A NEW EVALUATION METRICS AND HYBRID DEEP LEARNING MODEL TO ENHANCE PREDICTIVE PERFORMANCE OF TRAFFIC FLOW

Anonymous authors
Paper under double-blind review

ABSTRACT

Traffic flow prediction is essential in transportation management, as it can help reduce congestion and optimize the planning of transport. With the emergence of deep learning techniques, several models have been developed for traffic flow prediction. In this research paper, we proposed new evaluation metrics to calculate the gain of performance prediction models, based on the Root Mean Square Error (RMSE), Mean Squared Error (MSE), and R-squared (R^2) scores with balancing coefficients. Then, we proposed a new hybrid Long Short-Term Memory-Gated Recurrent Unit (LSTM-GRU) model. The experimental results of this study emphasize the effectiveness of deep learning models in predicting traffic flow. The hybrid models combining GRU (Gated Recurrent Unit) and LSTM (Long Short-Term Memory) show promising performances in forecasting tasks. The hybrid GRU-LSTM (Gated Recurrent Unit-Long Short-Term Memory) emerges as the best compromise vs. LSTM, GRU, and LSTM with the Wadam optimizer. This model exhibits stable performance across the train, validation, and test sets, indicating it does not suffer from overfitting, as proved by the gain evaluation metrics. Besides, it provides a good balance of accuracy, stability, and computational efficiency.

KEYWORDS : Deep Learning - Gated Recurrent Unit - Long Short-Term Memory - Hybrid model - Traffic Flow Prediction – Evaluation metrics - Balancing Coefficients - Gain metric.

1 INTRODUCTION

In recent years, the escalating complexity of urban landscapes and transportation systems has necessitated the development of evolution techniques to address the improvement posed by traffic management. Predicting traffic flows becomes crucial to help decision-making in the planification of transport to optimize transportation networks.

Deep Learning (DL) and Machine Learning (ML) methods have emerged as powerful tools in traffic prediction and forecasting tascks. Deep learning is a subset of machine learning, leverages artificial neural networks to automatically extract intricate features from vast datasets. ML provides a broader framework encompassing various algorithms that enable systems to learn and improve through training. Integrating DL and ML methods brings a paradigm shift in traffic flow prediction by enabling models to distinguish complex patterns, and provide predictions results. Also, these methods can process massive datasets, including real-time traffic information, historical patterns, and environmental factors, thereby enhancing the accuracy and reliability of predictions. DL characterized by its ability to automatically learn hierarchical representations of data, proves to be exceptionally adept at extracting intricate features from vast and complex datasets. Applying deep neural networks, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), enables the model to discern nuanced patterns within traffic data, facilitating more accurate predictions. ML encompassed a broader set of algorithms, complements DL by offering versatile tools for analyzing historical data, identifying trends, and building predictive models. Supervised learning techniques, such as regression methods, can be employed to develop robust models that capture the non-linear relationships inherent in traffic flow dynamics.

Ren et al. (2021) combined the sequential structure of LSTM with CNN's guidance on the local trend of traffic flow. To improve the performance of traffic flow prediction models, researchers have proposed combining multiple DL methods, which have shown promising results in terms of prediction accuracy and stability, Ren et al. (2021). LSTM model has emerged as a popular approach for short-term traffic prediction, Khan et al. (2023). This model is particularly effective in capturing the long-term dependencies in traffic flow datasets, making them well-suited for predicting traffic patterns over short time horizons, Redhu & Kumar (2023). A taxonomy of LSTM models for traffic prediction has been developed, categorizing models based on their input features, output targets, and other vital characteristics, Khan et al. (2023). This taxonomy provides a framework for understanding the different approaches to LSTM-based traffic prediction and their strengths and weaknesses. The applications of LSTM for short-term traffic prediction are numerous and varied, Khan et al. (2023). These models have been used to predict traffic flow, speed, and travel time also to detect congestion and abnormal traffic conditions, Abbas et al. (2018). These applications have essential implications for traffic management, allowing transportation authorities to optimize traffic flow, reduce congestion, and improve road safety. Additionally, LSTM-based traffic prediction can be integrated with other intelligent transportation systems, such as traffic signal control and route guidance, to create a more efficient and sustainable transportation network. Despite the advantages of LSTM-based traffic prediction, several research areas and future trends should be considered, Khan et al. (2023). In addition, more research is needed on the interpretability and ability of LSTM models to handle complex traffic scenarios, such as incidents and accidents. Therefore, there is a growing interest in using LSTM models in combination with other machine learning techniques, such as boosting techniques, to improve traffic prediction accuracy and efficiency, Khan et al. (2023).

In this paper, we propose a comparative analysis of four deep learning methods, LSTM, LSTM with Wadam optimizer, GRU, and Hybrid LSTM-GRU. Moreover, the combined model of LSTM and GRU has demonstrated outstanding prediction accuracy and stability performance. When analyzing prediction models, it is essential to consider prediction accuracy and stability. This paper is divided into four sections. The first section addresses the Traffic Flow Prediction and Forecasting Problem to provide its definition and motivation. The second section presents a literature review that focus on LSTM and GRU models in traffic flow prediction and applications of Hybrid Deep Learning Models. The next section 3 interprets Data Sets used in this study and introduces the proposed contributions. Firstly, the definition of a new metric called "gain" is based on new balancing coefficients. Secondly, we proposed a new hybrid model. The Experimental Results are presented afterwards in the fourth section, with a subsection dedicated to LSTM model, LSTM model with Wadam optimizer, GRU Model, and the Proposed Hybrid LSTM-GRU Model. Finally, we conclude with a comparison and analysis of the models tested by the new proposed compromise metric.

2 Traffic Flow Prediction and Forecasting Problem: Scientific Context and Motivations

Traffic flow prediction (TFP) is a crucial aspect of transportation engineering and traffic management, as it affects the safety, efficiency, and sustainability of road networks, urban planning, and public transport systems. Many variables influence TFP, including external factors such as precipitation, average wind speed, temperature, and weather types, Zeng et al. (2023). The multi-factor GRU model, the Attentive Attributed Recurrent Graph Neural Network (AARGNN), and the multi-graph convolutional-recursive neural network (MGC-RNN) are some models that consider various external factors for TFP, Zeng et al. (2023). Additionally, the volume and density of traffic flow are key variables that affect TFP. This problem is based on data obtained from various sources, including intelligent data acquisition terminals such as radar and GPS devices, which provide a large amount of data reflecting the state of the road network, forming the foundation for TFP, Chen et al. (2023). Other variables include holidays and seasonal information, Zhang et al. (2023). Historical traffic data is used to predict traffic flow parameters, and incorporating relevant factors into the model can improve prediction accuracy, Chen et al. (2023) and Zeng et al. (2023). ML methods, such as ARIMA and LSTM, have been applied to traffic flow prediction and online regression support vector machine models (OL-SVR) for highways under typical and atypical conditions, Chen et al. (2023). The optimization of the prediction method is essential to improve prediction efficiency and accuracy, Chen et al. (2023). Weather conditions are one of the many external factors that significantly impact TFP. Weather data is integrated as multi-source external data, which enhances the capture

ability of geospatial semantic information of the TFP model Chen et al. (2023). The meteorological factors include average temperature, wind speed, and maximum and minimum temperatures, Zhang et al. (2023). The prediction model considers meteorological factors for TFP, with expressions for the inputs at a time t, including whether it is a weekday, rainfall intensity, and temperature, Zhang et al. (2023) and Zeng et al. (2023). Weather conditions are considered external factors in traffic flow forecasting with missing data, Zeng et al. (2023). Temperature and rainfall intensity are used for data imputation and forecasting in TFP, Zeng et al. (2023). Moreover, considering whether it is a working day further improves the predictive performance of the models due to the strong influence of weekends on traffic flow. The proposed hybrid models that incorporate multiple factors, including weather conditions, outperform traditional models in terms of prediction accuracy, Zeng et al. (2023). However, including temperature and rainfall intensity somewhat reduces prediction accuracy, as their correlation with traffic flow could be higher, Zeng et al. (2023). In addition, weather conditions notably impact TFP and should be included in models for enhanced accuracy. Also, road infrastructure plays a crucial role in predicting traffic flow. One way to achieve this is by using the topology structure of the road network to construct a graph and extract spatial features, Chen et al. (2023). Besides, an urban traffic signal control system can help reduce traffic congestion and accidents. However, traditional signal lamps based on fixed traffic signal control strategies cannot deal with complex and changeable urban traffic states, leading to increased traffic congestion and accidents, Zhang et al. (2023). Inadequate road infrastructure can also contribute to traffic congestion, accidents, emissions of carbon dioxide, and substantial economic losses, Zhang et al. (2023).

