CS230

Predicting Delays in Flight Departure Time at SFO

Yanqiu Wang
Stanford University
yanqiu@stanford.edu

Yutong Sun
Stanford University
ycsun@stanford.edu

Abstract

As flying becomes a widely-adopted way of transportation, flight delays also begin to rise as one of the prominent problems for travellers. In this study, we explored applying deep learning to predict flight departure time delays at San Francisco International Airport with information such as departure time and weather conditions. Three models were implemented – a simple neural network, an LSTM, and a CNN. All models reached reasonable accuracy and F-scores, among which LSTM performed the best across all metrics. Most importantly, we found that a CNN – an architecture rarely valued for solving this problem – that takes in previous flight information performs just as well as a LSTM network, which is usually applied to predict flight delays.

1 Introduction

In the United States, more than 44,000 flights are serviced for 2.7 million passengers every single day [1]. San Francisco International Airport (SFO) carries more than 55.8 million passengers per year, a number that has continued to increase since 2011 [2]. With this growing number of flights and airplane travellers, flight delays have also become a common problem. U.S. Department of Transportation admits that it is "difficult for an airline to estimate how long a delay will be" ahead of time [3]. With the rapid development of deep learning recently, we were interested in seeing how different deep learning architectures could potentially help solve the problem of predicting flight delays. For both the simple NN model, we input 12 features relating to each flight such as time of the day and weather conditions, and outputs a delay class predicted by the model. For the LSTM and CNN, there are 13 input features for each flight (with the addition of delay class of previous flight/flights) and one output of the delay class predicted for the current flight of interest. For detailed description on input features and output classes, please refer to Section 3.

2 Related work

Most of the existing research done surrounding prediction of flight delays have focused on solving the binary classification problem of whether or not a flight will be delayed [4, 5, 6] where information on the current and previous flights is used to generate a prediction on whether the flight will be delayed for more than 15 minutes. Some have explored using the binary classes of "high delay" versus "low delay" [7] based on flight and weather information. Recognizing how a delayed prior flight influences the delay status of the next flight, many researchers have focused on using an LSTM in predicting delays in an effort to capture the time-series nature of the data [4], whereas others have explored using an artificial neural network [5, 7] or a random forest algorithm [8, 9]. While most algorithms were able to reach a high accuracy, they were unable to predict a value for the amount of delay. Our work aims to add onto the existing research by taking into account a comprehensive feature list that includes both flight information and weather data, and outputs a multi-class classification indicating

CS230: Deep Learning, Winter 2018, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

the approximate amount of time that a flight departure might be delayed. Additionally, we propose using a CNN architecture that is able to capture the time-dependency of the flight departure delays.

3 Dataset and Features

Our first dataset is the on-time performance of flights in the United States by the Bureau of Transportation Statistics [11], containing information such as flight date, flight number, departure airport, departure delay time, and much more for flights within the United States from 1987 until today. The second dataset is the hourly weather dataset for several major cities around the world between Janurary 2012 and October 2017 [12]. It includes weather information such as humidity, pressure, weather condition, wind speed, etc. After merging the two datasets with R, we extracted the corresponding entries for the months of January, March, May, July, August, October, and December of 2016 as representative data points. We split the 100k examples randomly into train, dev, and test sets by an approximate 90/5/5 ratio (see Figure 4 for exact numbers).

3.1 Data Preprocessing

We synthesized our data using R, where each entry corresponds to a single flight with all relevant information. To do so, we matched every flight with the hourly weather data of its scheduled take-off time. For example, a flight that was scheduled to take off at 6:35 am on January 2nd is matched with weather features of San Francisco labeled "1/2/2017 6:00 - 7:00" in the first dataset. With this processing, each set contains 13 variables: 6 weather features (humidity, temperature, wind speed, wind direction, weather description, and pressure) and 6 flight features (distance of flight, scheduled departure time, destination, day of month, day of week, flight number), and a final "Delay_Group", which classifies each flight into one of the 15 delay intervals (every 15 minutes from <15 minutes to >180 minutes). For the simple NN, each training example represents one flight entry with 12 features (all corresponding variables except for delay group). For the LSTM model, each flight's input is the 13 features of its previous flight, including delay group. For the CNN model, each training example is represented by all 13 variables of the 5 previous flights. All three models use the delay group of the current flight as the ground truth label.

