Applied Soft Computing Journal 184 (2025) 113783

Contents lists available at ScienceDirect

Applied Soft Computing

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

An Actor–Critic-based adapted Deep Reinforcement Learning model for multi-step traffic state prediction

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

[a] Faculdade de Engenharia, Universidade do Porto, Rua Dr. Roberto Frias, s/n, 4200-465 Porto, Portugal
[b] INESC-TEC, Departamento de Engenharia e Gestão Industrial, Faculdade de Engenharia, Universidade do Porto, Rua Dr. Roberto Frias, s/n, 4200-465 Porto, Portugal
[c] Instituto de Ciência e Inovação em Engenharia Mecânica e Engenharia Industrial, Departamento de Engenharia Mecânica, Faculdade de Engenharia, Universidade do Porto, Rua Dr. Roberto Frias, s/n, 4200-465 Porto, Portugal

ARTICLE INFO

Dataset link: Caltrans PeMS, PeMS-BAY and METR-LA

Keywords:
Intelligent transportation systems
Multi-step ahead prediction
Actor–Critic network
Accumulation of errors
Denoising Autoencoder

ABSTRACT

Traffic state prediction is critical to decision-making in various traffic management applications. Despite significant advancements in Deep Learning (DL) models, such as Long Short-Term Memory (LSTM), Graph Neural Networks (GNN), and attention-based transformer models, multi-step predictions remain challenging. The state-of-the-art models face a common limitation: the predictions' accuracy decreases as the prediction horizon increases, a phenomenon known as error accumulation. In addition, with the arrival of non-recurrent events and external noise, the models fail to maintain good prediction accuracy. Deep Reinforcement Learning (DRL) has been widely applied to diverse tasks, including optimising intersection traffic signal control. However, its potential to address multi-step traffic prediction challenges remains underexplored. This study introduces an Actor–Critic-based adapted DRL method to explore the solution to the challenges associated with multi-step prediction. The Actor network makes predictions by capturing the temporal correlations of the data sequence, and the Critic network optimises the Actor by evaluating the prediction quality using Q-values. This novel combination of Supervised Learning and Reinforcement Learning (RL) paradigms, along with non-autoregressive modelling, helps the model to mitigate the error accumulation problem and increase its robustness to the arrival of non-recurrent events. It also introduces a Denoising Autoencoder to deal with external noise effectively. The proposed model was trained and evaluated on three benchmark traffic flow and speed datasets. Baseline multi-step prediction models were implemented for comparison based on performance metrics such as Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE). The results reveal that the proposed method outperforms the baselines by achieving average improvements of 0.26 to 21.29% in terms of MAE and RMSE for up to 24 time steps of prediction length on the three used datasets, at the expense of relatively higher computational costs. On top of that, this adapted DRL approach outperforms traditional DRL models, such as Deep Deterministic Policy Gradient (DDPG), in accuracy and computational efficiency.

1. Introduction

Traffic state prediction is a fundamental task of Intelligent Transportation Systems (ITS) for managing traffic efficiently. This task involves predicting future states based on past observations, enabling better decision-making and resource allocation. Traditional methods, such as Autoregressive Integrated Moving Average (ARIMA) [1], Exponential Smoothing (ES) [2], Support Vector Regression (SVR), and K-Nearest Neighbours (KNN) [3] offer solutions for linear traffic time-series data but often fall short when handling complex, non-linear patterns present in real-world traffic datasets. Deep learning (DL) models, particularly Long Short-Term Memory (LSTM) networks, have proven effective in capturing such patterns due to their ability to process sequential data and model long-term dependencies.

Single-step prediction has limited practical applicability and is relatively manageable with these models. However, the limitations of single-step predictions underscore the growing need for multi-step-ahead prediction [4], which is significantly more challenging to model. Attention mechanism-based transformers have been employed to tackle

* Correspondence to: Departamento de Engenharia Mecânica, Faculdade de Engenharia, Universidade do Porto, Rua Dr. Roberto Frias, s/n, 4200-465 Porto, Portugal.
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.asoc.2025.113783
Received 30 January 2025; Received in revised form 7 August 2025; Accepted 9 August 2025
Available online 27 August 2025

these challenges successfully [5,6]. However, a fundamental limitation remains: accuracy declines as the prediction horizon extends. Consequently, multi-step prediction using current DL-based models faces significant hurdles as errors accumulate over time, making sustaining accuracy over longer horizons increasingly challenging. On top of that, these models are sensitive to external noise and fail to maintain good prediction accuracy with the arrival of extreme events such as accidents and adverse weather conditions.

Deep Reinforcement Learning (DRL) has emerged as a popular approach in robotics, self-driving vehicles, and adaptive road traffic signal control, demonstrating remarkable resilience in discovering optimal policies to tackle complex tasks in these domains effectively [7–9]. Building on this success, exploring whether DRL can effectively address the challenges associated with traffic state prediction is worthwhile. Despite its potential, there is a significant gap in the literature regarding applying DRL to traffic state prediction problems. By contrast, few studies have focused on using DRL for general-purpose time-series problems, such as energy consumption and trade forecasting. To accomplish the objectives, these studies used advanced DRL architectures, such as Soft Update Duelling Double Deep Q-learning (SU-D3QN) and DDPG. However, beyond stability issues, DDPG's reliance on a deterministic policy limited the exploration of diverse weight combinations, leading to suboptimal performance [10]. In addition, they were modelled for single-step predictions based on small datasets such as only 1000 samples [11].

This research proposes an Actor–Critic-based adapted DRL model to address the abovementioned limitations, where the Actor network makes the predictions (actions), and improves its policies depending on the Critic's judgements. At the same time, the Critic refines its evaluations based on the Actor's actions. The Actor uses layers of LSTMs and Bidirectional Long Short-Term Memory (BiLSTM) to capture temporal correlations in the sequence and predict future time steps. The Critic network also employs LSTMs to estimate Q-values, which guide the Actor optimisation. This technique does not necessarily represent a true DRL approach, but rather an adapted version. State-of-the-art baseline models were implemented for comprehensive performance comparisons, and three benchmark datasets were used for training and evaluation. Also, a state-of-the-art traditional DDPG model was developed and tested for comparisons. The experimental results demonstrated its superior performance, particularly for higher prediction lengths. The main contributions of this study are:

- This study presents a unique adapted DRL-based architecture for multi-step traffic state prediction, incorporating LSTM layers into Actor and Critic networks to efficiently capture temporal dependencies.
- The Actor focuses on making correct future predictions, whereas the Critic network reviews and refines these predictions using Q-values, leveraging RL paradigms.
- By integrating Supervised Learning and RL paradigms, the model outperforms baseline approaches in mitigating the error accumulation problem and retaining good accuracy in the face of uncertainty from non-recurring events.
- It also introduces a Denoising Autoencoder to improve robustness to external noise.

This article is organised as follows: Section 2 overviews selected state-of-the-art methods, highlighting their performance and limitations. In Section 3, the proposed model is described, followed by a description of the experimental setup and achieved results in Section 4. Section 5 discusses the overall performance of the proposed model and some of its features. Finally, conclusions are presented in Section 6.

2. Related works

Traffic state prediction has garnered significant attention in improving the efficiency of ITS, leading to the development of various methods. Many of these methods are focused on single-step prediction, which bears substantial limitations in practical applications [12]. Modelling multi-step prediction, although desirable, is considerably more challenging than single-step prediction due to the previously mentioned reasons. The use of DRL methods in addressing these issues is underexplored, and hence, the literature contains limited research on their usage for traffic state prediction. A few studies have investigated its applicability to different time-series forecasting challenges. This section reviews recent developments in multi-step prediction models, including their fundamental advantages and disadvantages.

Traditionally, statistical models, like Auto Regressive Integrated Moving Average (ARIMA) models, have been widely used due to their interpretability and effectiveness in capturing linear relationships. However, these models often struggle with complex, non-linear patterns in real-world traffic state data [13]. Recent advancements in DL have introduced models capable of modelling such complexities. Among them, LSTMs and Gated Recurrent Units (GRUs) have emerged as powerful approaches for sequential data due to their ability to capture temporal dependencies [14]. LSTMs use memory cells to mitigate the vanishing gradient problem, allowing them to learn long-term dependencies effectively. Research has shown that LSTMs outperform traditional models in traffic state prediction tasks, such as in traffic speed prediction [15], traffic volume prediction [16], and traffic congestion prediction [17]. However, these models process each sample at a time and thus might not be most effective in capturing long-term patterns. Hence, researchers explored attention mechanisms and Graph Neural Networks (GNN) to solve these limitations. For example, Du et al. [18] proposed an LSTM-based encoder–decoder architecture for multi-step traffic flow prediction. Their promising results are confined to short-term predictions only.

Due to the ability to extract long-term trends and dynamic dependencies, transformers based on the attention mechanism are being employed extensively to address this limitation. Feng et al. [19] incorporated an attention mechanism with CNN and BiLSTM to enable predictions up to 60 min ahead. Despite this strategy demonstrating good enhancements, short-term forecasts surpassed long-term predictions. Sattarzadeh et al. [20] proposed a spatial–temporal Autoencoder model incorporating a transformer architecture to improve these drawbacks of vanilla transformers for a prediction length of up to 60 min. The transformer encoder–decoder architecture combining GRU, CNN, and residual connections outperformed the MAE, RMSE and MAPE baselines. However, it lacked sufficient experimental validation to prove the robustness to noise and non-recurrent events. To address these problems, Xue et al. [21] proposed an adaptive spatial–temporal transformer model. It introduced a multi-view temporal attention module to capture short- and long-term dependencies and an adaptive gated fusion mechanism to achieve dynamic fusion of spatial–temporal features. Although it mitigated the influence of outliers, the prediction errors accumulated rapidly with increased prediction time steps. Another method was proposed by Liu et al. [22], which replaced the traditional multi-head self-attention mechanism with a time-environment-aware self-attention block and used a parallel spatial self-attention architecture to capture both short- and long-term patterns simultaneously to enhance the efficiency of conventional transformer-based models. However, it still suffered from the accumulation of errors while the prediction horizons increased.

GNNs are also extensively employed to enable more context-aware and accurate multi-step predictions. For example, Zou et al. [23] introduced a GNN-based model that leveraged spatial–temporal correlations in traffic states to tackle the aforementioned issue. Instead of generating one step at a time, their model predicted all desired future traffic states in a single forward pass, reducing error accumulation by avoiding

reliance on intermediate steps. However, this approach required a fixed prediction horizon during training, requiring retraining for different horizons. Furthermore, it faced challenges adapting to dynamic, unforeseen changes in real-time, such as sudden traffic surges caused by accidents or weather conditions. Combinations of GNNs and attention mechanisms were also applied, such as by Zhao et al. [24], which introduced a Graph Spatial–Temporal Transformer (GSTT) model for multi-step traffic state prediction. This model integrated a multi-view Graph Convolutional Network (GCN) to capture the spatial patterns and a multi-head transformer network to model temporal trends and random disturbances. On the METR-LA dataset, the model achieved an MAE of 2.506 and 3.342 for prediction horizons of 15 and 45 min, respectively. While it demonstrated a slight reduction in the error accumulation phenomenon, the MAE still increased by 25% when extending the prediction horizon from 15 to 45 min, indicating challenges in maintaining consistent accuracy over longer time horizons. Similarly, Luo et al. [25] proposed the Long-Short Term Transformer Network (LSTNN), which integrated a stacked 1D dilated CNN for modelling long-term trends, a dynamic GCN for periodic features, and a short-term trend extractor to capture fine-grained temporal details. Using the METR-LA dataset, the model achieved an MAE of 2.42 and 2.96 for prediction horizons of 15 and 60 min, respectively. Nonetheless, it exhibited an 18.24% rise in MAE as the prediction horizon increased, illustrating the issue of error accumulation. Hence, further improvements could be made to enhance its performance over longer time steps. Similar problems persist in the methods proposed by Geng et al. [26] and Li et al. [27].

