50.039: Theory and Practice of Deep Learning Project Report
Flight Departure Delay Prediction

Group 18
Lim Jia Hui (1006924) | Data Preprocessing
Pham Hong Quan (1007131) | Model Training
Shrinidhi Arul Prakasam (1007007) | Model Training

1. Introduction

GitHub repository link: https://github.com/Hypernating/50.039_FlightDelayPredictor
Flight delays are often influenced by weather conditions, impacting airlines, passengers, and airport operations. This project aims to develop a deep learning model that predicts flight departure delays (minutes) using weather data as hourly time series data. By analyzing historical weather patterns, including air temperature, wind speed and precipitation, the model will capture complex relationships between weather conditions and delays. Leveraging deep neural networks, this approach seeks to enhance forecasting accuracy, enabling airlines and airport authorities to make proactive decisions, improve scheduling efficiency, and minimize disruptions caused by adverse weather.

2. Dataset

We combined weather and flight data, focusing on Delta Air Lines Inc. flights departing from John F. Kennedy International Airport (JFK) from 2016 to 2021.
Data: https://drive.google.com/drive/folders/1FBSiQEvIoivnwkyswmi7aRI7V1qwlld5?usp=sharing

Weather Data

- Source: National Oceanic and Atmospheric Administration (https://observablehq.com/@observablehq/noaa-weather-data-by-major-u-s-city)
- Original Features: station, date, air_temp, dew_temp, sea_level_pressure, wind_direction, wind_speed_rate, sky_condition, precip_1hour, precip_6hour
- Engineered Features: Rounded Datetime (derived from date)

Flight Data

- Source: Bureau of Transportation Statistics (https://www.transtats.bts.gov/ontime/Departures.aspx)
- Original Features: Carrier Code, Date (MM/DD/YYYY), Flight Number, Tail Number, Destination Airport, Scheduled departure time, Actual departure time, Departure delay (Minutes)
- Engineered Features: Schedule Departure Datetime (derived from Date (MM/DD/YYYY) and Scheduled departure time), Rounded Datetime (derived from Schedule Departure Datetime)

3. Data Preprocessing

To prepare the datasets for training, we applied the following preprocessing steps:

1. Feature Selection
   - Dropped irrelevant features (station)
   - Dropped features with excessive missing data (precip_6hour)

2. Date Format Standardization
   - Standardized inconsistent date formats across the datasets to enable accurate time-based merging (Rounded Datetime)

3. Time Alignment
   - Rounded each flight's scheduled departure time down to the nearest hour to align with the hourly weather data (Rounded Datetime)

4. Handling Missing Values
   - Set negative precipitation values to 0 (precip_1hour)
   - Filled missing numerical weather data using linear interpolation, followed by backward fill, to preserve continuity in the time series weather data (air_temp, dew_temp, sea_level_pressure, wind_direction, wind_speed_rate, sky_condition, precip_1hour)

5. Data Normalization
   - Applied Min-Max Scaling to normalize numerical weather features to a 0–1 range, to improve model training stability and convergence (air_temp, dew_temp, sea_level_pressure, wind_direction, wind_speed_rate, sky_condition, precip_1hour)

6. Feature Engineering

To prepare the datasets for modeling, we implemented an extensive preprocessing pipeline that transforms raw flight and weather data into structured features suitable for deep learning:

We enriched the dataset with temporal features extracted from scheduled departure times. This included day of week, month, and hour components, which were subsequently one-hot encoded. Hours were additionally grouped into meaningful periods (morning, afternoon, evening, night) to capture time-of-day patterns. We also incorporated a binary indicator for US federal holidays to account for seasonal travel surges.

Another inclusion is historical performance metrics. For each flight, we calculated average historical delays by flight number, destination airport, and carrier. We also incorporated temporal patterns by adding average delays by hour, day of week, and month. These features provide the model with important context about systemic patterns in flight operations.

For weather data processing, we constructed fixed-length sequences capturing 5 hours of conditions prior to scheduled departure. The sequence construction function extracts key meteorological variables: air

temperature, dew temperature, sea level pressure, wind direction, wind speed rate, sky condition, and hourly precipitation. For flights with incomplete weather histories, we implemented appropriate padding strategies to ensure consistent sequence dimensions. Additionally, we calculated weather trend features by measuring the rate-of-change for each parameter, capturing the directional movement of weather conditions approaching departure time.

All numerical features were normalized using StandardScaler to improve model training. Weather trends were scaled independently, and we applied separate scaling to flight delay values while preserving the ability to convert back to the original scale for evaluation.

