Predict Flight Delays Based on Hybrid-LSTM

Jianyang Zhou
Duke University
Durham, NC
jianyang.zhou@duke.edu

Javier Pastorino
Duke University
Durham, NC
javier.pastorino@duke.edu

***Abstract*—Flight delays pose significant challenges to airline operations, passenger satisfaction, and economic efficiency. Accurately predicting flight delays is crucial for improving air traffic management and mitigating disruptions. Recent advancements in deep learning have enabled more sophisticated models capable of capturing the complex relationships between flight schedules, weather conditions, and airport congestion. In this study, we propose a Hybrid-LSTM (H-LSTM) model that integrates historical flight sequences and real-time flight attributes to predict arrival delay categories. Our model leverages LSTM networks to analyze sequential flight dependencies and a Dense network to incorporate independent flight features, including airport information and meteorological conditions. A sliding window approach was employed to structure historical flight sequences, enhancing the model's ability to capture delay propagation patterns. Experimental results demonstrate that our H-LSTM model achieves and accuracy of 89.4% outperforming baseline random forest methods (85.5%). We also analyzed the performance in different scenarios looking at the traffic of the airports, Among large airports, our model achieved a prediction accuracy of 90.6%, higher than the baseline of 88.4%.**

***Index Terms*—Deep Learning, machine learning, aviation, flight delay, prediction**

I. Introduction

Flight delays have long been a critical concern in the aviation industry, posing significant challenges to airline operations, passenger satisfaction, and economic performance. Each year, delays incur substantial financial losses for airlines, disrupt operational schedules, and negatively affect customer experiences. As air traffic networks grow increasingly complex and travel demand continues to rise, accurately predicting flight delays has become essential for enhancing efficiency and mitigating associated disruptions.

Recent advancements in machine learning (ML) and deep learning (DL) provide promising solutions for flight delay prediction. These technologies facilitate the analysis of large-scale, high-dimensional datasets and capture intricate relationships among key delay factors, including weather conditions, airport congestion, and air traffic flow. Deep learning models, such as Recurrent Neural Networks (RNNs) and Convolutional Neural Networks (CNNs), have demonstrated significant potential in addressing the temporal and spatial complexities inherent in flight data.

This study leverages deep learning methodologies, specifically Long Short-Term Memory (LSTM) networks, to predict flight delays by analyzing sequential flight data and modeling delay propagation across interconnected flights. By integrating flight attributes, weather conditions, and airport delay patterns, the proposed approach aims to generate accurate and actionable predictions. The research highlights the importance of understanding both macro-level factors (e.g., weather patterns, airport congestion) and micro-level attributes (e.g., aircraft-specific delay history) to improve predictive accuracy.

This paper contributes to existing research by introducing a hybrid modeling approach that integrates multiple data sources with advanced deep learning techniques. The findings aim to assist airlines, airports, and passengers in making data-driven decisions to mitigate delays, optimize operational efficiency, and enhance the overall air travel experience.

II. Literature Review

A. Machine Learning Approaches

Flight delays are a critical issue that impacts the economy, passenger satisfaction, and airline operations. With growing air traffic and interconnected networks, predicting and mitigating delays has become increasingly important. Early research about using machine learning to predict flight delay can be tracked back to 2008, where k-Means clustering was applied to analyze historical flight data, identifying five delay grades of delay —from no delay to heavy delay, based on similarities in key metrics [1].

Recent years have witnessed significant growth in the application of machine learning techniques to predict flight delays. Esmaeilzadeh and Mokhtarimousavi employed a Support Vector Machine (SVM) model, utilizing data such as flight schedules, airport demand/capacity, and weather conditions from three major New York City airports, achieving 85.57% accuracy for arrival delay prediction [2]. Li and Jing applied a random forest model using features such as airport

congestion, network congestion, and demand-capacity imbalance to U.S. domestic flight data from July 2018, achieving arrival delay prediction errors smaller than 30 minutes in 97% of cases [3]. Yu et al. introduced a deep belief network for Beijing International Airport data, incorporating explanatory variables such as prior flight delays, airline characteristics, aircraft capacity, and boarding options, which delivered high prediction accuracy [4]. Guo et al. developed a hybrid technique combining Random Forest Regression with the Maximal Information Coefficient, using inputs such as aircraft capacity, prior delays, turnaround times, and airport crowding, outperforming traditional methods like linear regression and neural networks [5]. Similarly, Gui et al. [6] evaluated LSTM and Random Forest models with inputs like traffic flow, weather, flight schedules, and ADS-B signals, identifying Random Forest as the most efficient approach.

Despite these advancements, existing models often fail to fully capture the temporal dependencies of the flight sequence. For instance, Rebollo and Balakrishnan [7] [8], and Gopalakrishnan and Balakrishnan [9] constructed delay networks based on original-destination pair (OD-pair) delays, while Güvercin, Ferhatosmanoglu, and Gedik [10] and Li and Jing [11] modeled the air traffic system as a network of nodes and edges to extract temporal characteristics. However, these methods are limited in representing the temporal continuous of the air traffic system.

B. RNN and LSTM

RNN are a class of artificial neural networks designed to model the behavior of dynamic systems through hidden states. LSTM networks, a specialized implementation of RNNs, offer improved speed and accuracy compared to standard RNNs. This section provides an overview of the general architectures of RNNs and LSTM networks.

Given an input sequence $x = (x_1, x_2, \ldots, x_k, \ldots, x_T)$, a RNN computes the evolution of hidden states $h = (h_1, h_2, \ldots, h_k, \ldots, h_T)$ and the output sequence $y = (y_1, y_2, \ldots, y_k, \ldots, y_T)$. This process iteratively solves the following equations for each time step $t = 1$ to T. Here, x_k, h_k, and y_k are arbitrary-sized vectors corresponding to the dimensions of the input space, hidden space, and output space.

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

$$y_t = \phi_o(W_{hy}h_t + b_y) \tag{2}$$

where W_{hh} represents the weight matrix for transitioning hidden states between consecutive time steps, W_{xh} denotes the weight matrix mapping input to the hidden layer, and W_{hy} denotes the weight matrix connecting the hidden layer to the output. The terms b_h and b_y are bias vectors for their respective equations. The functions ϕ_h and ϕ_o serve as activation functions for the hidden states and output, respectively. Typically, these activation functions are nonlinear, such as a logistic sigmoid or a hyperbolic tangent function, applied element-wise to the given vectors.

