Ohm Prakasanatham, 2026, 14:2
ISSN (Online): 2348-4098
ISSN (Print): 2395-4752

International Journal of Science, Engineering and Technology
An Open Access Journal

Route Analysis and Traffic Jam Prediction using Deep Learning Techniques: An Enhanced LSTM-GRU Comparative Study

Ohm Prakasanatham

Abstract- Traffic route analysis plays a critical role in daily urban mobility. This study proposes and evaluates deep learning models for predicting traffic jam status based on contextual weather and event data. Specifically, LSTM and GRU architectures are trained and tested on a dataset of weather conditions, air quality, road characteristics, and events. This study introduces a structured hyperparameter optimization framework and an interpretive analysis explaining observed performance gaps. Data preprocessing includes one-hot encoding, Min-Max normalization, and temporal sequence organization. Both models are evaluated using accuracy, precision, recall, F1-score, and ROC AUC. The LSTM model achieves 87.5% accuracy and 91.67% ROC AUC, significantly outperforming the GRU model (50% accuracy, 67% ROC AUC). The findings offer practical guidance for selecting appropriate recurrent architectures in traffic prediction systems. Traffic prediction, deep learning, LSTM, GRU, weather data, event data, traffic congestion, performance evaluation, hyperparameter optimization.

Keywords: Traffic Prediction, Deep Learning, LSTM, GRU, Weather Data, Traffic Congestion Prediction.

I. INTRODUCTION

The rapid expansion of urban infrastructure has made accurate traffic congestion prediction a critical challenge for city planners, commuters, and transportation authorities. Predicting traffic jams in advance enables better route planning, reduces fuel consumption, and improves emergency response times. As cities generate increasing volumes of sensor, weather, and event data, machine learning and deep learning techniques have emerged as powerful tools for modelling complex traffic dynamics [1].

Among deep learning approaches, recurrent neural networks (RNNs) - particularly Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) networks - have gained significant attention for their ability to capture temporal dependencies in sequential data [2], [3]. However, while numerous studies apply these architectures to traffic prediction, few provide a rigorous comparative evaluation accompanied by an explanation of performance differences, nor do they address the selection criteria between architectures for practitioners.

This paper addresses these gaps through three specific contributions: (1) a structured hyperparameter optimization framework that systematically searches across layer depth, hidden unit count, dropout rate, and learning rate; (2) a comprehensive multi-metric evaluation including accuracy, precision, recall, F1-score, and ROC AUC; and (3) an interpretive analysis explaining why LSTM consistently outperforms GRU on the given task, grounded in architectural differences and data characteristics.

II. LITERATURE REVIEW

Hochreiter and Schmidhuber [4] introduced LSTM as a solution to the vanishing gradient problem in standard RNNs, enabling the network to retain long-range temporal dependencies. Cho et al. [5] later proposed GRU as a computationally lighter alternative, consolidating the input and forget gates into a single update gate. Zhang et al. [6] applied LSTM-based models to citywide crowd flow

prediction, while Cui et al. [7] explored bidirectional LSTM for network-wide traffic speed prediction.

Li et al. [8] introduced the Diffusion Convolutional Recurrent Neural Network (DCRNN), combining graph convolutional networks with recurrent architectures to capture spatial dependencies in road networks. Recent literature has explored Transformer-based architectures; Xu et al. [9] proposed a spatial-temporal Transformer achieving state-of-the-art results, while Yu et al. [10] combined graph attention networks with temporal convolutions for joint spatial-temporal modelling.
Generally, LSTM outperforms GRU when longer temporal dependencies are present, while GRU performs comparably on shorter sequences with fewer parameters [11]. For event- and weather-driven traffic prediction where patterns span multiple time steps, LSTM's additional memory capacity through its separate cell state is theoretically advantageous - a hypothesis supported by the experimental results of the present study.

III. MATERIALS AND METHODS

Dataset Description

The dataset comprises 500 historical records of traffic conditions from an urban road network, spanning a 12- month period with hourly resolution. Each instance includes weather condition category, ambient temperature (deg C), precipitation level (mm), event type, road classification, visibility (km), wind speed (km/h), holiday indicator, and Air Quality Index (AQI). The target variable is binary (1 = traffic jam, 0 = no jam), with a class distribution of approximately 60% non-jam and 40% jam instances.