Although DRL methods, such as DDPG and Proximal Policy Optimisation (PPO), have demonstrated remarkable success in diverse domains, including finance [28] and robotics [29], they have been less explored in addressing the challenges of multi-step predictions. Nonetheless, Hassan et al. [30] presented an RL-based freight demand prediction model to enhance operational planning decisions, highlighting significant advantages, such as adaptability, dynamic responsiveness, and potential real-time application. However, its evaluation lacked thorough comparisons with other advanced DL models. Similar problems persist in the work by Li et al. [31], where an LSTM captured temporal dependencies in sales data and a DRL agent improved inventory policies by predicting future sales patterns. To address these problems, LSTM-based Actor and Critic networks were proposed, achieving 5–52% of improvement compared to the baselines [32,33]. Yet, the accuracy needs further improvements, and no robustness test to external noise was performed. In these contexts, Ren et al. [34] improved the accuracy and robustness of short-term traffic prediction in cases involving traffic bursts caused by extraordinary events (SEs) such as concerts or sporting events in the Beijing Workers' Stadium neighbourhood using a DDPG framework integrating the LSTM networks. However, these models are insufficient to address the mentioned drawbacks of the existing state-of-the-art.

The aforementioned studies indicate that the accumulation of error phenomenon and noise sensitivity persist in multi-step traffic state prediction tasks and thus require new approaches. Therefore, this research proposes a hybrid method combining Supervised Learning and RL paradigms to investigate their abilities to address the abovementioned problems.

3. Methodology

The proposed model aims to overcome the challenges associated with multi-step ahead prediction, where prediction errors propagate over time. Although the DL-based models have successfully solved many of the current challenges, they still have bottlenecks. Hence, this research adapted the RL concepts with DL frameworks to overcome them. This section presents a comprehensive formulation of the proposed model. As such, Table 1 presents important mathematical symbols and their descriptions necessary to understand the model formulation.

3.1. Problem statement

Let us assume that $X_t = [x_{t-L+1}, \dots, x_t]$ is a past sequence of traffic states where L represents input sequence length. Now, Suppose T is the length of prediction steps; the objective is to predict the corresponding future sequence $Y_{t+1:t+T} = [y_{t+1}, y_{t+2}, \dots, y_{t+T}]$. Hence, the proposed model aims to learn this mapping using:

$$Y_{t+1:t+T} = f_{model}(X_t; \theta), \tag{1}$$

where θ represents the model's parameters learned during training.

3.2. Long short-term memory

The LSTM models use a gating mechanism to address the vanishing gradient problem of Recurrent Neural Networks (RNN) and can capture long-term patterns in data sequences. An LSTM cell is made up of four basic components: (i) the input gate, (ii) the forget gate, (iii) the output gate, and (iv) the cell state update. Consider the preprocessed input $X \in \mathbb{R}^{N\times T\times F}$, where N, T, and F indicate the number of samples, prediction steps, and features, respectively. At time step t, the input gate will process $X_t \in \mathbb{R}^{B\times T\times F}$, where B is the batch size to determine what information to store in the cell state c_t. It uses a sigmoid function σ to determine which values are essential using:

$$I_t = \sigma(W_{xi}X_t + W_{hi}h_{t-1} + b_i). \tag{2}$$

The forget gate determines which part of the previous cell state c_{t-1} to forget according to:

$$F_t = \sigma(W_{xf}X_t + W_{hf}h_{t-1} + b_f). \tag{3}$$

Here, F_t is a vector comprised of values between $[0, 1]$ with 0 (zero) and 1 (one), meaning forget completely and retain completely, respectively. The candidate cell state $\hat{c}_t$ computes the candidate values to add to the cell state. It uses the tanh activation function to scale them between $[-1, 1]$ according to:

$$\hat{c}_t = \tanh(W_{xc}X_t + W_{hc}h_{t-1} + b_c). \tag{4}$$

The cell state is then updated by retaining old information with the help of $F_t \odot c_{t-1}$ and incorporating new information using $I_t \odot \hat{c}_t$ as:

$$c_t = I_t \odot \hat{c}_t + F_t \odot c_{t-1}, \tag{5}$$

where $\odot$ denotes the element-wise multiplication. Finally, the output gate and hidden state are computed using [35]:

$$O_t = \sigma(W_{xo}X_t + W_{ho}h_{t-1} + b_o), \tag{6}$$

$$h_t = O_t \odot \tanh(c_t), \tag{7}$$

where O_t and h_t denote the output gate and hidden state at time step t. The I_t, F_t, and O_t parameters control the flow of information and, thus, enable the network to focus on relevant parts of a data sequence.

3.2.1. Bidirectional long short-term memory

An expansion of LSTMs that can process an input sequence forward and backwards is called a BiLSTM. This phenomenon enables the network to capture dependencies from both past and future contexts through [36]:

$$\overrightarrow{h_t} = \overrightarrow{LSTM}(X_t, \overrightarrow{h_{t-1}}), \tag{8}$$

$$\overleftarrow{h_t} = \overleftarrow{LSTM}(X_t, \overleftarrow{h_{t+1}}), \tag{9}$$

$$h_t = concat(\overrightarrow{h_t}, \overleftarrow{h_t}), \tag{10}$$

where $\overrightarrow{h_t}$ and $\overleftarrow{h_t}$ are the hidden states of forward and backwards LSTMs, respectively.

Table 1
Description of important mathematical symbols used in the formulation of the proposed model.

Symbol	Meaning	Symbol	Meaning
X_t and Y_t	An input and an output sequence of traffic states	t	Time step
N	Number of samples	$\odot$	Element-wise multiplication
L	Length of an input sequence	O_t	The output gate at t
T	Length of the prediction steps	h_t	The hidden state at t
θ	Model's learnable parameters	W_i	Weight matrix
F	Number of features	b_i	Bias vector
B	Batch size	U_i	Dimensionality of the output space
c_t	Cell state	$\hat{c}_t$	The candidate cell state
σ	The sigmoid function	tanh	The hyperbolic tangent function
Y_t and Y_p	True and predicted values	s and a	The state and action
γ	Discount factor	s' and a'	The next state and action
λ	The Weighting factor	Q	Q-values

3.3. Actor model

The Actor network aims to produce the output sequence $Y_a \in \mathbb{R}^{B\times T_o\times F}$ by processing the input sequence $X_t \in \mathbb{R}^{B\times T\times F}$ at time step t using a combination of LSTM and BiLSTM architectures. $X_t \in \mathbb{R}^{B\times T\times F}$ is passed through the first LSTM to capture the temporal features. Then, Y_{a1} is processed by a BiLSTM to improve the temporal context using both forward and backwards dependencies. The second LSTM layer further refines the learned temporal features. Afterwards, two Dense layers are used to make predictions. The Actor network can be formulated as:

$$Y_{a1} \in \mathbb{R}^{B\times T\times U_1} = LSTM(X_t), \tag{11}$$

$$Y_{a2} \in \mathbb{R}^{B\times T\times 2U_1} = BiLSTM(Y_{a1}), \tag{12}$$

$$Y_{a3} \in \mathbb{R}^{B\times T\times U_2} = LSTM(Y_{a2}), \tag{13}$$

$$Y_{a4} \in \mathbb{R}^{B\times T\times U_d} = ReLU(W_d Y_{a3} + b_d), \tag{14}$$

$$Y_a \in \mathbb{R}^{B\times T_o\times F} = Linear(Y_{a4}), \tag{15}$$

where U_1, U_2 and U_d represent the dimensionality of the output space, W_d is the weight matrix, and b_d is the bias vector.

The initial LSTM handles raw sequential input $X_t \in \mathbb{R}^{B\times T\times F}$, extracting short-term relationships while keeping the temporal dimension for the following layer. The BiLSTM collects context from previous and future time steps in $Y_{a1} \in \mathbb{R}^{B\times T\times U_1}$, which improves the model's capacity to learn long-range bidirectional patterns. The second LSTM consolidates all time steps into a fixed-length context vector to output $Y_{a3} \in \mathbb{R}^{B\times T\times U_2}$, reducing dimensionality before Dense layers turn the high-level features into the final prediction. The first Dense layer with ReLU transform $Y_{a3} \in \mathbb{R}^{B\times T\times U_2}$ through a linear operation followed by a non-linear activation to output $Y_{a4} \in \mathbb{R}^{B\times T\times U_d}$. The objective is to enhance the abstraction of high-level features and incorporate non-linearity to facilitate the learning of complex mappings. The final Dense functions are used as an unconstrained output layer for regression, yielding $Y_a \in \mathbb{R}^{B\times T_o\times F}$.

The aforementioned sequence of layers was chosen by extensive experimental studies focusing on preventing underfitting and overfitting while maintaining manageable computational expenses. On top of that, this hierarchical structure was designed to enhance its robustness, i.e., to tackle the sensor errors efficiently.

3.4. Critic model

The Critic network evaluates the quality of the Actor's predictions by computing the Q-values to reflect how good the predictions are. It consists of an LSTM and two Dense layers according to:

$$Y_{c1} \in \mathbb{R}^{B\times U_{c1}} = LSTM(X_t), \tag{16}$$

$$Y_{c2} \in \mathbb{R}^{B\times U_{c2}} = ReLU(Y_{c1} W_{c1} + b_{c1}), \tag{17}$$

$$Q \in \mathbb{R}^{B\times 1} = Linear(Y_{c2} W_{c2} + b_{c2}), \tag{18}$$

where U_{c1} and U_{c2} represent the dimensionality of the output space, W_{c1} and W_{c2} are the weight matrix, and b_{c1} and b_{c2} are the bias vector. Here, Y_{c1} returns only the final hidden state for each sequence in the batch. Hence, the Critic model learns to assign Q-values that correlate with prediction errors.

The Critic network is now less complex than the Actor because it just needs to evaluate the states. The LSTM captures the temporal features from the input sequence X_t and returns $Y_{c1} \in \mathbb{R}^{B\times U_{c1}}$, which is its compressed representation. The first Dense layer with ReLU transforms $Y_{c1} \in \mathbb{R}^{B\times U_{c1}}$ using a linear operation. It is followed by a non-linear activation to learn complex value functions effectively. It acts as a feature extractor for the Critic's value estimation to provide $Y_{c2} \in \mathbb{R}^{B\times U_{c2}}$. The final Dense layer with linear activation outputs a scalar Q-value $Q \in \mathbb{R}^{B\times 1}$ which estimates the anticipated return from the states and is trained by a regression mechanism.

