features are on significantly different scales. As a result, data normalisation ensures that each data point has the same scale. However, features do not feed the data for transformers, but rather by themselves. Here, the normalisation enables a more smooth training process to avoid large gradients. As the raw data is stationary, the Min–Max normalisation method provided better performance. Other normalisation approaches were tested in the current study, like RobustScaler, which scales features using robust statistics to outliers, and StandardScaler, which standardise features by removing the mean and scaling to unit variance.

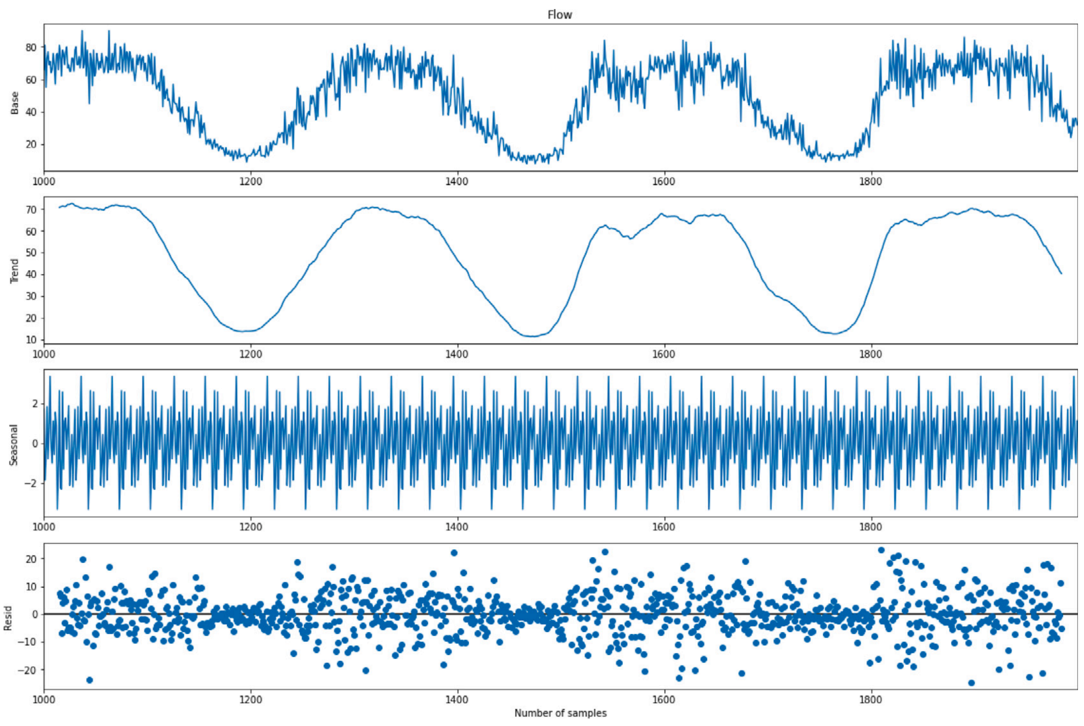


Fig. 5. Data visualisation in terms of seasonality and trends: (from the top to the bottom) 1st, 2nd, 3rd, and 4th graphs represent the base, trends, seasonality and residual data, respectively.

