A Deep Learning Approach to Flight Delay Prediction

Young Jin Kim, Sun Choi, Simon Briceno and Dimitri Mavris
Aerospace Systems Design Laboratory
Georgia Institute of Technology
Atlanta, Georgia 30332–0150

***Abstract*—Deep learning has achieved significant improvement in various machine learning tasks including image recognition, speech recognition, machine translation and etc. Inspired by the huge success of the paradigm, there have been lots of tries to apply deep learning algorithms to data analytics problems with big data including traffic flow prediction. However, there has been no attempt to apply the deep learning algorithms to the analysis of air traffic data. This paper investigates the effectiveness of the deep learning models in the air traffic delay prediction tasks. By combining multiple models based on the deep learning paradigm, an accurate and robust prediction model has been built which enables an elaborate analysis of the patterns in air traffic delays. In particular, Recurrent Neural Networks (RNN) has shown its great accuracy in modeling sequential data. Day-to-day sequences of the departure and arrival flight delays of an individual airport have been modeled by the Long Short-Term Memory RNN architecture. It has been shown that the accuracy of RNN improves with deeper architectures. In this study, four different ways of building deep RNN architecture are also discussed. Finally, the accuracy of the proposed prediction model was measured, analyzed and compared with previous prediction methods. It shows best accuracy compared with all other methods.**

I. Introduction

Flight delays in the National Airspace System (NAS) lead to a significant amount of costs according to a previous study [1]. In 2007, this accounted for approximately $33 billion as direct or indirect cost to passengers, airlines and other parts of the NAS. In order to reduce the wasted costs, various studies have been performed for the analysis and prediction of air traffic delays [2], [3], [4]. Based on the analysis, more efficient and mitigating air traffic management strategies could be established.

Many of the previous analyses are relying on modeling and simulation techniques. By defining a model emulating the actual behaviors of the components in the system, scenarios which need to be analyzed are regenerated via computer simulations. This simulation-based approach is valuable especially when we need to find interactions among the components and to analyze far future scenarios. The weaknesses of the simulation-based analyses are usually the slow speed of the simulation and possible inappropriate modeling assumptions. There have been several studies to improve the speed of simulation [5], [6], however it is still hard to select an appropriate level of abstractions for the quality of the analyses.

On the other hand, another group of analyses using data analytics and statistical machine learning has arisen motivated by the success of their techniques in many fields. Tu *et al.* [7] analyzed long-term and short-term patterns in air traffic delays using statistical methods. Xu *et al.* [8] proposed a Bayesian network approach to estimating delay propagation. Rebollo *et al.* [9] used machine learning techniques with air traffic network characteristics to predict air traffic delays. Choi *et al.* [10] proposed a machine learning model combined with weather data. However, there is still room for improvement in the accuracy.

In the meantime, deep learning paradigm which was inspired by the hierarchical structure of human perception has been widespread. Deep learning improves the accuracy of the classification and regression dramatically in many machine learning tasks such as image recognition, speech recognition, machine translation and etc [11], [12]. Furthermore, it is now utilized for the ground traffic flow prediction [13]. Especially, considering the current improvements of deep learning algorithms, it is meaningful to evaluate the applicability and the performance of a deep learning architecture for the flight delay prediction which is one of the air traffic data analytics applications.

There exists many deep learning architectures including stacked autoencoders, convolutional neural networks and recurrent neural networks. In this research, recurrent neural networks was selected as the architecture for the day-to-day delay status prediction task because it captured sequential and temporal relationships existing in the data. Intuitively, delay states of previous day's flights affect subsequent days' flight delays. Section II explains the deep learning algorithms used in this study and Section III explains the architecture of the networks trained in the study. Section IV presents the experiment results using the deep learning model and the conclusion is given in Section V.