TABLE I: Attributes and Data Types

Sr.	Attribute	Variable Type	Scale
1	Weather	Categoric al / Ordinal	Nominal
2	Temperature (deg C)	Continuous	Ratio
3	Precipitation (mm)	Continuous	Ratio
4	Event Type	Categoric al / Nomina l	Nominal
5	Road Type	Categoric al / Nomina l	Nominal
6	Visibility (km)	Continuous	Ratio
7	Wind Speed (km/h)	Continuous	Ratio
8	Holiday	Binary	Nominal
9	AQI	Continuous	Ratio
10	Traffic Jam (Target)	Binary	Nominal

Data Preprocessing

The raw dataset underwent a four-stage preprocessing pipeline prior to model training, as summarised in Table II.

- Data Collection: Historical records on date, time, weather, temperature, precipitation, events, traffic status, road type, visibility, wind speed, holiday status, AQI, and city location were assembled from publicly available repositories [6].
- Data Cleaning: Missing values were imputed using forward-fill for temporal continuity; outliers beyond three standard deviations were clipped; duplicate records were removed to ensure dataset integrity [8].
- Feature Engineering: Categorical variables (Weather, Event, Road Type, Holiday) were one-hot encoded. Continuous features (Temperature, Precipitation, Visibility, Wind Speed, AQI) were normalized to [0,1] using Min-Max scaling [7].
- Data Sequencing: Observations were organized into fixed-length temporal sequences of T=10 time steps to preserve temporal ordering and enable the recurrent models to learn time-dependent patterns [6].

TABLE II: Data Preprocessing Steps

Step	Description
Data Collection	Historical weather, event and traffic data assembled.
Data Cleaning	Missing values imputed; outliers clipped; duplicates removed.
Feature Engineering	Categorical variables one-hot encoded; continuous features normalized.
Data Sequencing	Data organized into T=10 temporal sequences.

Pseudocode for Data Preprocessing:

```
X, y = SplitFeaturesAndTarget(data)
X_train, X_test, y_train, y_test =
    TrainTestSplit(X, y, test_size=0.2,
                   random_state=42)
scaler = MinMaxScaler()
X_train = scaler.fit_transform(X_train) X_test  =
scaler.transform(X_test)
X_seq     = CreateSequences(X_train, T=10)
```

Model Construction

LSTM Model

The LSTM architecture was selected for its ability to model long-range temporal dependencies through a gating mechanism comprising an input gate, a forget gate, and an output gate, together maintaining a separate cell state that enables selective memory retention across many time steps [4]. This architecture is particularly suited to traffic prediction, where conditions may be influenced by events that occurred several time steps prior, such as sustained precipitation preceding peak congestion.

```
def BuildLSTMModel(layers, units, dropout, lr):
  model = Sequential()
  for i in range(1, layers + 1): if i ==
    1:
      model.add(LSTM(units=units,
        return_sequences=(layers > 1),
        input_shape=(T, n_features)))
    else:
      model.add(LSTM(units=units,
        return_sequences=(i < layers)))
    model.add(Dropout(dropout))
  model.add(Dense(1, activation='sigmoid')) return
  model
```

GRU Model

The GRU architecture simplifies LSTM's gating structure by combining the input and forget gates into a single update gate and eliminating the separate cell state [5]. This reduces the total parameter count, resulting in faster training. However, the reduced memory capacity limits its ability to capture long-range dependencies, which is directly reflected in the observed performance gap discussed in Section V.

```
def BuildGRUModel(layers, units, dropout, lr):
model = Sequential()
for i in range(1, layers + 1): if i == 1:
model.add(GRU(units=units,
return_sequences=(layers > 1), input_shape=(T,
n_features)))
else:
model.add(GRU(units=units, return_sequences=(i <
layers)))
model.add(Dropout(dropout)) model.add(Dense(1,
activation='sigmoid')) return model
```

Training and Hyperparameter Optimisation

The dataset was partitioned into training (70%), validation (15%), and test (15%) subsets using stratified sampling to maintain class distribution. Both models were trained using the Adam optimizer with binary cross-entropy loss. A systematic grid search was conducted over the hyperparameter space defined in Table III. The configuration yielding the highest validation accuracy was selected for final evaluation [12].