In addition, TFP is a significant research area in both industry and academia, with recent advancements in DL models highlighting their significance in facilitating efficient traffic management. The existing methodologies for predicting traffic flow can be broadly categorized into two methods. The first method is based on predicting traffic flow using past information on traffic flow, and ML and DL techniques such as Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) have been applied to improve prediction accuracy, Razali et al. (2021). The second method is based on TFP using intelligent traffic systems, which is an essential reference for vehicle route planning, contributes to sustainable traffic development at an economic level, and serves as a crucial foundation for traffic demand policies in intelligent transportation systems, Kim & Lee (2023). Machine Learning models, such as Support Vector Regression and K-Nearest Neighbor (KNN), have been used to predict traffic flow, with the KNN algorithm utilized to introduce a map model that captures the spatial and temporal dependencies in traffic flow data, Kim & Lee (2023).

The quality of traffic data is crucial for the efficacy of Intelligent Transportation Systems (ITS), Sayed et al. (2023). Accurate predictions are required for proactive responses in traffic control systems as road traffic deaths continue to rise, highlighting the importance of traffic forecasting to reduce congestion and ensure safer travel, Sayed et al. (2023) and Kim & Lee (2023). Accurately TFP requires using a DL model, Sayed et al. (2023). However, the lack of computationally effective methodologies, algorithms, and high-quality training data also influenced the prediction accuracy of traffic flow, Sayed et al. (2023). Additionally, the training data size can be tiny, so the training process must be reapplied to reflect the newer dataset, which takes additional time, Sayed et al. (2023). A benchmark for the size of uncertainty is essential, as it is necessary to determine the reliability of the predicted values through uncertainty estimation, especially when distributing resources to events that have yet to occur, Kim & Lee (2023).

Various solutions address the issue of TFP in urban areas. One approach is through traffic classification, traffic generation, and traffic forecasting, which are the three types of traffic prediction problems. These problems consider five features: road network, environmental data, spatial property, temporal property, and nonlinearity, Yuan & Li (2021). Traditional learning methods such as regression, decision trees, and hidden Markov models are applied in TFP, Yuan & Li (2021). Non-learning methods like kNN and historical averages are also considered for this purpose, Yuan & Li (2021). The road network structure is a considerable constraint when handling traffic prediction on roads or intersections, Yuan & Li (2021). Temporal features like holiday information and events are also helpful for effective traffic prediction. In addition, environmental data such as weather plays a vital role in TFP, Yuan & Li (2021). Deep architecture models are used to predict traffic flow in urban areas, Yuan & Li (2021).

3 LITERATURE REVIEW : DEEP LEARNING MODELS FOR TRAFFIC FLOW PREDICTION

Various models have been proposed to predict traffic flow and offer intelligent transportation systems, but traditional models suffer from the limitations of shallow learning architecture, Sroczyński & Czyżewski (2023).

Long Short-Term Memory (LSTM) neural network models have been widely adopted and studied for TFP in recent years. Compared to non-parametric models such as Support Vector Machines (SVMs), Kalman Filter, and Autoregressive Integrated Moving Average (ARIMA), LSTM models have shown superior prediction accuracy, Essien et al. (2021). LSTM models can be combined with deep belief networks to predict short-term traffic speed. Fusing weather and traffic data sources can improve the prediction performance of LSTM and deep belief network models in traffic flow prediction, Essien et al. (2021). Improved predictive accuracy can reduce frustration for road users, create cost savings for businesses, and cause less environmental harm, Essien et al. (2021). The LSTM-based model outperforms other statistical and machine learning (ML) models in improving prediction accuracy. Authors modeled the spatial dependence of the traffic data using Convolutional Neural Networks (CNN) and LSTM, demonstrating improved accuracy in comparison to traditional ML methods such as the ARIMA model, SVM, and SVR models, Essien et al. (2021) and Zhou et al. (2022a). Given their invariable model structures and weights, traditional machine learning methods cannot effectively extract the spatiotemporal dependencies between multi-source traffic data. The LSTM model overcomes these disadvantages by stacking neural network infrastructure and training the network with gradient descent, Zhou et al. (2022a). LSTM performs better than traditional ML methods in capturing temporal dynamics in TFP, as well as outperforming Deep Belief Networks (DBN) in capturing time series characteristics of traffic flow data, Zhou et al. (2022a) and Jia et al. (2017).

The architecture of a Gated Recurrent Unit (GRU) model is a type of Recurrent Neural Network (RNN) architecture that incorporates gating mechanisms. GRUs are similar to LSTM models but lack a context vector or output gate, resulting in fewer parameters than LSTM. The GRU model includes two specific gates: the update gate and the reset gate, which are two vectors that determine which information should be passed to the output. The update gate controls the degree to which new information is incorporated, while the reset gate controls the degree to which past information is forgotten. The gating mechanisms enable selective information update and resetting in the hidden state, allowing GRUs to retain essential information and forget irrelevant data. The gates decide what information is allowed through to the output, and they can be trained to keep information from going further back, which leads to better predictions. GRUs can pass relevant information down a chain of events to facilitate learning of long-term dependencies in sequential data processing. GRUs are a powerful solution for sequential data processing due to their innovative gating mechanisms that address the limitations of traditional RNNs. Furthermore, GRUs have a simpler architecture than LSTMs and have fewer parameters while being as efficient as LSTMs. They were introduced in 2014 by Kyunghyun Cho et al. GRUs have become increasingly popular in DL for various applications due to their ability to overcome the vanishing gradient problem in traditional RNNs, making them a valuable tool for researchers and practitioners, Cho et al. (2014).

LSTM models have become increasingly popular in TFP due to their ability to handle nonlinear and dynamic data. One of the strengths of LSTM models is their ability to predict traffic flow changes in a short period, Shi & Du (2022). Moreover, LSTM models can use incomplete traffic flow data from the past periods of the target prediction section and complete data from the adjacent sections to jointly predict traffic flow changes, Shi & Du (2022). The proposed MLR-LSTM method combines multiple linear regression with LSTM models for accurate prediction of future traffic flow changes, even when past traffic flow data of the target section is partially missing, Shi & Du (2022). Results have shown that the accuracy of the prediction results based on this method is similar to that of mainstream prediction results using continuous and non-missing target link flow data, Shi & Du (2022). However, there are limitations to using LSTM models in TFP. The proposed prediction method may need to be more feasible and widely used, Shi & Du (2022). While the proposed KNN-LSTM model effectively predicts traffic flow values almost coinciding with measured data, especially during morning and evening peak hours, there is only a small-scale improvement when the data time interval is short enough, Shi & Du (2022) and Luo et al. (2019). Furthermore, traditional models like ARIMA have limitations, as they assume traffic flow data is stationary, which is only

sometimes accurate in reality and can lead to poor prediction performance. In contrast, DL methods like the KNN-LSTM model have shown strengths in TFP, Luo et al. (2019).

In TFP, GRU models offer distinct advantages over LSTM models. One significant benefit is that GRU models improve the performance of LSTM models for TFP. Additionally, GRU networks have a simpler architecture than LSTM networks. Because of this simpler design, GRU networks can be easier to train and require fewer computational resources, allowing for faster training and predictions. Moreover, GRU networks are computationally less expensive than LSTM networks for TFP, making them a more efficient option for this task. These advantages make GRU models a promising approach for TFP. Several studies have evaluated the effectiveness of GRU models in predicting traffic flow. The Bi-GRU model has been introduced to capture temporal characteristics for TFP and has been compared with LSTM, Bi-LSTM, and GRU models using the same dataset, Wang et al. (2022). The prediction accuracy of the Bi-GRU model is relatively high, achieving more satisfactory traffic flow calculation results, Wen et al. (2023). Fusing fine-grained traffic flow statistical calculation methods significantly improves the robustness and generalization ability of the algorithm, Wen et al. (2023). Additionally, the GRU model is utilized to build a new DL neural network model to process the original data directly and predict traffic flow in various conditions, such as sunny days, cloudy days, rainy days, and evenings, Wen et al. (2023). The effectiveness of the GRU model is evaluated in comparison with other models for traffic prediction in urban areas, and it has been found that DL methods, such as RNNs, significantly enhance congestion prediction, Jang & Chen (2024) and Abdullah et al. (2023). The Bi-GRU model has slightly better prediction performance for overall traffic flow than the Bi-LSTM model, and the prediction performance of LSTM and GRU are comparable, but both are worse than Bi-GRU and Bi-LSTM, Wang et al. (2022). DL models outperform traditional ML models in terms of traffic volume prediction accuracy and the congestion level forecast, Abdullah et al. (2023).