II. Deep Recurrent Neural Networks

Recurrent Neural Networks (RNN) is an artificial neural networks that models the behaviors of dynamic systems using hidden states. Long Short-Term Memory (LSTM) networks is one kind of implementation of RNN architectures which is faster and more accurate than standard RNNs. In this section, general architectures of RNN and LSTM networks are explained. Then, the benefits of stacking these networks are discussed and the ways to make an architecture deeper using RNN are also discussed.

978-1-5090-2523-7/16/$31.00 ©2016 IEEE

A. Recurrent Neural Networks

Given an input sequence $x = (x_1, x_2, \ldots, x_k, \ldots, x_T)$, RNN computes the evolution of hidden states $h = (h_1, h_2, \ldots, h_k, \ldots, h_T)$ and output sequence $y = (y_1, y_2, \ldots, y_k, \ldots, y_T)$. This computation is performed iteratively solving the following equations for the time span from $t = 1$ to T. Here, x_k, h_k and y_k can be any arbitrary sized vectors which are represented as the dimension of input space, hidden space and output space.

$$h_t = \phi_h \left(W_{hh} h_{t-1} + W_{xh} x_t + b_h \right) \tag{1}$$

$$y_t = \phi_0 \left(W_{hy} h_t + b_y \right) \tag{2}$$

where W_{hh} denotes the weight matrix for the transition of hidden states from the previous time step to the current time step, W_{xh} denotes the weight matrix for the input to hidden layer and W_{hy} denotes the weight matrix for the hidden layer to output. b_h and b_y are capturing biases for each equation. ϕ_h and ϕ_o are activation functions for hidden states and output, respectively [14]. For theses activation functions, a saturating nonlinear function such as a logistic sigmoid function or a hyperbolic tangent function is applied element-wisely to the given vector usually.

B. LSTM

The LSTM architecture uses memory cells which will replace ϕ_h and ϕ_o of standard RNN architecture to store hidden layer information and it shows better performance for the sequence of long range than conventional RNN architectures. In this research, the LSTM memory cell proposed by Alex Graves *et al.* [15] was used. This single memory cell is repeated for the recurrence of the model. It has input gate(i), forget gate(f), output gate(o) and cell activation vectors(c), all of which are the same size as the hidden vector h. The following equations represent the computations of the model:

$$i_t = \phi \left(W_{xi} x_t + W_{ht} h_{t-1} + W_{ci} c_{t-1} + b_i \right) \tag{3}$$

$$f_t = \phi \left(W_{xf} x_t + W_{hf} h_{t-1} + W_{cf} c_{t-1} + b_f \right) \tag{4}$$

$$c_t = f_t c_{t-1} + i_t \tanh \left(W_{xc} x_t + W_{hc} h_{t-1} + b_c \right) \tag{5}$$

$$o_t = \phi \left(W_{xo} x_t + W_{ho} h_{t-1} + W_{co} c_t + b_o \right) \tag{6}$$

$$h_t = o_t \tanh \left(c_t \right) \tag{7}$$

where ϕ is the logistic sigmoid function. The cell architecture of LSTM module is illustrated in Fig. 1.

C. Deep architecture of RNN

From the past studies [15], [16], it has been shown that a deep and hierarchical model can be more efficient and accurate at representing some functions than a shallow one. Inspired by this hypothesis, a deep architecture of model is designed for the task of flight delay prediction. In order to make a RNN model deep, there exist four different ways which are deep input-to-hidden, deep hidden-to-output, deep hidden-to-hidden transition and stacks of hidden states. Each one strengthens the model in a different manner. First of all, the deep input-to-hidden architecture has the effect of non-linear dimensionality reduction which will reveal the underlying factors of variation from the original input. The deep hidden-to-output architecture may be useful to disentangle the factors of variations in the hidden state, making it easier to predict the output. The deep hidden-to-hidden transition architecture allows the RNN to learn a highly nonlinear and non-trivial transition between the consecutive hidden states. Lastly, the stack of hidden states enables a model to capture state transitions of different timescales.