3.5. Reinforcement learning framework

The Actor network learns the policy, i.e., which action to take, and the Critic evaluates the quality of the action. The training process uses the gradient ascent to improve the Actor's policy. The Actor relies on the Critic for assistance, yet operates independently in its optimisation process. The Critic is trained using pure supervised regression, whereas the Actor employs gradient ascent to optimise the predicted Q-value.

This research does not explicitly define the concept of a traditional RL agent, where it engages with the environment, executes actions, receives rewards, and acquires knowledge from experiences, but rather implicitly. The agent is the amalgamation of the Actor and Critic, who collaborate to acquire a policy.

Furthermore, the environment is implicit, and the training data serves the purpose. If $X \in \mathbb{R}^{B\times T\times F}$ represents the input sequence, then:

$$State = Input\ Sequence\ i.e.,\ X \in \mathbb{R}^{B\times T\times F}, \tag{19}$$

$$Action = Actor(X \in \mathbb{R}^{B\times T\times F}). \tag{20}$$

In conventional RL, the Q-value denotes the anticipated cumulative reward for executing an action in a state according to a policy. However, this research uses a non-conventional way to compute the Q-values, adapted for Supervised Learning tasks. Here, an action has no bearing on the Q-values. Rather, it calculates the current state's quality to produce precise predictions and is computed as:

$$Q(s) = Critic(X \in \mathbb{R}^{B\times T\times F}), \tag{21}$$

where s represents the state and unlike $Q(s, a)$, the Q-value is a function of only s. The reward function is not explicitly defined, though it is implicitly a combination of Supervised Learning with RL principles defined as:

$$Reward = -MSE(y, a) + Q(s), \tag{22}$$

where y presents true values, a represents actions. Now, in traditional RL, the Q-value is updated using temporal difference (TD) learning [37]. It involves (i) computing the TD target using $Reward$ +

$\gamma \max_{\acute{a}} Q(\acute{s}, \acute{a})$, (ii) calculating the loss between the target and $Q(s, a)$, and (iii) minimising loss using gradient descent to update Q-value. This research uses similar steps except for the computation of the TD target; here, the ground truth y serves as the target. Also, the discount factor γ is not explicitly defined but implicitly set to zero. Hence, the Q-value update is a supervised regression procedure in which the Critic learns to predict the imminent reward. The proposed mechanism is not a true RL approach but a simplified DRL adaptation.

3.6. Proposed model

The proposed model combines the representational ability of DL with the RL decision-making framework. For an input sequence $X \in \mathbb{R}^{B\times T\times F}$, a conventional DL-based model learns the temporal correlations aiming to predict an output sequence $Y \in \mathbb{R}^{B\times T_o\times F}$, where T_o is the prediction length. Thus, during training, it computes the difference between the actual and predicted sequences using a loss function, such as Mean Squared Error (MSE), defined as:

$$E = \frac{1}{N}\sum_{i=1}^{N}(Y_{t,i} - Y_{p,i})^2, \tag{23}$$

where N is the number of samples. The objective is to make the value of E as small as possible. Thus, the gradient of E is used to update the model's weight during back-propagation. In summary, a conventional DL-based model is optimised by minimising the value of E.

The proposed model uses an evaluation mechanism where the Actor makes predictions, and the Critic judges the prediction quality based on Q-values. The higher the Q-value, the better the prediction performance. For an input sequence $X \in \mathbb{R}^{B\times T\times F}$ at time step t, the Actor network generates the prediction using:

$$Y_t \in \mathbb{R}^{B\times T_o\times F} = Actor(X_t). \tag{24}$$

Then, the Critic network evaluates the prediction using the Q-value, which estimates the quality of the Actor's actions. It reflects how well the Actor's predictions align with the actual output. A higher Q-value suggests better predictions. Therefore, for a time step t, the Q-value can be computed as:

$$Q(s) \in \mathbb{R}^{B\times 1} = Critic(X_t). \tag{25}$$

Accordingly, during training, the network computes the predicted Q-values by combining the MSE loss, i.e., intermediate rewards and the Critic's estimate of long-term value using:

$$P_Q = -MSE(y, a) + \mathbb{E}[Q(s)], \tag{26}$$

where P_Q represents the predicted Q-values. The Actor tries to maximise it by using $-P_Q$ as the loss function during training, thus leveraging the Critic's guidance. Also, during training, the Critic learns how to associate Q-values with the predictions from the Actor network. Fig. 1 illustrates the full architecture of the proposed model according to the formulations mentioned above.

In summary, in this study, the Actor Network is a policy model that produces anticipated actions, i.e., output sequences derived from the preprocessed data input. At the same time, the Critic Network assesses these actions by calculating their Q-values. The feedback loop works as follows: (i) the Actor's loss is calculated as the negative of the predicted Q-value, i.e., P_Q, which combines an MSE between predicted and actual values, and the mean Q-values from the Critic, promoting behaviours that enhance accuracy and Q-value; (ii) the Critic's loss is the MSE between the Q-value and the actual target values, enhancing its ability to evaluate the quality of the Actor's actions; and (iii) both networks are updated via gradient descent, i.e., the Actor's gradients are designed to maximise the Critic's Q-values, while the Critic's gradients minimise its prediction error. The optimised network makes the final predictions. This adversarial dynamic, where the Actor improves its policy based on the Critic's evaluations and the Critic refines its evaluations based on the Actor's changing actions, is central to the adapted DRL model.

3.6.1. Loss functions

Let us assume that $Y_t \in \mathbb{R}^{B\times T\times F}$ is the ground truth or the target values, $Y_p \in \mathbb{R}^{B\times T\times F}$ is the predicted values, and $Q_t \in \mathbb{R}^{B\times 1}$ is the predicted Q-values. Thus, for multi-step-ahead prediction, if MSE is used as the loss function, then L_{pred} is computed as:

$$L_{pred} = \frac{1}{B}\sum_{i=1}^{B}\sum_{j=1}^{T\times F}(Y_{t,i,j} - Y_{p,i,j})^2. \tag{27}$$

Consequently, as stated above, the Actor network aims to maximise the Q-value while minimising the prediction error. The objective is:

$$L_{actor} = -\mathbb{E}[Q(s)] + \lambda L_{pred}, \tag{28}$$

where λ is the weighting factor. On the other hand, the Critic evaluates the Actor's predictions by applying the MSE loss function between the target y_i and the Q-value $Q(s_i)$ using:

$$L_{critic} = \frac{1}{B}\sum_{i=1}^{B}(y_i - Q(s_i))^2. \tag{29}$$

3.7. Data preprocessing

The raw data needs to be preprocessed to ensure proper model training. First, a normalisation step ensures that all the features have the same scale, facilitating faster convergence and better learning during training. It can be accomplished by rescaling each feature based on its minimum and maximum value using [38]:

$$X_{scaled} = \frac{X - X_{min}}{X_{max} - X_{min}}, \tag{30}$$

where X_{max} and X_{min} represents the minimum and maximum feature values. Each feature in the dataset is transformed into a range $[0, 1]$ to prevent large value ranges from dominating the learning process.

Finally, the data is transformed into sequences using a sliding window approach. Each input sequence consists of L past time steps, and the target is to predict the next T time steps. Assume that $X_t = [x_{t-L+1}, \ldots, x_t]$ represents the input sequence; then, $Y_{t+1:t+T} = [y_{t+1}, y_{t+2}, \ldots, y_{t+T}]$ is the multi-step ahead target sequence, which creates input–output pairs for multi-step prediction by sliding a fixed-length window across the dataset. Each cycle generates an input sequence of length L and an output sequence of length T. This method ensures temporal consistency, meaning input–output pairs remain in chronological sequence. It also generates sequences without duplicating data, improving data storage efficiency. Nonetheless, it may lead to some data loss as it eliminates the dataset's final samples to match the sequence lengths, which might provide challenges for short-term prediction.

4. Experiments

The proposed model requires extensive processing power to train. An NVIDIA DGX Station, including four NVIDIA Tesla V100 Tensor Core GPUs and 128 GB of RAM, was used. The code was developed using the open-source TensorFlow machine learning framework and CUDA (version 11.2) for GPU processing.

4.1. Code implementation

The proposed model was implemented in several steps using the TensorFlow platform. The data preprocessing involved filling in missing values using the forward-fill and backwards-fill techniques [39] to ensure temporal consistency. The dataset was scaled with the MinMaxScaler from *sklearn.preprocessing* to improve convergence during training. Input–output sequences were generated using a sliding window approach based on input and output time step lengths.

The architecture design incorporates a DRL framework consisting of Actor and Critic networks. The Actor network used several

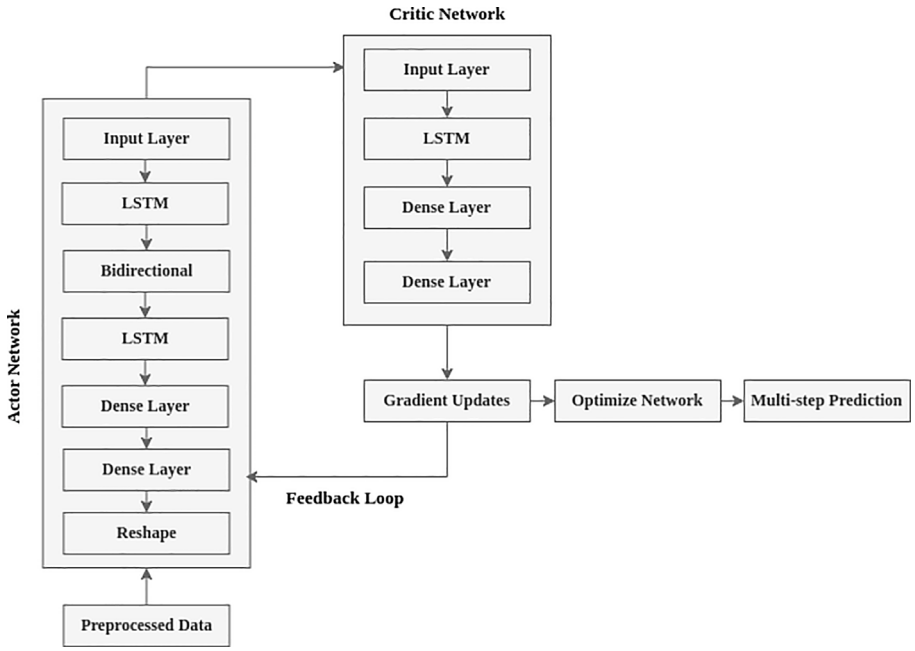

Fig. 1. Architecture of the proposed Actor–Critic-based adapted DRL model for multi-step traffic state prediction tasks.

