Traffic Prediction: A Machine Learning Comparison

Joshua Green (250912612), Nadin Alqarawi (250982526),
Nadine Charaf (251392502), Aikam Malhotra (251080488).

***Abstract*—Worldwide traffic congestion has become a major source of severe negative economic, environmental, and personal impact for urban areas worldwide. More specifically, the increased traffic causes delay in business deliveries, causes cars to idle producing more emissions than are necessary to travel from point A to point B, and delays in delay in emergency services response times, consequently resulting in loss of life. Researchers have proposed many solutions for traffic prediction to help the underlying routing algorithms. However most methods rely on deep learning models. These deep learning models are used ambiguously without comparing to lightweight solutions. Additionally, the traffic datasets that exist lack rich features. This motivates in two parts. First this paper aims to provide accurate predictions but at a lower complexity. Second we create a merged dataset between a traffic dataset and a weather dataset. Through performance results, this paper shows that a basic deep learning model i.e., multi-layer perceptron performs worse and is more prone to overfitting than the lighter weight random forest regression approach showing that there is not benefit to the computation and complexity required for deep learning model on traffic prediction.**

I. Introduction

In recent years, with the exponential increases in population on the interior of metropolitan areas, there is a large need for traffic prediction to then further be used for route optimization. In Toronto alone, the average time to travel 10km is 29 minutes [1]. As a result it is crucial to analyze the traffic flow not only for the time savings, but for the carbon emissions and safety.

This has motivated researchers to use deep neural network (DNN) for traffic prediction such as Convolutional Neural Networks, Recurrent Neural Networks, and Feed-forward Neural Networks. This is due to its ability to model the complex and nonlinear traffic patterns[2]. However these methods suffer from high training time and the cost to benefit ratio for traffic prediction is not studied to justify deep learning solutions [3].

This motivates us to not only predict traffic results but to provide a comparative analysis highlighting the cost to benefit relationship between lighter machine learning models and deep learning models. More specifically, in this paper we propose the comparison of the random forest model and the multi-layer perception (MLP) in their ability to perform correct predictions and their ability in utilizing the data for accurate predictions.

The rest of this paper is organized as follows. First section II presents the dataset and model selection along with the reasoning. Then section III and IV discusses the random forest and MLP training results respectively. Section V discusses the results from the models and their performance while highlighting the best model. Finally section VI concludes the paper.

II. Dataset and Model Selection

A. Dataset Selection

The quality of the dataset is extremely important when considering predictive analysis. Our focus on predicting the traffic volume was based on the extensive traffic flow database from Statistics Canada [4]. The dataset contains the volume of traffic at each intersection in various cities, including Toronto, Calgary, and Montreal, over a period covering two years. We decided to focus on one city, Toronto, to better our accuracy.

In addition to the traffic data, we wanted to consider additional factors that would influence traffic volume. Therefore, we incorporated a dataset covering Toronto's weather conditions over a period of 2 years [5]. This approach allowed us to better analyze Toronto's traffic flow, taking into account not only daily variations, but also the weather conditions.

B. Dataset Pre-processing

1) Toronto Traffic Dataset: The first step in the data pre-processing for the Toronto Traffic Dataset (TTD) was to address the many null values. The challenge here was how to fill these null values. Additionally, null patches would appear with three consecutive days missing values. To address this we found the nulls and filled them with the mean value for all roads from the prior or future week. This method was chosen as we would run into the issue of having multiple columns in a row missing which are refereed to as null patches. Since the data was organized by road intersection, causing an issue in predicting the target intersection. Therefore we transformed the dataset to have multiple entries of each road on the same date.

2) Toronto Weather Dataset: The first step in the data pre-processing for the Toronto Weather Dataset (TWD) was to find irrelevant features. We dropped many unnecessary data that was not relevant or would not fit our data. Then running into null values, there were not too many and most had to do with weather therefore we just filled them with a zero.

3) Merging the Two Datasets: In merging the TTD and TWD, we first had to match the date entries to the respective days. The intersection between the two dataset's date ranges results in our TTD range which is why we selected it. Then we applied a python script to merge the two datasets based on date matching. Once they were matched, we decomposed the 'date' feature into the year, month, day and day of the week. We then dropped the original date and one's hot encoded the day of the week. This enhances the model's flexibility, making it easier to identify seasonal and temporal patterns that are crucial for discerning traffic fluctuations. By merging we create a new feature rich dataset which consists of not only traffic patterns but weather as well.