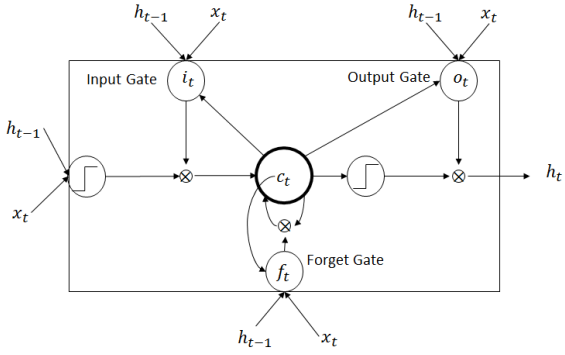

Fig. 1. Long Short-term Memory Cell [15]

For this research, deep input-to-hidden function, the deep hidden-to-output function and stacked RNN are applied. They are also illustrated in Fig. 3. In the case of the deep hidden-to-hidden transition, it is not used here because it can be seen as a duplication of LSTM. The non-linearity of the transition is already covered by LSTM architecture. The equations for deep input-to-hidden and deep hidden-to-output transitions just add more affine layers and non-linear transformation layers, thus the formulations are left out of this paper. The mathematical formulation of the stacked RNN is as follows:

$$h_t^{(l)} = f_h^{(l)} \left(h_t^{(l-1)}, h_{t-1}^{(l)} \right) = \phi_h \left(W_l h_{t-1}^{(l)} + U_l h_t^{(l-1)} \right) \tag{8}$$

where $h_t^{(l)}$ is the hidden state of the l-th level at time t. When $l = 1$, the state is computed using x_t instead of $h_t^{(l-1)}$. The hidden states of all the levels are recursively computed from the bottom level $l = 1$.

III. Network Training

The proposed model has a two-stage approach. The first stage is to predict daily delay status using deep RNN. The next stage is to predict delays of individual flights using daily delay status which is the output from the first stage, historical on-time performance data and weather data. For the training of the model, historical on-time performance data of the commercial airline flights and historical weather data for the ten major airports in the U.S. have been collected. Then, the historical data was grouped by airports so that the day-to-day sequence of arriving and departing flights at a specific airport can be fed into the first stage of the model. By computing hidden states sequentially, the delay status of subsequent days is predicted

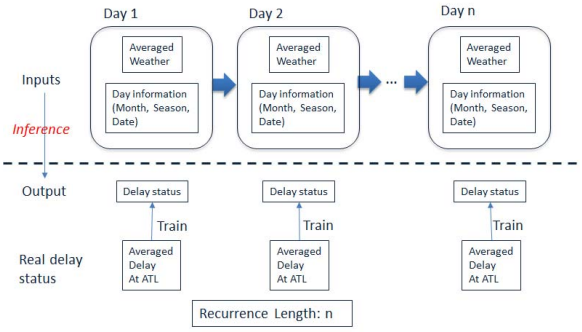

Fig. 2. Day-to-Day departure delay status model

as an output. For the second stage, the daily status, which comes from the first stage, is used as an input to a model to predict the delays of individual flights. The details of the actual network configuration and the methods used for the network training are described in this section.

A. Day-to-day delay status model

The purpose of the first stage is to get a day-to-day delay status model. From the Transtats database of U.S. Department of Transportation [17], on-time performance data of commercial airline flights are collected. Including the flight schedule, origin airport and destination airport, all the available data attributes are collected from the database. Table I shows the details of the flight data used. All of the departure delays and all of the arrival delays for each single day are averaged, respectively. The averaged values are used for representing the delay status of one single day. The binary status which is either not-delayed or delayed is acquired by applying threshold value to the averaged delay value. Several different threshold values are tested to analyze the most effective threshold value.

