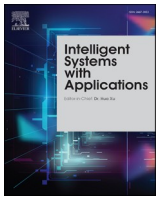
Intelligent Systems with Applications 30 (2026) 200656

Contents lists available at ScienceDirect

Intelligent Systems with Applications

journal homepage: www.journals.elsevier.com/intelligent-systems-with-applications



Deep liquid neural network for prediction of weather-impacted flight delay

Desmond B. Bisandu *, Majuran Chandrakumar, Irene Moulitsas *

Scientific Computing and Artificial Intelligence Laboratory, Centre for Computational Engineering Sciences/DARTeC, Cranfield University, College Road, MK43 0AL, UK

ARTICLE INFO

Keywords:
Flight delay prediction
Deep learning
Liquid neural networks
Long short-term memory
Multi-layer perceptron
Class imbalance
SMOTE

ABSTRACT

In this paper, we developed and analysed a model for predicting flight delays, focusing on those caused by weather conditions. Flight delays pose a significant problem for airlines, as the growth of air traffic often results in financial losses and passenger inconvenience. Unlike several previous research papers, this study provides a more detailed analysis of delays, distinguishing between weather-related and non-weather-related delays. Deep Learning (DL) models, including the proposed Liquid Neural Networks (LNN), Long Short-Term Memory (LSTM), and Multi-Layer Perceptron (MLP), were utilised to make predictions. We compare the LNN model, which has not been used previously for flight delay prediction, with the other two DL models. For this research, we employed two reliable datasets: flight data from the Bureau of Transportation Statistics and weather data from the Weather Underground site. Our methodological approach is comprehensive in addressing challenges encountered with the data. One major challenge was managing class imbalance, which can affect prediction accuracy, using the Synthetic Minority Over-sampling Technique algorithm. The selection of features for training and evaluation was based on mutual information scores and Pearson correlation coefficients. Hyperparameter optimisation was carried out systematically using grid search and trial-and-error. Models were evaluated with standard performance metrics, including precision, recall, F1-score, accuracy, and a confusion matrix. The results reveal that the LNN model outperformed the other DL models in predicting flight delays, particularly for the two classes related to delay occurrence. Overall, all three models produced excellent results in predicting on-time flights and acceptable outcomes when forecasting delays due to non-weather factors. However, the LNN model exhibited computational limitations when predicting weather-related delays, with notably lower scores for this class and significantly longer training durations. Despite these limitations, the LNN model demonstrated promising performance compared to prior models, even surpassing them on some metrics. These findings indicate that the LNN model has great potential for future flight delay prediction.

1. Introduction

Over the last few decades, the airline industry has experienced constant change. In fact, it has expanded rapidly, mainly because of globalisation. Growing demand for air travel fuels this expansion. According to the International Civil Aviation Organisation (ICAO), the number of air passengers rose from 1.67 billion in 2000 to over 4.5 billion in 2019 (Air transport, 2022). This increase in air passengers is clearly shown in Fig. 1.

Fig. 1 illustrates a decline in the number of air passengers in 2020. We can attribute this decline to the spread of COVID-19 worldwide and to the implementation of containment measures and travel bans. Despite this decrease, passenger numbers recovered in subsequent years, starting from 2021. By 2024, some experts anticipate passenger numbers will surpass pre-COVID-19 levels. This remarkable growth is also linked to a significant increase in flights, resulting in large volumes of air traffic. For example, the flight-tracking service Flightradar24 (Flight tracking statistics, 2024) reports that over 100,000 commercial flights have operated daily worldwide since 2021, as shown in Fig. 2.

Fig. 2 only considers flights tracked by Flightradar24. Therefore, this traffic density can complicate flight management and slightly raise the risk of delays. According to the Bureau of Transportation Statistics (BTS) (Airline on-time performance & causes of flight delays., 2021), a flight is considered delayed if an aircraft departs or arrives >15 min late. Such delays can range from a few minutes to several hours. Additionally, they negatively affect airlines, airports, and, especially, the passenger experience. In 2023, airlines faced significant difficulties. Indeed, around 20 % of flights were considered delayed according to BTS data (Airline on-time statistics & delay causes, 2024), as shown in Table 1.