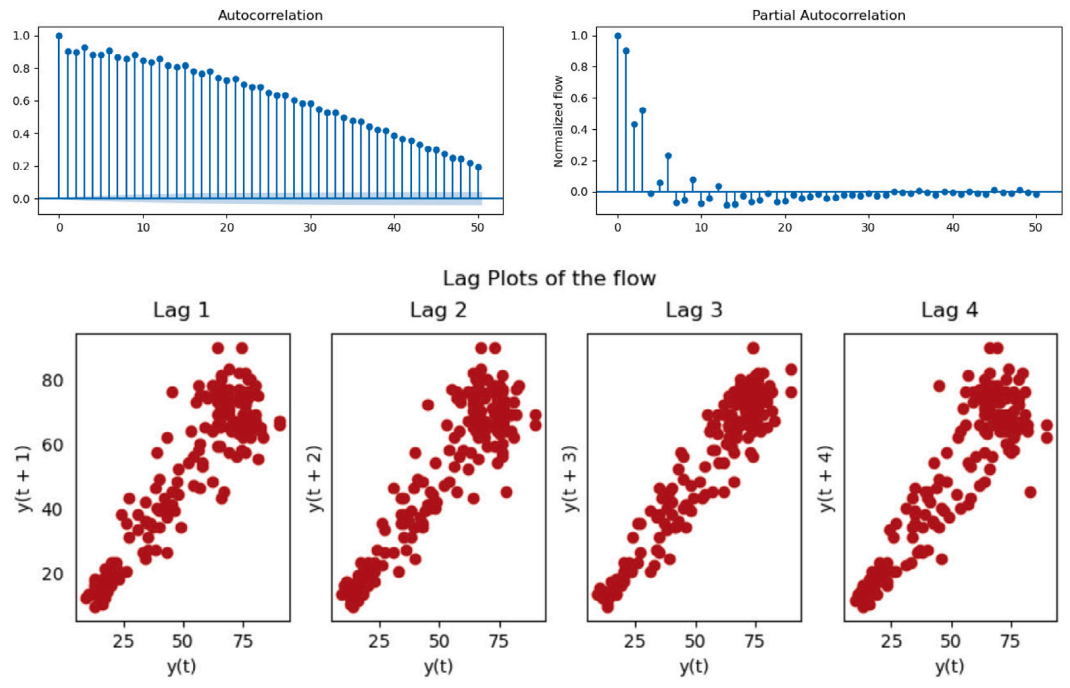


Fig. 6. Auto-correlation and partial autocorrelation analysis of the used dataset: the upper two graphs represent autocorrelation and partial autocorrelation analysis, while the bottom four graphs illustrate lag plot analysis.

However, the Min–Max normalisation technique showed its worthiness in the current context, resulting in the conversion of the minimal and maximum values into a 0 (zero) and 1 (one), respectively. Furthermore, the remaining values find their attributes to a decimal between 0 (zero) and 1 (one). Converting the data into input and target sequences for the model is crucial for the preprocessing data unit. A defined function named ‘generate_in_out_sequences’ was used to perform this step. The first 70k samples served as the training data, and the remaining were for testing. The construction sequences followed their dependencies on the size of the input window. For example, if 100 and 10 are the size of the data and of the input window, then the first 10 data elements would form the first sequence, and the rest would follow the process. In this approach, the length of the last sequence became very short, so removing it was the easiest option to follow. Different input window sizes allow the formation of varying sequence lengths, and an input window of size 200 provided the best result.

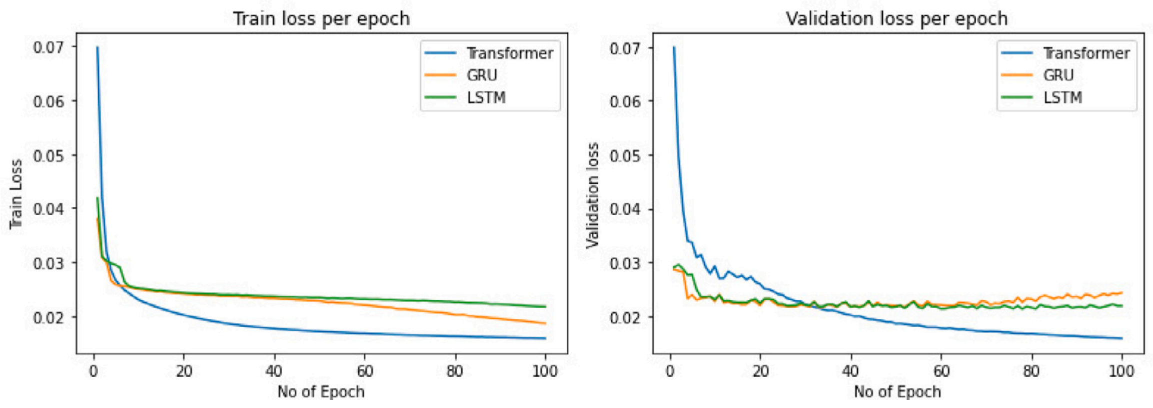


Fig. 7. Training loss and validation loss comparison between the three models: the left and right graphs represent training and validation loss comparisons between the three models under study, respectively.