We presumed that the weather conditions at the origin and the destination airports were important factors for the prediction task. Therefore, all the weather data related to the flight data were gathered from the Integrated Surface Database (ISD) of National Oceanic and Atmospheric Administration (NOAA) [18]. Similar to the historical flight data, all the available data attributes are collected from the weather database. Then, the weather data for a day is averaged. For both flight data and weather data, there is no pre-filtering for the available data attributes. By using the deep input-to-hidden architecture, it is expected that the most important features of the model are extracted automatically. The list of the weather attributes selected for the prediction task is shown in Table I. Based on the flight and weather input data, the class of delay is computed as an output. Then, this classification is repeated for the consequent days at an airport. In the sequence of the flight statuses, the delay of the preceding days will affect the delay of the following days, which make the recurrent relationships. Using an example of the departure delay sequence of the Atlanta airport, the concept for the model is illustrated in Fig. 2.

TABLE I
INPUTS AND OUTPUTS OF THE DAY-TO-DAY DELAY STATUS MODEL

Airports	ATL, LAX, ORD, DFW, DEN, JFK, SFO, CLT, LAS, PHX
Time period	Jan. 2010 - Aug. 2015
Attributes of (Input variables) Flight data	Day of week, Season, Month, Date
Attributes of (Input variables) Weather data	Wind direction, Wind speed, Cloud height, Visibility, Precipitation, Snow Accumulation, Intensity, Descriptor, Observation Code (Daily average)
Classification (Output variable)	Class of delay with different threshold values (15 minutes, 30 minutes) from averaged departure and arrival delay data

B. Deep architecture for the day-to-day delay status RNN model

In order to learn the sequential nature of the air traffic flight delays correctly, deep architectures described in the previous section are utilized. Deep input-to-hidden functions, deep hidden-to-output functions and stacked RNN architectures are merged into the designed model. The architecture of the network is illustrated in Fig. 3.

C. Individual flight delay model

Once, a delay status of one day is acquired, it is fed into the second stage model. The second stage is a layered neural networks (NN) model to compute a delay class of one specific flight using given delay status of the day of flight and historical delay class with historical weather data. For each depth, hyperbolic tangent function ($Tanh$) nonlinear transformation layer is followed by a fully connected linear layer. At the final depth, a logistic sigmoid function is used instead of $Tanh$ because the final output should be a binary class which is 0 and 1. The inputs and outputs of the model are summarized in Table II. The networks built for this stage is also illustrated in Fig. 4. The number of layers and the number of nodes in each layer of the NN model can vary. Impacts of those numbers will be discussed in Section IV.

D. Regularization

One of the most important issues that needs to be handled appropriately is how to prevent overfitting of the both day-to-day delay status and individual delay prediction neural networks models. As the complexity of the model increases with a large depth, the model is more prone to overfitting. It results in a serious degradation of the accuracy of the model. The dropout technique was used in this research proposed by Hinton *el al*. [19]. It has been proved that dropout enhances the accuracy of the deep learning model by randomly dropping units (along with their connections) from the neural networks during training [20]. As a result of the random drop, dropout