Furthermore, these delays result in significant extra costs for airlines.

* Corresponding authors.
E-mail addresses: desmond.bisandu@cranfield.ac.uk (D.B. Bisandu), i.moulitsas@cranfield.ac.uk (I. Moulitsas).

https://doi.org/10.1016/j.iswa.2026.200656
Received 8 July 2025; Received in revised form 27 October 2025; Accepted 23 March 2026
Available online 24 March 2026
2667-3053/© 2026 The Authors. Published by Elsevier Ltd. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

According to a report by AirHelp (Cost of disrupted flights to the economy, 2022), the annual expenses of flight delays in 2022 are estimated to range from $26 billion to $32 billion in the US and from $20 billion to $25 billion in Europe, as shown in Table 2.

Such costs include fuel, staff, and compensation for affected passengers (Cost of disrupted flights to the economy, 2022). From the passengers' perspective, such delays cause significant inconveniences, such as missed connections, altered travel plans, and wasted time, thereby affecting overall satisfaction. However, data imbalance remains a common problem in predicting flight delays. In fact, most flights arrive punctually, with only a small proportion experiencing delays, regardless of the year. For example, according to BTS data (Airline on-time statistics & delay causes, 2024), flight statistics for 2023 show that 78.34 % of flights arrive on time, while 20.12 % are delayed, as illustrated in Fig. 3.

This notable difference in samples between on-time and delayed flights creates a significant imbalance in the flight data. This imbalance can pose challenges for the prediction models. Indeed, it can make it harder for the models to recognise any potential patterns in the delayed flights. As a result, the models may become biased. Specifically, they may underpredict delays because of the influence of on-time flights. Therefore, it is essential to employ specific techniques, which will be explained in detail in the literature review and the methodology of this paper. Flight delays can be categorised into various types (Airline on-time performance & causes of flight delays., 2021; Understanding the reporting of causes of flight delays & cancellations, 2024). Below is a list of different causes of delays:

Air carrier delay (Airline on-time performance & causes of flight delays., 2021; Understanding the reporting of causes of flight delays & cancellations, 2024): Such delays result from internal operational problems such as unplanned aircraft maintenance, staff issues or logistical errors.

Extreme weather delay (Airline on-time performance & causes of flight delays., 2021; Understanding the reporting of causes of flight delays & cancellations, 2024): These delays are due to weather conditions unfavourable to the execution of a flight, such as tornadoes, storms or blizzards.

National Aviation System (NAS) delay (Airline on-time performance & causes of flight delays., 2021; Understanding the reporting of causes of flight delays & cancellations, 2024): Such delays result from operational factors related to the national aviation system. Such factors may include non-extreme weather conditions, airport operations, high traffic volumes and air traffic control.

Late-arriving aircraft delay (Airline on-time performance & causes of flight delays., 2021; Understanding the reporting of causes of flight delays & cancellations, 2024): These delays result from the late arrival of a previous flight operating with the same aircraft, leading inevitably to the late departure of the current flight.

Security delay (Airline on-time performance & causes of flight delays., 2021; Understanding the reporting of causes of flight delays & cancellations, 2024): Such delays may be caused by additional security measures or other security incidents. These include evacuating a terminal and reboarding an aircraft due to a security breach or a malfunction in control equipment.

