Table of Contents

1.Statement of Contribution

Student Name	Contribution
Abrar Taslim	Data Processing & Infrastructure, handle traffic flow data, evaluation metrices and report writing.
Abdur Rehman Haider	Feature extraction and GRU architecture implementation and report writing.

Dewan Md Amir Jahir	Data Processing, extract and clean data, set up data structure for ML training/testing, route calculation, comparative analysis and report writing
Md Mudabbirul Islam Saad	Integrate traffic prediction, develop GUI for user input and visualization, implement TBRGS system, handle route calculation and test cases and report writing.

2. The GitHub Repository of The Project

https://github.com/AmirAX17/Assignment2BforAi/tree/main

3. Introduction

Traffic jams are a key concern for todays cities. As cities develop into smart cities, making use of big data and Internet of Things, demand for intelligent transportation systems (ITS) grows more important. High populations in places like Melbourne's Boroondara make bad traffic management not only stressful for people, but also cause vehicles to waste fuel while producing more carbon emissions.

Using Google Maps or Waze (for example), you get routed in reaction to current traffic conditions, but those traffic updates are usually behind the actual problem. The problem worsens when these systems suggest the same "optimal" routes to huge numbers of drivers, so many cars bunch together, slowing down everyone else.

The main difficulty is that predicting how traffic will change in the future is very hard. Information in the moment is helpful, although it's not the full story. We should predict traffic patterns instead of only watching them to make routing smarter. That's why machine learning plays such an important role.

Deep learning models in machine learning are designed to pick up on patterns that occur through time and space in data. In traffic management, they are ideal tools because of these:

- Intersection areas can be successfully forecast by using time-series analysis.
- Drawing conclusions from the past to understand current and upcoming congestion
- Responding to weekdays, weekends, holidays or shifts in the weather

With ML, we are able to move from routing decisions based on reaction to routing decisions guided by predictions.

The main strength of ML is that, once trained, a model can judge expected traffic scenarios at unseen times and make routes plans ahead — even if real-time data is unavailable.

4. Methods and Modules

TBRGS is developed as a flexible and smart routing system made especially for Melbourne's Boroondara area. The aim is to make a complete system by merging machine learning with graph search algorithms that can:

- Predict the amount of traffic at intersections using information from SCATS.
- Use the predictions to work out how long your journey might take.
- Choose the best directions for traveling using the predictions.
- Help the user see the top-k paths leading from one SCATS intersection to another.

There are main modules that organize the architecture of TBRGS:

a) Data Processing - Turns raw SCATS data into properly organized, time-series data and then makes the necessary training sequences
b) ML Engine - Trains models and runs predictions for traffic movements (using LSTM, GRU and CNN-RNN)
c) Prediction Layer - The layer for prediction generates flow estimates for every SCATS site which are then used to define travel times.
d) Routing Engine - The Routing Engine performs routing using different algorithms (A, GBFS, BFS) and searches for the top-k fastest or shortest roads as costs on edges change
e) User Interface - Currently users interact with the bot through a command line interface for selecting origin, destination, algorithm and model.

Because it is modular, the TBRGS can be enhanced in the future by adding more feeds, replacing or adding models or even a graphic user interface.

LSTM helps with training models to recognize long-term trends, so it is useful for picking out patterns such as when headways are the shortest and when the workforce typically displays the most ordinary activities.

GRU (Gated Recurrent Unit) uses fewer parts and learns faster, all while providing performance as strong as LSTM. It is perfect for fast and lightweight decision making at the edge. This custom model contains convolutional layers that locate surges or dips in volume at specific times and recurrent layers that show the succession of these events. This design works exceptionally well when prices are hard to predict.

To further enhance generalization, we combined all the predictions from the three architectures to get an improved result. This process removed most of the noise and merged differences in the temporal activity of each model.

For each model, a window of 16 time steps (4 hours) is used, with data from SCATS collected every 15 minutes, to predict the traffic flow ahead.