Research has shown that the GRU model is a successful method for TFP, surpassing other models in efficiency and accuracy. Researchers have optimized this model to improve its performance, specifically regarding road network direction and DL implementation, Wen et al. (2023). In an urban TFP study, the GRU model was found to have higher accuracy than the LSTM model, Jang & Chen (2024). Additionally, a combination of the GRU model and backpropagation neural network was used to build a TFP model with US Twin Cities traffic data, which showed promising results, Zhang & Li (2021).

Hybrid deep learning models are a promising approach for TFP and have been proposed to solve road congestion challenges. These models provide information and decision support for solving congestion issues and promote cities' sustainable development, Zhou et al. (2022b). The proposed TFP model is based on a hybrid deep learning approach that utilizes DL models such as deep belief network (DBN), stacked autoencoder model, LSTM neural network, and path-based deep learning framework, Zhou et al. (2022b) and Rui et al. (2021). Nonparametric methods for TFP are shifting from ANNs to DL methods, as they can overcome the vanishing gradient problem of RNN and produce better traffic speed prediction on a city-wide scale Rui et al. (2021). Dynamic graphs provide a long-term TFP method based on hybrid deep learning models, Rui et al. (2021). However, DL methods have limitations, such as dependence on training data size and long training times, Rui et al. (2021). Various DL models have been proposed to predict traffic flow. These models have different architectures and employ various techniques to capture spatiotemporal dependencies and predict traffic flow. The proposed hybrid model (ELM-IBF) performs better in multistep-ahead traffic flow forecasting, leveraging an improved ensemble learning method and DBNs, Rui et al. (2021). Hybrid DL models are more suitable for dealing with complex and dynamic time series data and have been widely applied in TFP research, Zeng et al. (2023).

Hybrid deep learning models offer numerous advantages over traditional methods for TFP. For example, they can incorporate external factors such as weather and social event data to influence traffic flow, leading to more accurate predictions, Shi et al. (2019). Traditional shallow models, on the other hand, provide less accuracy in traffic prediction than deep neural network (DNN) models, Shi et al. (2019). Hybrid deep learning models can capture dependencies in multiple dimensions, allowing for a more comprehensive analysis of traffic patterns, Shi et al. (2019). Furthermore, hybrid deep learning models have been experimentally shown to outperform simple DNN models in terms of prediction performance and accuracy, Shi et al. (2019) and Zeng et al. (2023). Hybrid models that combine graph convolution and gated recurrent unit (GRU) neural networks have a distinctive advantage in spatiotemporal prediction, significantly improving over traditional methods that cannot

simultaneously capture spatial and temporal features, Zeng et al. (2023). In addition, hybrid deep learning models such as LSTM-I have achieved high accuracy in attribution and prediction, even with 80% of the data missing, Zeng et al. (2023). Hybrid deep learning models, such as fusion models, can perform data imputations and TFP, making them valuable tools for traffic management and planning purposes, Zeng et al. (2023). The major advantages of using hybrid deep learning models over traditional methods for TFP are clear in terms of accuracy, efficiency, and the ability to incorporate multiple factors. Applying hybrid deep learning models in TFP has excellent potential but faces various challenges and limitations. Missing data is one of the significant challenges in traffic flow forecasting, and existing studies often need to pay more attention to the relationship between data imputation and external factors, Zeng et al. (2023). The proposed hybrid models incorporate multiple factors for predicting traffic flow in data loss scenarios to address this gap and use Predictive mean matching (PMM) and k-nearest neighbors (KNN) for data imputation, Zeng et al. (2023). However, not considering multiple factors in TFP limits hybrid deep learning models. Including factors with low correlation can increase prediction error Zeng et al. (2023). Additionally, the stability and robustness of the models need to be examined, especially with limited data, Rui et al. (2021). The raw traffic flow data may also contain little noise or abnormal data, which can impact the accuracy of TFP. Therefore, missing and erroneous records must be appropriately remedied using the temporally adjacent records, Rui et al. (2021) and Zeng et al. (2023). Multiple factors can influence traffic flow, and ignoring their impact can lead to suboptimal results. Hybrid models often need to remember the impact of various factors on traffic flow, Zeng et al. (2023). However, considering whether it is a working day, it was shown to improve the predictive performance of the proposed models, Zeng et al. (2023). In terms of model selection, the experimentally demonstrated K-BiLSTM model was shown to be more accurate than other models in predicting traffic flow, Zeng et al. (2023). While hybrid deep learning models show promise in TFP, careful consideration of multiple factors, data imputation techniques, and model selection are crucial for achieving accurate predictions in real-world scenarios.

This study presents some recurrent neural network models, such as Long Short Term Memory, Gated Recurrent Unit and Hybrid LSTM-GRU. First, we based on fundamental deep learning models and the theoretical structures of LSTM, LSTMW GRU and Hybrid LSTM-GRU. Second, we interpreted and analyzed these models. Computational complexity and performance results are shown for time series data and natural language processing, and their results are compared. Deep Learning Models: LSTM, LSTM with Wadam, GRU and Hybrid LSTM-GRU in time series data are discussed, and their forecast results are compared. Artificial Neural Networks are algorithms consisting of a very large number of neurons that are internally connected with each other and function analogically to neurons in the human brain. Deep learning is a new area that aims to implement these techniques using structures such as artificial neural networks, convolutional neural networks, and recurrent neural networks, which have not been sufficiently successful before due to the incapability of both computing power speed and large amounts of training sets. In a more accurate definition, deep learning is a variety of artificial intelligence that involves a series of digital learning algorithms that are used to mimic the way humans gain some excess of learning.

LSTM exhibits a powerful characteristic: the capacity to retain or remove temporal information from each cell. It captures long-range temporal dependencies by utilizing the architecture of memory cells and several gate components. Each gate operates as a filter, controlling diverse aspects such as how much of the new input information should be considered for adding to the current cell information, removing current cell information with respect to the new input information, and dimensionally reducing the current cell to produce new output information. Thus, the properties of the LSTM gates enable the cell state to connect alongside and propagate to capture important information from long-range temporal patterns with respect to the depth of superposed LSTM architectural levels. Although LSTM succeeds in capturing important structures of the temporal dependencies via the architecture of memory cells and multiple gating components, it requires much additional parameter estimation to develop the requested complex features. Hence, this increases the training time complexity and the requirement for increased capacity. To address these issues, GRU was developed, exhibiting nearly the same performance as LSTM by utilizing fewer components.

Therefore, it is essential to carefully consider the amount and quality of data used when applying DL models for TFP purposes. Additionally, it is important to choose the effective models and to consider effective evaluation metrics.

4 CONTRIBUTIONS

In this section, we outline our contributions, most notably the introduction of a novel evaluation metric referred to as "gain".

4.1 CONTRIBUTION 1 : BALANCING COEFFICIENTS

We propose balancing coefficients to ensure that our model does not fall into overlearning. An equation to normalize the errors obtained on the train and test data sets is used to adjust the coefficients. For example, if we consider that $\text{RMSE}_{\text{test}}$ et $\text{RMSE}_{\text{train}}$, MAE_{test} et $\text{MAE}_{\text{train}}$ represent the errors, respectively, RMSE on the test and train sets and, MAE on the test and train sets, the balancing coefficients will be defined as follows:

$$\alpha = \frac{\text{RMSE}_{\text{test}}}{\text{RMSE}_{\text{train}}+\text{RMSE}_{\text{test}}}; \quad \beta = \frac{\text{MAE}_{\text{test}}}{\text{MAE}_{\text{train}}+\text{MAE}_{\text{test}}}; \quad \gamma = \frac{R^2_{\text{test}}}{R^2_{\text{train}}+R^2_{\text{test}}}.$$

α: This coefficient depends on the error on the test set $\text{RMSE}_{\text{test}}$. The larger RMSE error on the test set relative to that on the train, the higher the coefficient α, which means it gives more weight to the error coming from the test set.

β: This coefficient depends on the error on the test set MAE_{test}. If the MAE error on the train set is larger, the model could suffer from underlearning, and it would be important to increase this coefficient to balance the error on the train set.

γ: This coefficient depends on the stability of the test set R^2_{test}, in order to maintain a certain stability and not depend solely on absolute errors.

The proposed balancing coefficients serve to align training and testing outcomes, thereby reinforcing the reliability of the predictive model. When the error metrics and the variance R^2 values obtained from training and testing are closely matched, the coefficients converge toward 0.5, whereas greater discrepancies between these measures drive the coefficients toward 0. This formulation can mitigate the risk of overfitting. These balancing coefficients provide a foundation for the computation of the gain equation.

4.2 CONTRIBUTION 2 : NEW GAIN EQUATION

We propose a new gain equation that presents a weighted combination of aspects of RMSE and MAE, with a component that represents the explanation of variance (inspired by R-Squared R^2).

$$\text{Gain} = \frac{1}{2}\left[1 - (\alpha \cdot \text{RMSE} + \beta \cdot \text{MAE})\right] + \gamma \cdot R^2$$

Where, α, β, and γ are balancing coefficients.