In this paper, we examine weather-related flight delays for various reasons. Although they represent a smaller percentage, weather delays are crucial due to their unpredictability and potential severity. Weather conditions can cause significant disruptions, affecting individual flights and the entire air traffic system. Weather data typically includes temperature, wind speed and direction, precipitation, and atmospheric pressure. However, it is essential to recognise the challenge posed by weather data. The reliability of weather data can create additional difficulties for prediction models. Weather forecasting is often prone to uncertainty due to its unpredictable nature (How accurate are weather forecasts? 5 reasons for inaccuracy, 2024). Moreover, despite recent technological advances, uncertainty still exists in weather data, though to a lesser extent. This uncertainty can impact the accuracy of flight delay prediction models. Therefore, this should be considered when analysing results from various models. The uncertainty in one-day forecasts is about 2–4 % (How accurate are weather forecasts? 5 reasons for inaccuracy, 2024). For three-day forecasts, it rises to 10 %, and it continues to increase for long-term forecasts (How accurate are weather forecasts? 5 reasons for inaccuracy, 2024) Fig. 4

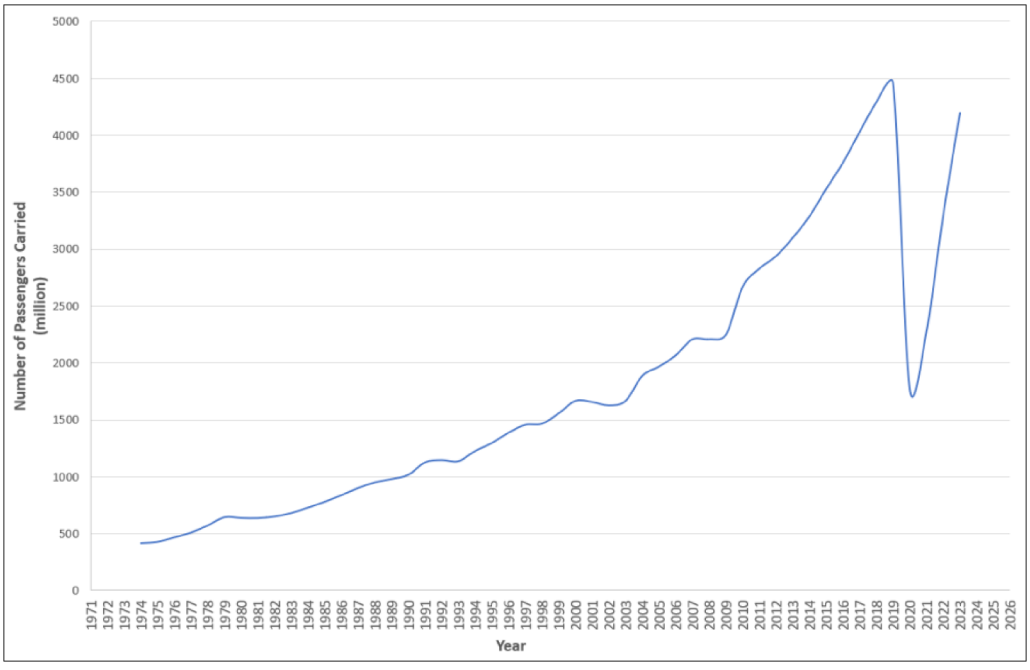


Fig. 1. World air passenger traffic from 1974 to 2023 (Air transport, 2022).

1.1. Motivations

The motivations for predicting flight delays are diverse and influenced by various factors. These include economic, environmental, operational, and customer experience considerations. The detailed overview of the key drivers emphasises the importance of predicting weather-related flight delays.

One of the main reasons for predicting flight delays is to reduce costs. Flight delays can incur substantial expenses for airlines, including extra fuel, staff wages, passenger compensation, and penalties for missed slots (Anupkumar 2023). For example, the Federal Aviation Administration (FAA) estimated that the yearly cost of delays was $33 billion in 2019 (U.s. passenger carrier delay costs, 2024). Therefore, it is vital to accurately forecast flight delays. Indeed, by anticipating these delays, airlines can plan more efficiently and minimise unforeseen expenses.