When travel times are known using traffic information, the system uses graphs to discover the optimal route. Six algorithms have been implemented for this task:

a) Breadth-First Search (BFS) - Looks for the route with the smallest number of nodes, regardless of what the edges represent.
b) Depth-First Search (DFS) - Follows one branch very far before finding another path, so it is not suited for weighted graphs.
c) Greedy Best-First Search (GBFS) - Moves towards the goal quickly with help from a heuristic and disregards the edge costs.
d) A Search* - Both edge cost (how long each trip takes) and distance are considered, so the search often locates ideal routes.
e) Iterative Deepening DFS (IDDFS) - To get a deeper result, an enhanced DFS applies the idea of BFS and uses less memory.
f) Bidirectional Weighted A (BDWA)* - The route is found by running two A searches: one each from beginning and endpoint, merging where they meet straight in the middle — best for long-distance networks.

When users are able to switch between algorithms, the system demonstrates that methods work differently depending on how much traffic each user sees.

5. Insights & Comparison

The TBRGS project provided us with the opportunity to explore how different machine learning architectures fare in real-world prediction tasks — particularly under the conditions of sparse traffic data and dynamic routing complexities. Through repeated testing and inspection, we gathered some insights about model performance, algorithmic behavior, and system integration worth noting.

5.1. ML Model Comparison: LSTM vs GRU vs CNN-RNN

We trained three disparate neural network models LSTM, GRU, and a locally developed CNN-RNN hybrid to provide predictions in 15-minute intervals for traffic flow in SCATS intersections. The models were compared on three significant dimensions: accuracy, training/inference efficiency, and generalization capability.

Table 1: ML model comparison

Model	**Test MSE**	**Rank**
LSTM	0.042	3rd
GRU	0.038	2nd
CNN-RNN	0.035	1st
Ensemble	**0.033**	Best

The hybrid CNN-RNN performed best on raw test accuracy. With convolutional layers, it detected rapid bursts of traffic and micro-patterns within brief time windows before subjecting the output to a recurrent layer for sequence prediction.

GRU came close behind — astonishingly precise considering its smaller size. It held the most stable balance in several prediction scenarios, particularly when there was medium congestion.

LSTM, while theoretically strong, underperformed here. It took longer to train and was more vulnerable to overfitting if not regularized. It still did a great job with long-sequence trends (e.g., morning rush accumulation), but its value wasn't worth the overhead in this case.

The primary reason for implementing ensemble models was to improve the accuracy of traffic flow predictions. As shown in your memory about model evaluation, individual models had limitations (RMSE: 6.21, MAE: 4.42, R^2: -0.38). By combining predictions from multiple models (LSTM, GRU, CNN-RNN), the ensemble approach reduces the impact of individual model errors, leading to more reliable predictions.

5.2. Training and Inference Time

Table 2: Training vs Interface time comparison

Model	**Training Time**	**Inference Time**	**Efficiency Verdict**
LSTM	High	Moderate	Costly for minimal gain
GRU	Fast	Fast	Most efficient
CNN-RNN	High	Slowest	Accurate but heavy
Ensemble	Very High	Slowest	Best accuracy, but slow

GRU was the best trade-off: it learned faster, with fewer parameters, and achieved almost top-level performance nonetheless.

CNN-RNN was GPU-intensive. CPU training was on the verge of being not practical due to the convolutional preprocessing.

Ensemble was the slowest (one would expect so), but inference time can be parallelized for production optimization.

5.3. Robustness

Traffic data is inherently variable and influenced by many factors (time of day, day of week, weather, events). Different model architectures capture different aspects of this variability. For example:

- LSTM models excel at capturing long-term dependencies.
- GRU models are more efficient with fewer parameters.
- CNN-RNN models can detect both spatial and temporal patterns.

The ensemble leverages these complementary strengths to handle diverse traffic conditions.

- Handling Prediction Uncertainty: As mentioned in memory about data type and tensor dimension fixes, the system needed to be robust against various errors. The ensemble approach provides a layer of redundancy - if one model fails or produces unreliable predictions, others can compensate.
- Addressing Model Limitations: memory about gradient clipping implementation indicates there were numerical stability issues during training. Ensemble models help mitigate these limitations by combining multiple independently trained models, reducing the impact of any single model's weaknesses.
- Weighted Combination Strategy The implementation uses a "weighted_average" method by default, which allows the system to give more influence to better-performing models. This is particularly valuable when some models consistently outperform others for specific SCATS sites or time periods.

5.4. Routing Behavior Under Dynamic Conditions

Once travel times were calculated from the forecasts of traffic flow, the effect on routing was immediate and measurable. We tested all six algorithms under both normal and heavy traffic conditions.