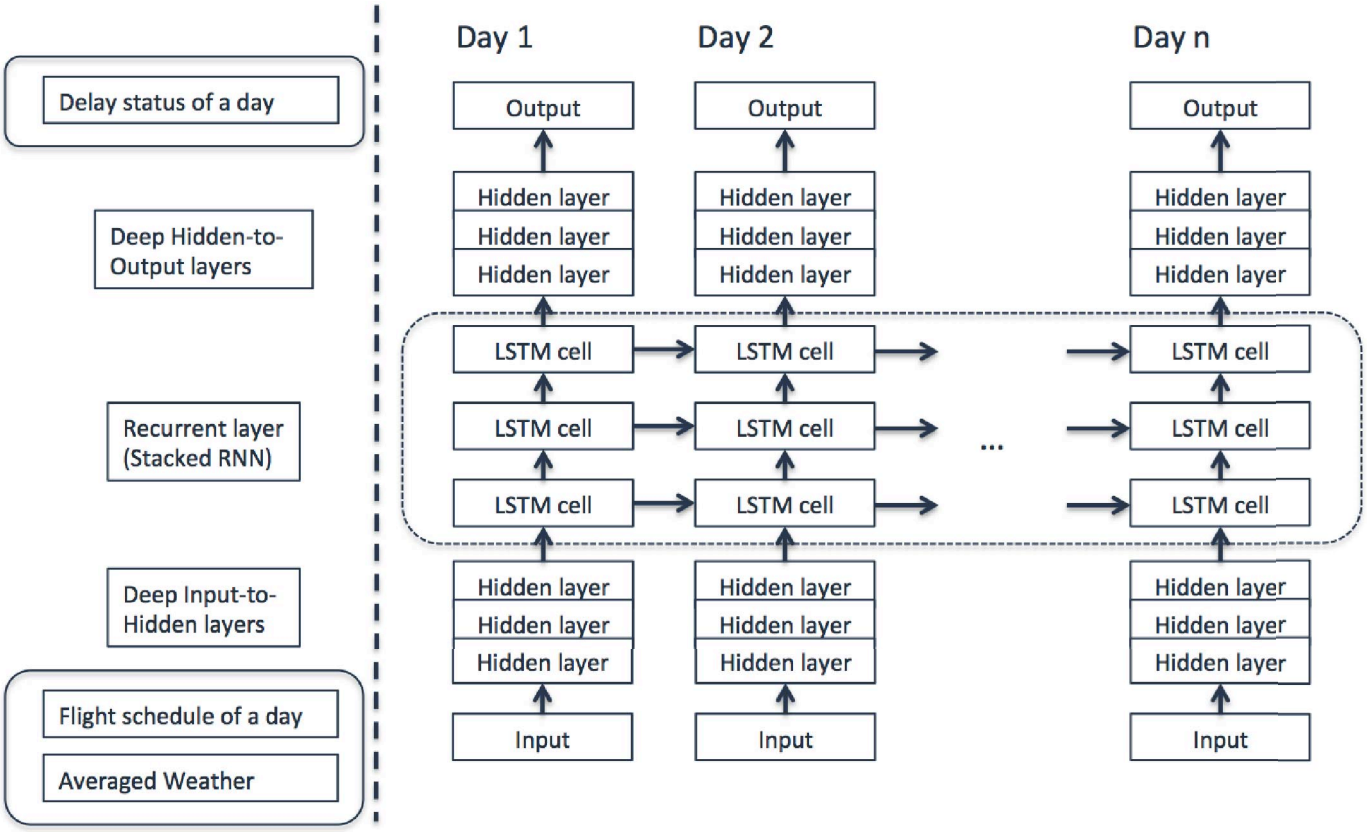

Fig. 3. Deep architecture for the RNN model

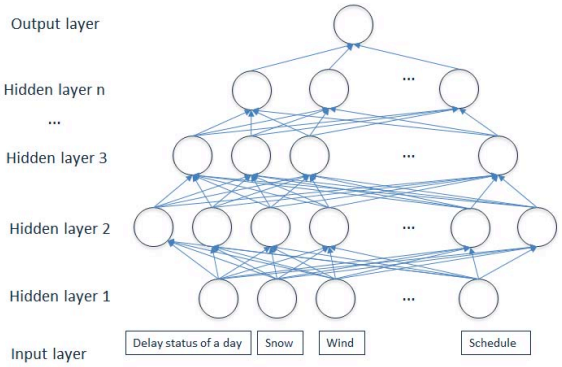

Fig. 4. Individual flight delay model

TABLE II
INPUTS AND OUTPUTS OF THE INDIVIDUAL FLIGHT DELAY MODEL

Attributes of (Input variables) Flight data	Day of week, Season, Month, Date Origin airport, Destination airport Scheduled departure time Scheduled arrival time Delay status of origin airport Delay status of destination airport
Attributes of (Input variables) Weather data	Wind direction, Wind speed, Cloud height, Visibility, Precipitation, Snow Accumulation, Intensity, Descriptor, Observation Code
Classification (Output variable)	Class of delay with different threshold values (15 minutes, 30 minutes)

samples from an exponential number of different "thinned" networks. At test time, it is easy to approximate the effect of averaging the predictions of all these thinned networks by simply using a single unthinned network that has smaller weights. This significantly reduces overfitting and gives major improvements over other regularization methods.

