Delta Airlines Pilot Shortage LSTM Modeling

Samuel Yong
Georgia Institute of Technology
samuel6269@gmail.com

Abstract

"With the current business uncertainty it is important to know that we have the necessary resources to support scheduled flight plans." The Delta airlines business model relies on their ability to transport their customers safely and reliably. Their current motto is to "Keep Climbing" and one of the most important aspects of operating a fleet is resource planning. They rely on predictive analytics to forecast changes in flight crew availability and plan accordingly to fill in the gaps. This project evaluated the performance of an LSTM Model to fulfill Delta's service goals and resource planning.

1. Introduction/Background/Motivation

Delta Airlines services over 200 million customers per year with a staff of around 35,000 pilots and flight attendants. The company uses predictive modeling to forecast flight crew availability up to 3 days prior to the departing flight. On the 3rd day, demand and resource count is finalized and may no longer be changed. Inaccurate forecasting may lead to flight delays or even cancellations, severely impacting their large customer base and tarnishing Delta's reputation.

The task at hand was to build a predictive model for the available resources and the demand for pilots for each day of the year. Delta currently uses a Machine Learning Tree based model for their forecasting. Ultimately, the goal is to attempt to build a model that performs better than their existing technique. Obtaining more accurate forecasting will directly impact Delta and their customers. As the aviation industry recovers from the impact of COVID-19 and the demand for air travel continues to increase, any improvement in resource forecasting will prevent travel disruptions.

2. Approach

2.1. Overview and Data Processing

The power of Deep Learning models were investigated given the large amount of high dimensional data available. This allows for using end to end learning without performing much feature engineering, and creating a singular model for all different type of data sequences. Because of the sequential nature of the data, a recurrent based neural network was built to take advantage of learning from data that has a time scale component [2]. However, due to some of the limitations of an RNN model, an LSTM model was chosen for better gradient flow during backpropagation and sequential memory management. All components of this model was implemented using PyTorch tools and libraries. The first step in formulating the LSTM model was to format the input data in a scheme that PyTorch could read, manipulate, and back-propagate with. It required transforming the input data to be shaped into a 3D PyTorch Tensor with dimensions (N, L, D). N represents the batch size or the number of sequences or operational dates used for training in a batch. L represents the sequence length, or the days prior for each operational date. Lastly, D represents the number of additional features of the data. Each datapoint in the tensor represents a sequence with a specific operational date, fleet, and seat. These 3 categories are one hot encoded into the tensor. Lastly, the most difficult aspect of processing the data was to account for a variable sequence length among the datapoints. With some operational dates having more historic data than others, some of the datapoints had to be padded in order for the input tensor to have a consistent sequence length [1].

2.2. LSTM Block

After processing the input data into the correct tensor format, an LSTM block was created to learn useful sequential properties from the data. Figure 1 portrays a flow diagram of the LSTM block.

Within this block, a neural network layer is created for data belonging to each days prior. For example, given an operational date datapoint with data for 10 days prior to 5 days prior, all data for the 10th day prior is fed into the first layer. This first layer outputs a hidden state that is fed into the second layer, along with input data for the 9th day prior. This process repeats until the end with the 5th day prior layer and outputs a final hidden state that is fed into the next

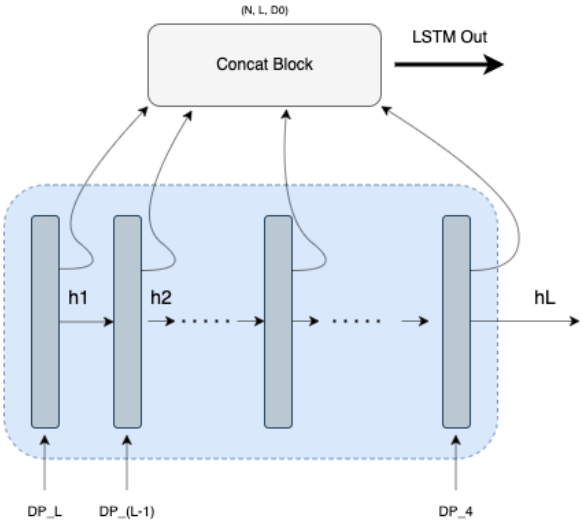

Figure 1. LSTM Neural Network Block

block for further processing. The LSTM block also has the option to keep track of each hidden state from each layer, and concatenate all the hidden states into a single output tensor. Using the PyTorch LSTM library, this block can be easily scaled to handle multiple datapoints with multiple dimensions simultaneously. One concept to note is that the layers within an LSTM block produce both a hidden and cell state that propagate from one layer to the next, but for simplification, further description of the hidden state also implies the cell state.