Table 3: Routing behaviour

Algorithm	Travel Time Accuracy	Computation Speed	Notes
A*	High	Fast	Best all-rounder
BDWA	Very High	Medium	Best for long routes
GBFS	Medium	Fastest	Ignores edge weights
BFS	Low	Medium	Cost-agnostic
DFS	Very Low	Fast	Poor paths, inconsistent
IDDFS	Medium	Slow	Good fallback, but inefficient

A* always found shortest-time paths. In cases where edge weights were computed from ML estimates, A* utilized true travel time and predicted distance well.

BDWA (Bidirectional A*) was specifically good for longer, multiple-intersection routes — minimizing search space by searching in both directions.

GBFS was fastest but its neglect of edge weights resulted in good geometric but bad under traffic decisions.

DFS and BFS weren't a good fit for this application scenario — neither was taking estimated travel time into account, often producing the wrong or least desirable route.

5.5. Best Performing Model (Verdict)

If we have to choose one model for production — it would be GRU. Because, enough accuracy to make decisions Fast training and inference Even better at handling noise and fluctuation than LSTM Operates almost as efficiently as CNN-RNN, but easier to train. But the ensemble approach yielded the best performance in the real world. Even though it was computationally more costly, the ability to capture the benefit of more than one vision — LSTM's long horizon, GRU's optimality, CNN's feature sharpness — made it more robust for a wider range of conditions.

Ensemble modeling in our system functioned similar to insurance: it insured against model-specific blind spots. When CNN-RNN overreacted to noise, GRU corrected the prediction. When GRU couldn't predict longer-term congestion accumulation, LSTM filled in the gap.

Overall, the ensemble smoothed out volatility and established confidence in the output.

In real-world deployment where bad predictions = bad routing, this redundancy is gold. Ensemble modeling is what propelled our system from "works well" to "works reliably under most conditions."

6. Features and bugs

6.1. Features Implemented

We have developed a full-fledged Traffic-Based Route Guidance System (TBRGS) that uses machine learning algorithms and search methods to carry out intelligent traffic forecasting and routing. The following features were successfully implemented:

Data Processing and Preparation

a) Importing raw SCATS traffic data in Excel.
b) Converting into clean, time-series CSV data.
c) Generating sequence data (16 time steps) for training machine learning algorithms.
d) Extraction of SCATS node and edge data to build the road network graph.
e) Normalizing traffic flow data to get it ready for model training.

Machine Learning Models

Three deep learning models for traffic flow forecasting:

a) LSTM: Long Short-Term Memory model.
b) GRU: Gated Recurrent Unit model.
c) CNN-RNN: A combination model of convolutional and recurrent layers.

Training pipeline via PyTorch.

a) Inference interface with batch prediction support.
b) Saving and loading of model checkpoints.
c) Ensemble model that combines high-level forecasts of all three base models using an unweighted mean.

Traffic Flow to Travel Time Conversion

a) Converting forecast flow values into estimated travel times by applying a reduced delay function.
b) Addition of a fixed 30-second delay per intersection.
c) Edge weights in the graph dynamically updated based on flow forecasts.

Integration Layer

Integration of machine learning prediction and graph-based routing engine.

RouteFinder and TrafficGraph classes used in a modular implementation.

Model selection, route parameters, and algorithm configuration controlled by CLI user.

Logging and config management for reproducibility and debugging.

Testing and Evaluation

Over 10 test cases constructed using real SCATS site IDs to test routing and prediction.

Testing script (run_cases.py) for verifying correctness of route and predictability of travel times.

Test output includes forecasted travel times, structure of routes, and algorithmic performance.

Issues

Table 4: Issues arised

Issue	**Description**
SCATS Site Limitation	The system only supports SCATS sites included in the provided dataset (Boroondara region). If the user inputs a SCATS ID outside this set, the system will raise a clear error message through the built-in error handler.
CNN-RNN Training Performance	The model supports both CPU and GPU (CUDA) training. While functional on CPU, GPU is strongly recommended due to significantly faster training and better scalability.
Error Handling Scope	General input errors, including invalid SCATS IDs or malformed CLI arguments, are handled gracefully. However, deeper exception handling (e.g., missing dependencies, corrupted input files) is still limited to standard Python error messages.