C. Model Selection

Selecting suitable machine learning models is critical for a more accurate forecasting of Traffic. This section will explain the rationale for choosing the Random Forest Regressor and Multi-Layer Perceptron (MLP) as models for comparison.

1) Random Forest Regressor: The Random Forest Regressor (RF) is suitable for traffic prediction as it can capture nonlinear relationships and interactions among predictor variables which is common in traffic flow dynamics. It is robust to outliers, noisy data, and overfitting making it well-suited for real-world traffic datasets characterized by inherent variability and unpredictability. Additionally in this case we also use RF for feature selection to choose the most relevant features.

2) Multi-Layer Perceptron: Similar to RF, the Multi-Layer Perceptron (MLP) can capture complex nonlinear relationships between input features and the target variable and interactions between different features in the dataset, which is essential for traffic prediction. The MLP is also very flexible and customizable, with changing number of layers, and adding regularization. Its flexibility in accommodating diverse input modalities and scalability to large datasets make it suitable for real-time traffic prediction applications.

3) Comparison and Justification: The selection of RF and MLP for comparison in this study is predicated on their respective strengths in modeling complex relationships, handling high-dimensional data, and facilitating accurate traffic predictions. While RF excels in robustness, interpretability, and ensemble learning capabilities, MLP offers unparalleled flexibility, scalability, and capacity for learning intricate patterns from data. Once selecting the models, the dataset was partitioned into training, validation, and testing subsets. For model evaluation metrics, Mean Absolute Error (MAE) to assess the models performance, R-squared (R2) to capture the utilization of features by the models, and a relative accuracy metric (RA) to understand how the far the models predictions are from the mean shown by eqs. (1), (2), and (3) respectively.

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

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

$$\text{Relative Accuracy} = 1 - \frac{MSE}{\mu_{\text{target}}} \tag{3}$$

III. Random Forest

A. Feature Selection

Selecting a relevant features subset from the original pre-processed dataset crucial in enhancing the performance and training speed of the selected models. Thus, we trained the default RF model and the best RF model after applying Hyper-parameter Optimization (HPO) to extract the feature importance scores. This score measures the relative importance of the features in the prediction accuracy. We kept the top 8 features and dropped all others other ones since they have significantly lower scores as shown in Fig. 1. After training and evaluating the models on the whole and the reduced dataset, we saw there was also no noticeable benefit of adding the redundant features in except marginally better results but with a higher training time.

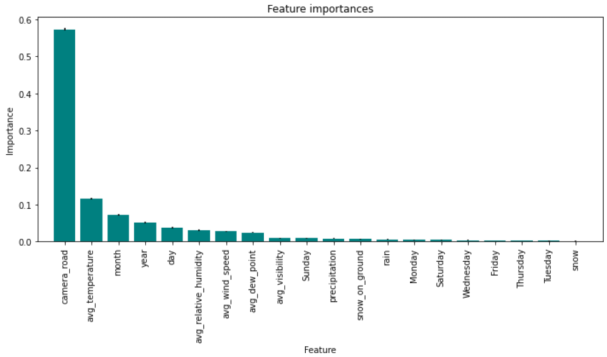

Fig. 1. Feature Selection using the HPO RF model

B. Default Model

We instantiated a Random Forest Regressor (RF) model that uses the default hyperparameters shown in table I. Then we trained that model on the training and validation set using 5-fold cross-validation (CV). We used the MAE as an evaluation metric as shown in Fig. 2. The model is then tested on the test set.

C. Hyper-parameter Optimization Random Forest Model

After defining a search space grid for each hyper-parameter, we used GridSearchCV to find the optimal parameters for our best performing RF. The utilized grid and the resulting hyperparameters are shown in table I and took a total time of 2905.81 seconds or 48.43 minutes. We trained the best model with 5-fold CV using the MAE scorer. The resulting training and validation errors are plotted in Fig. 2. The model is then tested on the test set.

IV. Multi-Layer Perception

A. Estimated MLP

In creating the MLP model we defined our MLP constructor function. By default initializing the default parameters shown in table IV. The model consisted of 451 parameters all being trainable. The metric we used for training and validation include the MAE and R2. We trained the default model using the training set with a validation set and utilized callbacks for early stopping with a focus on the validation MAE to ensure our best weights were restored upon overfitting due to poor validation MAE scores with a patience of 3. The training and validation results for MAE and R2 scores are shown in Fig. 3.