E. Training methods

For the training of the designed model, the stochastic gradient descent (SGD) algorithm is utilized. In contrast to the conventional gradient descent algorithm which is called batch gradient descent, it uses only one sample data at every iteration step of the training optimization. By using only one random sample at a time, it reduces computation time and memory space used for the training, significantly [21]. Sometimes, the algorithm is not converging to the direct descent direction of a local optimum because of the noise in a single data point. However, it is not a problem when a large amount of data is available. Furthermore, by adding random sampling procedure for selecting a sample data at every iteration step, SGD is another effective method to prevent overfitting and increase

general performance. Mini-batch gradient descent algorithm is in between batch gradient descent and SGD. It uses a subset of data for each iteration so reduces the time to converge. For some models in the study, mini-batch gradient descent algorithm was also utilized.

IV. Experiments

Using implemented day-to-day RNN model and individual flight NN model, experiments have been performed to analyze the effectiveness of the deep architectures. At first, day-to-day delay status model was trained with different deep RNN settings, then individual flight delay model was trained and tested by varying parameters. For these two experiments, historical data for the Atlanta airport was utilized. Finally, for evaluating the generalization performance of the model, one setting acquired from the day-to-day model experiment of the Atlanta airport was applied to other major airports and the accuracies were analyzed.

A. Day-to-day delay status model

For the first experiment, the departure delay status of the Atlanta airport was utilized. Two different sets of sequence lengths and delay threshold values were used. One is 7 days of sequence with 15 minutes threshold value and the other one is 9 days of sequence and 30 minutes threshold value. For both cases, the model was varied by different architectures which were deep input-to-hidden, stacked RNN and combined architecture as explained in Section II. The combined architecture is using deep input-to-hidden, stacked RNN and deep hidden-to-output architectures all together. As a reference model, a shallow model was also tested. The shallow model includes a single LSTM layer and input and output are directly linked to the LSTM layer. Table III shows the accuracies for different deep architectures of RNN. We achieved about 90% accuracy for the day-to-day delay status prediction. This means that we can get a fairly accurate delay status for a single day. Another observation was that the deep architectures were improving the accuracy of the model. In both cases, deep input-to-hidden architecture improved the accuracy slightly. And, by merging all deep architectures, we could get 3-5% improved accuracies compared to shallow one.

However, stacked RNN does not always guarantee an improvement. In the latter experiment case, it shows an improvement but it does not in the first case. It can be analyzed that its complexity makes the computation of the model difficult. It has been commonly observed in RNN architectures with long sequences. By stacking multiple LSTM cells, the complexity of the model is increasing too much and causing numerical difficulties. Even for some other parameter settings, the stacked LSTM did not converge numerically. One last observation from the experiment is that the model is a better predictor for the larger delay threshold value. This can be seen that the smaller threshold value would be noisier than larger threshold value so it is harder to predict smaller threshold value delays. In other words, the larger threshold value is classifying delay days with a more definite criterion.