The balancing coefficients are calculated from the performances on the test and train datasets and the errors obtained on the train and test datasets. The balancing coefficients α, β, and γ are proposed to assure the stability of models in a hybrid Gain function. These coefficients will be derived as a function of the test and train evaluation metrics in order to find an optimal balance. Regression learning methods are conventionally assessed through error metrics, such as MAE and RMSE, in conjunction with the coefficient of variance R^2. Building upon these measures, we introduce the gain equation, which jointly accounts for both error and variance. A gain value approaching 1 reflects a model of higher predictive performance.

The proposed Gain equation combines these different metrics to provide a balanced assessment of the model's performance. It allows errors to be integrated while taking into account the explanatory capacity of the model. The balancing coefficients α, β, and γ allow a different weight to be given to each metric depending on the observed performance. The balancing coefficients α, β, and γ are calculated to adjust the magnitude of the observed errors on the train and test datasets. The importance of the Gain function and the equilibrium coefficients lies in their ability to provide a more comprehensive and nuanced assessment of a model's performance. By taking into account both absolute errors (RMSE and MAE) and accuracy square (R^2), this approach provides a better balance of performance between different data sets. It also helps to identify weaknesses in a model and to adjust

hyperparameters to improve predictive accuracy and generalization on unknown data. It also helps to carry out a comparative study between models and analyze their performance. By integrating these elements, more robust and reliable models can be developed for practical applications.

4.3 CONTRIBUTION 3 : EXPERIMENTATION

Model	Train %			Validation %			Test %			Gain
	MAE	RMSE	R^2	MAE	RMSE	R^2	MAE	RMSE	R^2	Accuracy %
LSTM	4.09	6.04	95.17	4.88	7.99	91.51	5.00	8.10	91.33	91.03
LSTMW	3.24	4.80	96.95	4.09	6.88	93.69	4.09	7.09	93.35	92.53
GRU	4.13	5.84	95.48	4.86	7.26	92.99	4.84	7.31	92.93	92.49
Hybrid Model	3.61	5.59	95.86	4.27	6.81	93.82	4.37	6.93	93.64	**93.15**

Table 1: Comparison of evaluation metrics across training, validation, and test sets.

As reported in Table 1, the optimized LSTM model attains the highest coefficient of determination (R^2) on the training dataset, outperforming the alternative approaches. Conversely, the hybrid model demonstrates superior generalization, achieving the highest R^2 value on the test dataset. Furthermore, the proposed Gain Accuracy metric offers a rigorous and unambiguous criterion for model selection, ensuring both stability and predictive reliability.

5 DEEP LEARNING METHODOLOGY IN TRAFFIC FLOW PREDICTION

5.1 DATASET ANALYSIS

In our experimental tests, we use real traffic flow prediction datasets cases adopted from Kaggle.com. Data analysis involves preprocessing, feature selection, and data quality assessment to ensure the reliability and validity of training input data using deep learning models. This data provides information related to various factors that can influence traffic volumes. This data is crucial for conducting analysis and building predictive models to understand and forecast traffic patterns, which are influenced by many factors such as weather, environmental conditions, and time. The given dataset consists of 33750 rows and 15 columns. Each row represents a specific time and contains information related to conditional route, temperature, timing, weather, traffic volume, and other information. Some characteristics present general information such as timestamps, indicating the date, time of observation, and weather and holidays. Other characteristics provide details such as air pollution index, humidity, wind speed and direction, visibility, dew point, temperature, rainfall and snowfall amounts, cloud cover, and various descriptions of weather conditions. The dataset also includes the volume of traffic at each time point. This comprehensive dataset allows for exploring and analyzing the relationship between weather and timestamp factors and traffic volume.

The correlation of the dataset presented in Table 2 shows how different factors are related to the target "traffic volume". The strongest positive correlation is found with the "hours" feature, suggesting that certain times of the day have higher traffic volume. Other factors like temperature, clouds, and the day of the week also show some correlation, although it could be more robust. On the other hand, factors like rain, visibility, dew point, snow, and air pollution have very weak or almost no correlation with traffic volume. These results indicate that the time of day significantly impacts traffic volume, while other factors have a lesser influence.

In this section, we present the experimental results of our tests using deep learning models for traffic flow prediction. To evaluate and validate the models, we divided the dataset into three sets: a training set comprising 70% of the data, a validation set containing 20%, and a test set comprising 10%. This splitting process is commonly employed to guarantee separate datasets for training, fine-tuning hyperparameters, and assessing the model's performance on unseen data. The training set trains the models, allowing them to learn patterns and relationships within the data. The validation set is crucial for fine-tuning hyperparameters and evaluating the model's performance during development. The test set is an independent dataset to assess the model's generalization and ability to make predictions on new, unseen data. The objective is to balance having enough data for training and validation while ensuring a sufficient amount is set aside for unbiased testing and evaluation.

Table 2: Correlation of features with traffic volume

Feature	hours	temperature	clouds_all	is_holiday	day	humidity
Correlation	0.348867	0.127871	0.037922	0.037698	0.037264	0.016455
Feature	wind_speed	wind_direction	rain_p_h	visibility_in_miles	dew_point	snow_p_h
Correlation	0.016329	0.014278	0.005642	0.001348	0.001348	0.001197
Feature	air_pollution_index	year	month	weather_type	weather_description	Weekend
Correlation	-0.003751	-0.006714	-0.015371	-0.039530	-0.053461	-0.221928

This research compared the performance of LSTM, GRU, LSTM with Wadam optimizer, and Hybrid GRU-LSTM Models in the traffic flow prediction application. However, existing studies have reported that GRU-based models outperform LSTM-based models in computation time. GRU-based models consistently outperformed other models in terms of accuracy values in all but one task. This study analyzed and compared four RNN-based models.

5.2 Classical evaluation metrics

These metrics provide an overall view of prediction accuracy and help to understand how a model performs against real data.

The RMSE is a measure of the difference between predicted and observed values, taking into account the square root of the mean square error. It is more sensitive to large errors.

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

Where :

- n is the total number of observations (or examples).
- y_i represents the observed (real) values.
- $\hat{y}_i$ represents the values predicted by the model.
- $(y_i - \hat{y}_i)^2$ is the squared error for each observation.

The RMSE measures the difference between predicted and observed values by taking the square root of the mean squared error. This metric is particularly sensitive to large errors, making it a valuable indicator for applications where large errors are costly. The equation indicates that the lower the RMSE, the closer the predictions are to the true values. However, its sensitivity to large errors can also mask the overall performance of a model if it has a few significant errors.

The MAE is the mean of the absolute errors, measuring accuracy in terms of mean error regardless of direction (positive or negative).

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

The mean absolute error is a more robust indicator, as it does not take into account the direction of the errors. The formula indicates that the MAE provides a simple and direct measure of the mean error, making it easier to interpret the results.

The coefficient of determination R^2 evaluates the proportion of the variance of the dependent values explained by the model, thus giving an idea of the explanatory capacity of the model. Its formula shows that an R^2 close to 1 indicates a good fit, while an R^2 close to 0 suggests that the model fails to explain the variance in the data.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

5.3 DEEPLEARNING MODELS

5.3.1 LSTM MODEL

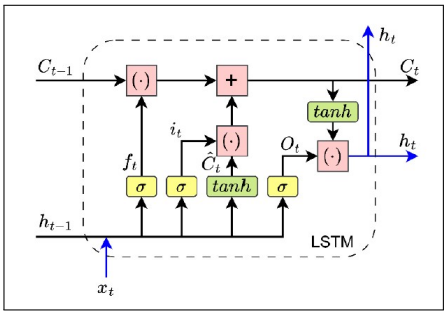

Figure 1: Structure of the Long Short-Term Memory (LSTM) architecture as described in Hussain et al. (2023).

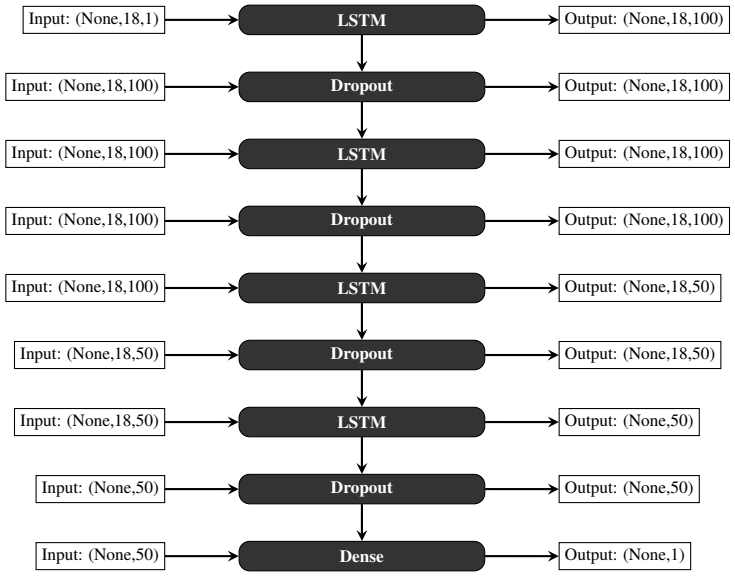

Figure 2: Architecture of the LSTM model.