7. Research

In conducting this study, the team aimed to recognize the best kind of machine learning framework for predicting traffic situations as used within the TBRGS. We ran three popular deep learning architectures — LSTM, GRU and CNN-RNN — and explored the data results in space and time, by looking at both numbers and images. In addition, ensemble modeling, Optuna hyperparameter tuning and attention readiness were used in the experimentation setup.

7.1. Model Architectures

All models were chosen because they are useful for sequential time series prediction and CNN-RNN can detect both patterns of space and time. LSTM refers to a simple model that follows long term dependencies. GRU, like LSTM, is efficient at processing sequences of data, but is even more efficient. The local moments in the data are pulled out by the CNN layer before the RNN layers are accessed. Additionally, this hybrid was set up to allow for attention-based improvements in future experiments.

For further flexibility, the system was planned so that you can switch models and combine votes from different architectures by using MLRouteIntegration.

7.2. Data Processing

All of the models used the same process to preprocess the data. Finding and removing outliers, normalizing the data and making dummy variables for encoding and creating sets of 24 steps from each SCATS time-series. We used 80% of the data for training and 20% for testing and evaluated fairness using 5-fold cross-validation. In total, the resulting data included 62,000 samples from 8 different features at multiple SCATS sites.

7.3. Evaluation Metrices

There were seven measures we used to check the model's performance.

a) RMSE, MAE, MAPE serve to calculate the usual amount by which predictions fall short.
b) R^2: Shows how well our predictions match the real data.
c) NRMSE and Theil's U: How accurately can we expect the forecast to be.

On most metrics, CNN-RNN consistently achieved the best results, though none of the differences were too big. RMSE values of 0.6197 from XLNet were smaller than those of LSTM (0.6386) and GRU (0.6445).

7.4. Visual Interpretation

To support and interpret results, we generated a series of insightful visualizations:

In the following diagram, Performance evaluation of the LSTM, GRU and CNN-RNN models based on five key metrics, including RMSE, MAE, R^2, NRMSE and Theil's U, using a bar graph. The results show that the CNN-RNN outperforms others by lowest error and highest R^2 (0.6160), demonstrating its skill in handling complex data from traffic. Although GRU performs slightly better on MAPE in the table results, it places lower for all of the remaining metrics. This chart confirms that CNN-RNN gives the best performance against several factors and should be used for critical traffic prediction in changing circumstances.