TABLE III
Accuracy of day-to-day RNN models for the Atlanta airport

Parameters	Shallow	Stacked RNN	Input-to-Hidden	Combined
Sequence: 7 days Threshold: 15 mins	78.55	77.41	79.70	80.63
Sequence: 9 days Threshold: 30 mins	87.07	90.86	90.92	90.95

TABLE IV
Accuracy of individual flight delay models

Layers	Number of hidden nodes for each layer	Epoch	Accuracy
1	133	22	85.32
2	$133 \rightarrow 100$	22	86.57
3	$133 \rightarrow 200 \rightarrow 15$	22	86.71
4	$133 \rightarrow 200 \rightarrow 100 \rightarrow 15$	22	86.93
5	$133 \rightarrow 300 \rightarrow 200 \rightarrow 100 \rightarrow 15$	22	86.99
5	$133 \rightarrow 300 \rightarrow 200 \rightarrow 100 \rightarrow 15$	228	87.40
5 (mini-batch)	$133 \rightarrow 300 \rightarrow 200 \rightarrow 100 \rightarrow 15$	228	87.42

B. Individual flight delay model

By combining the delay status of a single day, historical flight data and weather data, the model for individual OD pairs was trained. The networks described in Section III and Fig. 4 was utilized. In the deep layered fully connected nodes, the number of layers, the number of hidden nodes in each layer, epoch and the batch size are varied and the accuracy is tested. An epoch means one full pass through the training set. At every iteration, the number of samples used for the training is the batch size. Table IV shows the accuracies acquired for different settings. From the results, the deep model achieved high accuracy ranging from 86% to 87%. It also shows that the increased number of layers is contributing to improve accuracy. Thus, the deep learning approach to the flight delay prediction is effective. Another observation is that more epochs make the model more accurate which is a common observation in most of the previous data analytics tasks [22]. This suggests that deep learning approaches and models will perform better in the future by accumulating more data. Additionally, the accuracy for the model tested in this part might be used for the direct comparison with previous studies. Therefore, 87.42% accuracy is the best accuracy so far comparing with the previous best accuracy 83.4% [9], [10].

C. Comparison of day-to-day model for different airports

The day-to-day model has been applied to 10 different airports. In the first part of this section, the best performing parameter settings are found using the Atlanta airport's air traffic data. The settings are applied to other 9 airports and the accuracies of the models are evaluated again here. Table V shows the accuracy results for 10 airports. It shows all the accuracy values are over 85% except for PHX airport. Even for

TABLE V
ACCURACY OF DAY-TO-DAY MODEL FOR DIFFERENT AIRPORTS

Airport	Accuracy
Atlanta (ATL)	90.95
Los Angeles (LAX)	86.96
Chicago (ORD)	85.61
Dallas (DFW)	89.31
Denver (DEN)	89.82
New York (JFK)	86.51
San Francisco (SFO)	87.52
Charlotte (CLT)	91.80
Las Vegas (LAS)	91.81
Phoenix (PHX)	71.34

the CLT and LAS airports, the accuracies are higher than the ATL airport. Therefore, the model generalized well the day-to-day sequence and it can be concluded that the sequential delay impacts of previous days are modeled well in the LSTM RNN architecture for most of the airports. It is also observed that deep learning models need large amounts of data from the low accuracy of PHX airport which has much less air traffic data than the other airports. We presume there is a threshold value for the amount of data to make a model effective. In case of PHX airport, the amount of air traffic data has not exceeded the threshold value but the traffic data for other airports exceeds the threshold value.

V. CONCLUSIONS AND FUTURE WORKS

From this study, we have shown that the deep architecture can improve the accuracy of the airport delay prediction models. In particular, by applying deep LSTM RNN architecture to the prediction model, a reliable delay status of a single day could be acquired. Then, the most accurate delay states for individual flights have been acquired by feeding the delay status of a day to the individual flight delay model. It gives state-of-the-art results in predicting individual flight delays. The next steps are to apply other deep architectures to the prediction and analysis task of flight delays. It may yield important patterns in flight delay data.

REFERENCES

[1] M. Ball, C. Barnhart, M. Dresner, M. Hansen, K. Neels, A. Odoni, E. Peterson, L. Sherry, A. Trani, B. Zou *et al.*, *Total delay impact study*. Institute of Transportation Studies, University of California, Berkeley, 2010.