The LSTM architecture introduces memory cells that replace the activation functions ϕ_h and ϕ_o in standard RNNs to better retain long-term dependencies. LSTM networks exhibit superior performance in modeling long-range sequences compared to conventional RNN architectures.

In this research, we use the LSTM memory cell proposed by Alex Graves [15]. This single memory cell is recurrently applied within the model. It consists of an input gate (i), forget gate (f), output gate (o), and cell state vector (c), all of which share the same dimensionality as the hidden state h. The model's computations are defined as follows:

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

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

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

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

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

where ϕ represents the logistic sigmoid function. The LSTM module's cell architecture is illustrated in Figure below.

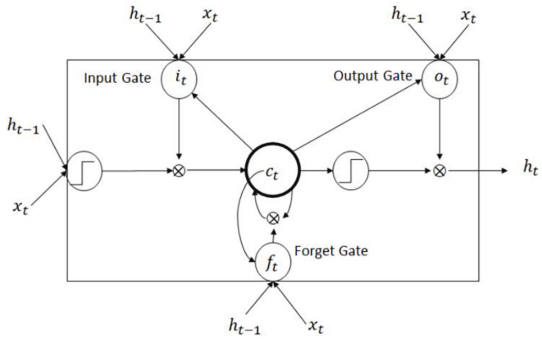

Fig. 1: Long hort-term memory cell[15]

C. Research Gaps

Delay propagation is a significant phenomenon in air transportation networks [12], occurring when an arrival delay at a destination airport impacts subsequent flights operated by the same aircraft, crew, or even transit passengers. Among these factors, aircraft are the primary contributors to delay propagation. Figure 2 illustrates the delay propagation of one aircraft over four flights. In this example, after the first flight from airport A to airport B, the aircraft required unexpected mechanical inspection delaying its departure at airport B. This generated a late arrival to airport C for 60 minutes and a consequent delayed departure. Every flight since then has been affected by the spread of the delay, until it eliminated by enough layover time.

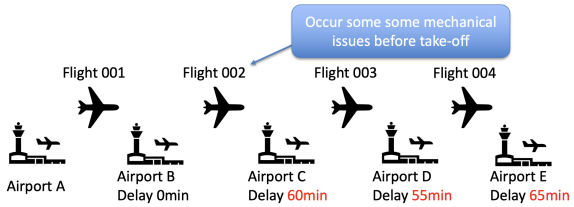

Fig. 2: Delay propagation in flight networks

To capture more information about an aircraft flight history, we developed an LSTM model to learn each aircraft's flight sequence, and find patterns in the delay sequences. We then integrated the information of the current flight to be predicted in order to make a more accurate prediction.

III. Experiment Data

In this section, we present the data, conduct an initial review, and outline the processing methodology.

A. Dataset Overview

We acquired flight data from the Bureau of Transportation Statistics (BTS) official website Airline On-Time Performance Data, which contains domestic flight records from the United States monthly [13]. The original monthly dataset covering approximately 500,000 flight entries for each month with 123 features, including departure and arrival airport, gate time, in-flight time, delay reason, etc, as shown in Appendix A. We manually downloaded data for 24 months from January 2023 to December 2024 and then concatenated them.

We gather weather data from the National Oceanic and Atmospheric Administration (NOAA) official website, which primarily records hourly weather conditions for each airport [14]. We manually downloaded data for 342 airports. The dataset contains 140 features for each airport, including weather conditions, temperature, wind speed, wind direction, etc, as shown in Appendix B.

B. Initial Data Exploration

1) Delay Analysis

In 2023-2024, the average arrival delay time for domestic flights was 22.3 minutes. We used Tableau to visualize the data and project it onto a national map shown in Figure 3.

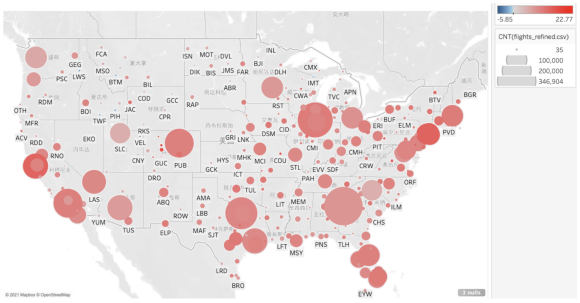

Fig. 3: Arrival delay overview by map

Figure 4 shows an distribution of the flights delay in the dataset. As it can be seen, approximately 71% of the flights were delayed less than 15 minutes, which are often considered on-time in civil aviation transportation. Further, 11% of the flights experienced delays of more than 60 minutes, classified as severe delays.

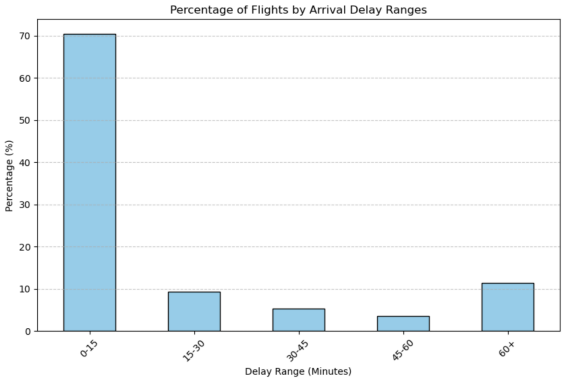

Fig. 4: Arrival delay distribution in 5 categories

In Figure 5, we analyzed delay reason for flight. The original dataset provides five categories for delays: `CarrierDelay`, `WeatherDelay`, `NASDelay`(National Air System Delay), `SecurityDelay`, `LateAircraftDelay`, each delayed flight is recorded with the specific delay reason time in minute. It showed that 35.6% of the delays had a root cause in the aircraft's previous flights.