Table 1
Performance comparison in terms of different metrics.

Metrics	SVR	LSTM	GRU	Transformer
MAPE	51.64%	12.37%	12.66%	8.36%
MSE	3.21	6.07	6.5	3.17

4.4. Training

To build the transformer encoder block, the following parameters demonstrated best performance: $d_{model} = in\_features = 250$, $nhead = 5$, $num\_layers = 5$, and $dropout = 0.1$. On the other hand, for the transformer decoder block, linear transformation of the incoming data with $in\_features = feature\_size = 250$ and $out\_feature = 1$ served the purpose. The mean squared error (squared $L2$ norm) loss function and the stochastic gradient descent (SGD) with a learning rate of 0.005 were used to calculate the loss and optimisation. For algorithm training, the batch size was 10, and at every epoch, the linear step function equated with a learning rate of $gamma = 0.98$ and $step\_size = 1.0$ found worthwhile. The number of model's parameters was 10, 275, 731. The model took 19.4 h for training to 100 epochs with an average training time per epoch of 700.1 s, which is 172.76% higher than of the LSTM or GRU models. However, compared to the baselines, the model seems to be more stable. Fig. 7 illustrates the comparison of the model's training and validation losses against the LSTM and GRU models.

The proposed model learns well without underfitting and overfitting problems relatively the baseline modes under comparison. Nevertheless, the LSTM and GRU based models suffered overfitting problems, as shown in Fig. 7.

4.5. Results

The training of the model consisted of 100 epochs, and after several trial–error checking, a model containing 5 heads with a stacked 5 identical encoder–decoder layers provided the best results. The parameter setup followed the described information given in the previous section. Two metrics: MAPE and MSE, were used to assess the models under comparison. A SVR, LSTM and GRU models were used as baselines. The same computational environment with identical hyper-parameters befitted their implementations and training. All the experiments used 200 time steps of observed data to forecast the next 200 time steps. Compared to the SVR model, the transformer-based model increased the MAPE value by 83.8%. While relatively to the LSTM and GRU models, it demonstrated 32.4 and 33.9% improvements, respectively. One can note that the SVR model demonstrated a MSE value of 3.21, only 1.24% less than of the proposed model. Table 1 presents the obtained results of this comparison.

Table 2
Effects of different parameters on the proposed model's performance.

Heads	Encoder layer	Decoder layer	Parameters	MAPE
10	10	10	20, 551, 211	10.33%
10	6	6	12, 330, 827	9.258%
10	5	5	10, 275, 731	8.43%
5	10	10	20, 551, 211	10.04%
5	6	6	12, 330, 827	9.09%
5	5	5	10, 275, 731	8.36%

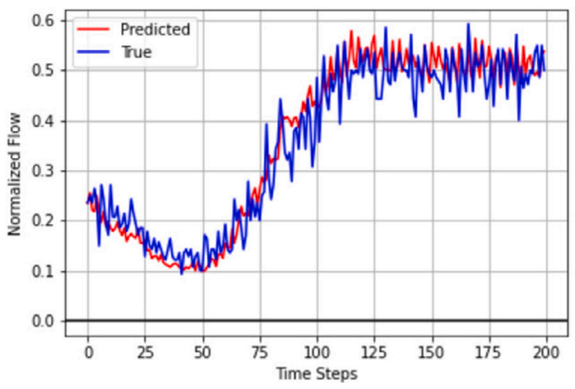
The performance of the proposed model was also compared against another state-of-the-art model: the TrafficBERT model (Jin et al., 2021). The used datasets were the same, but the volume was slightly less than the one used in this study. The authors used only 61/153 days of data from the year 2012; however, here, 6 months of data from the year 2021 were used. In the referred work, 12 observed data points were used to forecast the next 12 time steps (60 min only) and a MAPE of 7.72% was obtained (Jin et al., 2021). It is much more challenging to forecast over a longer time horizon, and hence a MAPE of 8.36 does not represent a drawback of the proposed model. Actually, this study aimed at forecasting much longer time steps, and, in fact, led to excellent results in forecasting over 1000 min time horizons.