[2] B. Manley and L. Sherry, "Analysis of performance and equity in ground delay programs," *Transportation Research Part C: Emerging Technologies*, vol. 18, no. 6, pp. 910–920, 2010.

[3] J. Ferguson, A. Q. Kara, K. Hoffman, and L. Sherry, "Estimating domestic us airline cost of delay based on european model," *Transportation Research Part C: Emerging Technologies*, vol. 33, pp. 311–323, 2013.

[4] C. N. Glover and M. O. Ball, "Stochastic optimization models for ground delay program planning with equity–efficiency tradeoffs," *Transportation Research Part C: Emerging Technologies*, vol. 33, pp. 196–202, 2013.

[5] Y. J. Kim, O. J. Pinon-Fischer, and D. N. Mavris, "Parallel simulation of agent-based model for air traffic network," in *AIAA Modeling and Simulation Technologies Conference*, 2015, p. 2799.

[6] F. Wieland, "Parallel simulation for aviation applications," in *Proceedings of the 30th conference on Winter simulation*. IEEE Computer Society Press, 1998, pp. 1191–1198.

[7] Y. Tu, M. O. Ball, and W. S. Jank, "Estimating flight departure delay distributionsa statistical approach with long-term trend and short-term pattern," *Journal of the American Statistical Association*, vol. 103, no. 481, pp. 112–125, 2008.

[8] N. Xu, G. Donohue, K. B. Laskey, and C.-H. Chen, "Estimation of delay propagation in the national aviation system using bayesian networks," in *6th USA/Europe Air Traffic Management Research and Development Seminar*. Citeseer, 2005.

[9] J. J. Rebollo and H. Balakrishnan, "Characterization and prediction of air traffic delays," *Transportation Research Part C: Emerging Technologies*, vol. 44, pp. 231–241, 2014.

[10] S. Choi, Y. J. Kim, S. Briceno, and D. N. Mavris, "Prediction of weather-induced airline delays based on machine learning algorithms," in *Digital Avionics Systems Conference (DASC), 2016 IEEE/AIAA 35th*. IEEE, 2016.

[11] M. M. Najafabadi, F. Villanustre, T. M. Khoshgoftaar, N. Seliya, R. Wald, and E. Muharemagic, "Deep learning applications and challenges in big data analytics," *Journal of Big Data*, vol. 2, no. 1, pp. 1–21, 2015.

[12] H. Kashyap, H. A. Ahmed, N. Hoque, S. Roy, and D. K. Bhattacharyya, "Big data analytics in bioinformatics: A machine learning perspective," *arXiv preprint arXiv:1506.05101*, 2015.

[13] Y. Lv, Y. Duan, W. Kang, Z. Li, and F.-Y. Wang, "Traffic flow prediction with big data: A deep learning approach," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 16, no. 2, pp. 865–873, 2015.

[14] K.-i. Funahashi and Y. Nakamura, "Approximation of dynamical systems by continuous time recurrent neural networks," *Neural networks*, vol. 6, no. 6, pp. 801–806, 1993.

[15] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 6645–6649.

[16] R. Pascanu, C. Gulcehre, K. Cho, and Y. Bengio, "How to construct deep recurrent neural networks," *arXiv preprint arXiv:1312.6026*, 2013.

[17] "Bureau of transports statistics, http://www.transtats.bts.gov."

[18] "National oceanic and atmospheric administration, http://www.noaa.gov/."

[19] G. E. Dahl, T. N. Sainath, and G. E. Hinton, "Improving deep neural networks for lvcsr using rectified linear units and dropout," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 8609–8613.

[20] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[21] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of COMPSTAT'2010*. Springer, 2010, pp. 177–186.

[22] X.-W. Chen and X. Lin, "Big data deep learning: challenges and perspectives," *Access, IEEE*, vol. 2, pp. 514–525, 2014.