A DEEP LEARNING MODEL TO FORECAST THE IMPACT OF COVID-19 ON TRAFFIC DEMAND IN SALT LAKE COUNTY

Aaron Wang

Introduction

Since early March 2020, the global COVID-19 pandemic has had an enormous impact on various aspects of society. In the United States, there have been over 79.2 million cases and 957 thousand deaths, as of March 7, 2022. In addition to illness and loss of life, the pandemic has had a significant effect on traffic across the United States. During the early stages of the pandemic, traffic was significantly reduced due to travel restrictions imposed by governments, fear of getting sick, lower levels of economic and social activity, and shutdowns of schools and businesses. Later on, as the pandemic progressed, traffic volume demonstrated a gradually increasing trend. It is clear that the COVID pandemic has had a significant effect on traffic demand.

This project seeks to predict the impact of COVID-19 on vehicular traffic in Salt Lake County during the pandemic. Traffic data from freeways in Salt Lake County will be used to quantify the traffic patterns. Deep learning models will be developed to accurately forecast the traffic demand patterns in the near future, using a novel approach integrating machine learning with graph theory. Traditional machine learning forecasting methods will also be tested. These models will potentially be able to help responsive agencies prepare for near-future traffic patterns and demands.

Data Collection

This study focuses on freeway traffic patterns in Salt Lake County. County-wide vehicle miles traveled (VMT) of freeways within the county were used to quantify traffic flow. The VMT data was collected from the UDOT Performance Measurement System (PeMS) from January 2019 to the last week of November 2021, providing traffic data before and during the pandemic.

Various factors related to the pandemic and vehicular traffic were also collected as explanatory variables:

• Daily new COVID-19 cases and the percentage of fully vaccinated individuals over five years of age were collected from the Utah Department of Health.

• The daily news sentiment index, obtained from the Federal Reserve Bank of San Francisco, is based on analysis of economic-related news from 24 major U.S. newspapers, providing information about overall sentiment as a quantifiable number. The monthly unemployment rate of Salt Lake County was obtained from the Utah Department of Workforce Services.

• Weather conditions, including temperature, precipitation, and snow depth were collected from the National Oceanic and Atmospheric Administration. The weather observing station at the Salt Lake International Airport was selected for weather data in Salt Lake County.

• Pandemic-related policies, including stay at home orders, mask mandates, school closings, and a state of emergency, can restrict or influence travel. If a policy was effective at a certain time, a dummy variable was set to 1; otherwise, it was set to 0.

• Dates of holidays were collected from the Utah State Government. If there was a public holiday, the dummy variable was set to 1; otherwise, it was set to 0.

All data were aggregated by week. Economic and weather data were collected from January 2019 to the last week of November 2021, while pandemic related data and policies were collected once they became available until the last week of November 2021.

We can see that at the start of the pandemic in early March 2020, there was a sharp drop in VMT. Since about June 2020, VMT has gradually increased. There was another drop that coincided with the highest COVID case count around November and December 2020, although that may be influenced by other factors as well, such as winter weather. More recently, we can see a decreasing trend in VMT as COVID cases rise and Omicron spreads.

Stationarity tests were also conducted on the explanatory variables. Only the snow depth variable, health emergency, and holiday variables were stationary, while all other variables were non-stationary. After second order differencing, all explanatory variables were stationarized

The Python programming language and the Numpy, Pandas, and pmdarima libraries were used to process and analyze the data. The Matplotlib library was used to create graphs and visualizations of the data.

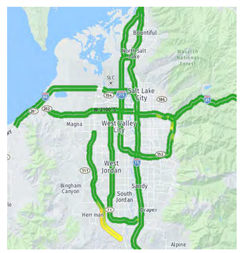

FIGURE 1. Freeways in Salt Lake County

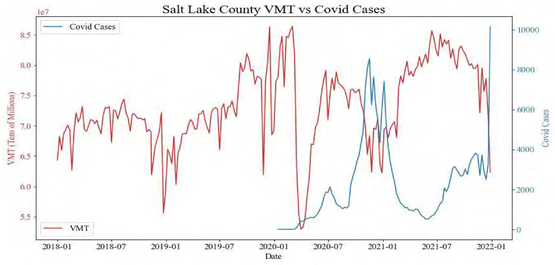

FIGURE 2. Salt Lake County VMT Versus Number of New Covid Cases

Models

ARIMA

An AutoRegressive Integrated Moving Average (ARIMA) model is a classic time-series forecasting model, combining AutoRegressive and Moving Average models.

The AutoRegressive (AR) component of the ARIMA models the current value based on a linear combination of previous values. It can be denoted as AR(*p*), where *p* is the number of previous values to consider. An AutoRegressive model of order *p* is defined as

$$x_t = \Phi_1 x_{t-1} + \Phi_2 x_{t-2} + \cdots + \Phi_p x_{t-p} + \varepsilon_t$$

where *ε_t* is noise, and *Φ_1, Φ_2, … , Φ_p* are parameters.