Predicting flight delays helps enhance environmental sustainability. We recognise that aircraft on the ground or waiting on runways consume fuel unnecessarily, thereby increasing CO_2 and other pollutant emissions (What exactly is the environmental impact of flight disruptions?, 2023). Therefore, accurate prediction of flight delays could help reduce avoidable emissions and fuel use. Moreover, this is especially important at a time when the aviation industry is actively seeking ways to significantly lower its carbon footprint and comply with stricter environmental regulations (What exactly is the environmental impact of flight disruptions?, 2023).

Furthermore, predicting flight delays can significantly enhance resource management. This advantage can be realised through better flight-schedule optimisation and improved air traffic control, leading to more efficient operations (Mamdouh et al., 2023; Hatıpoglu & Tosun, 2024). Such efficiency also encourages better utilisation of assets and personnel. For instance, based on forecasts, airlines can adjust their staff allocations to prevent crew overwork or underemployment. Similarly, airports can manage their runways and boarding gates more effectively, thereby enabling smoother operations.

Finally, improving the passenger experience is another reason for predicting flight delays. This enhancement in passenger satisfaction is crucial, especially in a competitive market. Flight delays can cause significant inconvenience, such as missed connections, longer wait times, and uncertainty about arrival times. By accurately predicting and communicating these delays, airlines can inform passengers in real time.

Table 1
Flight performance in 2023 (Airline on-time statistics & delay causes, 2024).

On-time Performance	Number of Operations	% of Total Operations
On-time	5702,506	78.34%
Delayed	1464,537	20.12%
Others (Cancelled or Diverted)	111,694	1.54%

Table 2
Impact of flight delays on the US and Europe economies in US dollars in 2022 (Cost of disrupted flights to the economy, 2022).

Delay Impact	USA	Europe
Airline cost (billion)	9–11	8–10
Passenger cost (billion)	12–15	8–10
Spillover cost (billion)	5–6	4–5
Total cost (billion)	26–32	20–25

This enables travellers to alter their flights or reorganise their travel plans (Song et al., 2020; Efthymiou et al., Dec 2018). Additionally, it helps airlines to create a smoother travel experience and foster stronger customer loyalty.

Overall, flight delay prediction addresses airlines' key strategic and operational needs, from reducing costs to enhancing passenger satisfaction. These motivations are interconnected and together contribute to the overall improvement of the airline industry.

1.2. Contributions

The main contribution of this paper is the development of a Deep Learning (DL) model, specifically the Liquid Neural Network (LNN). This model has not been previously used to predict flight delays. Additionally, the paper distinguishes between weather-related delays and other types, enabling a more focused analysis of the factors influencing each delay category. Moreover, advanced techniques are employed to address imbalances in the flight data. Consequently, the model can effectively handle minority classes and deliver more reliable predictions. Overall, the approach presented here offers a new perspective on flight delay analysis and forecasting.