Again, for the first 200 time steps, the transformer predicted the traffic flow much better than the other baselines, which indicates the model's worthiness to capture the hidden dynamics of the traffic flow data more effectively and provide better prediction accuracy. Fig. 8 illustrates the predicted and actual values of traffic flow of the proposed transformer-based model over a 200 time steps. For forecasting the future traffic flow, the transformer demonstrated that it is efficient in gaining long range dependencies of the traffic flow, as is shown in Fig. 8. For the next 200 time steps, this figure illustrates that the model can forecast the traffic flow over a higher time horizon.

Table 2 depicts the effects of different parameters on the performance of the proposed model. The model containing 5 heads with 5 identical encoder–decoder layers led to the best results. The increased number of layers within the encoder–decoder architecture deteriorated the model's performance. On the other hand, the increase of the number of heads within the attention mechanism did not provide evidence of performance gain.

5. Discussion

Building traffic forecasting models is challenging because their states are highly dynamic and sometimes difficult to predict. Capturing the Spatio-temporal features of the traffic states is a critical aspect of building a robust model. For SVR, it is difficult to capture those features, because capturing Spatio-temporal features require a large



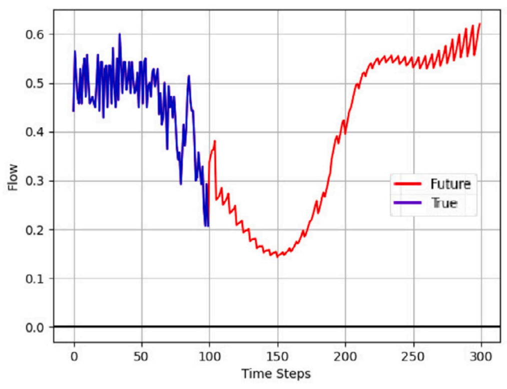
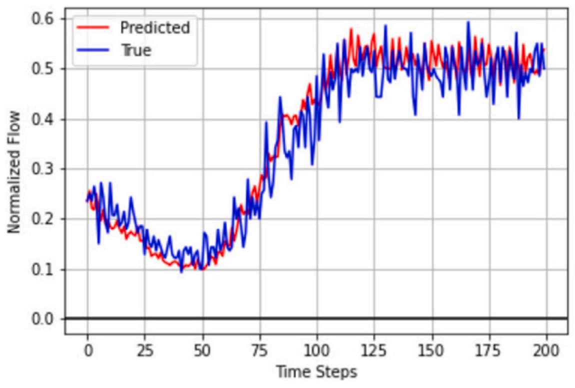


Fig. 8. Performance of the proposed transformer model: the left and right graphs represent the prediction performance and future forecasting, respectively.



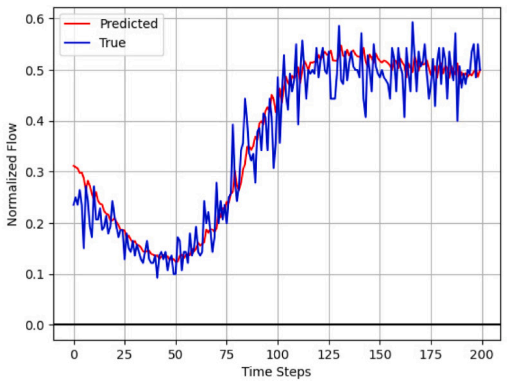


Fig. 9. Performance of the proposed model in terms of the dataset size: the left and right graphs are as to datasets of 100k and 10.2k samples, and test set sizes of 30k and 3k, respectively.

amount of data, and SVR is less capable of dealing with it. However, it is reasonably memory efficient and effectively when there is a clear margin of separation between classes. On the other hand, the proposed model is sufficiently robust when the dataset contains more noise and target classes overlap, and the results provide evidence to confirm its worthiness.