The Moving Average (MA) component of the ARIMA models the current value based on a linear combination of previous error terms. It can be denoted as MA(*q*), where *q* is the number of previous error terms to consider. A Moving Average model of order *q* is defined as

$$x_t = \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \cdots + \theta_q \varepsilon_{t-q} + \varepsilon_t$$

where *ε_t* is noise, and *θ_1, θ_2, … , θ_q* are parameters.

The ARMA model is a combination of the AR and MA, taking into account both past values and past error terms. It can be denoted as ARMA(*p,q*), where *p* is the order of the AR component and *q* is the order of the MA component. An ARMA model of order (*p, q*) is defined as

$$x_t = \Phi_1 x_{t-1} + \Phi_2 x_{t-2} + \cdots + \Phi_p x_{t-p} + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \cdots + \theta_q \varepsilon_{t-q}$$

where *ε_t* is noise, and *Φ_1, Φ_2, … , Φ_p* and *θ_1, θ_2, … , θ_q* are parameters.

The ARIMA model is just an ARMA modeled on a differenced series, taking into account the difference between two observations. The differenced series can be calculated as follows:

$$x_t = x_t - x_{t-1}$$

We can also difference the series multiple times, in which case we take the difference between two terms of the previously differenced series. An ARIMA model can be denoted as ARIMA(*p, d, q*), where *d* is the order of differencing, and *p* and *q* remain the same as previously described. The ARIMA is then formally defined as

$$x_t = \Phi_1 x_{t-1} + \Phi_2 x_{t-2} + \cdots + \Phi_p x_{t-p} + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \cdots + \theta_q \varepsilon_{t-q}$$

where *x_t* denotes the differenced series, and all other parameters are as previously defined.

Recurrent Neural Network (RNN)

A Recurrent Neural Network (RNN) is a type of artificial neural network adapted for sequential data. They are networks with feedback loops that allow them to retain memory of previous inputs to generate the next output. Unrolled, RNNs can be thought of as a chain of recurrent layers. The data is input sequentially, from each considered timestep. Each layer passes information on to the next layer, until the current timestep is reached and a value is outputted.

The equation for the hidden state at timestep *t+1* and activation function *f* is:

$$h_{t+1} = f(w_x \cdot x_t + w_h \cdot h_t + b_h)$$

Where *w_x* are the weights of the inputs in the recurrent layer, *w_h* are the weights of the hidden units, *x_t* is the input at timestep *t*, *h_t* are the hidden units at timestep *t*, and *b_h* is the bias unit of the recurrent layer. The output *y* at time *t* is computed by

$$y_t = f(w_y \cdot h_t + b_y)$$

where *w_y* are the weights of from the hidden to output layer and *b_y* is the bias unit of the feedforward layer.

A diagram of an unrolled RNN is shown in Figure 3.

Long Short-Term Memory (LSTM)

A Long Short-Term Memory (LSTM) model is a type of recurrent neural network (RNN) that can handle long-term memory better than traditional RNNs. It is commonly used for time series forecasting. An LSTM has a chain of repeating modules of neural networks, with each module containing a cell, input gate, output gate, and forget gate.

The LSTM model utilizes its forget gates to remove bias toward recent events. A given LSTM unit will take in the previous cell state and perform various activations, multiplications, and concatenations. Then, the forget gate calculates how much of the current cell state and its input should be passed along, forgetting the rest. A diagram of an LSTM cell is shown in Figure 4.

Note that while the activation function normally used in LSTM is the hyperbolic tangent function (tanh), the rectified linear unit (ReLU) activation function was adopted for better performance in this model.

Graph Convolutional Network—LSTM (GCN-LSTM)

Graph Neural Networks (GNN) are increasing in popularity due to their ability to incorporate graphs and capture the relationships between various factors.

The Graph Convolutional Network (GCN) can be used to perform machine learning while incorporating network structures such as graphs. As shown in Figure 5, the model consists of two parts: the graph convolution network (GCN) layer and the LSTM layer. After being passed through the GCN layers, the output is used as input to the LSTM layers, which then output the final result. The GCN layer has trainable parameters weight matrix **W** and bias vector **b**, with inputs of node features matrix **F** and the normalized graph adjacency matrix **A'**, as shown in Figure 6. The graph adjacency matrix **A'** represents the connections between the nodes. **A'** is normalized so that each node's contribution is proportional to how connected the node is in the graph. The output of a layer can be calculated as

$$z = \sigma(A'FW + b)$$

where *σ* is some non-linear activation function. The output can then be used as the input to another neural network layer.