TABLE III: Hyperparameter Search Space

Hyperparameter	Candidate Values
Number of Layers	1, 2, 3
Number of Units	64, 128, 256
Dropout Rate	0.2, 0.3, 0.4
Learning Rate	0.001, 0.01, 0.1

```
best_model, best_acc = None, 0.0 for layers in [1, 2, 3]:
for units in [64, 128, 256]:
for dropout in [0.2, 0.3, 0.4]:
for lr in [0.001, 0.01, 0.1]:
model = BuildModel(layers,units,
dropout, lr)
model.compile( optimizer=Adam(lr), loss='binary_crossentropy')
model.fit(X_train, y_train, epochs=50, batch_size=32, validation_data=(X_val,y_val))
acc = Evaluate(model, X_val, y_val) if acc > best_acc:
best_acc = acc best_model = model
```

Evaluation Metrics

Model performance was assessed using five standard classification metrics. The F1-score and ROC AUC were prioritized as primary metrics given the mild class imbalance (60/40 distribution), as they are more informative than accuracy alone under such conditions. Table IV summarizes the metrics used.

TABLE IV: Evaluation Metrics

Metric	Formula
Accuracy	(TP+TN) / (TP+TN+FP+FN)
Precision	TP / (TP+FP)

IV. RESULTS

LSTM Model Results

The optimally configured LSTM model (2 layers, 128 units, dropout=0.3, lr=0.001) achieved 87.5% accuracy, 100% precision, 83.33% recall, 90.91% F1-score, and 91.67% ROC AUC. The 100% precision indicates every predicted jam was a true jam, making the model highly reliable for alert generation. The 83.33% recall confirms it successfully detected five out of every six actual jam events. The 91.67% ROC AUC demonstrates excellent discrimination between jam and non-jam conditions.

GRU Model Results

The optimally configured GRU model (1 layer, 64 units, dropout=0.2, lr=0.001) achieved 50.0% accuracy, 100% ,precision, 33.33% recall, 50.0% F1-score, and 66.67% ROC AUC. Although precision remained perfect, the 33.33% recall reveals the model detected only one-third of actual traffic jams. This conservative prediction behavior results in low overall utility despite zero false positives.

TABLE V: Comparative Performance of LSTM and GRU Models

Model	Accuracy	Precision	Recall	F1-Score	ROC AUC
LSTM	87.5%	100%	83.33%	90.91%	91.67%
GRU	50.0%	100%	33.33%	50.0%	66.67%

V. DISCUSSION

Performance Comparison

The results demonstrate a clear and consistent performance advantage for LSTM over GRU across all metrics except precision, where both achieve perfect scores due to identical conservative threshold behavior. The LSTM advantage is most pronounced in recall (83.33% vs. 33.33%), F1-score (90.91% vs. 50.0%), and accuracy (87.5% vs. 50.0%), collectively reflecting a far superior ability to detect actual traffic congestion.

Explaining the LSTM Advantage

Memory Capacity: The LSTM architecture maintains a dedicated cell state updated independently through input and forget gates. This allows LSTM to retain information over longer time horizons. Traffic jam occurrences in this dataset are influenced by weather and event patterns developing over multiple preceding time steps, making LSTM's richer

memory structure better suited for capturing these extended dependencies.

Gating Complexity: The GRU consolidates LSTM's three gates into two (update and reset) and merges the cell and hidden states. While computationally efficient, this reduces expressive power for distinguishing what to retain. For the multi-feature, temporally rich traffic prediction task considered here, this manifests as substantially lower recall.

Dataset and Regularization: With 500 instances, LSTM's additional parameters - when well-regularised through dropout - provide sufficient capacity to distinguish positive instances reliably. GRU's lower parameter count results in insufficient capacity for reliable congestion detection on this dataset.

Perfect Precision and Class Imbalance

Both models achieve 100% precision, warranting careful interpretation. This results from both models predicting very few positive instances overall, all of which are true positives. The GRU's recall of 33.33% indicates it labels only a small subset of actual jams as positive, keeping false positives at zero while missing the majority of congestion events. Threshold adjustment or class-weighted loss could improve GRU recall and is recommended for future work. ROC AUC and F1-score were prioritized as primary metrics precisely because they are robust to this imbalance pattern.