On top of that, the LSTM and GRU models remember every piece of information across time. Hence, they are suitable for time series prediction, mainly because of their ability to remember past inputs. However, in both cases, the path lengths between two positions in a sequence always grow proportionally with distance and support less parallelisation between training samples. Hence, the proposed model is more suitable for gaining long-range features than the LSTM and GRU models.

This study investigated the suitabilities of a multi-head attention-based transformer to overcome the limitations of the conventional models for traffic flow forecasting problems. However, the best outcomes rely on carefully modelling of the attention mechanism. One of the findings of this study depicts that transformers need to be fed with big data to get good performance. The proposed model performed less effectively in gaining the dependencies with less number of data samples (around 10k samples), and prediction followed a mean distribution of the frequency components of the flow values, as shown in Fig. 9, which indicates that the model failed to capture the hidden patterns of the data. Hence, in the case of a multi-attention based transformer, a reasonable volume of traffic states data is required to get a good result.

The selection of an input window, i.e., creating input-target sequences, is vital for good performance. The input data requires proper analysis to find out the seasonality patterns, and the input sequence size needs to contain seasonality or trends. For this study case, an input window of size 200 led to the performances mentioned above.

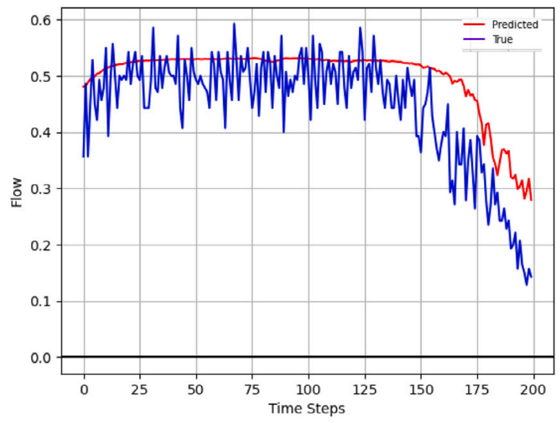


Fig. 10. Performance of the proposed model when a different input window is applied (the hyper-parameters as well as the evaluation criteria were identical to the ones of previous experiment).

This study also confirmed that the proposed model designed for input windows with a size less than 200 time steps (inappropriate selection of input-target sequences) led to inferior results, as shown in Fig. 10.

The main objectives of this study were to verify the usefulness and validity of the multi-head attention based transformer for traffic flow forecasting and compare its performance against the LSTM and GRU based models, which have been the-state-of-the-art in this field for a few years. The results demonstrate that the transformers are well suited for time series forecasting when fed with large data samples and carefully selecting input-target sequences. Also, a multi-head attention mechanism instead of self-attention provides a better performance, especially for multi-step forecasting tasks. The suitability of implementation of such models, both in the private and public sector is wide and ranges from applications in autonomous vehicles (Song, Guo, Li, & Zhang, 2021) or mobility management to the development of adaptive traffic signal controlling systems (Kim & Jeong, 2019). Besides, its implementation may be helpful in highway construction projects (Tong, Guo, & Fang, 2021), dealing with climate management policies (Nejad, Haghdadi, Bruce, et al., 2020), or even modelling road accidents reduction policies (Rashidi, Keshavarz, Pazari, Safahieh, & Samimi, 2022). However, it might require more verification or improvements to increase the model's efficiency and robustness to make it more competent in real-world sectors.

6. Conclusion

Traffic flow forecasting is crucial for intelligent transportation systems. As traffic state is dynamic and sometimes unpredictable, the conventional statistical models fail to provide efficient performance. On the other hand, machine learning approaches, particularly deep neural network-based models, can deal with these problems in manageable ways. This article proposed a multi-head attention based transformer model for traffic flow forecasting tasks. It also presented a comprehensive performance comparison against support vector regression, long short-term memory, and gated recurrent unit based models on the PeMS database. The mean absolute percentage error and mean squared error were used as evaluation metrics. The results confirmed the superior performance of the proposed multi-head attention based transformer model by improving the MAPE value by (32.4–83.8)% over the baselines under comparison. However, the SVR based model shows an MSE value of 3.21, only 1.24% less than of the proposed model. One of the main limitations of the proposed model is its prediction accuracy, and adding more features might solve this problem. Another drawback is that the proposed model is solely effective for a specific road and correspondent data used to train it. Thus, it would require pre-training on other datasets in order to be employed for other roads, which is one of the future developments of this study. Nonetheless, the proposed model does not require rainfall data or adjacent road information to provide a compelling performance, which is crucial in order to be integrated in competent intelligent transportation systems.