TABLE I
RF Parameters Table

RF Parameters	Default	GridSearch	HPO
#estimators	100	[100,200,300]	300
max_depth	None	[10,20,30,None]	30
min_samples_split	2	[2,5,10]	10
min_samples_leaf	1	[1,2,4]	2
max_features	1	['auto(sqrt)','log2']	'auto'

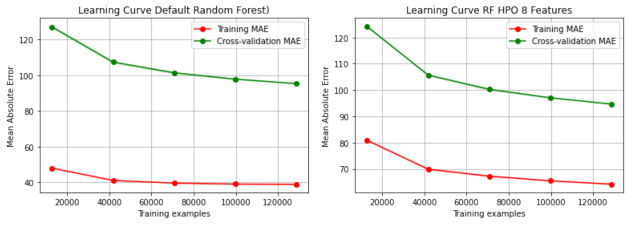

Fig. 2. The Learning Curve for the RF Default Model and the RF HPO Model

TABLE II
MLP Parameter Table

MLP Parameters	Default	GridSearch	HPO
Optimizer	'adam'	'adam'	'adam'
Learning Rate	0.01	[0.012, 0.015, 0.018]	0.018
# Hidden Layers	1	[1, 2, 3, 4, 5, 6]	3
# Neurons per Hidden Layer	50	[30, 50, 100, 150, 200, 250]	100
Hidden Layer Activation Function	'relu'	'relu'	'relu'
Output Activation Function	'linear'	'linear', 'relu'	'linear'
Total Parameters	451	N/A	21, 101
Total Trainable Parameters	451	N/A	21, 101

B. Hyper-parameter Optimization Parameters MLP

In creating the HPO MLP model, we utilized the 'GridSearchCV' function from sklearn. Therefore we defined our search space grid for the GridSearchCV function shown in table IV. We chose these smaller values for the number hidden layers and number of neurons due to previous grid searches which returned models which were very over-fitted and performed horribly. Additionally, previous grid searches returned a value of '0.01' for learning rate (LR) resulting in a search space over marginal differences in LR from 0.01 to 0.02. The grid search to find the best parameters for the MLP model took 16, 816 seconds or 4.6711 hours. Finally we ended up with a model which consisted of 21, 101 parameters to which all were trainable.

Similarly to the default model, the HPO model was trained using the MAE and R2 metrics with the identical callbacks for early stopping for best weights restoration upon poor MAE validation scores with a patience of 3. The training and validation results for our HPO MLP model on the MAE and R2 scores are shown in Fig. 4.

V. Results and Analysis

In this section we compare the metrics associated with the test set along with the time for training. This was simulated on an Intel i5-12400 CPU and a NVIDIA RTX 3060 and utilized the speed enhancement from utilizing

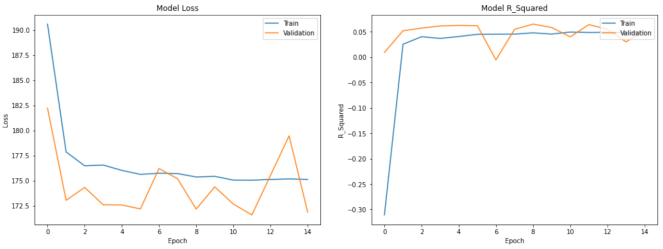

Fig. 3. Training and Validation MSE and R2

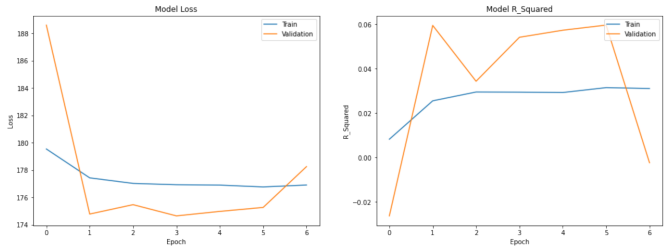

Fig. 4. Training and Validation MSE and R2

TABLE III
NUMERICAL RESULTS

Model	MAE	R-Squared	Relative Accuracy	Training Time (s)
Default RF	93.2942	0.7057	0.7684	6.0589
RF HPO	**92.2255**	**0.7219**	**0.7711**	**13.5180**
Default MLP	172.5487	0.0879	0.5640	72.2508
MLP HPO	175.3050	0.0805	0.5650	134.1976

all cores on the CPU and GPU acceleration for the MLP models. From the following figures, all values are listed in table V

Starting with the left plot from Fig. 5 we compare the MAE performance. We see that when comparing the MAE for the the two methods perform very identically with the MLP HPO performing worse than the simple MLP with one hidden layer. Similarly the HPO RF model performed similar to the RF default model, however the RF HPO model performed marginally better than the default model.