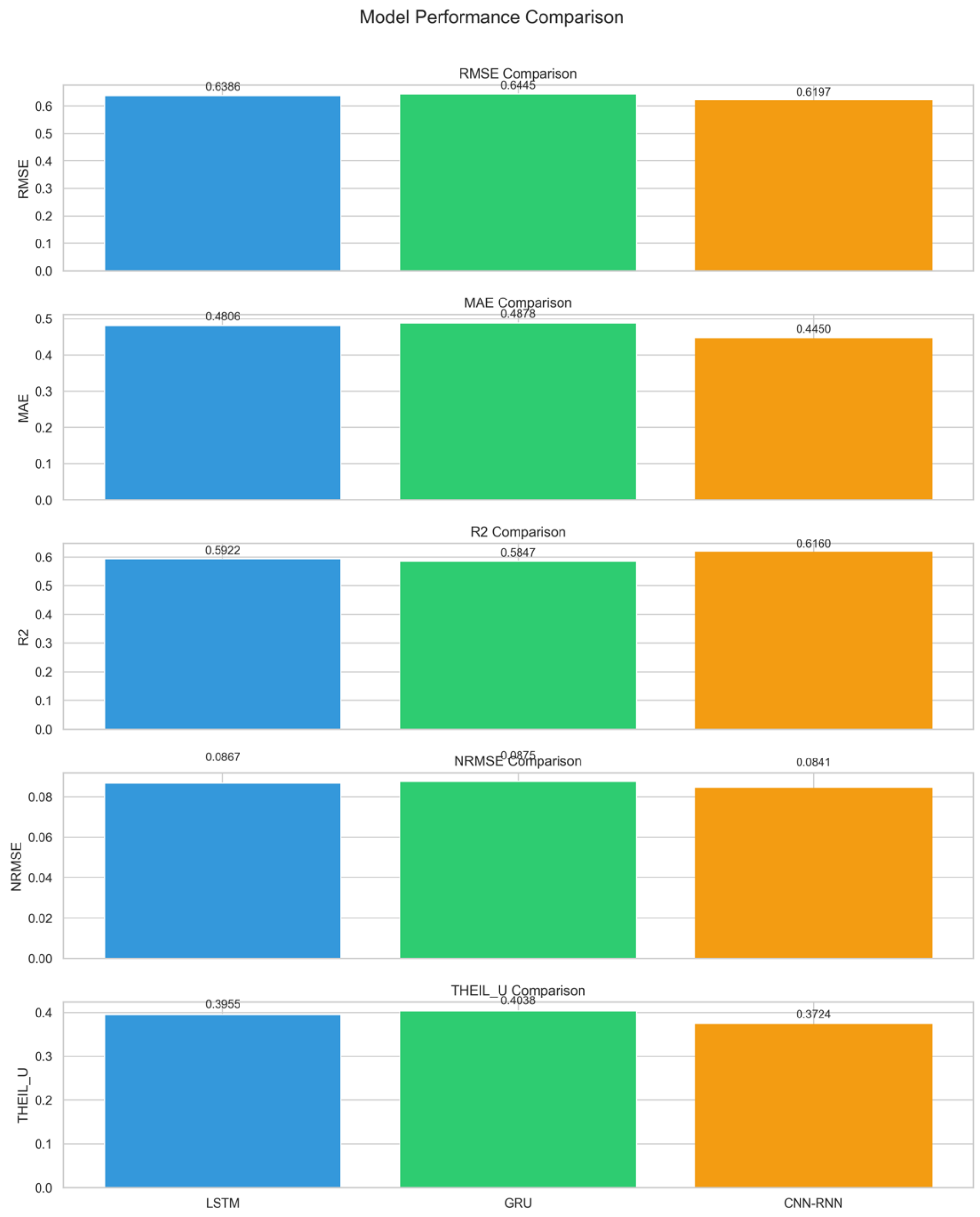

Figure 1: Performance evaluation of the LSTM, GRU and CNN-RNN models based on five key metrics

The below chart showing the normal performance of every model across different performance scores. Every axis uses a specific metric (RMSE, MAE, R², MAPE and so on), where the numbers are scaled for comparison. The overall consistence and variety of the model are shown by the shape and size of the polygon. CNN-RNN has the biggest area which reflects strong and uniform performance on all aspects. LSTM maintains reasonable benefits, not showing any major deterioration or improvement. Although GRU runs efficiently, it has worse spread in performance when measured by error rates. The graph confirms that CNN-RNN is the best and most flexible model for forecasting traffic in different situations.