CRediT authorship contribution statement

Selim Reza: Investigation, Data collection, Code implementation, Formal analysis, Original draft preparation. **Marta Campos Ferreira:** Writing – review and editing. **J.J.M. Machado:** Writing – review and editing. **João Manuel R.S. Tavares:** Conceptualisation, Funding acquisition, Supervision, Writing – review and editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This article results from the project Safe Cities– ''Inovação para Construir Cidades Seguras'', with reference POCI-01-0247-FEDER-041435, co-funded by the European Regional Development Fund (ERDF), through the Operational Programme for Competitiveness and Internationalization (COMPETE 2020), under the PORTUGAL 2020 Partnership Agreement.

References

Awad, M., & Khanna, R. (2015). Support vector regression. In *Efficient learning machines* (pp. 67–80). Springer.

Bartlett, Z., Han, L., Nguyen, T., & Johnson, P. (2019). Prediction of road traffic flow based on deep recurrent neural networks. (pp. 102–109). http://dx.doi.org/10.1109/SmartWorld-UIC-ATC-SCALCOM-IOP-SCI.2019.00060.

Baum, C. (2018). KPSS: STata module to compute Kwiatkowski-Phillips-Schmidt-Shin test for stationarity.

Chen, C. (2002). *Freeway performance measurement system (PeMS)*. Berkeley: University of California.

Cholakov, R., & Kolev, T. (2021). Transformers predicting the future. Applying attention in next-frame and time series forecasting. arXiv preprint arXiv:2108.08224.

Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2015). Gated feedback recurrent neural networks. In *International conference on machine learning* (pp. 2067–2075). PMLR.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

Dey, R., & Salem, F. M. (2017). Gate-variants of gated recurrent unit (GRU) neural networks. In *2017 IEEE 60th international midwest symposium on circuits and systems (MWSCAS)* (pp. 1597–1600). IEEE.

Ding, X., Liu, J., Yang, F., & Cao, J. (2021). Random radial basis function kernel-based support vector machine. *Journal of the Franklin Institute*, *358*(18), 10121–10140.

Elhenawy, M., & Rakha, H. (2017). Spatiotemporal traffic state prediction based on discriminatively pre-trained deep neural networks. *Advances in Science, Technology and Engineering Systems*, *2*(3), 678–686.

Fu, R., Zhang, Z., & Li, L. (2016). Using LSTM and GRU neural network methods for traffic flow prediction. In *2016 31st youth academic annual conference of chinese association of automation (YAC)* (pp. 324–328). IEEE.

Grigsby, J., Wang, Z., & Qi, Y. (2021). Long-range transformers for dynamic spatiotemporal forecasting. arXiv preprint arXiv:2109.12218.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780.

Jia, Y., Wu, J., Ben-Akiva, M., Seshadri, R., & Du, Y. (2017). Rainfall-integrated traffic speed prediction using deep learning method. *IET Intelligent Transport Systems*, *11*(9), 531–536.

Jin, K., Wi, J., Lee, E., Kang, S., Kim, S., & Kim, Y. (2021). TrafficBERT: PRe-trained model with large-scale data for long-range traffic flow forecasting. *Expert Systems with Applications*, *186*, Article 115738.

Khan, Z., Khan, S., Dey, K., & Chowdhury, M. (2019a). Development and evaluation of recurrent neural network-based models for hourly traffic volume and annual average daily traffic prediction. *Transportation Research Record*, *2673*(7), 489–503. http://dx.doi.org/10.1177/0361198119849059.