LSTM layers, including a BiLSTM according to the formulation of Section 3.3, to capture sequential dependencies and temporal patterns using the *tensorflow.keras.layers* module. Dense layers with ReLU activations from *tensorflow.keras.activations* module enhanced the feature extraction process. The Critic network, designed to evaluate the Actor's predictions, used an LSTM layer followed by Dense layers according to Section 3.4 from the *tensorflow.keras.layers* module, to estimate scalar reward signals to guide the Actor during training.

During training, the DRL framework optimised the Actor network to maximise Q-values while the Critic network minimised the prediction errors. The gradients for both networks were computed using *tensorflow.GradientTape*, and the updates were applied via the Adam optimiser using *tf.keras.optimisers* module. Performance metrics, such as MAE and RMSE, were used to monitor the model's outcomes.

4.2. Datasets

The proposed model was trained and evaluated using three publicly available traffic state datasets: the PeMS-BAY, METR-LA [40] and PeMS [41] datasets:

(a) PeMS-BAY dataset: It contains six months of traffic speed data collected from 325 sensors in the San Francisco Bay Area in the USA. The data collection lasted from January 1, 2017, to June 30, 2017, with 52,116 samples. The DateTimeIndex runs from 2017-01-01 00:00:00 to 2017-06-30 23:55:00. This dataset consists of 325 columns, each reflecting readings from a specific sensor, with sensor IDs ranging from 400 001 to 414 694, and does not contain any missing values.

(b) METR-LA dataset: It contains historical traffic speed data from 207 sensors throughout Los Angeles County, California, in the USA. The data collection occurred between March 1, 2012, and June 27, 2012, generating 34,272 samples. The DateTimeIndex ranges from 2012-03-01 00:00:00 to 2012-06-27 23:55:00. This dataset also has 207 columns containing sensor IDs ranging from 773 869 to 769 373, and does not contain any missing values.

(c) PeMS dataset: It consists of traffic flow measurements acquired over 23 days, 19 h, and 5 min, from January 1, 2012, 00:00:00 to January 24, 2012, 19:05:00. It includes 6854 data entries and 154 features. The first column contains the timestamps of each data point, with no missing values. The remaining 153 columns represent traffic flow data acquired by three individual sensor IDs, mainly 200, 201, and 300. A few sensor columns contain minimal missing values, with a maximum of one per column.

4.2.1. Anomaly analysis

This study employed an anomaly detection [42] analysis independently on the PeMS-BAY and METR-LA datasets, utilising an LSTM Autoencoder model [43]. The reconstruction error was computed for both datasets, and abnormalities were detected by establishing a threshold at the 95th percentile of the reconstruction error distribution. The datasets had an anomaly rate of 5.00%, as shown in Fig. 2. These findings indicate the challenges a model faces in detecting and predicting unexpected patterns in traffic data with comparable sensitivity.

4.3. Model training and evaluation

The presented model was trained in two stages: (i) the Actor learned to make predictions that maximise the Critic's Q-value while minimising prediction error, and (ii) the Critic learned to evaluate the Actor's predictions reliably. Gradients are computed so that the two models can be updated. Fig. 3 illustrates the training process of the proposed model. Here, Actor_vars and Critic_vars represent trainable weights of the Actor and Critic networks, respectively. The datasets were split into train, validation, and test sets with a ratio of 70:10:20%, respectively, to perform extensive experiments within the current context.

4.3.1. Hyperparameters

A Random Search method [44] was used to fine-tune the model's hyperparameters based on the search space of Table 2. It performed the task by randomly selecting combinations. Unlike grid search, it explored the parameter space more effectively by prioritising random selection over evaluation, resulting in lower computational costs. The evaluation criterion used during the hyperparameter tuning was the validation MAE, which was selected for its direct and interpretable assessment of prediction accuracy in time series prediction tasks. The upper limit of trials was set to 50, and the method required 41 to acquire the optimal set of hyperparameters. Here, Units_LSTM1_Actor, Units_BiLSTM_Actor, Units_LSTM2_Actor, Units_LSTM_Critic, Units_Dense, LR_Actor, and LR_Critic represent the number of units of the first LSTM layer, the BiLSTM layer, the second LSTM layer of the Actor network, the LSTM units of the Critic network, the units of the

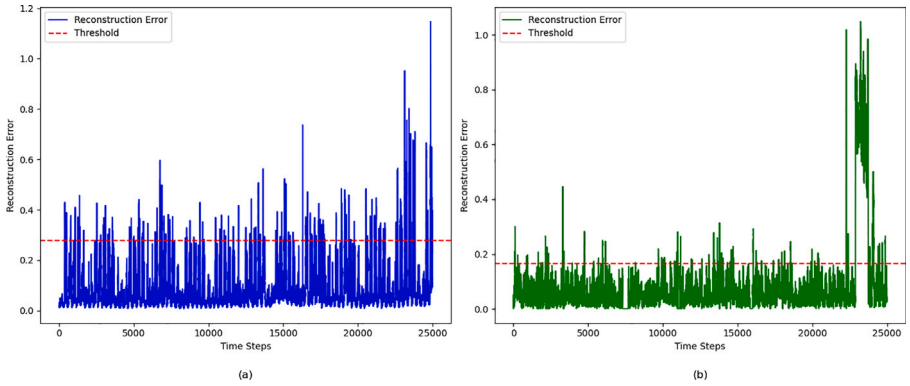

Fig. 2. Reconstruction error, i.e., Anomaly Detection, for the (a) PeMS-BAY and (b) METR-LA datasets.

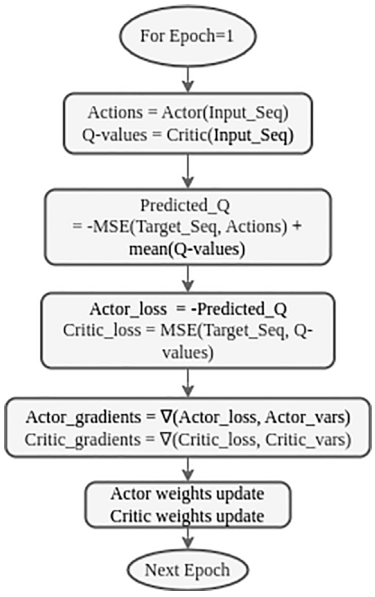

Fig. 3. Illustration of simplified training process of the proposed model for an epoch.

Table 2
Hyperparameter search space explored in this study (the values of the obtained optimal hyperparameters are in bold).

Hyperparameter	Values
Units_LSTM1_Actor	32, 64, 96, 128, 160, 192, 224, 256, **312**
Units_BiLSTM_Actor	32, 64, 96, 128, 160, 192, 224, 256, **312**
Units_LSTM2_Actor	32, 64, 96, **128**, 160, 192, 224, 256, 312
Units_LSTM_Critic	32, **64**, 96, 128, 160, 192, 224, 256, 312
Units_Dense	32, 64, 96, **128**, 160, 192, 224, 256, 312
LR_Actor	$\mathbf{1e^{-4}}, 5e^{-4}, 1e^{-3}$
LR_Critic	$\mathbf{1e^{-4}}, 5e^{-4}, 1e^{-3}$

Fully Connected layers, and the learning rate of both networks. The values of the obtained ideal hyperparameters are indicated in bold in Table 2.

The values in the search space were selected based on best practices reported in relevant literature before being passed into the fine-tuner. The Actor network's first LSTM and BiLSTM layers performed best with 312 memory units. However, its second LSTM layer required 128 memory units. For the Critic network, 64 memory units provided the optimal performance. The optimal learning rate for each network was $1e^{-4}$.

4.4. Performance metrics

The performance of the proposed model was evaluated using standard metrics for traffic state prediction, mainly (i) Mean Absolute Error (MAE) and (ii) Root Mean Squared Error (RMSE). Assume that x_i and y_i represent true and predicted values, respectively. MAE measures the average magnitude of the errors in a set of predictions without considering their direction. It indicates the average deviation of predictions from the actual values according to [45]:

$$MAE(x, y) = \frac{1}{N}\sum_{i=1}^{N}|x_i - y_i|. \tag{31}$$

RMSE penalises larger errors more heavily, measuring overall prediction accuracy. It calculates the square root of the mean of the squared deviations between x_i and y_i using [45]:

$$RMSE(x, y) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - y_i)^2}. \tag{32}$$

4.5. Results

The results of the proposed model were obtained following comprehensive training and evaluation using three state-of-the-art datasets as aforementioned. Also, it was necessary to implement state-of-the-art baseline models to make extensive performance comparisons. Hence, five previously published similar algorithms were implemented to serve this purpose. These models were chosen based on their popularity and ability to model multi-step predictions, and were trained and evaluated using the same settings as the proposed model. The selected models for comparison are:

(a) LSTM-BiLSTM [46]: The authors amalgamated LSTM, BiLSTM, and Dense layers to tackle multi-step-ahead traffic flow prediction challenges. A comparable model was developed, trained, and evaluated using the same datasets used in this study.

(b) CNN-LSTM [47]: This method combined CNN, LSTM, and Dense layers to enhance the prediction accuracy. An identical model was developed, trained, and evaluated on the same datasets used in this study.

(c) Encoder–Decoder [48]: A multi-step-ahead traffic speed prediction model was proposed using an LSTM-based Encoder–Decoder architecture. A Time-Distributed Dense layer was used to make the final prediction. This model was implemented, trained, and evaluated using the same datasets used in this study.

(d) Autoformer [49]: The authors presented a decomposition architecture that included an auto-correlation mechanism for discovering traffic state relationships and aggregating their representations at the subseries level. An identical model was developed, trained, and evaluated from scratch on the three datasets used in this study.

(e) PatchSTG [50]: The authors presented a spatial–temporal patch learning model leveraging the transformer architecture that efficiently

Table 3
Performance of the proposed and baseline models for different prediction time steps (3–24) on the PeMS test dataset based on the MAE and RMSE scores (best values in bold).

Models	MAE					RMSE				
	3	6	12	18	24	3	6	12	18	24
LSTM-BiLSTM	**1.86**	**2.20**	2.78	4.12	4.33	**4.47**	5.62	6.90	7.69	8.61
CNN-LSTM	3.53	3.25	3.44	3.26	3.30	5.93	5.60	5.81	5.51	5.62
Encoder–Decoder	3.44	3.24	3.28	3.32	3.29	5.80	5.50	5.54	5.64	5.55
Autoformer	3.46	3.43	3.41	3.43	3.46	5.83	5.83	5.79	5.76	5.82
PatchSTG	2.62	2.59	2.67	2.84	2.70	4.68	4.86	4.86	4.89	4.88
Proposed	3.42	2.80	**2.60**	**2.65**	**2.54**	5.74	**4.58**	**4.25**	**4.34**	**4.20**

Table 4
Performance of the proposed and baseline models for different prediction time steps (3–24) on the PeMS-BAY test dataset based on the MAE and RMSE scores (best values in bold).