2) Flight Sequence Data

Studying how many flights each aircraft performs per day is also important for our research. We calculated the number of flights each aircraft performs per day and found that the number varies between 1 and 12. Notably, 32% of aircraft operate 6 flights per day, making

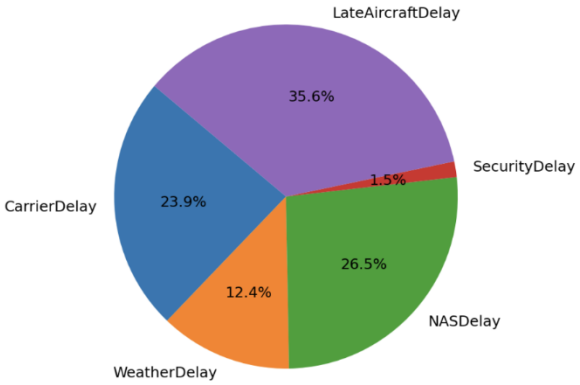

Fig. 5: Proportion of each delay reason

it the most common value. Figure 6 briefly illustrates this distribution.

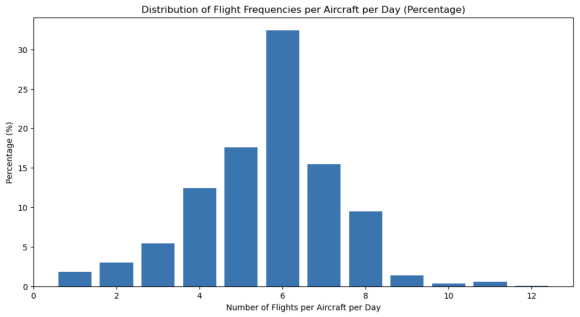

Fig. 6: Number of flights aircraft operates in one day

C. Data Pre-processing

1) Data Cleaning

In the original dataset, not all of the features are useful, for example, to describe the departure airport, the dataset provides nine different features, including `OriginAirporID`, `OriginAirportSeqID`, `OriginCityMarketID`, `Origin`(the IATA airport code), `OriginCityName`, `OriginState`, `OriginStateFips`, `OriginStateName`, and `OriginWac`. After checking the definition of these features, only the IATA airport code is relevant, thus we remove all but `Origin` attributes. After reviewing all 120 features, we removed unnecessary and duplicate features that represented the same attributes, and finally retained 42 useful features.

The same problem exists with weather data: most features represent daily or monthly weather patterns, which are not relevant to the hourly conditions corresponding to a single flight. Therefore, we excluded them and retained only hourly weather data.

2) Add Delay Category

In order to make the prediction more accurate, rather than just predicting whether the flight will be delayed or not, we categorize arrival delays into five numerical categories: 1 (less than 15 minutes), 2 (15 to 30 minutes), 3 (30 to 45 minutes), 4 (45 to 60 minutes), and 5 (more than 60 minutes). This classification allows future predictions to focus on identifying the specific delay category.

3) Add Average Airport Departure Delay

Flights departing from the same airport within the same time period tend to have similar delay characteristics [9]. Based on this observation, we incorporated statistics on the average airport departure delay into the dataset. First, we grouped the flight data by departure airport and sorted them by scheduled departure time. Then, for each flight, we calculated the average airport departure delay over the time ranges $\{[0, 30), [30, 60), [60, 90), [90, 120)\}$ prior to its scheduled departure time, adding the feature `AvgDelay0_30`, `AvgDelay30_60`, `AvgDelay60_90`, `AvgDelay90_120`. Additionally, we counted the number of departures within these time periods, adding the feature `DepCount0_30`, `DepCount30_60`, `DepCount60_90`, `DepCount90_120`. If no flights were recorded in a given time range, we assigned a value of 0 to both the average airport departure delay and the departure count.

These counts are then normalized so that their sum equals 1:

$$\text{DepCount\_Normalized}_i = \frac{\text{DepCount}_i}{\sum_{j=1}^{4} \text{DepCount}_j}$$

where $i \in \{1, 2, 3, 4\}$ corresponds to the four time ranges. This produces a normalized distribution reflecting the relative congestion in each interval.

To emphasize the influence of more recent departures, we apply a weighted average using weights of 4:3:2:1 for the four intervals, respectively. The final weighted feature is calculated as:

$$\text{DepCount\_Weighted} = \sum_{i=1}^{4} w_i \cdot \text{DepCount\_Normalized}_i$$

where $w = [4, 3, 2, 1]$. This weighted value is used as a feature to capture short-term congestion trends at the departure airport. Below figure 7 is an example showing how we handle the DepCount parameter.

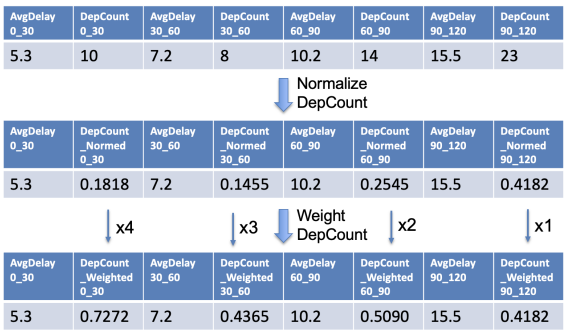

Fig. 7: Example of how we handle the DepCount

4) Merging Flight and Weather Data

After processing the weather and flight data, they are ready to be combined. For each flight record, the weather data closest in time to the actual departure and arrival times at the respective airports is selected and added to the flight dataset.

5) Normalize Data

Before training the deep learning model, it is important to prepare the input features so that the model can learn well. One key step in this process is data normalization, which helps bring all input features to a similar scale. This makes the training more stable and helps the optimizer work better.

We used *standardization* to normalize all numeric and encoded features. This was done using the `StandardScaler` from the `sklearn.preprocessing` package. Standardization changes each feature so that it has a mean of zero and a standard deviation of one. This is done using the formula:

$$X_{\text{scaled}} = \frac{X - \mu}{\sigma}$$

where μ is the average value of the feature and σ is its standard deviation. This step ensures that no single feature dominates the learning process due to its larger values.

Normalization was done after we changed the categorical values to numbers and before we built the input sequences for the model. To avoid data leakage, we fitted the scaler only on the training data. Then we used the same scaler to transform the validation and test data.

Using normalization in the workflow helped the model learn faster, avoid unstable training, and perform better on unseen flight delay predictions.