Khan, Z., Khan, S. M., Dey, K., & Chowdhury, M. (2019b). Development and evaluation of recurrent neural network-based models for hourly traffic volume and annual average daily traffic prediction. *Transportation Research Record*, *2673*(7), 489–503.

Khodabandelou, G., Kheriji, W., & Selem, F. (2021). Link traffic speed forecasting using convolutional attention-based gated recurrent unit. *Applied Intelligence: The International Journal of Artificial Intelligence, Neural Networks, and Complex Problem-Solving Technologies*, *51*(4), 2331–2352. http://dx.doi.org/10.1007/s10489-020-02020-8.

Kim, D., & Jeong, O. (2019). Cooperative traffic signal control with traffic flow prediction in multi-intersection. *Sensors*, *20*(1), 137.

Li, S., Jin, X., Xuan, Y., Zhou, X., Chen, W., Wang, Y.-X., et al. (2019). Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. *Advances in Neural Information Processing Systems*, *32*, 5243–5253.

Li, J., Wang, X., Tu, Z., & Lyu, M. R. (2021). On the diversity of multi-head attention. *Neurocomputing*, *454*, 14–24.

Lim, B., Arık, S. O., Loeff, N., & Pfister, T. (2021). Temporal fusion transformers for interpretable multi-horizon time series forecasting. *International Journal of Forecasting*.

Lu, H., Ge, Z., Song, Y., Jiang, D., Zhou, T., & Qin, J. (2021). A temporal-aware LSTM enhanced by loss-switch mechanism for traffic flow forecasting. *Neurocomputing*, *427*, 169–178. http://dx.doi.org/10.1016/j.neucom.2020.11.026.

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

Nejad, M. F., Haghdadi, N., Bruce, A., et al. (2020). Climate policy and intelligent transport systems: application of new transport technologies to reduce greenhouse emissions. In *2020 national conference on emerging trends on sustainable technology and engineering applications (NCETSTEA)* (pp. 1–9). IEEE.

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

Pu, Z., Cui, Z., Wang, S., Li, Q., & Wang, Y. (2020). Time-aware gated recurrent unit networks for forecasting road surface friction using historical data with missing values. *IET Intelligent Transport Systems*, *14*(4), 213–219.

Rashidi, M. H., Keshavarz, S., Pazari, P., Safahieh, N., & Samimi, A. (2022). Modeling the accuracy of traffic crash prediction models. *IATSS Research*.

Reza, S., Oliveira, H. S., Machado, J. J., & Tavares, J. M. R. (2021). Urban safety: an image-processing and deep-learning-based intelligent traffic management and control system. *Sensors*, *21*(22), 7705.

Sherstinsky, A. (2020). Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network. *Physica D: Nonlinear Phenomena*, *404*, Article 132306.

Song, X., Guo, Y., Li, N., & Zhang, L. (2021). Online traffic flow prediction for edge computing-enhanced autonomous and connected vehicles. *IEEE Transactions on Vehicular Technology*, *70*(3), 2101–2111.

Tong, B., Guo, J., & Fang, S. (2021). Predicting budgetary estimate of highway construction projects in China based on GRA-LASSO. *Journal of Management in Engineering*, *37*(3), Article 04021012.

Varaiya, P. P. (2007). *Freeway performance measurement system (PeMS), PeMS 7.0*: *Technical report*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., et al. (2017). Attention is all you need. In *Advances in neural information processing systems* (pp. 5998–6008).

Xu, M., Dai, W., Liu, C., Gao, X., Lin, W., Qi, G.-J., et al. (2020). Spatial-temporal transformer networks for traffic flow forecasting. arXiv preprint arXiv:2001.02908.

Yan, H., Ma, X., & Pu, Z. (2021). Learning dynamic and hierarchical traffic spatiotemporal features with transformer. *IEEE Transactions on Intelligent Transportation Systems*.

Yu, B., Yin, H., & Zhu, Z. (2017). Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. arXiv preprint arXiv:1709.04875.

Zhao, Z., Chen, W., Wu, X., Chen, P. C., & Liu, J. (2017). LSTM Network: a deep learning approach for short-term traffic forecast. *IET Intelligent Transport Systems*, *11*(2), 68–75.