Models	MAE					RMSE				
	3	6	12	18	24	3	6	12	18	24
LSTM-BiLSTM	3.58	3.69	4.01	4.18	5.72	6.57	6.95	7.38	7.77	9.57
CNN-LSTM	3.04	3.35	3.19	3.65	4.50	5.26	5.87	5.64	6.54	7.92
Encoder–Decoder	2.98	3.12	3.20	3.45	3.49	4.86	5.22	5.41	5.82	6.05
Autoformer	3.32	3.41	3.53	3.76	3.70	5.61	5.92	6.14	6.51	6.42
PatchSTG	**2.69**	**2.76**	2.90	2.98	2.96	**4.80**	5.04	5.43	5.57	5.56
Proposed	3.19	2.97	**2.82**	**2.68**	**2.90**	5.35	**4.90**	**4.60**	**4.33**	**4.75**

divided the input data into spatial and temporal patches to capture local and global dependencies in both dimensions. It achieved remarkable accuracy and computing efficiency by leveraging transformer-based methods and patch embeddings. A similar model was developed, trained, and tested for comprehensive comparisons.

Table 3 displays the performance of the proposed model along with the selected baselines obtained for the PeMS traffic flow test dataset for different prediction lengths, i.e., 3 to 24 time steps, using the MAE and RMSE metrics. In this table, the best results are highlighted in bold. For higher prediction horizons, i.e., 12, 18, and 24 time steps, the proposed model outperformed its counterparts by achieving 2.62 to 5.93% lower MAE values than the second-best model. In terms of RMSE, it obtained a 5.8 to 13.93% improvement compared to the second-best model. However, for a prediction length of 3 time steps, the proposed model showed higher errors in terms of MAE and RMSE than its counterparts.

Table 4 presents the obtained results of the models under study regarding MAE and RMSE for different prediction lengths on the PeMS-BAY traffic speed test dataset. In this table, the best results are represented in bold. The proposed Actor–Critic obtained an MAE value of 2.97, 2.82, 2.68, and 2.90 for 6, 12, 18, and 24 prediction time steps, respectively. As to statistics, 2.02 to 10.07% improvements in MAE were achieved compared to the second-best PatchSTG model. However, it underperformed the PatchSTG model for 3 and 5 time steps prediction lengths by 15.67 and 7.07% higher error. On the other hand, in terms of RMSE, compared to the second-best model, 2.78, 15.26, 22.26, and 14.57% improvements were observed for 6 to 24 time steps, while the proposed model also exhibited a 10.29% decrease in performance for a prediction length of 3 time steps.

Table 5 presents the performance of the considered models for prediction horizons of 3 to 24 time steps in terms of MAE and RMSE on the METR-LA traffic speed test dataset. For the three time-step prediction lengths, the Autoformer model demonstrated the best MAE, which is 1.66% lower than the proposed model. However, in other prediction lengths, the proposed model comprehensively outperformed its counterparts both in terms of MAE and RMSE by achieving 21.29% average improvement relative to the second-best model.

Visualising the proposed model's capacity to capture temporal patterns on the test dataset across various network locations is necessary. Fig. 4 plots its prediction performance on the PeMS traffic flow test dataset with a prediction duration of 24 steps. The proposed architecture was modelled to simultaneously predict the traffic states of 153, 325, and 207 sensors in the three datasets used. This figure illustrates the actual (in blue) and predicted (in red) traffic states of four randomly chosen sensors' first 48 time steps. The more similar the two plots are, the better the performance. Consequently, these plots help to visualise the proposed model's ability to achieve state-of-the-art prediction accuracy for longer prediction horizons.

Scatter plots are particularly advantageous in this context as they offer a distinct visual evaluation of prediction accuracy, i.e, the closer the points are to the 45-degree diagonal line, the better the performance. Fig. 5 illustrates scatter plots of four randomly selected sensors comparing the actual and predicted (in blue) values obtained by the proposed model for an output length of 24 time steps on the PeMS traffic flow test dataset. The ideal prediction is represented by the 45-degree diagonal lines (in red), and the closer the blue dots are to the red lines, the better the prediction accuracy. For instance, if the actual traffic flow at a specific time step is 45 vehicles and the model predicts 44.5, the coordinate (45, 44.5) will be positioned slightly below the diagonal line, demonstrating a minor prediction error. From this illustration, it can be observed that the predicted values are closer to the ideal prediction lines, demonstrating the excellent performance of the proposed model. However, for the sensors of subplots (a) and (c), the prediction errors were less significant compared to the subplots (b) and (d). This indicated that different locations in a road network have different traffic characteristics, and the proposed model's performance was not uniform throughout the network.

5. Discussion

In contrast to single-step prediction, which offers merely a short-term perspective, multi-step ahead prediction facilitates proactive measures by elucidating the anticipated evolution of the road network throughout different time steps. However, it is more challenging to model such problems because of the mentioned phenomenon [51,52]. Numerous DL-based models, such as LSTMs, GNNs, and transformers, have been proposed to address these problems. However, these models are sensitive to external noise and unable to maintain good prediction performance due to the arrival of non-recurrent events. Additionally, the errors accumulate with the increase in prediction lengths.

DRL has demonstrated tremendous abilities to enhance the performance of dynamic decision-making tasks such as intersection traffic signal control [9,53]. However, they have been explored less in managing multi-step traffic state prediction problems. This study proposed an Actor–Critic-based model to address the issues mentioned above. Conventional DL models are optimised by minimising the loss function computed by the difference between the actual and predicted sequences during training. The models aim to make the loss as small as possible, and their gradient is used to update the model's weights during back-propagation. On the contrary, the proposed model works so that the Actor network is modelled to make predictions, and the Critic network evaluates the prediction quality based on Q-values. Here, the Actor network, on the one hand, aims at maximising the Q-value while, on the other hand, minimising the prediction error. Also, the Critic network learns to generate Q-values based on the observed Actor network's prediction performance. Three state-of-the-art datasets were used for model training and evaluation. Five related state-of-the-art previously published models were also implemented, trained, and evaluated using the same datasets for comprehensive performance comparisons.

The results demonstrated that the proposed model requires extensive computational resources to complete training while outperforming the baselines on most commonly used metrics, particularly for higher prediction lengths. However, short-term prediction, i.e., 3 and 6 time steps, showed less impressive performance. That phenomenon could have occurred for a variety of reasons. Both the Actor and Critic used LSTMs and had overly complex architectures. It could result in (i)

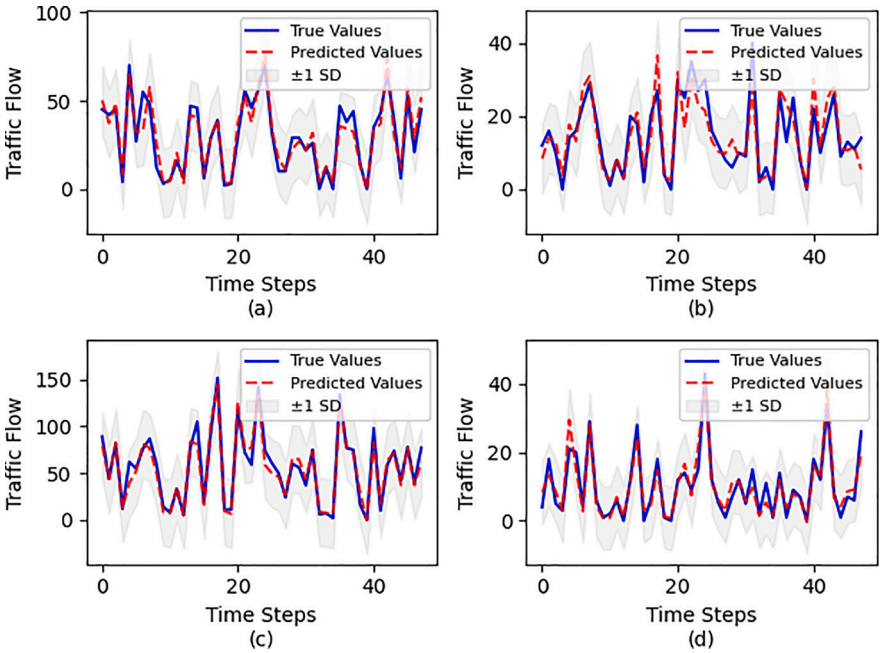

Fig. 4. Traffic flow prediction for an output length of 24 time steps on the PeMS test dataset obtained by the proposed model for four distinct sensors: (a), (b), (c), and (d). (The *x*-axis and *y*-axis represent time steps and traffic flow, respectively. The blue lines demonstrate the true values; the predicted values are in red. The grey band represents the confidence intervals around the projected values, i.e., ±1 Standard Deviation (SD). The smaller the band, the more precise the predictions.)

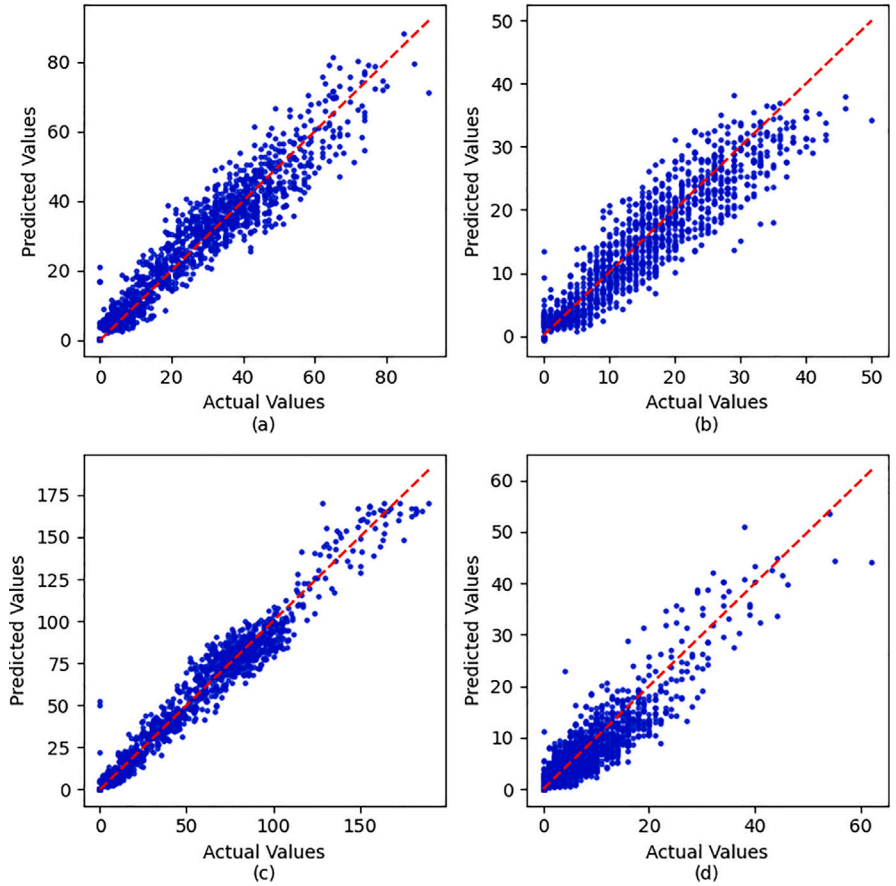

Fig. 5. Scatter plots comparing the actual and predicted traffic flows for a step ahead of 24 time steps on the PeMS test dataset obtained by the proposed model. (The plots represent predictions for four different sensors, (a), (b), (c), and (d), with the red dashed line indicating the ideal case of perfect predictions.)