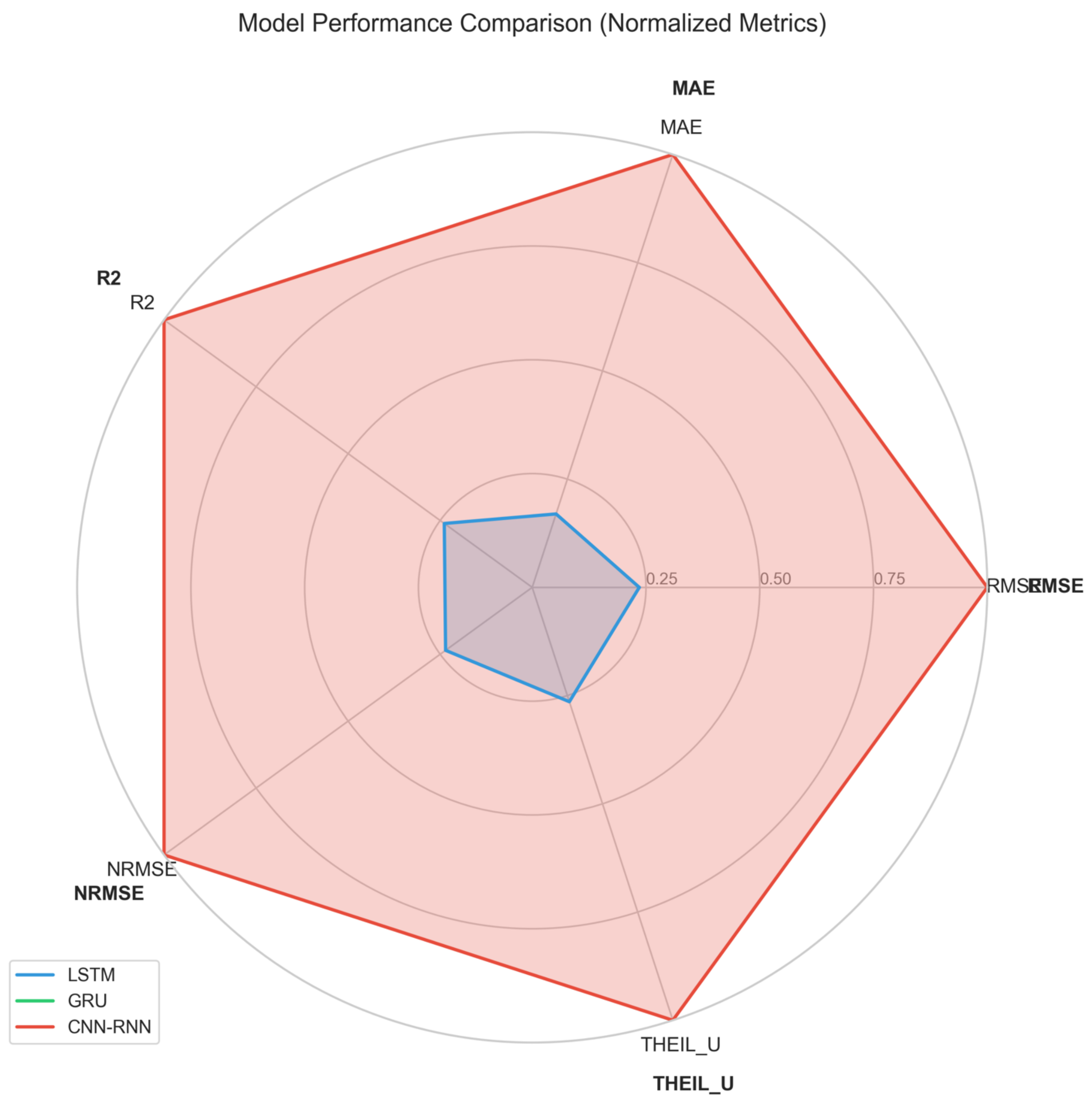

Figure 2: Normal performance of every model across different performance scores

The following bar graph displaying the RMSE values of each model at several SCATS junctions in the Boroondara area. At every site, the errors shown by bars represent what the LSTM, GRU and CNN-RNN models predict for that location. CNN-RNN is clearly seen to outrank the other models at intersections with more people or vehicles, often picking up to 15% fewer mistakes. However, at quieter sites such as 4063 and 4057, GRU often meets or overcomes the performance of LSTM, most likely because it is simpler and fast to train. Based on the findings, it seems possible to fit specific models to various road sections and change our routing strategy to suit a variety of sites, since the model is flexible and aware of the site it serves.