This study provides three main contributions that push the

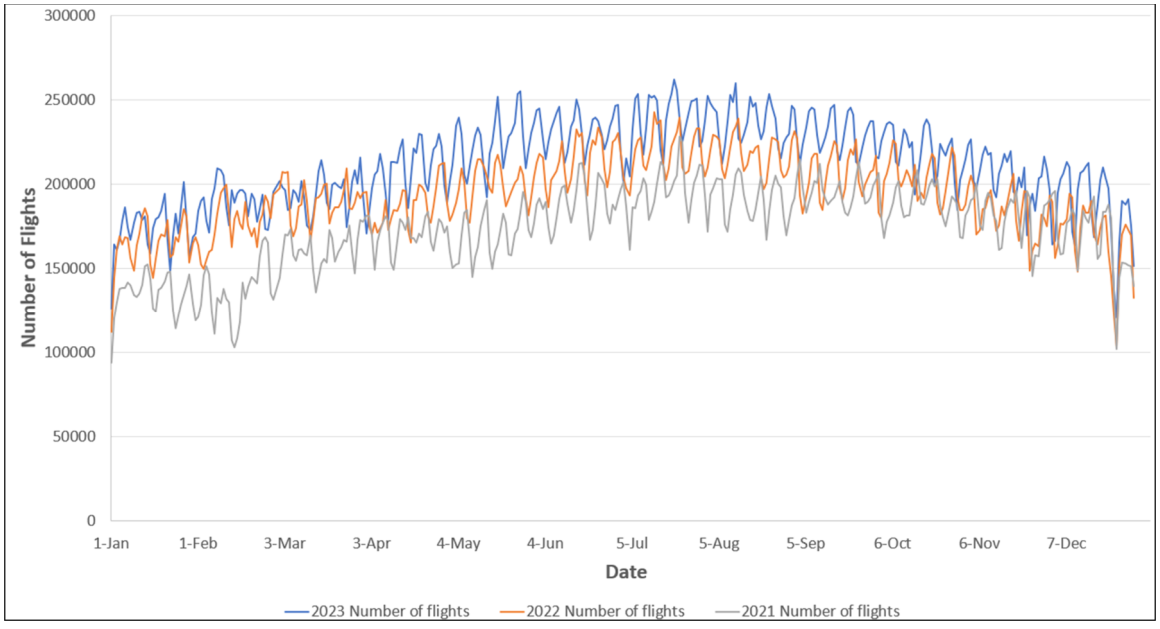
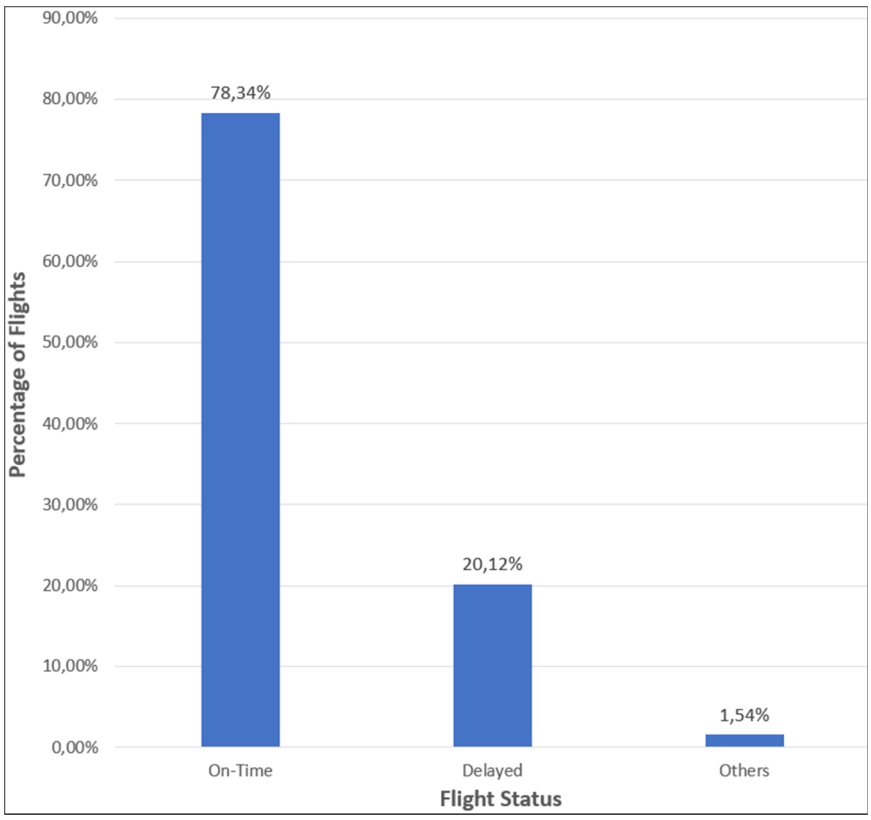


Fig. 2. Total number of flights tracked by Flightradar24 from 2021 to 2023 (Flight tracking statistics, 2024).