6) Overview of Input Data to Model

After completing all preprocessing steps, we obtain flight data from January 1, 2023, to December 31, 2024, comprising a total of 85 columns and 12,543,321 rows, which are used as input for the subsequent deep learning model.

IV. Methodology

A. Hybrid-LSTM

The Hybrid-LSTM aims to predict the arrival delay category (`ArrDelayCategory`) of a flight using both historical flight sequences and known features for the predicting flight. Given a sequence of past n flights for an aircraft, the objective is to predict the arrival delay category of the $(n+1)^{\text{th}}$ flight by leveraging both temporal dependencies and independent flight attributes. To achieve this, a hybrid deep learning model is developed, consisting of a LSTM network to capture historical delay trends and a Dense neural network to process known attributes of the target flight.

B. Model Architecture

The proposed Hybrid-LSTM model consists of two parallel input networks: Delay Propagation Network to capture delay sequential patterns from past flights, and a Dense Network to capture and process known contextual attributes of the target flight.

Figure 8 shows an overview of this model, while figure 9 shows the detailed layers in each sub-models.

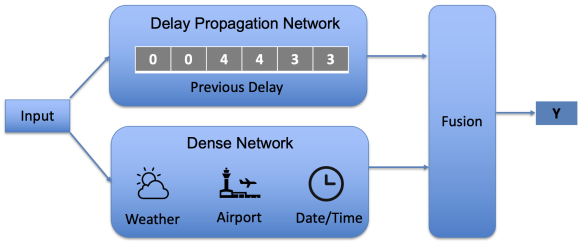

Fig. 8: Model Overview

1) Delay Propagation Network for Sequence Features

The Delay Propagation network in this model is designed to capture sequential dependencies in historical flight delays and extract meaningful patterns to improve arrival delay prediction. The network takes the `ArrDelayCategory` as input of the past five flights for a given aircraft and processes these sequential patterns through two stacked LSTM layers.

The first LSTM layer consists of 256 units and is configured with `return_sequences=True`, allowing it to pass sequential information to the next LSTM layer. Batch normalization is applied to stabilize activations and improve convergence speed. A dropout layer (30%) follows, helping to prevent overfitting by randomly deactivating neurons during training.

The second LSTM layer, also with 256 units, refines the sequential dependencies and extracts higher-level features. This layer is set with `return_sequences=False`, meaning it produces a fixed-size output vector representing the learned

time-dependent delay patterns. Another round of batch normalization and dropout (30%) is applied to further enhance stability and generalization.

2) Dense Network for Contextual Features

The Dense network processes all available flight attributes, including both historical and real-time features. Unlike the LSTM network, which captures sequential dependencies, the Dense network learns meaningful representations from independent flight data, such as scheduled departure time, airport information, and weather conditions.

The Dense Network has three dense layers to extract and refine features from the input data. The first layer has 512 units with ReLU activation, followed by batch normalization to stabilize learning and a 30% dropout to prevent overfitting. The second layer has 256 units with ReLU, along with batch normalization and a 20% dropout to keep the model general. The final layer has 128 units with ReLU, also followed by batch normalization and a 20% dropout, helping to create a strong and stable feature representation for classification.

3) Feature Fusion and Prediction

The outputs from the Delay Propagation and Dense networks are combined in the feature fusion layer to integrate both historical delay patterns and real-time flight attributes. This step ensures that the model can leverage both time-dependent information and contextual factors to make a more informed prediction.

The concatenation layer merges the output of the Delay Propagation network, which captures sequential dependencies in past arrival delay categories, with the output of the Dense network, which processes independent flight attributes. This combined feature representation provides a comprehensive understanding of flight delays by incorporating both historical trends and current flight conditions.

After fusion, the model passes the combined features through a dense layer with 128 units and a ReLU activation function, which refines the representation and enhances the model's ability to learn meaningful patterns. A dropout layer with a rate of 30% is applied to prevent overfitting by reducing reliance on specific neurons during training.

The final dense layer applies a softmax activation function to output a probability distribution over the possible arrival delay categories. This allows the model to classify each flight into one of the predefined delay categories based on the learned feature representations.

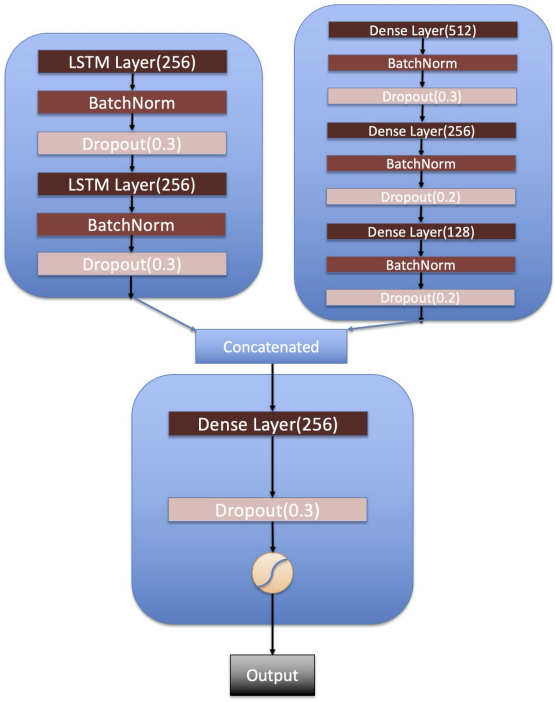

Fig. 9: The detailed model structure

V. Experiment Design and Training

A. Input and Target Variable Generation

Each flight record is associated with an aircraft (`Tail_Number`) and sorted in chronological order. The model's input features are categorized into two groups:

- **Sequential Features**: These features represent historical flight trends and are derived from the previous n flights. They include:
  - Historical delay trends of past flights (`ArrDelayCategory`)
- **Independent Features**: These are attributes of the $(n+1)^{\text{th}}$ flight, which are known before departure and ensure real-time data availability, including:
  - **Airport Information**: Departure and arrival airports, airlines
  - **Time Information**: Date, Day of Week, Scheduled Departure and Arrival Times
  - **Weather Information**: Temperature, precipitation, wind speed, humidity, and visibility at the departure and arrival airport

To maintain temporal dependencies, the dataset is structured using a sliding window approach: The previous n flights **Sequential Features** serve as the input to the Delay Propagation network. The $(n+1)^{\text{th}}$ flight's **Independent Features** are processed separately by a Dense neural network. Also, The $(n+1)^{\text{th}}$ flight's `ArrDelayCategory` serves as the target variable.