2.3. Overall LSTM Prediction Model

After creating the LSTM block, the last step in the model creation process is to leverage the power of other neural network layers for more accurate predictions by using the features and parameters of other different neural network layers. Experimentation of different type of Deep Learning network layers was done to see what networks aided the prediction process. After many trials, Figure 2 represents a diagram of the overall LSTM prediction model that was created.

The LSTM model first takes the input data, and transforms the data through a batch normalization layer. This normalizes the features of the input data before further processing to avoid any features having a heavier weight than the rest. The batch normalization layer also have learnable parameters that adjust the mean and standard deviation of each feature during normalization based on the overall loss function. After batch normalization, the data is fed into consecutive LSTM blocks where the hidden state of one block is fed as an input into the next LSTM block and so on. The number of consecutive LSTM blocks is a hyperparameter that can be tuned. After the main LSTM blocks, the last LSTM block will concatenate all the layer outputs seen in Figure 1 and feed this, along with the final hidden state into a linear layer. The reason why both the concatenated outputs and the last hidden layer is fed into the the next linear layer is because the LSTM block only creates predictions for all prediction dates up to 4 days prior. However, to generate a 3 day prior prediction, we need to incorporate an additional slot in the data tensor in order to compare our predictions for all snap dates plus the final 3 day prior prediction to our data. Figure 2 shows the 2nd dimension of the data tensor increase by 1 after the LSTM and linear layer. After this, another batch normalization and ReLU layers were added to incorporate some non-linearity advantages, and finally, two more linear feed-forward layers were added to make a supply and demand prediction for each prediction date plus one for the final 3 days prior. Figure 2 also lists out the output dimension of each layer. It can be seen that a starting tensor with shape (N, L, D0) is transformed into (N, N+1, D1), (N, N+1, D2), and then eventually transformed into a final prediction of shape (N, L+1, 1). D0 represents the original number of features from the raw data, and D1 and D2 are hyperparmameters that represent embedding size. After the batched input data is fed forward through the entire model, a loss result is calculated based on the prediction and the ground truth label given by the raw data. A loss function then optimizes this loss value by backpropagating this loss during training with the goal of minimizing this loss. Lastly, this model was created to make predictions for both pilot supply and demand simultaneously. However, after experimentation, better performance was seen by training this model for each one of this response features separately and training two different models with the same architecture.

3. Experiments and Results

The main goal of this project is to generate predictive models that can predict the pilot supply and demand count more accurately compared to the baseline benchmark calculated from the raw data. The baseline benchmark is calculated using the MAE metric. All models are trained using operational date data before March 1st, 2022, and accuracy is determined using a testing data set with operational date data that happens after this date.

Given the baseline metric used a MAE calculation, an L1 loss function was utilized during training in order for the model to be optimized based on an absolute error calculation. Table 1 illustrates the final optimal hyperparameters based on testing data results. The L1 loss curve for the supply prediction loss and the demand prediction loss during training can be seen in Figure 3 and Figure 4 respectively.

The prediction MAE for a specific operational date is calculated by taking all the previous predictions plus the final