. 3. Imbalance in historical flight performance data in 2023 (Airline on-time statistics & delay causes, 2024).

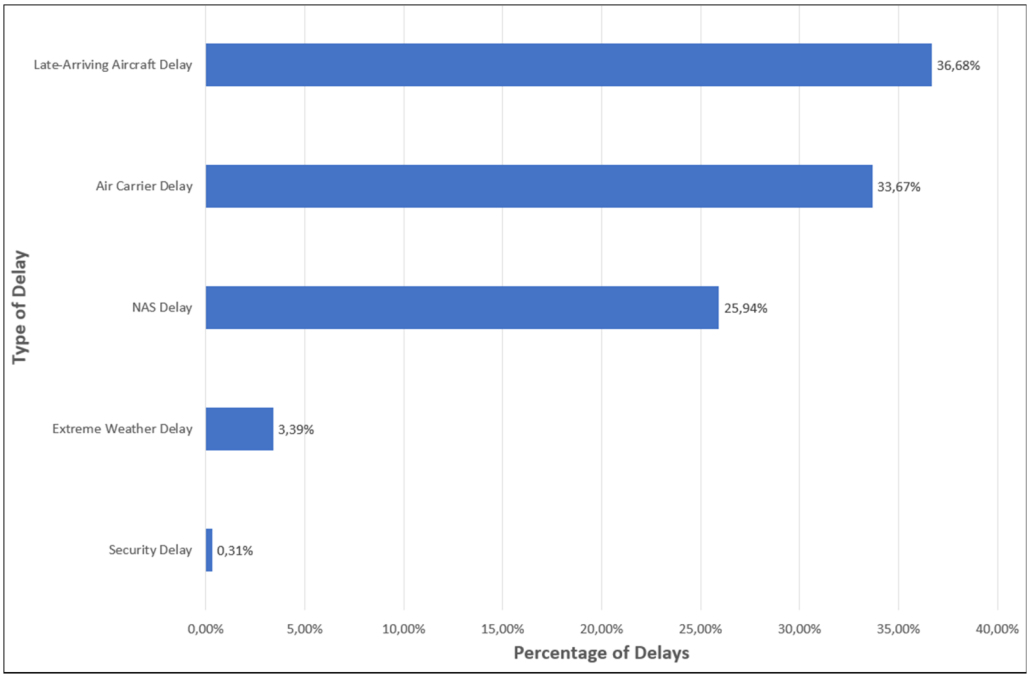


Fig. 4. Distribution of delayed flight types for 2023 (Airline on-time statistics & delay causes, 2024).

boundaries of DL-based flight delay prediction. First, we introduce the Liquid Neural Network (LNN), a new, dynamically adaptable neural architecture inspired by continuous-time models, applied to the aviation sector for the first time. Unlike static models, LNNs offer temporal flexibility and computational efficiency, enabling them to capture complex, evolving patterns in air traffic systems effectively. Second, we create a disaggregated prediction framework that clearly separates weather-related delays from other disruptions, facilitating category-specific learning—an ability missing in earlier methods. Third, we implement advanced data-balancing techniques to systematically tackle the severe class imbalance common in aviation datasets, allowing our model to maintain high predictive performance across both frequent and

rare delay types. Overall, this work paves a new methodological path for intelligent air traffic prediction and presents a highly generalisable neural model with the potential to transform real-time decision support systems in global aviation networks.

The paper is organised as follows: Section 2 reviews the literature; Section 3 describes the proposed methodology; Section 4 discusses the results; and Section 5 concludes with suggestions for future research.

2. Related work

The aviation industry has experienced a significant issue for decades. Flight delays affect airline operations and customer satisfaction. Furthermore, these delays are costing airlines billions of pounds. Consequently, researchers have focused on developing accurate methods to predict flight delays. However, they have consistently faced challenges in these predictions. Nevertheless, recent studies indicate that a trend in flight delay prediction is emerging. Indeed, many researchers have applied and used ML or DL techniques to achieve better performance results.