Table 5
Performance of the proposed and baseline models for different prediction time steps (3–24) on the METR-LA test dataset in terms of MAE and RMSE scores (best values in bold).

Models	MAE					RMSE				
	3	6	12	18	24	3	6	12	18	24
LSTM-BiLSTM	4.87	5.80	6.21	7.38	8.16	8.75	9.36	9.81	10.28	11.47
CNN-LSTM	6.57	6.69	7.65	8.66	10.96	11.70	12.55	14.05	15.67	18.69
Encoder–Decoder	5.79	6.41	6.89	7.57	7.96	10.67	12.26	13.04	13.84	14.49
Autoformer	**4.74**	6.91	7.53	7.47	7.81	8.95	11.24	12.10	13.72	14.52
PatchSTG	4.83	5.33	5.88	6.47	6.84	**7.23**	7.72	8.13	8.51	8.91
Proposed	4.82	**4.59**	**3.98**	**4.09**	**3.79**	8.09	**7.35**	**5.94**	**5.97**	**5.75**

Table 6
Performance of the proposed and DDPG models for different prediction time steps (3–24) on the PeMS dataset regarding MAE and RMSE scores.

Models	MAE					RMSE					Time (s/it)
	3	6	12	18	24	3	6	12	18	24	
DDPG	11.31	11.78	11.34	11.22	11.47	19.66	20.16	19.79	19.72	19.94	365.68
Proposed	3.42	2.80	2.60	2.65	2.54	5.74	4.58	4.25	4.34	4.20	59.49

overfitting of training data, (ii) inefficiency in learning simple short-term patterns, and (iii) vanishing gradients caused by deep recurrent layers. Additionally, it might lack short-term feature extraction skills. Finally, the training technique individually treated data samples in each sequence, which may have limited batch normalising benefits and resulted in unstable gradients. These issues can be addressed by introducing the attention mechanism in the Actor–Critic networks.

5.1. Comparisons with DDPG

A state-of-the-art DDPG model was implemented following a conventional RL approach, combining Actor–Critic methods with DL to validate the proposed model comprehensively. It includes an environment mimicking traffic data interactions, assigning states, and evaluating actions with MAE as rewards. The DDPG agent employs various neural networks: (i) Actor to predict actions, i.e., multi-step predictions, (ii) Critic to evaluate action quality through Q-values, combining state and action inputs, (iii) Target networks, i.e., Target-Actor and Target-Critic to stabilise training via soft updates, (iv) Exploration using the Ornstein–Uhlenbeck (OU) noise [54], which adds temporally correlated noise to actions during training, and (v) Replay Buffer to store experiences, i.e., state, action, and reward for batch sampling. The training technique consists of three steps: (i) updating the Critic network to minimise the TD error between predicted and target Q-values, (ii) updating the Actor to maximise Q-values via policy gradient ascent, and (iii) softly updating the Target networks to ensure stability. After training, the agent provides predictions that are evaluated using MAE and RMSE.

Table 6 tabulates the performance comparisons between the DDPG and the proposed model for 3 to 24 prediction time steps based on the PeMS dataset. Time (s/it) represents training time in seconds per epoch or iteration. Although the DDPG was free from the error accumulation problem, it underperformed compared to the proposed model in all prediction lengths, both in MAE and RMSE. On top of that, it required significantly more computational costs.

While the proposed model took 59.49 s for every epoch and achieved convergence at around 120 epochs, the DDPG required around 500 epochs and 365.68 s per epoch to complete training. These outcomes were obtained for the PeMS traffic flow dataset with only 6854 entities and 153 sensors. The other used datasets, i.e., PeMS-BAY and METR-LA, are much bigger in size and dimensions. Hence, the DDPG took even more computational resources for those cases. Suppose the DDPG was trained on the PeMS-BAY dataset with 52,116 entities and 325 sensors. For a prediction length of 24 time steps, the dimension of the action space would be 7800 (24×325), whereas for a conventional DDPG agent that aims to optimise the traffic signal controlling scheme, the action space dimension is up to four [55,56]. This phenomenon might have some adverse effects because (i) DDPG utilises the OU noise; however, in high-dimensional spaces, noise becomes useless, resulting in gibberish predictions; as a result, the agent failed to engage in significant exploration and became locked in suboptimal policies, and (ii) minor errors in the Actor are exacerbated by the Critic, resulting in divergence, and (iii) agent rarely converges on an appropriate policy. A potential solution to these problems might be an Autoencoder to compress actions and a Hierarchical DDPG.

5.2. Computational costs

This study also investigated the proposed model's computational cost and the considered baselines. The outputs exhibited that the proposed approach significantly improved its prediction accuracy at higher computational costs. Fig. 6 illustrates the performance comparisons between the proposed model and a model similar to the Actor network regarding MAE and RMSE. From this figure, one can confirm that the proposed model based on an adapted DRL approach provided better prediction accuracy than a conventional Supervised Learning approach. However, it required 6.29 times more training time to achieve convergence and 1.5 times more epochs to obtain optimal outputs compared to the PitchSTG model. On top of that, it required 1.11 times more average memory usage during training. Table 7 presents the comparisons of computational costs of the models under consideration. These increases in computational costs were primarily due to the presence of two separate neural networks, i.e., the Actor and Critic networks, and the iterative way of their optimisation. Also, because the Actor and Critic networks were optimised simultaneously, the proposed model required more epochs to obtain optimal performances because of the slower convergence. In terms of statistics, the total computational costs of the proposed model would be around 10.47 times higher than those of the PitchSTG model.

If the computational complexity of a conventional DL model is $\mathcal{O}(N \cdot U \cdot T)$, where N, U and L denote the number of layers, dimension of the hidden states and sequence length, respectively, then, for the proposed model, the computational complexity would increase to at least $\mathcal{O}((N_{actor} + N_{Critic}) \cdot U \cdot T)$, where N_{Actor} and N_{Critic} are the number of Actor and Critic networks' layers, respectively.

Several strategies can be followed to reduce the computational costs. For example, reducing the input–output sequence length and only selecting a handful of sensors. However, these would reduce the prediction accuracy, as was shown in the Results section for a prediction length of three time steps. Also, the model would be less effective if fewer sensors were considered. Another method is batch processing, which uses a batch of input samples through a neural network simultaneously during training or inference, rather than processing samples

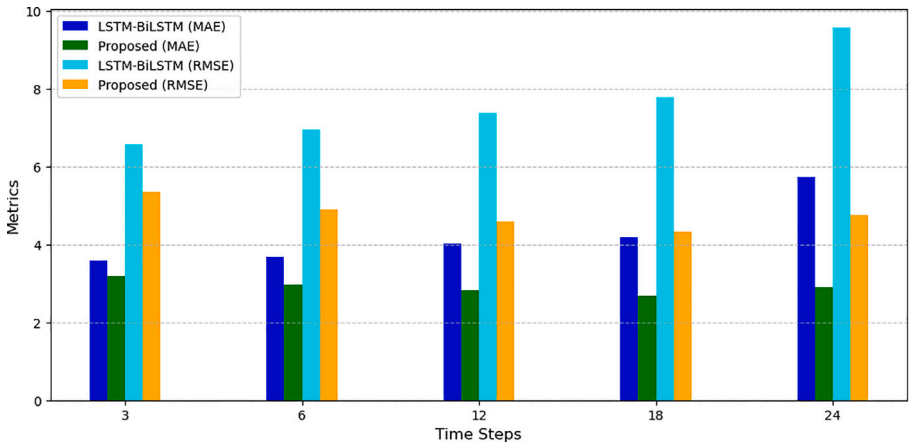

Fig. 6. Traffic flow prediction performance of the proposed model and a conventional DL architecture similar to the Actor network for different prediction lengths on the PeMS test dataset. (The MAE and RMSE values obtained in both cases were used to construct this bar chart.)

Table 7
Computational complexity comparisons between the models under study based on the PeMS dataset for a prediction length of 24 time steps.

Models	Time (s)	Memory (MiB)	Parameters (M)
LSTM-BiLSTM	381.22	1725.79	2.55
CNN-LSTM	271.35	639	0.13
Encoder–Decoder	88.47	599	0.10
Autoformer	117.73	703	0.14
PatchSTG	1134.67	2699	4.6
Proposed	7138.76	2986.79	Actor: 2.56; Critic: 0.06

Table 8
Comparison between the originally proposed model and the model which used a light-weighting technique based on the PeMS dataset for a prediction length of 24 time steps.

Feature	Original	Light-weighting	Change (%)
MAE	2.54	3.38	+33.07
RMSE	4.20	5.68	+35.24
Time (s)	7138.76	243.81	−96.58
Actor's parameters (M)	2.56	0.27	−89.45

one at a time, to use parallel computation. It would increase computational efficiency by better using GPUs, lowering memory overhead, and stabilising gradient updates by averaging losses across batches, resulting in faster convergence. Furthermore, optimisation techniques such as pruning and quantisation could be explored for lower computational costs.

In this study, a model light-weighting technique was investigated by including (i) a Depth-Wise Separable Convolution layer, (ii) decreasing the LSTM units from 312 to 128 and 128 to 64, and (iii) employing Dropout layers with a dropout rate of 0.2 in both Actor and Critic networks following the Input layer. It significantly reduced the computational costs. The training time was reduced from 7138.76 s to 243.81 s based on the PeMS dataset for a 24-timestep prediction length. However, the MAE and RMSE grew from 2.54 to 3.38 and 4.20 to 5.68, respectively, as shown in Table 8.

5.3. Robustness to noise and non-recurrent events

The current state-of-the-art models are sensitive to external noise [57] and fail to maintain good prediction accuracy with the arrival of non-recurrent events such as adverse weather, accidents and public events [58]. To address these issues, a simulation of the arrival of non-recurrent events was performed by adding several sudden spikes and dips to the test dataset to observe the proposed model's performance. The idea was to mimic sharp increases and decreases in traffic flows, usually due to those events, as illustrated in Fig. 7. This figure depicts the traffic flow for the first 250 samples from the test dataset of a randomly selected sensor. Four spikes and dips were created by increasing/decreasing the existing flow values by 2.5 times and 0.3 times to obtain a new test dataset. The proposed model was evaluated on the original and simulated test datasets. The results demonstrated that it can cope with the arrival of non-recurrent events by maintaining good prediction accuracy.

Table 9 presents the obtained MAE and RMSE values on different simulated scenarios of non-recurrent events. Three cases were considered: (i) sharp increase, (ii) sharp decrease, and (iii) a combination of both. Due to these scenarios, the MAE and RMSE values fluctuated by a maximum of 1.96 and 4.52%, revealing the achievements of the robustness goal.