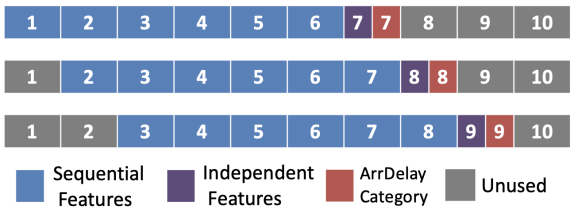

Fig. 10: Example of sliding window(N=6)

The figure 10 is an example of how this method works when input sequence length n=6. For the first window, we choose flight 1-6th's **Sequential Features** and flight 7th's **Independent Features** as input variable, and the 7th's flight's `ArrDelayCategory` as the target variable. For the second window, we choose flight 2-7th's **Sequential Features** and flight 8th's **Independent Features** as input variable, and the 8th's flight's `ArrDelayCategory` as the target variable, etc.

B. Data Split

Once all sequences were created, the dataset was split into two main parts. The most recent 15% of the samples were held out as a final *test set*, never seen during training or validation. This test set was used exclusively for final evaluation to provide an unbiased estimate of the model's real-world performance.

The remaining 85% of the data was used for model training and hyperparameter tuning via forward chaining cross-validation. We employed `TimeSeriesSplit` [16] with 5 folds. In each fold, the training set consisted of a growing window of earlier sequences(Figure 11), while the validation set consisted of the next block of future sequences. This strategy ensured that every validation fold only used information from the past to predict future events, preserving temporal causality. The model was trained and validated in each fold, and the average validation accuracy across all folds was reported. After cross-validation, the final model was evaluated on the held-out test set to assess generalization performance.

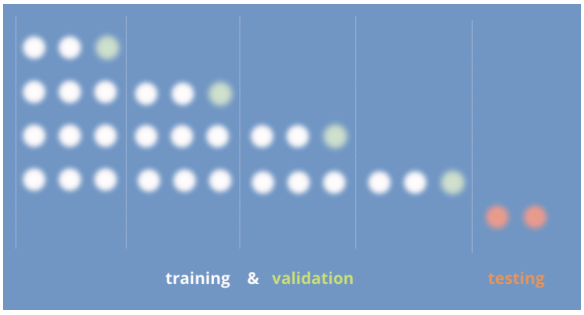

Fig. 11: Example of forward chaining cross-validation

C. Model Training

During each fold, the model was trained using the Adam optimizer with a learning rate of 0.0005, which can balancing training stability and training speed, and a batch size of 64. The categorical cross-entropy loss function was used, appropriate for the multi-class classification setting. To prevent overfitting and improve generalization, early stopping was applied with a patience of 7 epochs, monitoring the validation loss. The model with the lowest validation loss in each fold was retained.

We conducted experiments with input sequence length N = 6, figure 12 shown the validation accuracy across the 5 folds, the accuracy increases with training. Figure 13, this is the learning curve of the model in the last fold. The learning curve shows a sharp decline in both training and validation loss during the initial epochs, indicating rapid learning. As training progresses, the loss continues to decrease gradually, suggesting that the model is refining its understanding of the data. Around epoch 15, the validation loss stabilizes with minor fluctuations, while the training loss continues to decrease slightly, showing that the model is converging. The close alignment between training and validation loss throughout the process indicates minimal overfitting and good generalization. The slight fluctuations in validation loss are expected and may be attributed to variations in the dataset.

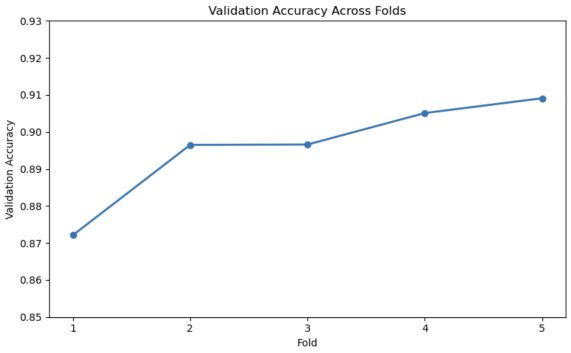

Fig. 12: Folds accuracy

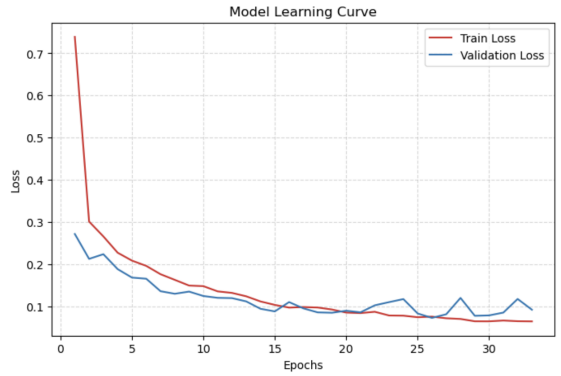

Fig. 13: Learning curve for the last fold

D. Performance Metrics

The evaluation is conducted using accuracy, precision, recall, and F1-score to assess classification performance. Additionally, a confusion matrix is presented to analyze misclassifications across different delay categories. For the binary classification task, we categorize samples based on their actual and predicted labels. The classification outcomes include True Positive (TP), False Positive (FP), True Negative (TN), and False Negative (FN). The total number of samples is given by $TP + FP + TN + FN$.

Accuracy measures the proportion of correctly classified samples among all samples and is defined as:

$$accuracy = \frac{TP + TN}{TP + FN + FP + TN} \tag{8}$$

Precision quantifies the proportion of predicted positive instances that are truly positive, as calculated by:

$$precision = \frac{TP}{TP + FP} \tag{9}$$

Recall indicates the fraction of actual positive cases correctly identified, given by:

$$recall = \frac{TP}{TP + FN} \tag{10}$$

The F1-score is a metric that balances precision and recall, providing a single value that represents the harmonic mean of both. It is particularly useful when dealing with imbalanced datasets, where a high precision or recall alone may not fully capture model performance. The F1-score is computed as:

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{11}$$

Table I displays the confusion matrix used in this study.