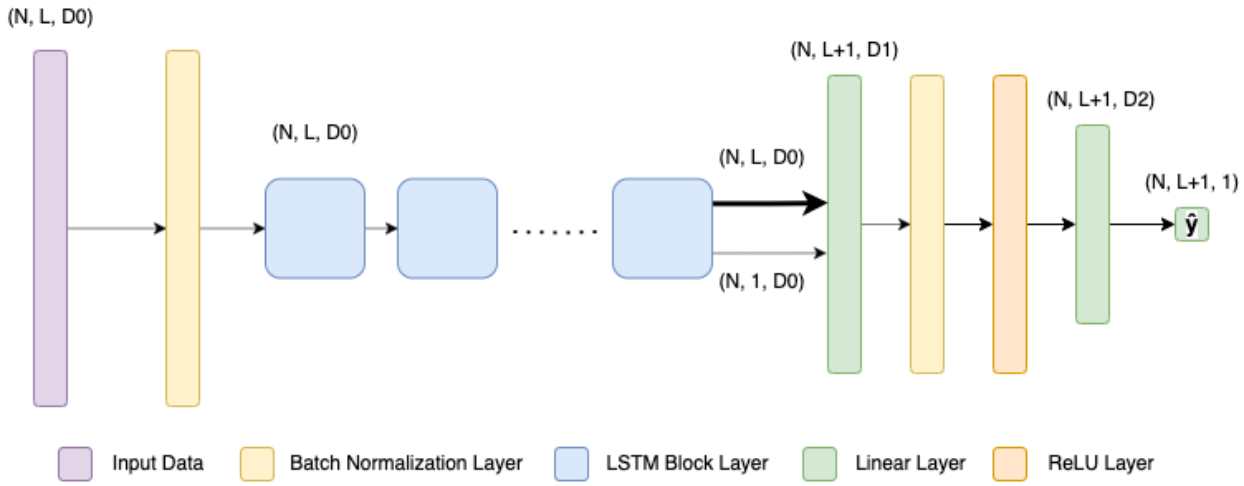

Figure 2. Overall LSTM Prediction Model

Hyperparameter	Optimal Value
Epochs	25
Learning Rate	0.05
Batch Size	32
LSTM Embedding Size	8
Linear Layer 1 Embedding Size	6
Linear Layer 2 Embedding Size	4

Table 1. LSTM Optimal Hyperparameters

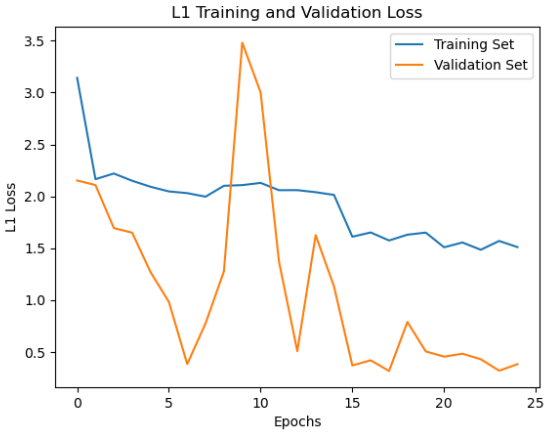

Figure 3. LSTM Demand Count Model Loss Curve

3 days prior prediction and calculating the absolute error for each one of these predictions when compared to the ground truth label in the data. Then the MAE is calculated by taking the mean of all these absolute errors. The final prediction MAE is then calculated as the average of all the operational date MAEs in the testing dataset.

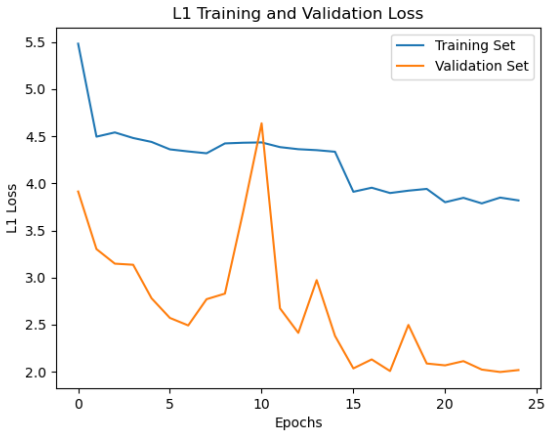

Figure 4. LSTM Supply Count Model Loss Curve

4. Concluding Remarks

The LSTM model was able to take into consideration all the features included in the data set as well as process the data more efficiently compared to the baseline model. This could be due to the difference in architecture in which the LSTM model involves training over a much larger number of parameters across its many layers. More importantly, the LSTM model had a higher prediction accuracy for any given operational date across all airline fleet types compared to the baseline regression model when using the MAE calculation. In conclusion, a LSTM prediction model proved to predict the pilot availability of a specific flight more accurately than a regression based forecast prediction given the size and type of the data available, and important features of the LSTM block.

References

[1] Jason Brownlee. Data preparation for variable length input sequences. https://machinelearningmastery.com/data-preparation-variable-length-input-sequences-sequence-prediction/, 2019. 1

[2] Charlie O'Neill. Pytorch lstms for time-series data. https://towardsdatascience.com/pytorch-lstms-for-time-series-data-cd16190929d7, 2022. 1