On the other hand, to address the issue of noise sensitivity, this study examined the proposed model on the test dataset, which was added with various levels of Gaussian random noise. Table 10 presents the results of this investigation based on the traffic flow PeMS test dataset for a prediction length of 24 time steps. With the increase in noise level from 0 (zero) to 0.3, the MAE and RMSE rose sharply, indicating that the proposed model is sensitive to various noise levels, which is a drawback of the proposed approach and was addressed by incorporating an LSTM-based Denoising Autoencoder model.

5.3.1. Denoising Autoencoder

A Denoising Autoencoder model was introduced to eliminate noise from data sequences, improving the robustness against random noise. The encoder contained two LSTM layers with 128 and 64 units, followed by a Dense layer with ReLU activation, which compressed input sequences into a lower-dimensional latent representation. The decoder reconstructed the original sequence from the latent space utilising a Dense layer, two LSTM layers (64 and 128 units), and a Time-Distributed Dense layer to align with the input dimensions. The model used a RepeatVector layer to restructure the latent representation before LSTM decoding, ensuring dimensional compatibility for sequence generation. The Adam optimiser minimises the MSE between reconstructed outputs and original clean sequences during training. The denoising task forced the model to learn robust feature representations, requiring it to discern between signal and noise in the data sequences.

The experimental results showed significant robustness improvements compared to the previously obtained ones, as is tabulated in Table 11. Here, for different noise levels, i.e., 0.00 to 0.50, the MAE and RMSE were computed with and without introducing the Denoising Autoencoder. Without the denoising technique, the proposed model's MAE increased dramatically from 2.54 to 41.40 for increasing noise levels. Similar increments were observed for RMSE. However, when

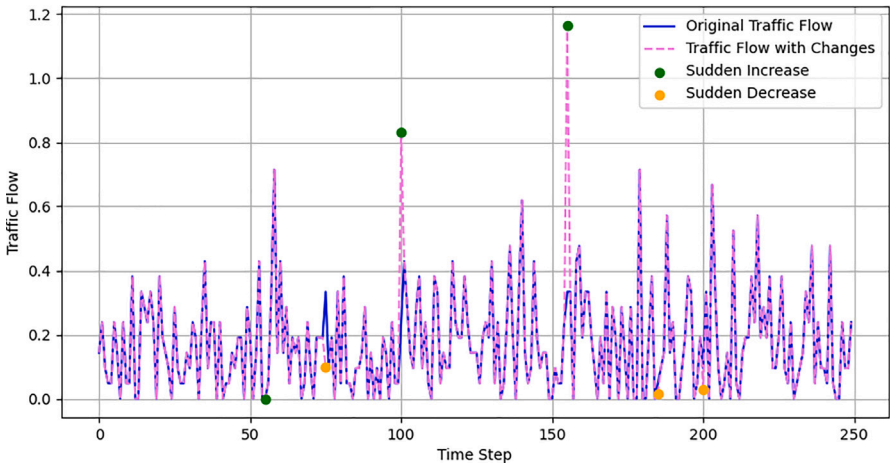

Fig. 7. Graphical illustration of sudden sharp increase and decrease in traffic flows represented by several spikes (green dots) and dips (orange dots) on the PeMS test dataset.

Table 9
Evolution of the model's performance with the arrival of non-recurrent events based on the PeMS test dataset for a 24-step prediction horizon.

Traffic flow	MAE (Original)	MAE (Simulated)	RMSE (Original)	MAE (Simulated)
Sudden increase	2.54	2.57	4.20	4.29
Sudden decrease	2.54	2.57	4.20	4.27
Combined	2.54	2.59	4.20	4.39

Table 10
Exploration of the sensitivity of the proposed model on various noise levels based on the PeMS test dataset for a 24-step prediction horizon (values in bold correspond to the best performance).

Noise level	MAE	RMSE
0.00 (Original)	**2.54**	**4.20**
0.10	9.61	12.28
0.20	17.88	22.59
0.30	26.30	32.76

Table 11
Exploration of the sensitivity of the proposed model on various noise levels based on the PeMS test dataset for a 24-step prediction horizon after incorporating a denoising Autoencoder. The headers 'After' and 'Before' denote the performance metrics obtained with and without the introduction of the Denoising Autoencoder, respectively.

Noise level	Before		After	
	MAE	RMSE	MAE	RMSE
0.00	2.54	4.20	2.89	6.15
0.10	9.61	12.28	2.93	6.27
0.20	17.88	22.59	2.99	6.39
0.30	26.30	32.76	3.11	6.61
0.40	33.35	41.85	3.28	7.01
0.50	41.40	51.96	3.66	8.32

the Denoising Autoencoder was applied, it stabilised significantly, with MAE and RMSE values ranging from 2.89 to 3.66 and 6.15 to 8.32, respectively, concerning the used noise levels.

5.4. Accumulation of errors

Currently available models suffer from the accumulation of error phenomenon [59] because the prediction errors accumulate with the increase in prediction lengths. This drawback significantly reduces the effectiveness of multi-step prediction models, particularly for higher prediction time steps. Compared to the baselines under study, the proposed model demonstrated more efficiency in reducing the accumulation of the error phenomenon, particularly for higher prediction horizons. This improvement was achieved because of two main reasons: (i) non-autoregressive modelling, where all desired future states were computed in a single pass, and (ii) the ability to capture long-term temporal patterns more efficiently. Fig. 8 represents the stacked area plots highlighting the evaluation of the MAE and RMSE values over prediction time steps. It allows one to visualise the cumulative error contributions of the models under study based on the obtained MAE and RMSE values to show how each model's performance aggregates over the prediction time steps. Based on this figure, one can realise that the proposed model outperformed the baselines, particularly for higher prediction steps in terms of MAE and RMSE values, which are indicated by the evolution of the areas' sizes.

This study aimed to explore the DRL in multi-step traffic state prediction to address (i) sensitivity to external noise, (ii) maintaining good prediction accuracy when non-recurrent events arrive, and (iii) the accumulation of errors phenomenon. The proposed model demonstrated that it is robust against sudden changes in traffic states and is less affected by the error accumulation problem. However, it is susceptible to external noise, which was tackled by incorporating a Denoising Autoencoder. Also, its prediction accuracy was less impressive than the baselines for shorter prediction horizons, mainly for 3 time steps. Another drawback of the proposed model is the computational cost. Despite these issues, the presented adapted DRL approach should be explored further by the research communities to address various multi-step traffic state prediction problems.

6. Conclusion

This article presented a novel approach to address the multi-step traffic state prediction problem by proposing an Actor–Critic architecture. The Actor network serves as the base predictor by capturing the temporal dependencies in the input dataset. On the other hand, the Critic network judges the quality of the predictions with the help of Q-values learned during the model's training. The Actor aims to maximise the Q-value and minimise the prediction error. The Critic

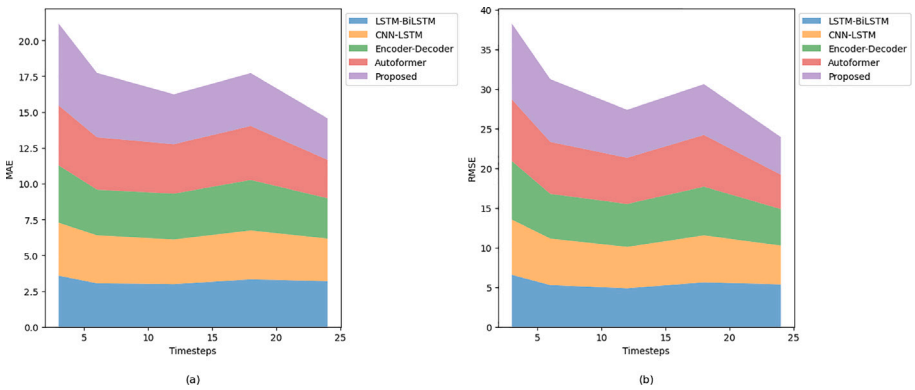

Fig. 8. Stacked area plots of the traffic flow prediction performance of the models under study in terms of (a) MAE and (b) RMSE metrics for different prediction lengths on the PeMS test dataset.

network learns to generate the Q-values associated with the corresponding prediction error. The higher the Q-values are, the better the prediction accuracy. The proposed model demonstrated excellent performance compared to state-of-the-art baselines through experimental validation on three state-of-the-art datasets, particularly for higher prediction lengths. Regarding statistics, it obtained average improvements of 0.26, 3.21 and 21.29%, the MAE and RMSE metrics for the three used datasets, respectively. It demonstrated promising outcomes in minimising the accumulation of errors while maintaining accuracy in introducing non-recurrent events. On top of that, a Denoising Autoencoder was proposed, which helped the model overcome the sensitivity to external noise. This use of DRL paradigms combined with DL in solving multi-step prediction tasks opens new avenues for research. Future work could improve the model's efficiency by reducing the computational costs. Exploring advanced models, such as spatial–temporal GNNs and transformers, like the Actor network, could improve short-term prediction performances. Moreover, comprehensive comparisons with more advanced methods can further validate the usefulness of the proposed model.

CRediT authorship contribution statement

Selim Reza: Writing – original draft, Methodology, Investigation, Formal analysis. **Marta Campos Ferreira:** Writing – review & editing. **J.J.M. Machado:** Writing – review & editing, Supervision, Conceptualization. **João Manuel R.S. Tavares:** Writing – review & editing, Supervision, Funding acquisition, Conceptualization.

Source code

The source code is available at GitHub Repository.

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

The authors would like to thank "Fundação para a Ciência e a Tecnologia" (FCT) for the PhD grant with reference 2022.12391.BD was awarded to the first author, of which this work is a part. This article partially results from the project "Sensitive Industry", co-funded by the European Regional Development Fund (ERDF) through the Operational Programme for Competitiveness and Internationalisation (COMPETE 2020) under the PORTUGAL 2020 Partnership Agreement.

Data availability

The used datasets are publicly available at CaltransPeMS and PeMS-BAYandMETR-LA.