TABLE I: The confusion matrix.

Actual	Predicted	
	Positive	Negative
Positive	TP	FN
Negative	FP	TN

VI. Experimental Results

A. Compare to the baseline models

The improved Hybrid-LSTM model demonstrates significant enhancements over the baseline model, it outperform 89.4% accuracy on the predicting, higher than baseline 85.5%, achieving higher classification accuracy across 0-15 min, 15-30 min, and 60+ min Class, 14 shows the prediction accuracy of the two models for different types of delays. Figure 15 and 16 shows the confusion matrix of them. Figure 17 shows the F1-Score for each delay category, most results are above 0.8. These improvements suggest that the updated model better captures feature representations, generalizes more effectively, and achieves superior predictive performance.

The ROC curve for 0-15 min delay prediction(Figure 18) indicates that the model performs well in classification tasks, achieving both high sensitivity and specificity. This level of performance is suitable for practical deployment in flight delay prediction scenarios, where distinguishing delay categories is critical.

Delay Category	Random Forest/%	H-LSTM/%
0-15 min	84.92	90.39
15-30 min	85.74	89.89
30-45 min	91.10	89.30
45-60 min	94.17	88.79
60+ min	84.06	90.22

Fig. 14: Model comparison

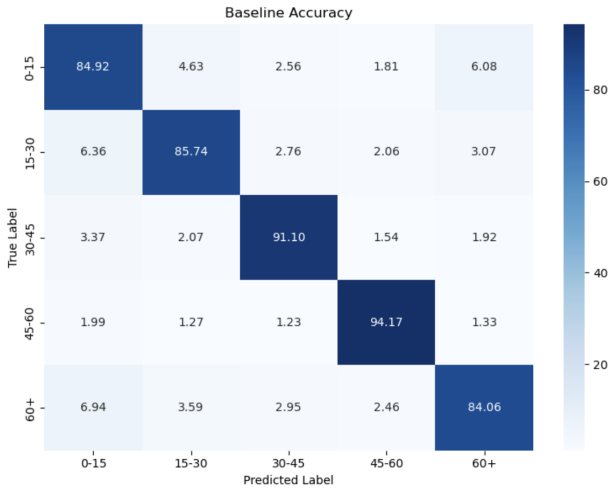

Fig. 15: Confusion Matrix - Baseline Model

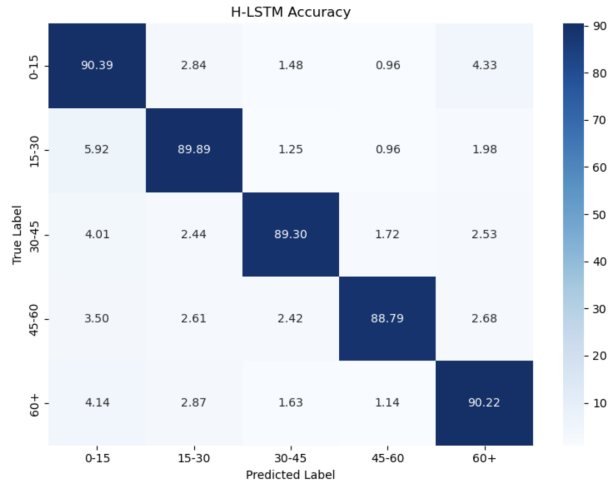

Fig. 16: Confusion Matrix - Hybrid-LSTM Model

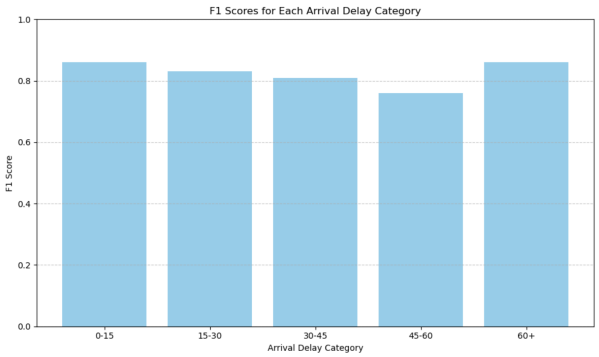

Fig. 17: F1-Score for each delay category

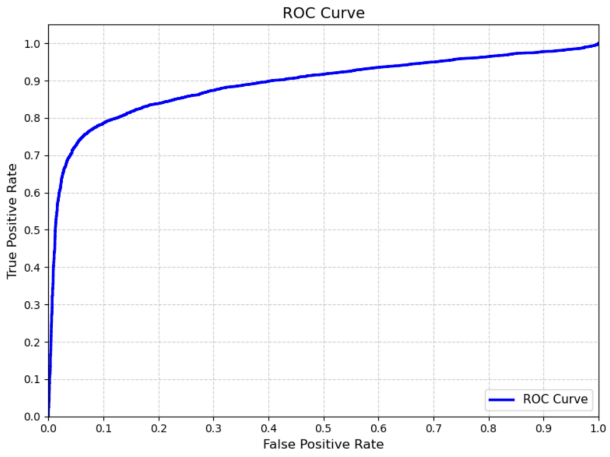

Fig. 18: ROC curve

B. *Performance on Different Size of Airport*

We compared the Accuracy and F1-Score for the Baseline Model and the H-LSTM Model across three airport categories: *L*arge Airports (more than $100,000$ annual flights), *Medium Airports* (between $10,000$ and $100,000$ annual flights), and *Small Airports* (with less than $10,000$ annual flights). In the US only 10% percent of the airports fit under the *L*arge Airports category; however, they account for 70% of the traffic. In figure 19 we can see the performance of both models using these categories. We can see that adding features about average departure delay at the airports, significantly improved the accuracy ($+2.18\%$) and F1-Score($+2.25\%$) for large airports. It also improved the accuracy for *medium* and *small* airports with a smaller gain ($+1.29\%$& $+0.39\%$).