LSTM introduced a new unit called a memory cell, which allows the network to store and access information over an extended period. The memory cell of an LSTM consists of several gates: an input gate, an output gate, and a forget gate. These gates regulate the flow of information within the memory cell, thereby controlling which information to retain and which to forget. This allows LSTMs to memorize important information over long sequences and ignore less relevant elements. h_t is the usual hidden state of RNNs, but in LSTM networks, we add a second state called C_t. Here, h_t represents the short-term memory of the neuron, and C_t represents the long-term memory.

This model is a deep learning model based on Recurrent Neural Network designed for sequence-to-one predictions, commonly used in tasks such as time series forecasting. The structure of this model is presented in Figure 2. The model consists of four LSTM layers, each followed by a dropout layer with a 20% dropout rate to prevent overfitting. The first three LSTM layers return sequences, allowing the model to capture long-term dependencies, while the fourth LSTM layer returns a single vector summarizing the sequence. The final dense layer outputs a single continuous value for regression. The model has 171,651 trainable parameters and uses the Adam optimizer with mean squared error as the loss function. This architecture is tailored to handle temporal dependencies in data and to generalize well, thanks to its regularization through dropout layers. After compilation with the

Adam optimizer and the RMSE loss function, the model is trained over 32 epochs with performance monitoring via callback.

5.3.2 LSTM WITH OPTIMIZATION MODEL

This model is an RNN based on the LSTM structure for regression. A custom function of the metric R^2 is defined to calculate the coefficient of determination R^2:

$$R^2 = 1 - \frac{SS_{\text{res}}}{SS_{\text{tot}}}$$

Where SS_{res} is the sum of the squares of the residuals and SS_{tot} is the total sum of the squares. A class WAdam, which inherits the Keras Adam optimizer, is defined to include a weight decay term. The optimizer adopted is expressed as :

$$\text{grad} += \text{weight\_decay} \times \text{var}$$

Where grad is the gradient and var is the variable to be updated.

The model is trained on the training set for 50 epochs with a batch size of 32. Predictions are made on the training, validation, and test sets, followed by the calculation of evaluation metrics : MAE, RMSE and R^2 score. These metrics are normalized with respect to the amplitude of the data to better assess performance. Figure 5 shows the training and validation results for the loss and R^2 of the training and validation curves during 50 epochs. This variant of the LSTM model is designed to solve regression problems, taking into account data sequences. The structure of the model, coupled with a custom optimizer and regularization techniques, enables robust performance on complex data sets. This model is an RNN using the LSTM structure to perform regression tasks on sequential data. It begins with data preparation, where a dataset is divided into training, validation, and test sets and normalized to ensure that the input values are centered around zero and have unit variance. This is essential to improve model convergence during training. The input data is remodeled into a 3D tensor, conforming to the requirements of the LSTM layers, which take into account not only the characteristics of the data but also its temporal order.

The model architecture comprises several LSTM layers, allowing temporal dependencies to be captured at different scales. Each LSTM layer is followed by a dropout layer to avoid overlearning by reducing the complexity of the model. The final layer is a dense layer that generates a single prediction, representing the output of the regression. To optimize learning, a custom optimizer, WAdam, is used, which is a variant of the Adam optimizer incorporating weight regularization to improve the generalization of the model.

5.3.3 GRU MODEL

The model is designed to make predictions on sequential data using a GRU-based recurrent neural network architecture.

The model architecture comprises three GRU layers. The first layer uses 200 units with a sigmoid activation function, allowing it to learn complex relationships in the data. The second layer with 100 units and a hyperbolic tangent activation further refines the learned representations, while the third layer, consisting of 50 units with a ReLU activation, produces an output that is then passed to the output layer.

The model is trained for 32 epochs, and performance is evaluated at each epoch using a validation set. Metrics such as mean absolute error, mean square error, and coefficient of determination R^2 are calculated to provide an indication of the accuracy of predictions over the validation and test sets.

The model is capable of predicting values based on temporal input data, providing a robust solution for sequence regression tasks. The structure and hyperparameters chosen favor good generalization, as demonstrated by the diversity of performance metrics obtained on the training, validation, and test datasets.

The structure of the GRU model is designed to capture complex temporal dependencies in sequential data. Data normalization and reshaping ensure that the model receives appropriate inputs. The

use of multiple GRU layers allows the model to learn hierarchical representations, with each layer learning features at a different level of complexity. The choice of MSE loss function is suitable for regression tasks, and the Adam optimizer is effective for updating weights in deep networks. Performance monitoring on validation sets enables the generalization of the model to be assessed and hyperparameters to be adjusted.

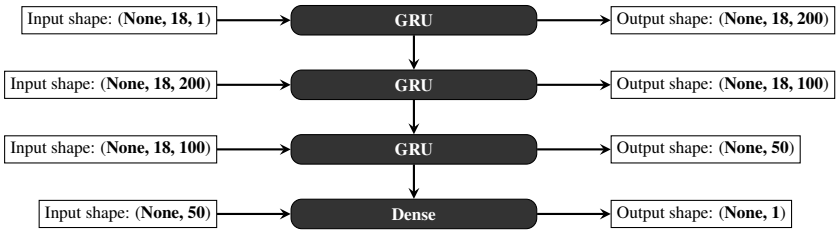

Figure 3: Architecture of the GRU Model

5.3.4 HYBRID MODEL

This model is a hybrid RNN-based model that combines GRU and LSTM layers for continuous prediction. It is designed to process temporal or sequential data and to make continuous regression predictions. The architecture starts with a data input of the form ($timesteps$,1) where $timesteps$ represents the number of time steps. The first layer is a GRU with 128 units, using the tanh activation function and returning complete sequences. This is followed by a second GRU layer with 64 units, also using $tanh$ and returning complete sequences. Next, a Batch Normalization layer is added to stabilize and speed up training. A dense layer with 64 units and $tanh$ activation follows this normalization, before a Flatten layer that flattens the output to a one-dimensional vector. The data is then restructured by a Reshape layer to be ready for an LSTM layer with 128 units, also using $tanh$ and returning the complete sequences. A final dense layer with one output unit is used for continuous regression prediction. The model uses the RMSE loss function and the Adam optimizer, with an initial learning rate fixed at 0.01, adjusted dynamically during training. In addition to the loss function, the mean absolute error (MAE) is used as an evaluation metric. To improve learning, a learning rate adjustment is implemented via the Learning Rate Scheduler, and early stopping is activated to avoid overlearning if validation performance does not improve after five consecutive epochs.

The model starts with two GRU layers, which capture the temporal dependencies of the data with 128 and 64 units, respectively. Each layer uses a tanh activation, which allows the model to better handle non-linear relationships in the data. These layers return complete sequences to allow the information to be passed on to subsequent layers. Batch normalization is applied to stabilize the training, followed by a dense layer of 64 units. Next, a flattening operation is performed to convert the output to a vector, and the vector is resized for processing by the next LSTM layer. This LSTM layer also consists of 128 units with tanh activation. Finally, a dense output layer with 1 unit predicts a continuous value, which is suitable for a regression task. The model is compiled with the Adam optimizer, which dynamically adjusts the learning rate, and the RMSE is used as the loss function. The model is trained with dynamic learning and an early stopping strategy to avoid overfitting.

The model is made up of several distinct blocks. It starts with an input in the form of a three-dimensional tensor of size (n samples, timesteps, features), where timesteps is the number of time steps and features is the dimension of each input vector. The first layer is a 128-unit GRU, using the tanh activation function and returning the full sequence of outputs (with return sequences=True). The output of this layer is in the form of (timesteps, 128). The second GRU layer, with 64 units, works in a similar way, with an input of (timesteps, 128) and an output of (timesteps, 64). To stabilize and speed up training, a batch normalization layer is added, followed by a dense layer with 64 units, activated by tanh and regularized by L2 with a regularization coefficient of 0.01. The Flatten and Reshape layers then transform the data, the latter restructuring the tensor into (1,-1) to prepare the final sequence. The LSTM layer, with 128 units and also using tanh, returns the complete sequence of outputs, resulting in a shape tensor (timesteps, 128). Finally, a dense output layer produces a single unit, suitable for a regression task, with no activation function.