4 Methods

4.1 Simple Neural Network

We first built a simple neural network as a baseline for our problem. The inputs are 12-feature vectors for each flight. We used two hidden layers and a final output layer with softmax activation to make predictions about one of the fifteen delay classes. Here is the detailed architecture:

Layer	Parameters	Activation
Hidden Layer 1	W = (25,12) b = (25,1)	ReLU
Hidden Layer 2	W = (12, 25) b = (12, 1)	ReLU
Output Layer	W = (25, 12) b = (25, 1)	Softmax

Figure 1: Architecture of Simple Neural Network

4.2 LSTM Model

We then implemented an LSTM model in order to capture the time-dependency of the flight delays. Inputs are 12-feature vectors for each flight and outputs a prediction about one of the fifteen delay classes (same as the simple NN). In training, we used a combination of LSTM cells, Dropout, BatchNorm and Dense layers in our model. Using this architecture, at each point of data processing, we are also utilizing the predictions of delay group made for the previous flight as an additional input, which is then able to capture the time-dependency nature. Here is the detail of the architecture:

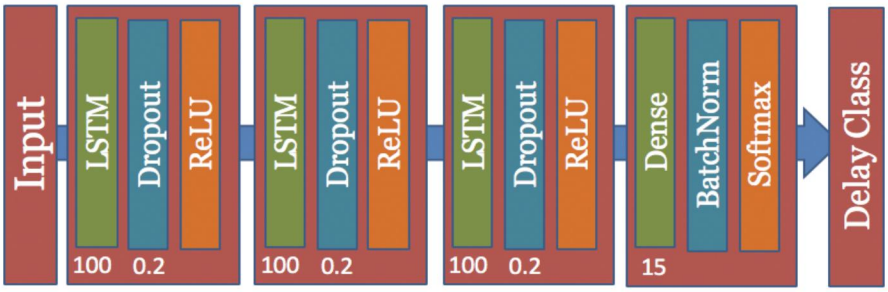

Figure 2: Architecture of the LSTM Model

4.3 CNN Model

We explored with various ways to implement a CNN to solve this problem. We initially considered implementing a 1-D CNN, but then decided to try and capture the time-dependency by using a 2-D CNN structure and reformatting our inputs. For this model, each flight is represented by the features of the five preceding flights. The output variable is the delay class of the current flight of interest. The following figure aims to clarify this structure:

MONTH	DAY OF MONTH	DAY OF WEEK	CARRIER FL NUM	DEST STATE FIPS	CRS DEP TIME	DISTANCE	Temperature	Wind Speed	Wind Direction	Humidity	Pressure	DEP DELAY GROUP
Input Feature												
1	1	5	408	51	754	2419	275.917434	3	11	75	1026	0
1	1	5	1798	41	758	550	275.917434	3	11	75	1026	-1
1	1	5	5268	30	759	807	275.917434	3	11	75	1026	0
1	1	5	5392	6	800	363	275.917434	3	11	75	1026	4
1	1	5	174	34	800	2565	275.917434	3	11	75	1026	2
1	1	5	1930	6	800	337	275.917434	3	11	75	1026	1
												Ground Truth

Figure 3: Data Manipulation for CNN Model

In training, we used a combination of convolutional, pooling, and fully connected layers, with a final output layer using a softmax activation to output a classification for the delay group. For each of the Conv layers, we use a filter size of (5, 5) so that it is a horizontally-sliding window. We believe this architecture helps capture the time dependency of delay since it takes in information from the past five flights in predicting the current flight delay group.

Layer	f	s	n	p	Activation
CONV	5	1	4	same	ReLU
MaxPool	5	1	/	same	/
CONV	5	1	8	same	ReLU
MaxPool	5	1	/	same	/
CONV	5	2	16	same	ReLU
CONV	5	3	32	same	ReLU
CONV	5	4	64	same	ReLU
FC	/	/	15	/	Softmax

Figure 3: Architecture of the CNN Model

4.4 Loss Function

All three models used the same loss function: the softmax cross-entropy loss. Because this is a multi-class classification problem, we decided to use this standard loss function that usually used to optimize for such problems.

$$\text{L} = -\sum_i y_i * \log(p_i), \text{ where}$$
$$p_i = \frac{e_i^a}{\sum_{k=1}^{8} e_k^a}.$$

5 Results & Discussion

5.1 Evaluation Metric

We used three metrics to evaluate our models and results: Multi-class Accuracy (15 classes), Binary Accuracy (delayed > 15 minutes or not), and Binary F1 Score. The formula of these metrics are shown below (Note: for binary metrics, "positive" is defined as "the flight is not delayed (i.e. delay time < 15min as deemed by Federal Aviation Administration [13])", and vice versa):

$$\text{Multi-class Accuracy} = \frac{\text{Number of Correct Prediction of Class}}{\text{Total Number of Examples}}$$
$$\text{Binary Accuracy} = \frac{\text{True Positive + True Negative}}{\text{Total Number of Examples}}$$
$$\text{Binary Precision} = \frac{\text{True Positive}}{\text{True Positive + False Positive}}$$
$$\text{Binary Recall} = \frac{\text{True Positive}}{\text{True Positive + False Negative}}$$
$$\text{Binary F1 Score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