The knowledge graph used in this project is displayed in Figure 7, where the edge from node *i* to node *j* exists if the node *i* has a possible impact on node *j*. The adjacency matrix **A'** is a binary matrix showing the existence of edges between nodes.

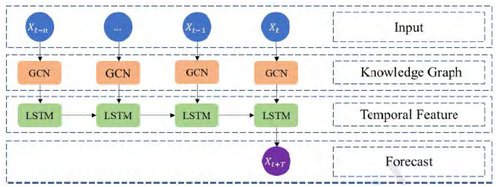

FIGURE 5. GCN-LSTM Model Structure

Model Architecture and Training

Data since the start of the pandemic were used to develop the model. One hundred forty weeks of data, from January 2019 to September 2021, were used for training, eight weeks were used for cross validation, and the most recent four weeks of data were used for testing.

To build the ARIMA model, the exogenous variables were made stationary through second order differencing. The AutoArima function of the pmdarima Python library was employed to find the optimal hyperparameters for the ARIMA model. The Akaike Information Criterion (AIC), which calculates information loss, was used as the metric for optimization. It is calculated as

$$AIC = -2ln(L) + 2k$$

where *L* is the log likelihood, and *k* is the number of parameters.

The hyperopt library from Python was utilized to build the optimal RNN, LSTM, and GCN-LSTM models, using Tree of Parzen Estimators (TPE) to search for the best hyperparameters. During training, an early stopping technique was employed to prevent overfitting. The loss function used in training was the mean squared error between the predicted factors and the observed ones. The Adam optimizer was selected to minimize the loss.

A persistence model was selected as the benchmark model. It calculates the future value of a time series under the assumption that nothing changes between the current time and the forecast time.
Two evaluation metrics were employed, root mean squared error (RMSE) and mean absolute percentage error (MAPE), which are calculated as follows:

$$RMSE = \sqrt{\frac{1}{n}\sum_{t=1}^{n}(VMT_t - \widehat{VMT}_t)^2}$$

$$MAPE = \frac{1}{n}\sum_{t=1}^{n}\left|\frac{VMT_t - \widehat{VMT}_t}{VMT_t}\right|$$

The Python programming language was used to create the models, with the use of the Tensorflow, Keras, pmdarima, hyperopt, and StellarGraph machine learning libraries. Table 1 and 2 show the hyperparameters of the models.

TABLE 1. Model Configurations

Model		RNN	LSTM	GCN-LSTM
Layer Configuration	RNN	[48, 88]	N/A	N/A
	GCN	N/A	N/A	[10, 24]
	LSTM	N/A	[128, 56, 64]	[108, 104]
Activation Function		ReLU	ReLU	ReLU
Learning Rate		0.0282	0.0032	0.0042
Dropout Rate		0.0123	0.2858	0.2255
Batch Size		4	4	4
Optimizer		Adam	Adam	Adam
Look Back		9	2	7

TABLE 2. ARIMA Parameters

Parameter	Value
Order (*p, d, q*)	(0, 1, 1)
Seasonal Order (*P, D, Q, m*)	(0, 1, 0, 52)

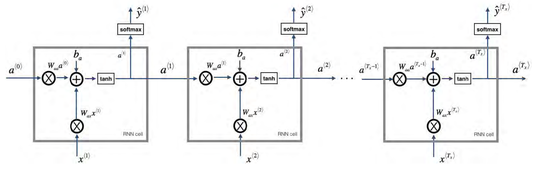

FIGURE 3. Diagram of an RNN

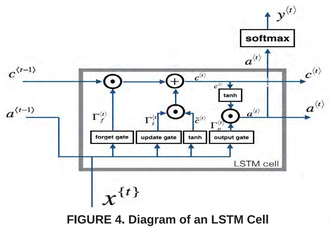

FIGURE 4. Diagram of an LSTM Cell

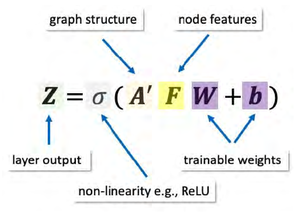

FIGURE 6. GCN Trainable Parameters

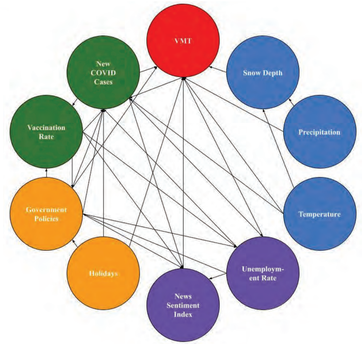

FIGURE 7. GCN Knowledge Graph Depiction

Results

Tables 3, 4, and 5 show the performance of the models. The GCN-LSTM model significantly outperformed the other models in terms of both evaluation metrics, improving on the persistence model results by 48.44% and 49.12% in terms of MAPE and RMSE, respectively.