The model is compiled with the MSE loss function, which is commonly used for regression tasks. MSE is defined by the formula :

$$\text{MSE}(y, \hat{y}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Where y_i is the true value and $\hat{y}_i$ is the prediction value. The optimizer used is Adam, with an initial learning rate of 0.01, which combines the advantages of learning rate adaptation and stochastic gradient descent (SGD) regularization. L2 regularization is applied to certain dense layers to prevent overlearning by adding a penalty to the loss term proportional to the sum of the squared weights. Batch normalization layers also help to normalize activations, reducing the risk of gradients exploding or disappearing.

In terms of callback strategy, early stopping is implemented to stop training if performance on the validation set does not improve after five consecutive epochs, thus avoiding overtraining. In addition, a learning rate scheduler is used to progressively reduce the learning rate (0.01 for the first 10 epochs, 0.001 thereafter, and 0.0001 after the 20th epoch). Training results are evaluated on the validation and test sets, using standard regression metrics such as Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and Coefficient of Determination (R^2 Score), which measures the proportion of variance explained by the model. Additionally, we calculate the gain value to evaluate the results.

The input data is normalized using a standard scaler before training, and the model predicts a continuous value corresponding to the regression task. This model implements an advanced architecture with deep GRU and LSTM layers, combined with regularization techniques (L2, dropout, batch normalization) and optimized training through dynamic scheduling of the learning rate.

The model is made up of several dense recurrent layers. Firstly, the first GRU layer has 128 units and uses the tanh activation function, returning a complete sequence, meaning that each temporal unit of the input produces an output. The second GRU layer has 64 units and also uses tanh, in turn returning a complete sequence. These GRU layers are essential for modeling long-term dependencies in time series, efficiently capturing the information present in sequences of data.

Next, a batch normalization layer is added to normalize activations, stabilize training, make the model more robust, and accelerate convergence by intervening after the GRU layers. A dense layer with 64 units, activated by tanh and incorporating L2 regularization, further refines the representations learned by the recurrent layers by penalizing excessive weights to reduce the risk of overlearning. The next layer, Flatten, flattens the dimensions of the tensor, transforming the data into a one-dimensional vector, thus facilitating the transition to a classical dense layer. The Reshape layer then reorganizes the tensor to prepare the data for the next recurrent layer.

The LSTM layer, with 128 units, is then integrated to capture additional information. LSTMs are particularly effective for tasks requiring long-term memory, complementing GRUs for more powerful sequence modeling. The final layer, a dense single-unit layer, produces the final prediction. As this is a regression task, this layer has no activation function, allowing a continuous output to be generated.

For optimization, the model uses the Adam optimizer with an initial learning rate of 0.01. Adam is an efficient optimizer that combines the advantages of gradient descent with momentum and regularization. The learning rate is dynamically adjusted at each epoch using a learning rate scheduler: it is 0.01 from epoch 0 to 10, 0.001 from epoch 10 to 20, and 0.0001 after epoch 20.

To avoid overlearning, several regularization techniques are used. L2 regularization is applied to the dense layer to constrain the weights and prevent them from reaching excessive values, while batch normalization helps to reduce the risk of overtraining.

The model is trained for 40 epochs with a batch size of 64, and two callbacks are used to optimize this process. Early stopping stops training when the performance on the validation set stops improving after five consecutive epochs, thus preventing overtraining. At the same time, the learning rate scheduler adjusts the learning rate over time to facilitate model convergence.

The model is evaluated on the training, validation, and test sets, using standard metrics for regression tasks, such as Mean Absolute Error (MAE), which measures the average error between predicted

and actual values, Root Mean Squared Error (RMSE), which penalizes larger errors, and Coefficient of Determination (R^2 Score), which indicates the proportion of data variance explained by the model.

This model combines the strengths of the GRU and LSTM architectures for modeling complex sequences while incorporating advanced regularization and optimization techniques. The combination of recurrent and dense layers effectively captures temporal relationships in the data while regulating the model to avoid overtraining.

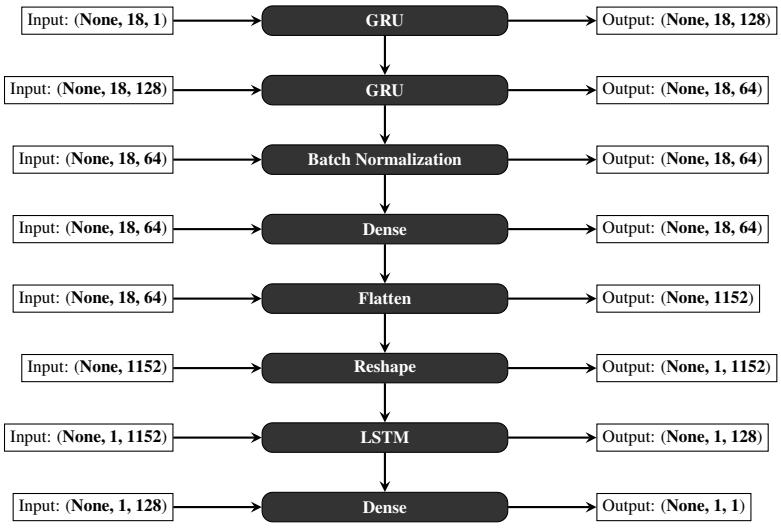

Figure 4: Architecture of the Proposed Hybrid LSTM-GRU Model

6 EXPERIMENTAION RESULTS

6.1 LSTM MODEL

The obtained results are visualized in Figure 5. The obtained prediction values are evaluated on the test set. We interpret the results presented in table 1 obtained, focusing on each metric and the value of the gain. The MAE represents the mean absolute error of the predictions on the test set. An MAE of 05.00% means that, on average, the model's predictions are within 0.0500 units of the true value. This indicates good model accuracy, as a lower MAE indicates a better fit. The RMSE measures the mean square root error between predicted and real values. With an RMSE of 08.10%, this means that, on average, the prediction errors are slightly larger, which is typical since the RMSE penalizes larger errors more heavily. However, this value remains reasonable, suggesting that the model is reliable. The coefficient of determination R^2 indicates the proportion of the variance in the data that is explained by the model. An R^2 means that 91.33% of the variance in the test data is explained by the model. This indicates good explanatory power and robust performance.

The MAE on the training set is 04.09%, which is slightly lower than that on the test set. This indicates that the model fits the training data well. With an RMSE of 0.0604 on the training set, the model shows a slightly smaller error on this data compared to the test set, which is expected. This may also indicate a good fit. An R^2 on the training set indicates that 95.17% of the variance is explained by the model, which is excellent and suggests that the model has learned to represent the training data well. The gain value of 91.03% is relatively high, indicating that the model performs well overall on the training and test sets. This value is the result of a combination of the errors and the R^2, suggesting that the model maintains good performance and generalization, even with variations in the data. The results show that the model is well fitted, with relatively low errors and high R^2 on both the training and test sets. This indicates a good ability to generalize, which is reinforced by the value of the gain.

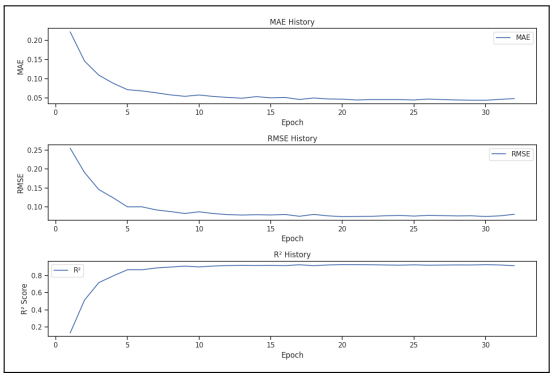

Figure 5: Evaluation Metrics History for the LSTM Model

6.2 LSTM WITH OPTIMIZATION MODEL

The model is compiled with the RMSE loss function and the R^2 metric, allowing an accurate assessment of its performance during training with the gain value. After training, predictions are made on the training, validation, and test datasets, with the evaluation metrics calculated to assess the model's performance. The training and validation results are visualized in Figure 6 to analyze the evolution of loss and performance over time.

Also in table 1, the results show that the model is well trained, with a strong ability to predict both on the training, the validation, and the test datasets. Although performance on the validation and test sets is slightly lower than on the training set, it is still very acceptable, indicating good generalization. The low loss values and high R^2 coefficients, as well as the gain value, reinforce the reliability of the model for practical applications.