Finally, we created a custom PyTorch Dataset class that efficiently handles both static features and sequential weather data. This class returns dictionary-like samples with 'static', 'weather_seq', and 'target' keys, converting all data to PyTorch tensors with appropriate data types. The dataset was split into training (80%) and testing (20%) sets, and we created DataLoader objects with appropriate batch sizes for efficient training.

4. Model

Model 1: CNN with LSTM

1. Convolutional Neural Networks with dense layers and pooling (CNNs) for feature extraction
    - CNN layers will be applied to detect short term variations in weather conditions like sudden temperature changes or wind gusts
    - 2D convolutional filters will extract local weather patterns before passing the data into sequential models
    - Dense connections allowing better capturing of long-term trends and mitigate the vanishing gradient problem
    - Pooling reduces dimensions of features while retaining crucial information, reducing computation and memory required as well as increasing invariance of features
    - 3 convolutional blocks with batch normalization, ReLU and dropout
    - Input shape : (1, 5, 7) - 1 channel, 5 time steps, 7 weather features
    - A skip connection from the first conv layer to the output of the third, to preserve lower-level features
    - Uses global average pooling and two fully connected layers to output a single predicted delay
2. Dataset Preparation (for CNN)
    - Input : weather dataframe and a delay series (from flight data)
    - Output : Pairs of input weather sequences and normalized delay values
    - Normalize the weather features and target (delay) using StandardScaler
    - Construct sequences of 5 consecutive time steps (shape : 5 x 7 per sample)
    - Each sequence becomes a training input, and the next delay value becomes the target
3. Long Short-Term Memory (LSTM) for capturing long-term dependencies
    - LSTM networks will capture long-term dependencies in hourly weather data, identifying persistent weather patterns leading to delays

- stacked, bidirectional LSTM model : 3 LSTM layers with hidden sizes of 100, 60, and 50
- After LSTM layers, applies global average pooling across time, then batch normalization, dropout, and a final fully connected layer

Prediction and Decision Layer

1. Batch Normalization: stabilizes training and improves generalization to unseen weather conditions
2. Output layer with linear activation: predicts the estimated departure delay in minutes
3. Global average pooling (CNN): converts spatial weather feature maps into a vector of size equal to the number of channels by averaging each feature map
4. Global average pooling (LSTM): aggregating the bi-directional LSTMs outputs over all time steps into a single representation for each sequence

Model 2: LSTM with Attention

The model processes weather data through a multi-layer LSTM (Long Short-Term Memory) network that excels at capturing temporal dependencies in sequential data. This allows the model to detect patterns in how changing weather conditions affect flight delays, such as rapid shifts in temperature or evolving precipitation. We implemented a bidirectional approach with configurable hidden dimensions to capture both forward and backward temporal dependencies in the weather sequence.

A critical innovation in our architecture is the attention mechanism that enables the model to focus on the most relevant time steps in the weather sequence. Rather than treating all weather observations equally, the attention layer learns to assign importance weights to different time steps, allowing the model to emphasize periods of weather that are most predictive of delays. This is implemented as a sequence of linear transformations with a tanh activation and softmax normalization to produce attention weights.

For static flight features, we designed a deep neural network with multiple fully-connected layers. This network gradually refines the representation of flight information through ReLU activations and progressively decreasing layer dimensions. To prevent overfitting, we incorporated batch normalization after each layer and applied dropout regularization with a rate of 0.3. This ensures the model generalizes well to unseen flight configurations rather than memorizing training examples.

The architecture fuses these two feature types by concatenating the output of the static feature network with the context vector produced by the attention mechanism. This combined representation then flows through additional fully-connected layers that learn to interpret the interaction between weather patterns and flight characteristics. The final prediction layer outputs a single continuous value representing the predicted delay in minutes.