The Multi-class Accuracy shows the percentage of correct classification of flights into the 15 delay categories, a direct measurement of the performance on our multi-class classification task. The binary metrics capture the concern of travellers: "will my flight be delayed or not?" This reasoning behind setting "not delayed" as positive is to optimize in our fine tune process in the direction of maximizing the probability of a not delayed flight labeled as not delayed. We believe labelling a not-delayed flight as delayed is more harmful (traveller may get to the airport later thinking it's going to be delayed) than labelling a delayed-flight as not delayed.

5.2 Hyperparameter Tuning

One hyperparameter we focused on was the number of layers to include in each model. For all three models, we explored using more number of layers than what we eventually decided on. For example, we experimented with using more Dense layers in the LSTM model but found that it did not increase the accuracy but instead took longer to train, and thus settled on a model with only one dense layer before the final activation. Similarly, we tried utilizing more CONV and POOL layers than our final model for the CNN, but found no improvement on accuracy.

Additionally for the CNN model, although we designed the filter sizes to be 5x5, we experimented with different stride sizes to find a balance. We initially set all stride sizes to 1 as we were wary of a larger stride size losing important information in our data. However, after increasing stride sizes in the later CONV layers, we found that little accuracy was sacrificed for a much faster training time. Therefore, we decided to increase the stride sizes in later parts of the model. For the number of filters at each layer, we were inspired by the VGG-16 model [10] where the number of channels is continually doubled by using more filters at each CONV layer.

To prevent over-fitting the training set, we kept track of the training cost and test cost for every 10 epochs. When training each model for the first time, we set the epoch (iteration) size to be 200 and looked for the instance where test-set cost starts to rise. For the simple-neural network, the threshold was 50 epochs, so we used this critical value as our new epoch size and re-trained the NN model to obtained the results. For both the CNN and LSTM models, the losses converged at the 60th epochs. Since no over-fitting was observed before these two thresholds, we chose them to be the epoch sizes and trained the models again for analysis.

We also experimented with other hyperparameters such as mini-batch size, learning rate and number of neurons in different layers of the models. The current version of our models give the highest accuracy without taking up unreasonable amount of computational power with our tuned hyperparameters.

5.3 Results

Model	Set	Accuracy (Delay Class)	Accuracy (Binary)	F1 (Binary)
Simple NN	Train m = 88616	45.11%	84.94%	91.80%
	Test m = 6129	36.63%	78.00%	87.50%
LSTM	Train m = 88614	**64.09%**	**89.99%**	**94.93%**
	Test m = 6127	**56.70%**	**83.23%**	**90.50%**
CNN	Train m = 88610	52.71%	89.83%	94.70%
	Test m = 6123	43.75%	81.64%	90.49%

Figure 4: Evaluation Metrics of Simple NN, LSTM, and CNN Models

Overall, the LSTM performed the best in all three of our evaluation metrics, and, as expected, the simple NN performed the worst in all three. Zooming in on the performance of both the LSTM and CNN models, however, there are interesting results. Both models perform exceptionally well on the binary task – both achieving an accuracy above 80% and F1 above 90% on the test set. Although LSTM does perform slightly better, the difference is almost negligible.

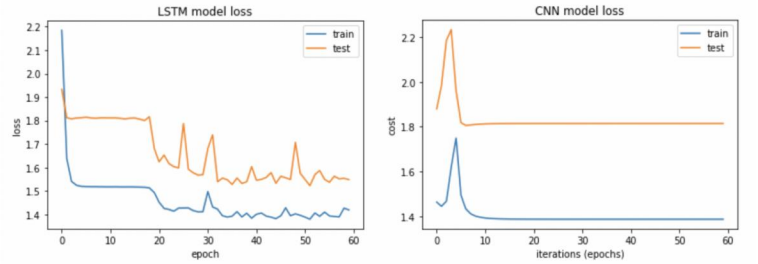

Figure 5: Loss of the LSTM model (left) and the CNN (model) during training

Looking at the multi-class classification task, LSTM performs significantly better than the CNN. Through the loss diagram during training, we found that the LSTM hits a plateau leading up to 18 epochs (test loss ≈ 1.82), but after running for longer, it is able to break this plateau and eventually reach a much lower test cost of around 1.52. On the other hand, the CNN model seems to quickly hit a plateau (test loss ≈ 1.81) and cannot break out of it (even after we ran it for 1000 epochs with more added layers). Further, when we early-stop the LSTM model at the first plateau, we find its performance accuracy to be extremely similar to that of our CNN model. Thus, we hypothesize that our CNN model is not robust enough to break out of the plateau which our LSTM model can, thus resulting in the difference in performance.