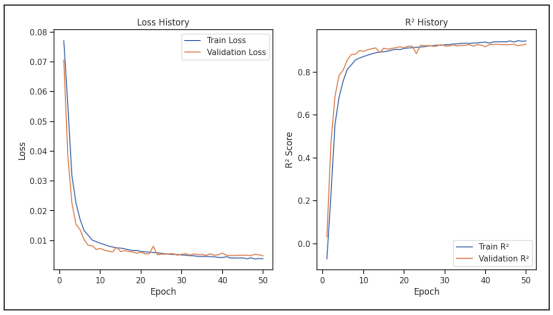

Figure 6: Historical Evaluation Metrics for the Optimized LSTM Model

6.3 GRU MODEL

The implementation of this model gives an average MAE error on the validation set is about 4.87%, on the test set about 4.84%, and on the training set, the error is lower at 4.13%.

A small difference between the training, validation, and test sets indicates that the model is well generalized. Performance is relatively similar across the three sets, indicating that there is no significant overfitting.

The RMSE is a measure that penalizes larger errors more than the MAE because the errors are squared. On the validation and test datasets, the RMSE is around 7.26% and 7.31%, respectively, while on the training set it is lower at 5.84%. These results also show that the model is well balanced and makes no major errors.

The fact that the RMSE is slightly higher than the MAE is normal, as it reflects the greater weighting of large errors in the calculation of the RMSE.

The R^2 indicates the proportion of the variance in the data explained by the model. A score close to 1 means that the model explains the variations in the target data well. On the validation set, the model explains 93% of the variance, on the test set, it explains 92.9%, and on the training set, it reaches 95.5%. These values suggest that the model captures the dynamics of the data well and that there is no under-learning or over-learning problem.

The obtained results represent an MAE of 4.843% compared to the actual values on the predictions, which indicates good overall performance.

The RMSE of 7.314% on the test set is a measure that evaluates the error, like the MAE, but it is more sensitive to large errors. This evaluation value shows that the model has no major errors and that the predictions are generally accurate.

The R^2 score indicates the proportion of the variance in the target values that is explained by the model. A score of 92.93% on the test set of the variance in the data is acquired by the model. This shows that the model fits the test data well, with only 7.07% of the variance not acquired. Over the training set, the MAE is slightly lower than the test MAE (4.131% vs. 4.843%), which is a sign of good fit and proves the stability of this model. This indicates that the model makes slightly fewer errors on average on the training data than on the test data, which is expected.

The RMSE on the training set is 5.84%, lower than that on the test set. This suggests that the model is slightly more accurate when evaluated on the training data, which is generally the case as the model has been optimized for this data.

The R^2 score on the training set is 95.48% of the variance in the training data. This score is slightly higher than that of the test set, which is normal, but it is still close to the performance observed on the test data, which is a good sign of generalization.

The MAE and RMSE are slightly lower on the train set than on the test set, which is normal in most machine learning models. The difference remains reasonable, indicating that the model has not overlearned too much on the training data. The R^2 scores are very similar for the train (95.48%) and test (92.93%) sets, which shows that the model does not have a major overlearning problem and that it generalizes well to the new data. Too large a gap between these two scores could indicate an overlearning problem, but here the gap is small, showing a good compromise between bias and variance.

The gain equation value of 92.49% is an overall indicator of the model's performance. It takes into account the errors (RMSE and MAE) and the explanatory capacity (R^2), weighted by equilibrium coefficients. A value of 92.49% is quite high and shows that the model is well balanced between its various metrics. This reflects a high-performance model that manages to explain the data well while maintaining a low margin of error.

These results show that the model performs well, with low errors and high accuracy, indicating a good explanation of the variance in the data. The relatively small difference between performance on the train and test sets shows that the model is well balanced and generalizes well without being overfitted to the training data. The Gain function highlights this robustness by providing a favorable weighted evaluation of the model's performance on several metrics.

In terms of model complexity, the GRU model has 705,755 parameters, of which 235,251 are trainable. It uses a GRU architecture with three GRU layers (200, 100, and 50 units) and a final dense layer for regression. This relatively complex architecture appears to be well suited to the data, as it achieves good results without showing signs of overfitting, despite a large number of parameters.

The absence of a large difference between the training, validation, and test performances indicates that the model has not overfitted and is capable of making robust predictions on never-before-seen data. Our GRU model shows excellent predictive performance and appears to generalize well without overfitting, with high accuracy close to that of LSTMW on the validation and test sets.

6.4 HYBRID MODEL

This hybrid model is a complex and well-balanced model that combines GRU and LSTM layers to capture both long-run and short-run dependencies in time series.

It incorporates regularization techniques such as batch normalization and uses learning rate scheduling to optimize training.

The training log shows an interesting evolution of the model over the first 29 epochs. From the first epoch, the model starts with a training loss of 0.1479 and an MAE of 0.2047, while validation shows a much lower loss at 0.0099 and an MAE of 0.0735. These initial results, together with a high learning rate of 0.01, indicate that the model generalizes well from the outset. Between epochs 2 and 5, the model oscillates slightly, but the loss and MAE remain around 0.01 for both training and validation, although the validation MAE is sometimes slightly higher than the training MAE. From epoch 6 onwards, a more noticeable improvement occurs, with the validation MAE dropping to 0.0637 and the validation loss continuing to decrease to 0.0073 by epoch 10.

At epoch 11, the learning rate decreases to 0.001, helping to stabilize training. Loss and MAE continue to fall, and the validation MAE reaches 0.0499, showing steady progress. By epoch 20, the validation shows a loss of 0.0049 and an MAE of 0.0443, signalling continued improvement in model accuracy. From epoch 21 onwards, with the learning rate reduced to 0.0001, the model achieves stability. The validation losses and MAE stabilize at around 0.0047 and 0.0428-0.0431, respectively, indicating that the model appears to have converged at this stage without much further improvement.

The model shows good progression with a continuous reduction in loss and MAE throughout training. Convergence is reached around epoch 20, with stable performance and a validation MAE around 0.0430-0.0431. The gradual adjustment of the learning rate, particularly after epoch 10, enabled the training to be refined effectively and improved the accuracy of the model.

The model is composed of several key elements. Firstly, it includes two GRU layers, with 128 and 64 units, respectively, which use sequences (with return sequences = True) to preserve the output at each time step. Secondly, batch normalization is applied after the GRU layers to stabilize the training. The model also incorporates a dense layer with 64 units, activated by a tanh function. The Flatten and Reshape layers transform the output before it is passed to the 128-unit LSTM layer, which processes the final sequence after transformation. A final dense layer produces a single output, indicating that the model is well suited to a regression task. With around 2.24 million parameters, the model shows good convergence and fits the task efficiently, with MAE validation remaining in a satisfactory range, while adjustments to the learning rate have helped to stabilize this convergence.

The model shows excellent fit on both training and validation data, with a high R^2 score across the board. The slight drop in validation and test scores compared to training suggests mild overfitting, but it's well-controlled. The learning rate adjustments are well-timed. The early larger learning rate allowed quick convergence, and the gradual reductions smoothed the training process. The sequential architecture of two GRU layers followed by LSTM, batch normalization, and dense layers seems effective for this task, balancing complexity with performance. Given the high performance, further tuning may not be necessary unless you aim to marginally improve generalization. You might explore techniques like dropout to prevent overfitting further or test the model with different initialization schemes or optimizers.

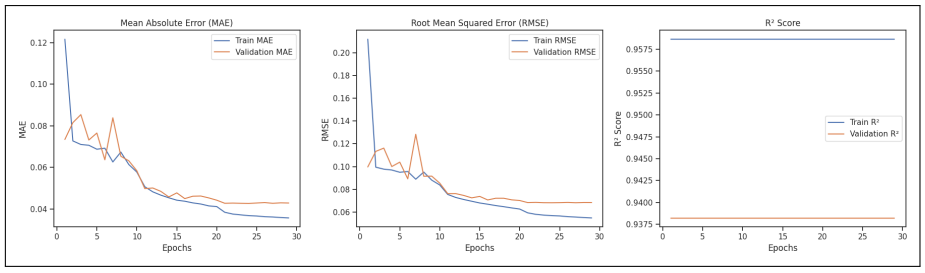

Figure 7: Variation of evaluation metrics for training and validation sets.

6.5 Interpretation and Analysis Results

The table 1 compares four models—LSTM, LSTMW (LSTM with optimization), GRU, and a hybrid model (combining LSTM and GRU)—based on evaluation metrics (MAE, RMSE, R^2, and gain

accuracy) across training, validation, and test datasets. The hybrid model consistently achieves the best performance on validation and test datasets, with the lowest MAE (4.27% on validation, 4.37% on test), the lowest RMSE (6.81% on validation, 6.93% on test), and the highest R^2 (93.82% on validation, 93.64% on test). It also records the highest overall accuracy at 93.15%, demonstrating superior generalization and predictive capabilities. LSTMW performs exceptionally well on the training set, achieving the lowest MAE (3.24%) and RMSE (4.80%), reflecting the effectiveness of optimization techniques. However, its slightly higher errors on validation and test datasets suggest limited adaptability and a potential overfitting tendency. GRU shows balanced performance across all datasets, providing competitive results and reliability. While the LSTM model delivers acceptable performance, it is outperformed by the other models, particularly in generalization, with higher errors and lower accuracy.