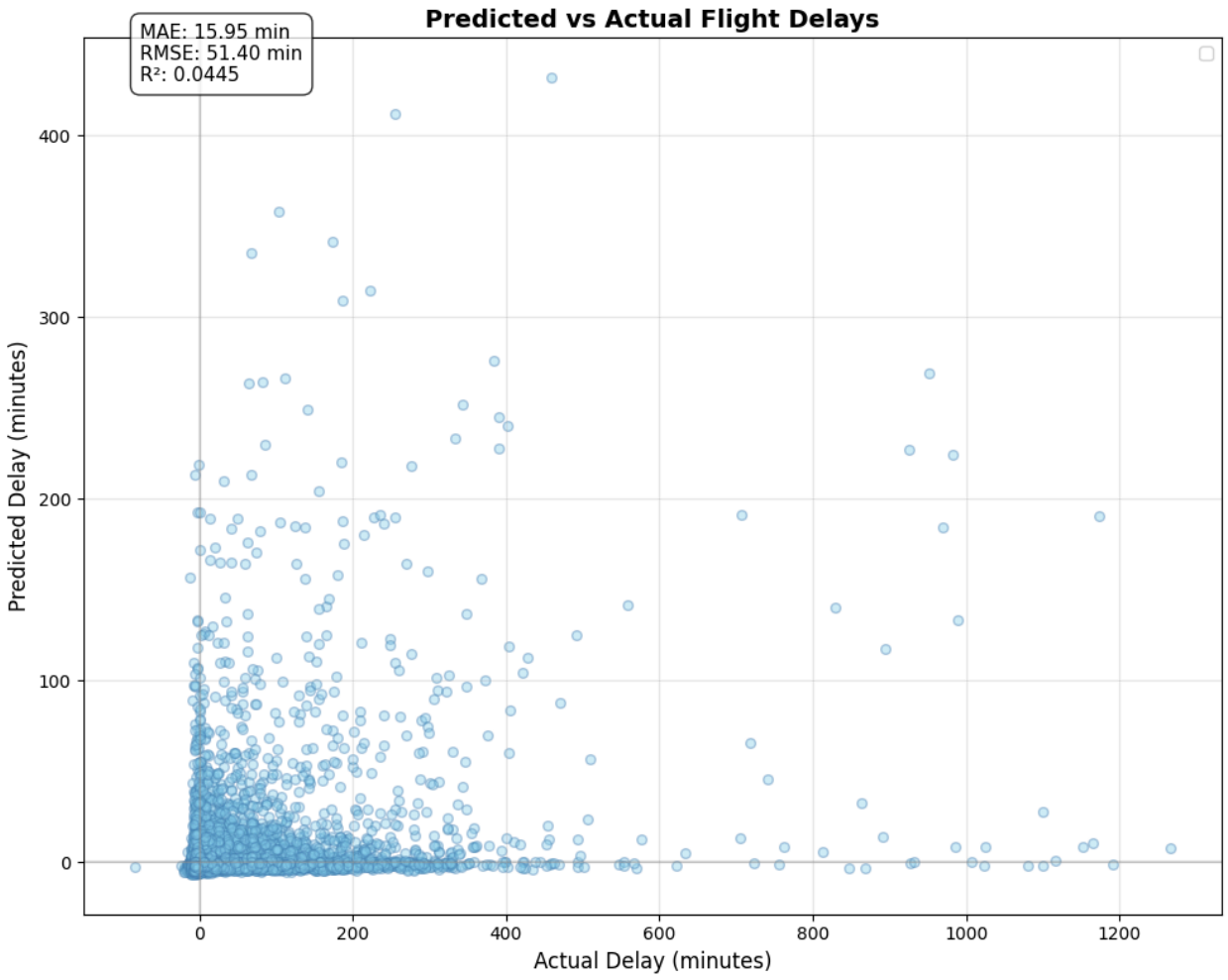
During the forward pass, the model processes static features and weather sequences in parallel. The LSTM processes the weather sequence and generates hidden states for each time step. The attention mechanism then computes weights for these hidden states and produces a context vector. This context vector is concatenated with the processed static features, and the combined representation is fed through the final prediction layers to produce the delay estimate.