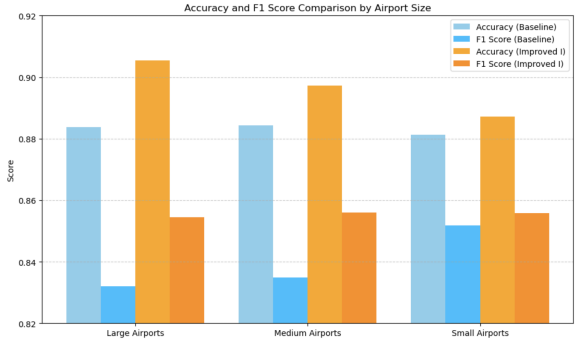

Fig. 19: Comparison of performance in different size of airport

VII. Discussion and Future Works

This study demonstrates that deep learning architectures can significantly improve the accuracy of flight delay prediction models. Specifically, by utilizing a Hybrid-LSTM architecture, a reliable delay status for a single flight can be obtained. The sequence of previous flights serves as input to the individual flight delay model, enabling more precise predictions for specific flights.

Although our model accounts only for propagated delays caused by the aircraft, in real-world flight operations, delays may also result from other factors, such as crew availability and transfer passengers. Future work will incorporate these additional factors to further enhance the model's predictive capability.

References

[1] Zonglei, L., Jiandong, W., & Guansheng, Z. (2008, December). A new method to alarm large scale of flights delay based on machine learning. In *2008 International Symposium on Knowledge Acquisition and Modeling* (pp. 589-592). IEEE.

[2] Esmaeilzadeh, E., & Mokhtarimousavi, S. (2020). Machine learning approach for flight departure delay prediction and analysis. *Transportation Research Record*, 2674(8), 145-159.

[3] Li, Q., & Jing, R. (2022). Flight delay prediction from spatial and temporal perspective. *Expert Systems with Applications*, 205, 117662.

[4] Yu, B., Guo, Z., Asian, S., Wang, H., & Chen, G. (2019). Flight delay prediction for commercial air transport: A deep learning approach. *Transportation Research Part E: Logistics and Transportation Review*, 125, 203–221.

[5] Guo, Z., Yu, B., Hao, M., Wang, W., Jiang, Y., & Zong, F. (2021). A novel hybrid method for flight departure delay prediction using Random Forest Regression and Maximal Information Coefficient. *Aerospace Science and Technology*, 116, 106822.

[6] Gui, G., Liu, F., Sun, J., Yang, J., Zhou, Z., & Zhao, D. (2019). Flight delay prediction based on aviation big data and machine learning. *IEEE Transactions on Vehicular Technology*, 69(1), 140-150.

[7] Rebollo, J. J., & Balakrishnan, H. (2014). Characterization and prediction of air traffic delays. *Transportation Research Part C: Emerging Technologies*, 44, 231-241.

[8] Rebollo de la Bandera, J. J. (2012). *Characterization and prediction of air traffic delays* (Doctoral dissertation, Massachusetts Institute of Technology).

[9] Gopalakrishnan, K., & Balakrishnan, H. (2017). A comparative analysis of models for predicting delays in air traffic networks. In *Proceedings of the 12th USA/Europe ATM R&D Seminar (ATM Seminar)*.

[10] Güvercin, M., Ferhatosmanoglu, N., & Gedik, B. (2020). Forecasting flight delays using clustered models based on airport networks. *IEEE Transactions on Intelligent Transportation Systems*, 22(5), 3179-3189.

[11] Li, Q., & Jing, R. (2021). Generation and prediction of flight delays in air transport. *IET Intelligent Transport Systems*, 15(6), 740-753.

[12] Kafle, N., & Zou, B. (2016). Modeling flight delay propagation: A new analytical-econometric approach. *Transportation Research Part B: Methodological*, 93, 520-542.

[13] Bureau of Transportation Statistics, "Airline on-time performance data," U.S. Department of Transportation. [Online]. Available: https://www.transtats.bts.gov/Tables.asp?QO_VQ=EFD&QO_anzr=Nv4yv0r%FDb0-gvzr%FDcr4s14zn0pr%FDQn6n&QO_fu146_anzr=b0-gvzr. [Accessed: Mar. 20, 2025].

[14] National Centers for Environmental Information (NCEI), "Data Access," National Oceanic and Atmospheric Administration (NOAA), [Online]. Available: https://www.ncei.noaa.gov/access/search/index. [Accessed: Mar. 20, 2025].

[15] Graves, A., Mohamed, A. R., & Hinton, G. (2013, May). Speech recognition with deep recurrent neural networks. In 2013 IEEE international conference on acoustics, speech and signal processing (pp. 6645-6649). Ieee.

[16] https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.TimeSeriesSplit.html

Appendix

Appendix A: Flight Features From Raw Data

The following flight-related features were used in the model:

- Year
- Quarter
- Month
- DayofMonth
- DayOfWeek
- FlightDate
- Marketing_Airline_Network
- Operated_or_Branded_Code_Share_Partners
- DOT_ID_Marketing_Airline
- IATA_Code_Marketing_Airline
- Flight_Number_Marketing_Airline
- Originally_Scheduled_Code_Share_Airline
- DOT_ID_Originally_Scheduled_Code_Share_Airline
- IATA_Code_Originally_Scheduled_Code_Share_Airline
- Flight_Num_Originally_Scheduled_Code_Share_Airline
- Operating_Airline
- DOT_ID_Operating_Airline
- IATA_Code_Operating_Airline
- Tail_Number
- Flight_Number_Operating_Airline
- OriginAirportID
- OriginAirportSeqID
- OriginCityMarketID
- Origin
- OriginCityName
- OriginState
- OriginStateFips
- OriginStateName
- OriginWac
- DestAirportID
- DestAirportSeqID
- DestCityMarketID
- Dest
- DestCityName
- DestState
- DestStateFips
- DestStateName
- DestWac
- CRSDepTime
- DepTime
- DepDelay
- DepDelayMinutes
- DepDel15
- DepartureDelayGroups