References

[1] S. Deepan, M. Saravanan, Air quality index prediction using seasonal autoregressive integrated moving average transductive long short-term memory, ETRI J. (2024) e12658.
[2] F. Harrou, A. Zeroual, F. Kadri, Y. Sun, Enhancing road traffic flow prediction with improved deep learning using wavelet transforms, Results Eng. (2024) 102342.
[3] X. Luo, D. Li, Y. Yang, S. Zhang, Spatiotemporal traffic flow prediction with KNN and LSTM, J. Adv. Transp. 2019 (1) (2019) 4145353.
[4] J. Deng, X. Song, I.W. Tsang, H. Xiong, The bigger the better? rethinking the effective model scale in long-term time series forecasting, 2024, arXiv preprint arXiv:2401.11929.
[5] J. Chen, L. Zheng, Y. Hu, W. Wang, H. Zhang, X. Hu, Traffic flow matrix-based graph neural network with attention mechanism for traffic flow prediction, Inf. Fusion 104 (2024) 102146.
[6] N.S. Chauhan, N. Kumar, A. Eskandarian, A novel confined attention mechanism driven bi-GRU model for traffic flow prediction, IEEE Trans. Intell. Transp. Syst. (2024).
[7] M. Taghian, S. Miwa, Y. Mitsuka, J. Günther, S. Golestan, O. Zaiane, Explainability of deep reinforcement learning algorithms in robotic domains by using Layer-wise Relevance Propagation, Eng. Appl. Artif. Intell. 137 (2024) 109131.
[8] J. Zhang, J. Ge, S. Li, S. Li, L. Li, A bi-level network-wide cooperative driving approach including deep reinforcement learning-based routing, IEEE Trans. Intell. Veh. (2023).
[9] T. Wang, Z. Zhu, J. Zhang, J. Tian, W. Zhang, A large-scale traffic signal control algorithm based on multi-layer graph deep reinforcement learning, Transp. Res. Part C: Emerg. Technol. 162 (2024) 104582.
[10] Y. Fu, D. Wu, B. Boulet, Reinforcement learning based dynamic model combination for time series forecasting, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 36, 2022, pp. 6639–6647.
[11] H. Zheng, Y. Cao, D. Sun, M. Wang, B. Yan, C. Ye, Research on time series prediction of multi-process based on deep learning, Sci. Rep. 14 (1) (2024) 3739.
[12] Z. Wan, Y. Kang, R. Ou, S. Xue, D. Xu, X. Luo, Multi-step time series forecasting on the temperature of lithium-ion batteries, J. Energy Storage 64 (2023) 107092.
[13] Y. Liu, T. Feng, S. Rasouli, M. Wong, ST-DAGCN: A spatiotemporal dual adaptive graph convolutional network model for traffic prediction, Neurocomputing 601 (2024) 128175.
[14] T. Fischer, C. Krauss, Deep learning with long short-term memory networks for financial market predictions, European J. Oper. Res. 270 (2) (2018) 654–669.
[15] C. Ounoughi, S.B. Yahia, Sequence to sequence hybrid Bi-LSTM model for traffic speed prediction, Expert Syst. Appl. 236 (2024) 121325.
[16] Z. Li, H. Xu, X. Gao, Z. Wang, W. Xu, Fusion attention mechanism bidirectional LSTM for short-term traffic flow prediction, J. Intell. Transp. Syst. 28 (4) (2024) 511–524.
[17] K.N. Kumar, D. Roy, T.A. Suman, C. Vishnu, C.K. Mohan, TSANet: Forecasting traffic congestion patterns from aerial videos using graphs and transformers, Pattern Recognit. 155 (2024) 110721.
[18] S. Du, T. Li, Y. Yang, X. Gong, S.-J. Horng, An LSTM based encoder-decoder model for MultiStep traffic flow prediction, in: 2019 International Joint Conference on Neural Networks, IJCNN, IEEE, 2019, pp. 1–8.

[19] H. Feng, X. Zhang, Y. Xu, Multi-step ahead prediction of traffic speed based on attention-based CNN-LSTM-BiLSTM, in: 2022 5th International Conference on Data Science and Information Technology, DSIT, IEEE, 2022, pp. 1–6.
[20] A.R. Sattarzadeh, P.N. Pathirana, V.T. Huynh, Traffic state estimation with spatio-temporal autoencoding transformer (STAT model), IEEE Access (2025).
[21] Z. Xue, L. Huang, Q. Ning, ASTTN: An Adaptive Spatial–Temporal Transformer Network for traffic flow prediction, Eng. Appl. Artif. Intell. 148 (2025) 110263.
[22] S. Liu, X. Wang, An improved transformer based traffic flow prediction model, Sci. Rep. 15 (1) (2025) 8284.
[23] G. Zou, Z. Lai, T. Wang, Z. Liu, Y. Li, MT-STNet: A novel multi-task spatiotemporal network for highway traffic flow prediction, IEEE Trans. Intell. Transp. Syst. (2024).
[24] Z. Zhao, G. Shen, L. Wang, X. Kong, Graph spatial-temporal transformer network for traffic prediction, Big Data Res. 36 (2024) 100427.
[25] Q. Luo, S. He, X. Han, Y. Wang, H. Li, LSTTN: A Long-Short Term Transformer-based spatiotemporal neural network for traffic flow forecasting, Knowl.-Based Syst. 293 (2024) 111637.
[26] Z. Geng, J. Xu, R. Wu, C. Zhao, J. Wang, Y. Li, C. Zhang, STGAFormer: Spatial–temporal Gated Attention Transformer based Graph Neural Network for traffic flow forecasting, Inf. Fusion 105 (2024) 102228.
[27] Z. Li, J. Zhou, Z. Lin, T. Zhou, Dynamic spatial aware graph transformer for spatiotemporal traffic flow forecasting, Knowl.-Based Syst. 297 (2024) 111946.
[28] Y. Mao, et al., Deep reinforcement learning for trading: A survey, in: Proceedings of the 2020 International Conference on Artificial Intelligence and Statistics, AISTATS, PMLR, 2020.
[29] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal policy optimization algorithms, in: Proceedings of the 34th International Conference on Machine Learning, ICML, PMLR, 2017, pp. 37–45.
[30] L.A.H. Hassan, H.S. Mahmassani, Y. Chen, Reinforcement learning framework for freight demand forecasting to support operational planning decisions, Transp. Res. Part E: Logist. Transp. Rev. 137 (2020) 101926.
[31] C. Li, et al., Deep reinforcement learning for inventory management, Int. J. Prod. Res. (2019).
[32] Y. Li, P. Ni, V. Chang, Application of deep reinforcement learning in stock trading strategies and stock forecasting, Computing 102 (6) (2020) 1305–1322.
[33] J. Zou, J. Lou, B. Wang, S. Liu, A novel deep reinforcement learning based automated stock trading system using cascaded lstm networks, Expert Syst. Appl. 242 (2024) 122801.
[34] Y. Ren, H. Jiang, N. Ji, H. Yu, TBSM: A traffic burst-sensitive model for short-term prediction under special events, Knowl.-Based Syst. 240 (2022) 108120.
[35] K. Zhou, S.-K. Oh, J. Qiu, W. Pedrycz, K. Seo, J.H. Yoon, Design of hierarchical neural networks using deep LSTM and self-organizing dynamical fuzzy-neural network architecture, IEEE Trans. Fuzzy Syst. (2024).
[36] Y. Huang, X. Wan, L. Zhang, X. Lu, A novel deep reinforcement learning framework with BiLSTM-Attention networks for algorithmic trading, Expert Syst. Appl. 240 (2024) 122581.
[37] Y. Zhang, J. Liu, C. Li, Y. Niu, Y. Yang, Y. Liu, W. Ouyang, A perspective of q-value estimation on offline-to-online reinforcement learning, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 38, 2024, pp. 16908–16916.
[38] A. Mahmoud, A. Mohammed, Leveraging hybrid deep learning models for enhanced multivariate time series forecasting, Neural Process. Lett. 56 (5) (2024) 223.
[39] R. Hyndman, Forecasting: Principles and Practice, OTexts, 2018.
[40] Y. Li, R. Yu, C. Shahabi, Y. Liu, Diffusion convolutional recurrent neural network: Data-driven traffic forecasting, in: International Conference on Learning Representations, ICLR'18, 2018.
[41] C. Chen, Freeway Performance Measurement System (PeMS), University of California, Berkeley, 2002.
[42] S. Schmidl, P. Wenig, T. Papenbrock, Anomaly detection in time series: a comprehensive evaluation, Proc. VLDB Endow. 15 (9) (2022) 1779–1797.
[43] Y. Wei, J. Jang-Jaccard, W. Xu, F. Sabrina, S. Camtepe, M. Boulic, LSTM-autoencoder-based anomaly detection for indoor air quality time-series data, IEEE Sens. J. 23 (4) (2023) 3787–3800.
[44] J. Bergstra, Y. Bengio, Random search for hyper-parameter optimization, J. Mach. Learn. Res. 13 (2) (2012).
[45] Z. Xing, M. Huang, W. Li, D. Peng, Spatial linear transformer and temporal convolution network for traffic flow prediction, Sci. Rep. 14 (1) (2024) 4040.
[46] C. Ma, G. Dai, J. Zhou, Short-term traffic flow prediction for urban road sections based on time series analysis and LSTM_BILSTM method, IEEE Trans. Intell. Transp. Syst. 23 (6) (2021) 5615–5624.
[47] H.-J. Lee, D.-J. Park, Collision evasive action timing for MASS using CNN–LSTM-based ship trajectory prediction in restricted area, Ocean Eng. 294 (2024) 116766.
[48] S. Mostafi, T. Alghamdi, K. Elgazzar, Interconnected traffic forecasting using time distributed encoder-decoder multivariate multi-step LSTM, in: 2024 IEEE Intelligent Vehicles Symposium, IV, IEEE, 2024, pp. 2503–2508.
[49] H. Wu, J. Xu, J. Wang, M. Long, Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting, Adv. Neural Inf. Process. Syst. 34 (2021) 22419–22430.
[50] Y. Fang, Y. Liang, B. Hui, Z. Shao, L. Deng, X. Liu, X. Jiang, K. Zheng, Efficient large-scale traffic forecasting with transformers: A spatial data management perspective, 2024, arXiv preprint arXiv:2412.09972.
[51] K. Fu, H. Li, X. Shi, An encoder–decoder architecture with Fourier attention for chaotic time series multi-step prediction, Appl. Soft Comput. 156 (2024) 111409.
[52] E. Strøm, O.E. Gundersen, Performance metrics for multi-step forecasting measuring win-loss, seasonal variance and forecast stability: an empirical study, Appl. Intell. 54 (21) (2024) 10490–10515.
[53] S. Swapno, S. Nobel, P. Meena, V. Meena, A.T. Azar, Z. Haider, M. Tounsi, A reinforcement learning approach for reducing traffic congestion using deep Q learning, Sci. Rep. 14 (1) (2024) 1–20.
[54] H. Jin, Z. Li, H. Cheng, Y. Xia, H. Hu, Sum-rate maximization for uplink multi-user NOMA with improper Gaussian signaling: A deep reinforcement learning approach, IEEE Trans. Veh. Technol. (2025).
[55] G. Yang, X. Wen, F. Chen, Multi-agent deep reinforcement learning with graph attention network for traffic signal control in multiple-intersection urban areas, Transp. Res. Rec. (2025) 03611981241297979.
[56] H. Hu, S. Lin, P. Wang, J. Xu, Control of traffic network signals based on deep deterministic policy gradients, Appl. Intell. 55 (6) (2025) 381.
[57] Y. Liu, S. Rasouli, M. Wong, T. Feng, T. Huang, RT-GCN: Gaussian-based spatiotemporal graph convolutional network for robust traffic prediction, Inf. Fusion 102 (2024) 102078.
[58] X. Zheng, S.A. Bagloee, M. Sarvi, TRECK: Long-term traffic forecasting with contrastive representation learning, IEEE Trans. Intell. Transp. Syst. (2024).
[59] Y. Ji, Y. Luo, A. Lu, D. Xia, L. Yang, A. Wee-Chung Liew, Galformer: a transformer with generative decoding and a hybrid loss function for multi-step stock market index prediction, Sci. Rep. 14 (1) (2024) 23762.