5. Evaluation

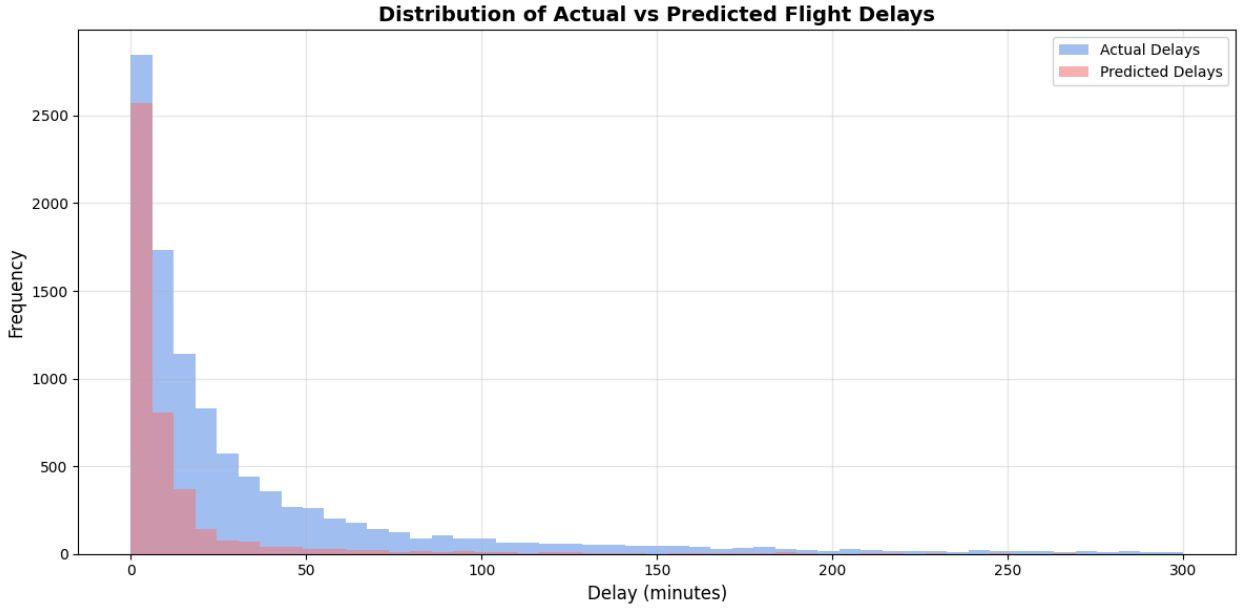
Model 2:

We evaluated our flight delay prediction model using standard regression metrics and created visualizations to analyze its performance patterns. Our evaluation reveals both strengths and limitations of the current approach.

To visualize model performance, we created two primary visualizations. The first is a scatter plot showing predicted versus actual flight delays. This was generated using matplotlib by plotting each flight's actual delay on the x-axis and its predicted delay on the y-axis. We added a diagonal line representing perfect prediction, along with performance metrics in the top-left corner for reference.



The second visualization is a dual histogram comparing the distribution of actual delays versus predicted delays. This visualization helps us understand how well the model captures the overall distribution of delays.



Our evaluation reveals several important patterns in model performance. The scatter plot shows that the model struggles significantly with predicting larger delays (over 100 minutes). While it performs reasonably well for on-time or slightly delayed flights, there is a clear tendency to underpredict severe delays. This is evident in the scatter plot where predictions rarely exceed 200 minutes even when actual delays extend to 1200 minutes.

The distribution histogram further confirms this limitation, showing that the model fails to capture the long tail of the delay distribution. The actual delay distribution extends well beyond 50 minutes with a long tail, while predicted delays are heavily concentrated below 30 minutes. This indicates that while the model captures the central tendency of delays, it struggles with the variability and extreme values that are characteristic of flight delay data.

Several factors can contribute to these limitations: class imbalances between on-time or slightly delayed flights compared to severely delayed flights, delay factors that might not be related to weather at all (such as mechanical issues or downstream effects from other delayed flights) or weather sequence separated by hour might not capture fine-grained variances in weather.

Despite these limitations, the model's MAE of approximately 16 minutes indicates that it provides useful predictions for the majority of flights, which experience relatively minor delays. For operational planning, this level of accuracy could still provide valuable information for short-term scheduling and resource allocation.

6. Conclusion

Our research developed a neural network model that predicts flight delays by combining static flight information with sequential weather data. The model architecture integrates an LSTM-based weather processor with attention mechanisms and a deep neural network for static features, capturing both temporal weather patterns and flight-specific variables.

Evaluation revealed that the model achieved reasonable accuracy for minor delays with an MAE of 15.95 minutes, but struggled significantly with predicting extreme delays exceeding 100 minutes. This limitation is evident in both the scatter plot analysis and distribution comparison, where the model tends to underpredict severe delays and fails to capture the long tail of the delay distribution.

Several factors contribute to these performance limitations, including class imbalance in the training data, unpredictable operational factors not captured in our dataset, insufficient weather sequence length, and the complex non-linear relationship between conditions and extreme delays. The model's R^2 score of 0.0445 indicates limited explanatory power over the entire range of delay variability.

Despite these challenges, the model provides valuable predictions for the majority of flights experiencing moderate delays, making it useful for short-term operational planning and resource allocation. Future improvements should focus on addressing class imbalance through techniques like oversampling or weighted loss functions, incorporating additional data sources related to major delay causes such as mechanical issues or crew availability, extending the weather sequence length to capture developing weather systems, and exploring more sophisticated model architectures designed to better handle extreme events.