All tested models improved on the benchmark persistence model. Surprisingly, the LSTM was outperformed by the RNN, with a 31.68% and 11.59% higher MAPE and RMSE. The ARIMA model, despite performing the worst out of the four tested models, still produced decent results, with a MAPE of 3.92% and a RMSE of approximately 3.86 million miles.

As shown in Figure 8, there is a sudden drop of VMT from the second week to the third week in the testing data, and then a sharp rise from the third week to the fourth week. Only the ARIMA and GCN-LSTM were able to accurately predict the drop, and only the ARIMA was able to forecast the subsequent rise.

TABLE 3. Model Performance

Model	Persistence		ARIMA		RNN		LSTM		GCN-LSTM	
Metric	RMSE (10^6 Miles)	MAPE	RMSE (10^6 Miles)	MAPE	RMSE (10^6 Miles)	MAPE	RMSE (10^6 Miles)	MAPE	RMSE (10^6 Miles)	MAPE
Performance	5.4317	5.28%	3.8612	3.92%	3.6978	3.26%	4.7324	3.90%	2.7637	2.72%

TABLE 4. Model Performance Improvement (MAPE)

Model	Persistence	ARIMA	RNN	LSTM	GCN-LSTM
Persistence	0.00%	-25.81%	-38.27%	-26.16%	-48.44%
ARIMA	34.79%	0.00%	-16.79%	-0.46%	-30.50%
RNN	61.99%	20.18%	0.00%	19.62%	-16.48%
LSTM	35.42%	0.46%	-16.40%	0.00%	-30.18%
GCN-LSTM	93.96%	43.89%	19.73%	43.23%	0.00%

TABLE 5. Model Performance Improvement (RMSE)

Model	Persistence	ARIMA	RNN	LSTM	GCN-LSTM
Persistence	0.00%	-28.91%	-31.92%	-12.87%	-49.12%
ARIMA	40.67%	0.00%	-4.23%	22.56%	-28.42%
RNN	46.89%	4.42%	0.00%	27.98%	-25.26%
LSTM	14.78%	-18.41%	-21.86%	0.00%	-41.60%
GCN-LSTM	96.53%	39.71%	33.80%	71.23%	0.00%

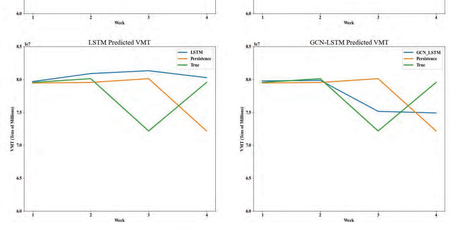

FIGURE 8. Model Predictions

Conclusion and Future Work

All tested models perform significantly better than the persistence model. Persistence models fail to accurately forecast future VMT due to the rapid developments of the pandemic, as well as the variation of weekly VMT in general. For example, the VMT of Salt Lake County dropped by 9.9% (7.9 million miles) from the second to third week of the test dataset. While this occurred near the Thanksgiving holiday, which may have influenced VMT, it suggests that it will be difficult to make long-term traffic forecasts during the pandemic.

Surprisingly, the RNN outperformed the LSTM, by 16.40% and 21.86% in terms of MAPE and RMSE, respectively. In theory, the opposite should be true, as the LSTM should be able to capture long-term dependencies much more accurately. However, the LSTM only used the past two weeks of data, while the RNN used the past nine weeks of data. Given that the GCN-LSTM also used the last seven weeks of data, perhaps the LSTM was not optimized well.

There is a sudden drop of VMT from the second to third week, then a sharp rise from the third to fourth week. Only the ARIMA and GCN-LSTM were able to accurately predict the drop, and only the ARIMA was able to forecast the subsequent rise. This suggests that the ARIMA model may also be a viable option for future VMT forecasting, as the prediction graphs show that it captures the changes in VMT much more accurately than the other models.

As expected, the GCN-LSTM performs desirably, with a MAPE of 2.72%. It performed significantly better than the traditional LSTM, demonstrating its ability to combine inter-variable relationships and incorporate human knowledge to produce more accurate results than traditional machine learning forecasting methods.

With certain data only available up to November 2021, this project was not able to take into account the recent developments of the pandemic, including the continuing Omicron variant, recent school closings, and the availability of a booster shot. Further research could utilize more recent data to improve the models.

Vehicle class data, such as Truck VMT, was not utilized in this project. Future work could investigate the trends in these data, and it might help improve the performance of the model as well.
In addition, the Gated Recurrent Unit (GRU), another common forecasting method, was not tested in this project, and could potentially perform well in this scenario.

While this model may not be extremely useful on a county-wide level, as it is too general, it could be applied to local freeway sections. Departments of transportation in the State of Utah and the United States could utilize the model to predict and plan for changes in traffic demand in the near future.