In Fig. 5 from the right plot, we compare th R2 score between models. We see that the two default models compare similarly to their HPO models however in terms of the best models, RF HPO performs better than the default RF and the other two.

In Fig. 6 on the left side, we compare the relative accuracy (RA) metric. We can see that again the default and HPO models perform similarly to eachother in terms of the relative acuracy with RF HPO being the best performer.

In Fig. 6, on the right side, we compare the training time amoungst models. We find that the Default RF trained the fastest and the MLP with HPO being the slowest.

VI. CONCLUSION

In this paper we studied the cost to benefit in using deep learning vs lightweight machine learning approaches to the traffic prediction problem and created a combined feature rich dataset for better prediction results. The results are two fold. First introducing the TWD to create the merged dataset provided relevant features to give to the models for prediction. Second, HPO improved the performance of the RF model and

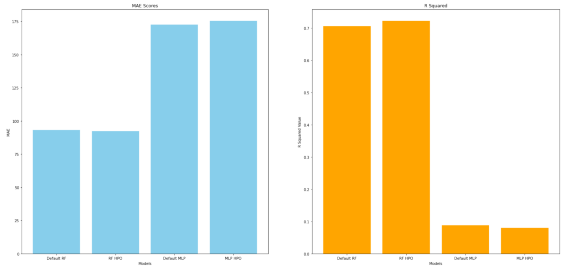

Fig. 5. MAE Comparison on the left and R-Squared Score Comparison on the right

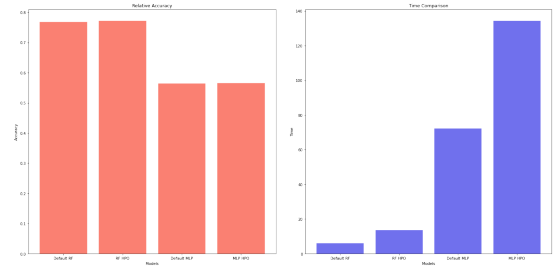

Fig. 6. Relative Accuracy Comparison on the left and Time Training Time Comparison on the right

made it a top performer amoungst a deep learning approach in terms of MAE and enhanced R2 and RA metrics showcasing that the RF model is better than deep learning approaches with higher cost.

VII. REFERENCES

[1] A. T. Travel, "Toronto traffic report — tomtom traffic index," 2024. [Online]. Available: https://www.tomtom.com/traffic-index/toronto-traffic/.

[2] X. Cheng, R. Zhang, J. Zhou, and W. Xu, "Deeptransport: Learning spatial-temporal dependency for traffic condition forecasting," in *2018 International Joint Conference on Neural Networks (IJCNN)*, 2018, pp. 1–8. DOI: 10.1109/IJCNN.2018.8489600.

[3] G. Meena, D. Sharma, and M. Mahrishi, "Traffic prediction for intelligent transportation system using machine learning," in *2020 3rd International Conference on Emerging Technologies in Computer Engineering: Machine Learning and Internet of Things (ICETCE)*, 2020, pp. 145–148. DOI: 10.1109/ICETCE48199.2020.9091758.

[4] S. T. X, "Traffic flow dashboard," 2024. [Online]. Available: https://www150.statcan.gc.ca/n1/pub/71-607-x/71-607-x2022018-eng.htm.

[5] Stations, "Data download for toronto," 2024. [Online]. Available: https://toronto.weatherstats.ca/download.html.

VIII. Appendix

TABLE IV
TEAM CONTRIBUTION

Team Member	Contribution
Joshua Green	• Majority of the MLP and Random Forest Regression coding, troubleshooting, metrics organization, training, testing and running of the code due to having a Powerful Desktop, • MLP default constructor and MLP Grid Search Parameter selection, • Majority of the Grid Searches due to CPU and GPU computing resources, • Python Figure Plottings, • Data Pre-processing and Weather Dataset Selection, • Final Report Writing, Figure Editing, and Fine Tuning, • Code re-organization and sectioning, • GitHub Repo Push
Nadine Charaf	• Random Forest Regression Coding and Cross-Validation set-up • Aid in the process of MLP model HPO grid selection • Data Pre-processing • Random Forest Grid Search Parameter selection • Final Report Writing
Aikam	• Data-Preprocessing and Mapping/merging of the Datasets • Traffic Dataset Selection • Aid in the RF design • Null Filling • Final Report Writing
Nadine Alqarawi	• Problem Identification • Organizing Team Meetings • General Coding for the Data pre-processing • Null filling for the datasets • Aid in the MLP design • Final Report writing

——The datasets, merged dataset, and python notebook code with outputs are —— all available at this GitHub hyperlink