Comparison with Related Work

The LSTM model's 87.5% accuracy and 91.67% ROC AUC are competitive with comparable binary traffic jam classification results. Li et al. [8] reported 80-90% accuracy using DCRNN with full spatial graph inputs; the present model achieves comparable discriminative performance using only environmental and event features without graph-structured road data. Transformer-based approaches [9] achieve higher accuracy on large-scale benchmarks but require significantly larger datasets and computational resources beyond the scope of this study.

VI. CONCLUSION

This study presented a comparative evaluation of LSTM and GRU deep learning architectures for binary traffic jam prediction using contextual weather and event data. The LSTM model demonstrated consistently superior performance, achieving 87.5% accuracy, 90.91% F1-score, and 91.67% ROC AUC, compared to 50.0% accuracy, 50.0% F1-score, and 66.67% ROC AUC for the GRU. The performance gap is explained by LSTM's greater memory capacity and gating expressiveness, which better accommodate the long-range temporal dependencies present in the dataset.

Key limitations include: (1) the dataset is limited to 500 instances; (2) spatial dependencies between road segments are not modelled; and (3) only two recurrent architectures are evaluated. Future work should expand the dataset, incorporate graph-based spatial modelling (e.g., DCRNN), benchmark against Transformer variants, and explore ensemble approaches combining LSTM with attention mechanisms. Threshold Calibration for improved recall under operational conditions is also recommended.

Acknowledgements

The author would like to thank the institutions and data providers who made the traffic and environmental datasets available for this research. No external funding was received for this study.

REFERENCES

1. Y. LeCun, Y. Bengio, and G. Hinton, Deep learning, Nature, vol. 521, no. 7553, pp. 436-444, 2015.
2. F. A. Gers, J. Schmidhuber, and F. Cummins, Learning to forget: Continual prediction with LSTM, Neural Computation, vol. 12, no. 10, pp. 2451-2471, 2000.
3. Y. Gal and Z. Ghahramani, Dropout as a Bayesian approximation: Representing model uncertainty in deep learning, Proc. ICML, pp. 1050-1059, 2016.

4. S. Hochreiter and J. Schmidhuber, Long short-term memory, Neural Computation, vol. 9, no. 8, pp. 1735-1780, 1997.
5. K. Cho, B. Van Merrienboer, D. Bahdanau, and Y. Bengio, On the properties of neural machine translation: Encoder-decoder approaches, arXiv:1409.1259, 2014.
6. J. Zhang, Y. Zheng, and D. Qi, Deep spatio-temporal residual networks for citywide crowd flows prediction, Proc. AAAI, vol. 31, 2017.
7. Z. Cui, R. Ke, Z. Pu, and Y. Wang, Deep bidirectional and unidirectional LSTM recurrent neural network for network-wide traffic speed prediction, arXiv:1801.02143, 2018.
8. Y. Li, R. Yu, C. Shahabi, and Y. Liu, Diffusion convolutional recurrent neural network: Data-driven traffic forecasting, arXiv:1707.01926, 2017.
9. C. Xu et al., Spatial-temporal Transformer networks for traffic flow forecasting, arXiv:2001.02908, 2020.
10. B. Yu, H. Yin, and Z. Zhu, Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting, Proc. IJCAI, pp. 3634-3640, 2018.
11. K. Greff et al., LSTM: A search space odyssey, IEEE Trans. Neural Netw. Learn. Syst., vol. 28, no. 10, pp. 2222-2232, 2017.
12. J. Bergstra, R. Bardenet, Y. Bengio, and B. Kegl, Algorithms for hyper-parameter optimization, Advances in NIPS, vol. 24, 2011.
13. N. Srivastava et al., Dropout: A simple way to prevent neural networks from overfitting, J. Mach. Learn. Res., vol. 15, no. 1, pp. 1929-1958, 2014.
14. D. P. Kingma and J. Ba, Adam: A method for stochastic optimization, arXiv:1412.6980, 2014.
15. I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. Cambridge, MA: MIT Press, 2016.
16. J. Zhang et al., DNN-based prediction model for spatio-temporal data, Proc. 24th ACM SIGSPATIAL, pp. 1-4, 2016.

Author's Details

1Ohm Prakasanatham, [Assistant Professor], [Department of Computer Science and Engineering], PSIT], [Uttar Pradesh], [India], [ohm@psit.ac.in]