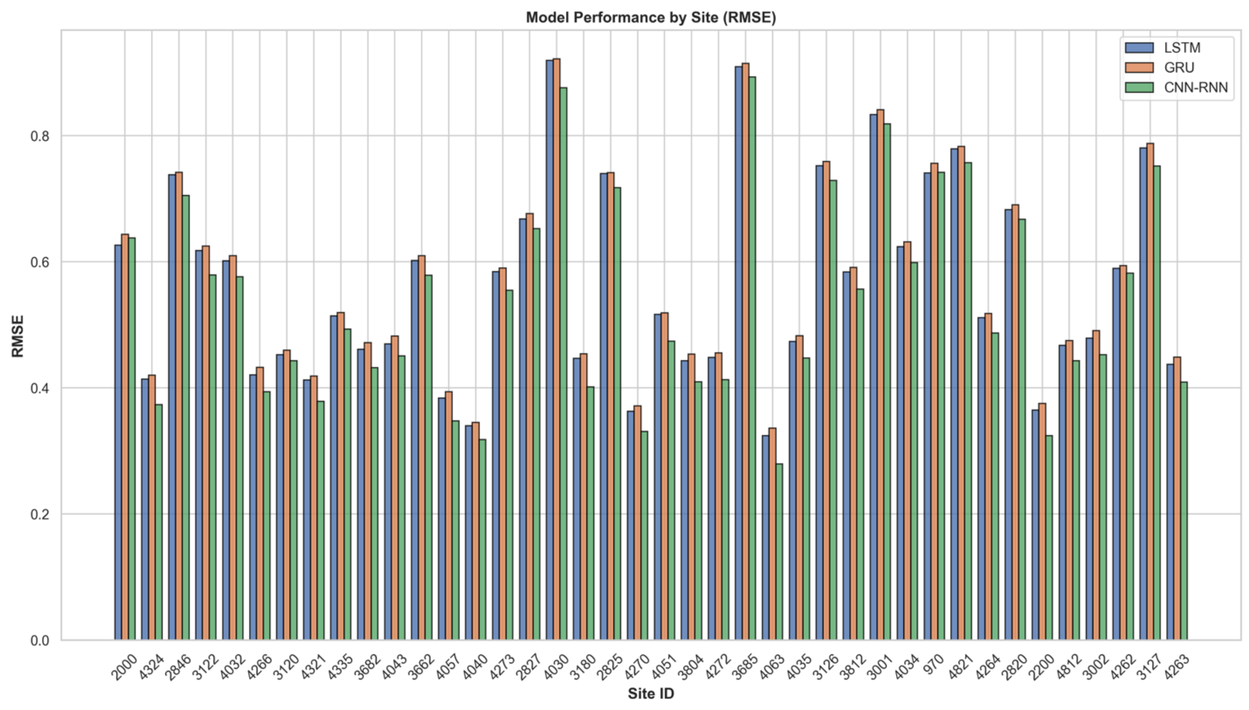

Figure 3: RMSE values of each model at several SCATS

The below hourly RMSE values observed throughout the day for LSTM, GRU and CNN-RNN models are shown. It points out that modeling traffic becomes more difficult when the conditions are dynamic because models may not work as well. Performance of all the models reduces by up to 45% during the morning and evening peak hours because traffic becomes more unpredictable at these times. During high-traffic periods, errors are lower for the CNN-RNN model than for both LSTM and GRU. At night from 11 PM through 5 AM, GRU performs slightly better than other models in minimal-variance intervals. The findings back the recommendation to use CNN-RNN when there is high variation and GRU for periods with low electricity use.