The hybrid model emerges as the most effective approach, leveraging the strengths of both LSTM and GRU to achieve robust generalization and accurate predictions, making it the best-suited model for the task.

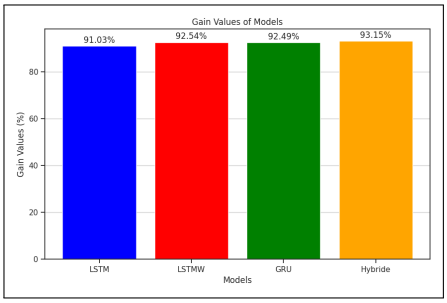

Figure 8: Comparison of models based on gain values.

The model is evaluated on the training, validation, and test sets, using standard metrics for regression tasks, such as Mean Absolute Error (MAE), which measures the average error between predicted and actual values, Root Mean Squared Error (RMSE), which penalizes larger errors, and Coefficient of Determination (R^2 Score), which indicates the proportion of data variance explained by the model.

This model combines the strengths of the GRU and LSTM architectures for modeling complex sequences while incorporating advanced regularization and optimization techniques. The combination of recurrent and dense layers effectively captures temporal relationships in the data while regulating the model to avoid overtraining.

7 Conclusion

This article investigates the problem of traffic flow prediction using deep learning models, which is essential to improve urban traffic management, planning routes, and reducing congestion. The study compares several models, including LSTM, GRU, optimized LSTM, and a proposed hybrid model that combines the strengths of these architectures.

The analysis shows that the traditional LSTM and GRU models effectively capture temporal dependencies in traffic data, with LSTM generally performing better for long-term sequences due to its robust handling of long-range dependencies. The optimized LSTM model further improves the prediction accuracy by fine-tuning the hyperparameters and implementing optimization techniques that improve the convergence and overall performance of the model.

However, the hybrid model outperforms all others. By integrating the best features of LSTM, GRU, and additional enhancements, it provides superior accuracy and generalization capabilities. This performance is demonstrated by lower prediction errors and better adaptation to complex traffic patterns.

In addition, the paper contributes a novel evaluation metric that combines RMSE, MAE, and R^2 with the introduction of an equilibration coefficient. This new metric provides a more comprehensive evaluation of prediction performance, balancing between error size, consistency, and model reliability.

Generally, the results show that the hybrid model not only achieves lower prediction errors but also adapts more effectively to traffic variations, demonstrating its superiority for traffic flow prediction tasks. The new evaluation metric reinforces these findings by providing a balanced and nuanced assessment of performance, demonstrating that the hybrid model achieves the best results across all metrics.

Openness to future areas of research is essential to improve methods for predicting traffic flow and other related areas. One potential avenue would be the integration of real-time data from IoT sensors and intelligent transport systems to improve the responsiveness and accuracy of deep learning models. In addition, the exploration of new neural network architectures, such as Transformers, which have shown good results in other sequential domains, could offer additional gains in long-term prediction. In addition, the use of model federation or distributed learning techniques would enable collaborative models to be developed without compromising data confidentiality. These avenues open up promising prospects for more robust prediction and better management of intelligent transport systems.

REFERENCES

Z. Abbas, A. Al-Shishtawy, S. Girdzijauskas, and V. Vlassov. Short-term traffic prediction using long short-term memory neural networks. In *2018 IEEE International Congress on Big Data (BigData Congress)*, pp. 57–65. IEEE, July 2018.

Sura Mahmood Abdullah, Muthusamy Periyasamy, Nafees Ahmed Kamaludeen, et al. Optimizing traffic flow in smart cities: Soft gru-based recurrent neural networks for enhanced congestion prediction using deep learning. *Sustainability*, 15(7):5949, 2023.

Yuguang Chen, Jintao Huang, Hongbin Xu, et al. Road traffic flow prediction based on dynamic spatiotemporal graph attention network. *Scientific reports*, 13(1):14729, 2023.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, et al. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

A. Essien, I. Petrounias, P. Sampaio, and S. Sampaio. A deep-learning model for urban traffic flow prediction with traffic events mined from twitter. *World Wide Web*, 24(4):1345–1368, 2021.

A. H. A. Hussain et al. Urban traffic flow estimation system based on gated recurrent unit deep learning methodology for internet of vehicles. *IEEE Access*, 11:58516–58531, 2023.

Hung-Chin Jang and Che-An Chen. Urban traffic flow prediction using lstm and gru. *Engineering Proceedings*, 55(1):86, 2024.

Y. Jia, J. Wu, and M. Xu. Traffic flow prediction with rainfall impact using a deep learning method. *Journal of Advanced Transportation*, 2017.

A. Khan, M. Fouda, Do M., D. T., A. Almaleh, and A. U. Rahman. Short-term traffic prediction using deep learning long short-term memory: Taxonomy, applications, challenges, and future trends. *IEEE Access*, 2023.

Mingyu Kim and Donghyun Lee. Why uncertainty in deep learning for traffic flow prediction is needed. *Sustainability*, 15(23):16204, 2023.

X. Luo, D. Li, Y. Yang, and S. Zhang. Spatiotemporal traffic flow prediction with knn and lstm. *Journal of Advanced Transportation*, 2019.

Noor Afiza Mat Razali, Nuraini Shamsaimon, Khairul Khalil Ishak, et al. Gap, techniques and evaluation: traffic flow prediction using machine learning and deep learning. *Journal of Big Data*, 8(1):1–25, 2021.

P. Redhu and K. Kumar. Short-term traffic flow prediction based on optimized deep learning neural network: Pso-bi-lstm. *Physica A: Statistical Mechanics and its Applications*, 625:129001, 2023.

C. Ren, C. Chai, C. Yin, H. Ji, X. Cheng, G. Gao, and H. Zhang. Short-term traffic flow prediction: A method of combined deep learnings. *Journal of Advanced Transportation*, pp. 1–15, 2021.

Yikang Rui, Wenqi Lu, Ziwei Yi, et al. A novel hybrid model for predicting traffic flow via improved ensemble learning combined with deep belief networks. *Mathematical Problems in Engineering*, 2021:1–16, 2021.

Sayed A. Sayed, Yasser Abdel-Hamid, and Hesham Ahmed Hefny. Artificial intelligence-based traffic flow prediction: a comprehensive review. *Journal of Electrical Systems and Information Technology*, 10(1):13, 2023.

Ruizhe Shi and Lijing Du. Multi-section traffic flow prediction based on mlr-lstm neural network. *Sensors*, 22(19):7517, 2022.

Yan Shi, Haoran Feng, Xiongfei Geng, et al. A survey of hybrid deep learning methods for traffic flow prediction. In *Proceedings of the 2019 3rd international conference on advances in image processing*, pp. 133–138, 2019.

A. Sroczyński and A. Czyżewski. Road traffic can be predicted by machine learning equally effectively as by complex microscopic models. *Scientific reports*, 13(1):14523, 2023.

S. Wang, C. Shao, J. Zhang, Y. Zheng, and M. Meng. Traffic flow prediction using bi-directional gated recurrent unit method. *Urban Informatics*, 1(1):16, 2022.

W. Wen, D. Xu, and Y. Xia. A novel traffic optimization method using gru-based deep neural network for the iov system. *PeerJ Computer Science*, 9:e1411, 2023.

Haitao Yuan and Guoliang Li. A survey of traffic prediction: from spatio-temporal data to intelligent transportation. *Data Science and Engineering*, 6:63–85, 2021.

Wenbao Zeng, Ketong Wang, Jianghua Zhou, et al. Traffic flow prediction based on hybrid deep learning models considering missing data and multiple factors. *Sustainability*, 15(14):11092, 2023.

Xijun Zhang and Jiwen Li. Traffic flow prediction based on gru-bp combined neural network. In *Journal of Physics: Conference Series*, pp. 012060. IOP Publishing, 2021.

Yulin Zhang, Ke Shang, Zhiwei Cui, et al. Research on traffic flow prediction at intersections based on dt-tcn-attention. *Sensors*, 23(15):6683, 2023.

Q. Zhou, N. Chen, and S. Lin. Fastnn: A deep learning approach for traffic flow prediction considering spatiotemporal features. *Sensors*, 22(18):6921, 2022a.

Shenghan Zhou, Chaofan Wei, Chaofei Song, et al. A hybrid deep learning model for short-term traffic flow prediction considering spatiotemporal features. *Sustainability*, 14(16):10039, 2022b.