- DepTimeBlk
- TaxiOut
- WheelsOff
- WheelsOn
- TaxiIn
- CRSArrTime
- ArrTime
- ArrDelay
- ArrDelayMinutes
- ArrDel15
- ArrivalDelayGroups
- ArrTimeBlk
- Cancelled
- CancellationCode
- Diverted
- CRSElapsedTime
- ActualElapsedTime
- AirTime
- Flights
- Distance
- DistanceGroup
- CarrierDelay
- WeatherDelay
- NASDelay
- SecurityDelay
- LateAircraftDelay
- FirstDepTime
- TotalAddGTime
- LongestAddGTime
- DivAirportLandings
- DivReachedDest
- DivActualElapsedTime
- DivArrDelay
- DivDistance
- Div1Airport
- Div1AirportID
- Div1AirportSeqID
- Div1WheelsOn
- Div1TotalGTime
- Div1LongestGTime
- Div1WheelsOff
- Div1TailNum
- Div2Airport
- Div2AirportID
- Div2AirportSeqID
- Div2WheelsOn
- Div2TotalGTime
- Div2LongestGTime
- Div2WheelsOff
- Div2TailNum
- Div3Airport
- Div3AirportID
- Div3AirportSeqID
- Div3WheelsOn
- Div3TotalGTime
- Div3LongestGTime
- Div3WheelsOff
- Div3TailNum
- Div4Airport
- Div4AirportID
- Div4AirportSeqID
- Div4WheelsOn
- Div4TotalGTime
- Div4LongestGTime
- Div4WheelsOff
- Div4TailNum
- Div5Airport
- Div5AirportID
- Div5AirportSeqID
- Div5WheelsOn
- Div5TotalGTime
- Div5LongestGTime
- Div5WheelsOff
- Div5TailNum
- Duplicate

Appendix B: Weather Features From Raw Data

The following weather-related features were used in the model:

- STATION
- DATE
- LATITUDE
- LONGITUDE
- ELEVATION
- NAME
- REPORT_TYPE
- SOURCE
- HourlyAltimeterSetting
- HourlyDewPointTemperature
- HourlyDryBulbTemperature
- HourlyPrecipitation
- HourlyPresentWeatherType
- HourlyPressureChange
- HourlyPressureTendency
- HourlyRelativeHumidity
- HourlySkyConditions
- HourlySeaLevelPressure
- HourlyStationPressure
- HourlyVisibility
- HourlyWetBulbTemperature
- HourlyWindDirection
- HourlyWindGustSpeed
- HourlyWindSpeed
- Sunrise
- Sunset
- DailyAverageDewPointTemperature
- DailyAverageDryBulbTemperature
- DailyAverageRelativeHumidity
- DailyAverageSeaLevelPressure
- DailyAverageStationPressure
- DailyAverageWetBulbTemperature
- DailyAverageWindSpeed
- DailyCoolingDegreeDays
- DailyDepartureFromNormalAverageTemperature
- DailyHeatingDegreeDays
- DailyMaximumDryBulbTemperature
- DailyMinimumDryBulbTemperature
- DailyPeakWindDirection
- DailyPeakWindSpeed
- DailyPrecipitation
- DailySnowDepth
- DailySnowfall
- DailySustainedWindDirection
- DailySustainedWindSpeed
- DailyWeather
- MonthlyAverageRH
- MonthlyDaysWithGT001Precip
- MonthlyDaysWithGT010Precip
- MonthlyDaysWithGT32Temp
- MonthlyDaysWithGT90Temp
- MonthlyDaysWithLT0Temp
- MonthlyDaysWithLT32Temp
- MonthlyDepartureFromNormalAverageTemperature
- MonthlyDepartureFromNormalCoolingDegreeDays
- MonthlyDepartureFromNormalHeatingDegreeDays
- MonthlyDepartureFromNormalMaximumTemperature
- MonthlyDepartureFromNormalMinimumTemperature
- MonthlyDepartureFromNormalPrecipitation
- MonthlyDewpointTemperature
- MonthlyGreatestPrecip
- MonthlyGreatestPrecipDate
- MonthlyGreatestSnowDepth
- MonthlyGreatestSnowDepthDate
- MonthlyGreatestSnowfall
- MonthlyGreatestSnowfallDate
- MonthlyMaxSeaLevelPressureValue
- MonthlyMaxSeaLevelPressureValueDate
- MonthlyMaxSeaLevelPressureValueTime
- MonthlyMaximumTemperature
- MonthlyMeanTemperature
- MonthlyMinSeaLevelPressureValue
- MonthlyMinSeaLevelPressureValueDate
- MonthlyMinSeaLevelPressureValueTime
- MonthlyMinimumTemperature
- MonthlySeaLevelPressure
- MonthlyStationPressure
- MonthlyTotalLiquidPrecipitation
- MonthlyTotalSnowfall
- MonthlyWetBulb
- MonthlyAverageWindSpeed
- CoolingDegreeDaysSeasonToDate
- MonthlyCoolingDegreeDays
- MonthlyNumberDaysWithSnowfall
- HeatingDegreeDaysSeasonToDate
- MonthlyHeatingDegreeDays
- MonthlyNumberDaysWithThunderstorms
- MonthlyNumberDaysWithHeavyFog
- NormalsCoolingDegreeDay
- NormalsHeatingDegreeDay
- ShortDurationEndDate005
- ShortDurationEndDate010
- ShortDurationEndDate015
- ShortDurationEndDate020
- ShortDurationEndDate030
- ShortDurationEndDate045
- ShortDurationEndDate060
- ShortDurationEndDate080
- ShortDurationEndDate100
- ShortDurationEndDate120
- ShortDurationEndDate150
- ShortDurationEndDate180
- ShortDurationPrecipitationValue005
- ShortDurationPrecipitationValue010

- ShortDurationPrecipitationValue015
- ShortDurationPrecipitationValue020
- ShortDurationPrecipitationValue030
- ShortDurationPrecipitationValue045
- ShortDurationPrecipitationValue060
- ShortDurationPrecipitationValue080
- ShortDurationPrecipitationValue100
- ShortDurationPrecipitationValue120
- ShortDurationPrecipitationValue150
- ShortDurationPrecipitationValue180
- REM
- BackupDirection
- BackupDistance
- BackupDistanceUnit
- BackupElements
- BackupElevation
- BackupEquipment
- BackupLatitude
- BackupLongitude
- BackupName
- WindEquipmentChangeDate
- Airport
- Missing_Airports