6 Conclusion/Future Work

Our result shows that the LSTM model outperforms the other two with a multi-class accuracy of 56.70%, a binary accuracy of 83.23%, and a F1 score of 90.53% (on the test set). The CNN model performed almost as well as the LSTM on the *binary* classification task with an accuracy of 81.64% and F1 Score of 90.49%. We believe both the CNN and LSTM models are able to tackle the binary task sufficiently well by capturing the time dependency, but the LSTM model performs significantly better on the harder task of multi-class classification. It is important to note that all three algorithms, including the simple NN, were able to predict the delay class at reasonable accuracy.

For future work, it would be interesting to explore whether with more computational power and a much deeper model, a CNN architecture would be able to break the plateau and eventually converge to a loss that is similar to the current LSTM model in the multi-class classification, although practically such a model may not be best-fitted for real-world application due to its large size. Furthermore, a deeper LSTM model could be trained for longer with more computational power to see whether it may break the current plateau that it converges to and reach an even higher multi-class accuracy.

7 Contributions

The authors worked together through every step of this project. During the implementation phase, both authors contributed to designing, implementing, and tuning the model architectures. After obtaining the results, the authors collaborated on analyzing, visualizing, and summarizing the results.

We would like to thank Weini Yu for her continued support throughout the quarter and Ashwin Sreenivas for his helpful guidance during office hours at the very early stages of this project.

Our code is on GitHub at: https://github.com/cocosun1/CS230-Project

References

[1] FlySFO | San Francisco International Airport. (2019). SFO Served an All-Time Record 55.8 Million Passengers in 2017 | San Francisco International Airport. [online] Available at: https://www.flysfo.com/media/press-releases/sfo-served-all-time-record-558-million-passengers-2017 [Accessed 9 Jun. 2019].

[2] US Department of Transportation. (2019). Flight Delays & Cancellations. [online] Available at: https://www.transportation.gov/individuals/aviation-consumer-protection/flight-delays-cancellations [Accessed 9 Jun. 2019].

[3] Faa.gov. (2019). Air Traffic By The Numbers. [online] Available at: https://www.faa.gov/air_traffic/by_the_numbers/ [Accessed 9 Jun. 2019].

[4] Y. J. Kim, S. Choi, S. Briceno and D. Mavris, "A deep learning approach to flight delay prediction," 2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC), Sacramento, CA, 2016, pp. 1-6. doi: 10.1109/DASC.2016.7778092

[5] Gopalakrishnan, Karthik and Hamsa Balakrishnan. "A Comparative Analysis of Models for Predicting Delays in Air Traffic Networks." Air Traffic Management Research and Development Seminar, June 2017, Seattle, Washington, USA, ATM Seminar, June 2017 © 2017 ATM Seminar

[6] Horiguchi, Yuji, Baba, Yukino, Kashima, Hisashi, Suzuki, Masahito, Kayahara, Hiroki, AND Maeno, Jun. "Predicting Fuel Consumption and Flight Delays for Low-Cost Airlines" Innovative Applications of Artificial Intelligence (2017): n. pag. Web. 8 Jun. 2019

[7] Rebollo, Juan Jose, and Hamsa Balakrishnan. "Characterization and prediction of air traffic delays." Transportation research part C: Emerging technologies 44 (2014): 231-241.

[8] Takeichi, Noboru, et al. "Prediction of delay due to air traffic control by machine learning." AIAA Modeling and Simulation Technologies Conference. 2017.

[9] Choi, Sun, et al. "Prediction of weather-induced airline delays based on machine learning algorithms." 2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC). IEEE, 2016.

[10] Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556 (2014).

[11] "On-Time : Reporting Carrier On-Time Performance (1987-Present)." Bureau of Transoportation Statistics , www.transtats.bts.gov/DL_SelectFields.asp?Table_ID=236&DB_Short_Name=On-Time.

[12] Gene, Selfish. "Historical Hourly Weather Data 2012-2017." Kaggle, 28 Dec. 2017, www.kaggle.com/selfishgene/historical-hourly-weather-data#wind_speed.csv.

[13] Defining and Measuring Aircraft Delay and Airport Capacity Thresholds. Transportation Research Board, 2014.