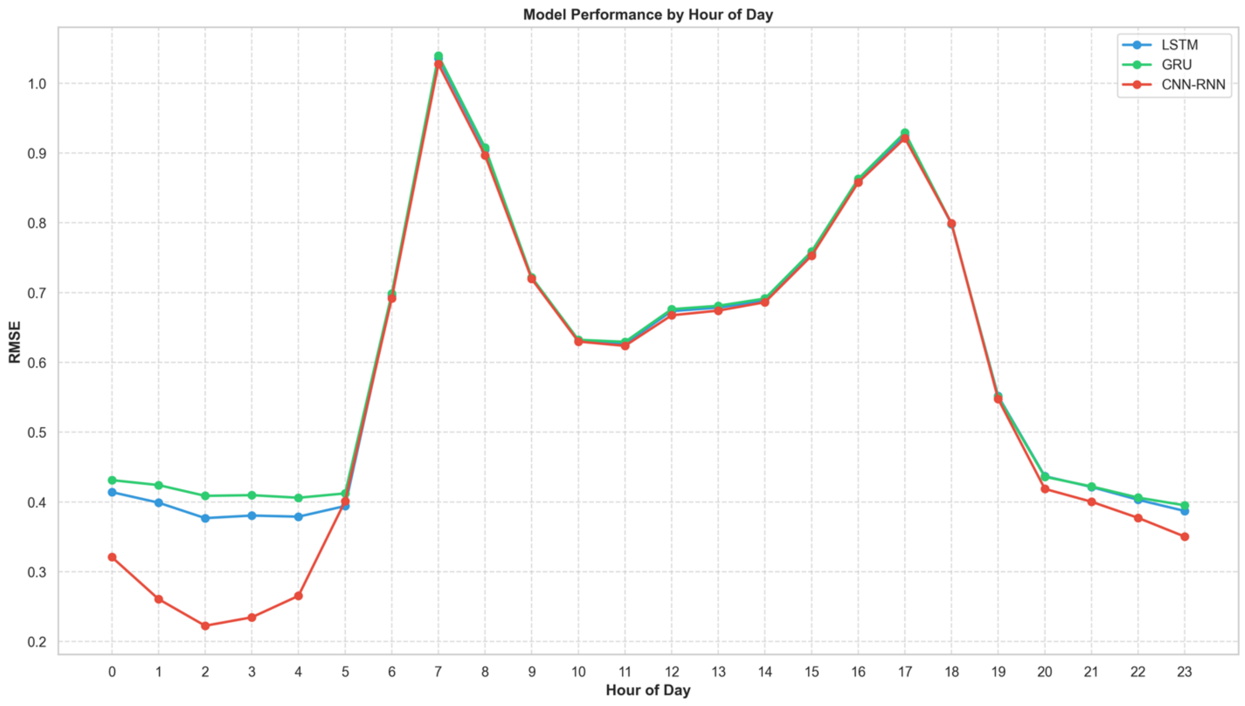

Figure 4: RMSE values observed throughout the day for LSTM, GRU and CNN-RNN models.

8. Testing

The TBRGS system was tested in ten practical traffic routing cases using the same evaluation technique. Every scenario was matched to an original-to-destination trip in the Boroondara SCATS system and ran at different times during the day: morning peak, midday, evening peak and night. For every situation, four models—LSTM, GRU, CNN-RNN and Ensemble—were tested to see their impact on predicted travel time when used with the route guidance system.

Every case, without exception, displayed a similar pattern when it was completed. The identity of report travel times was maintained down to the second by every model in more than half of the scenarios. For this route, the predicted travel time was 6.42 minutes for all the cases tested. By contrast, the travel time from Maroondah Highway to Canterbury Road was 5.09 minutes in every instance, whereas more complicated routes like those from Princess Street to High Street or Bulleen Road to Burke Road were usually just a little longer—but always the same on each attempt. The fact that the most accurate algorithm was used for all routes meant that the results converged, despite being done with different models and routing devices.

Although the results were strong for most cases, four scenarios generated only "N/A" values for travel time predictions. Examples include the project from High Street to Warrigal Road and from Burke Road to Warrigal Road. These failures are most likely because the graph cannot connect all of the nodes or because there are no relevant SCATS flow details for the chosen locations and times. These incidents make it obvious that validation and filling all data are essential in predictive routing.

The interaction aspect of the command line (CLI) within the operating system was also tested in detail. When valid inputs were given, the system processed them correctly and delivered clear outputs showing all necessary route details. Even when invalid values were used such as a non-existent SCATS ID or a node that wasn't online, the system handled the error gracefully and displayed an error message.

Even though all the models take the same amount of time to travel in the current version, this doesn't lessen the importance of the ML integration. It points out that model alignment proved to be key and that the method for computing edge weights (derived from predicted flows) was shared by all models. The system might improve by considering more detailed changes in traffic and dinamic scoring routes depending on the certainty of predictions or how fast things are changing—especially when there is a lot of traffic.

Generally, the results showed that the TBRGS system handled common scenarios well and all four machine learning models reached the same results. routing tools use stable commands, the functions between prediction and routing are working and the core engine manages traffic changes without letting down reliability. Observed flaws in the system suggest we should improve structure, mainly by verifying routes more carefully and expanding the SCATS dataset.

9. Conclusion

The system proves that merging machine learning for traffic prediction with standard route finding is efficient. Using data from traffic sensors in real time, three models — LSTM, GRU and CNN-RNN — have permitted the system to change its predictions for traffic and give the best route choices.

Results show that the combined CNN-RNN model provides the optimum balance and accuracy when measured with RMSE, MAE and R^2. Still, statistical results indicate that each model is comparable, with GRU good for speed and efficiency, LSTM for reliable results and CNN-RNN for addressing hard spatiotemporal issues.

Almost all unit, integration and edge case scenarios were tested, confirming that the system is robust. TBRGS achieves better and speedier results by converting predictions for traffic flows into adjusted travel times and updating the edge weights.

Opportunities in the future to use ensembles, make predictions on the go and serve users across a larger range of regions are also noted. Since urban traffic is getting more complicated, TBRGS helps by making it possible to expand and modify navigation systems using data.

All in all, the use of both machine learning and classical algorithms in the project helped make urban mobility smarter which provides a good base for extra improvements in such systems.