To investigate the significance of weather data for prediction, S. Choi et al. (Choi et al., 2016) developed ML/DL models and trained and evaluated them with and without weather information. We collected data from the US BTS to obtain historical flight performance data and from the National Oceanic and Atmospheric Administration (NOAA) to get weather data. The data preprocessing methodology was quite conventional: first, cancelled or diverted flights were classified as delayed flights. For data selection, they retained only the most relevant variables for flight delay prediction. They used linear interpolation to handle missing data effectively and encoded categorical variables as numerical values to improve model performance. Finally, they normalised the variables to prevent any feature from dominating the models. The authors implemented models including Decision Tree (DT), Random Forest (RF), AdaBoost, and K-Nearest Neighbour (KNN). To optimise hyperparameters, they used a trial-and-error approach. They also employed 10-fold cross-validation to minimise overfitting. The results indicated that incorporating weather data during training and evaluation significantly enhanced model performance. This finding was corroborated by another study conducted by N. Etani (Etani, 2019), which, despite utilising a different methodology, showed the same trend. The results in Table 3 further confirmed that models using weather data performed better, especially the RF-based model.

Other researchers in this field have recognised the importance of weather data. By integrating weather data, they aimed to improve the accuracy of flight delay predictions. However, despite the apparent imbalance in the data, some researchers have not taken any serious steps to address it. V. Sujay et al. (Sujay et al., 2024) implemented the RF, Gradient Boosting, and Support Vector Machine (SVM) models. They employed a standard methodology, handling missing data with interpolation for categorical variables and mean imputation for numerical variables. They encoded categorical variables into numerical form and normalised the features. To optimise hyperparameters, the authors used grid search, random search, and Bayesian search algorithms. Additionally, to enhance the reliability of the models, they applied cross- validation. The authors achieved very good results; indeed, the proposed model scored at least 92 % across all performance metrics, namely accuracy, precision, recall, and F 1- score. In this way, they presented a method to classify flight delays from on- time flights easily. D. Mavris et al. (Kim et al., 2016) adopted a two- stage methodology centred on a Recurrent Neural Network (RNN) with Long Short- Term Memory (LSTM) architectures. The first stage predicts the day- to- day delay status at the airport, using historical flight performance and weather data for ten major US airports. The second stage refines these predictions for each flight delay by combining the daily delay status with flight performance and weather data, using a fully connected multi- layer neural network. Some regularisation techniques, such as Dropout, are employed to reduce overfitting. Furthermore, the models are trained with stochastic gradient descent and mini- batch algorithms. The authors demonstrated that deep models achieved accuracies of 80. 63 % and 90. 95 % for sequences of seven and nine days, respectively. These results outperform those of simpler models. For individual delays, a multilayer neural network achieved an accuracy of 87. 87.42 %. Additionally, the authors developed a model trained on one airport that could be generalised to over 85 % accuracy for all but one of the other airports.

Table 3
Summary of previous research on flight delay prediction methods/data.

Authors	Methods	Datasets	Time Span
N. Etani (Etani, 2019)	Models: RF, Gradient Boosting, SVM, DT, AdaBoost Imbalance: Balanced parameter set to class weight parameter	Flight: FLIGHTSTATS database Weather: Japan Meteorological Agency	1 Mar 2012 - 3 Dec.2018
V.Sujay et al. (Sujay et al., 2024)	Models: RF, Gradient Boosting, SVM Imbalance: None	Flight ant Weather: APIs and databases	Unknown
D. Mavris et al. (Kim et al., 2016)	Models: RNN with LSTM architectures Imbalance: None	Flight and weather: ten major US airports	Jan. 2010 - Aug. 2015
R. Hendrickx et al. (Hendrickx et al., 2021)	Models: RF, MLP Imbalance: SMOTE, RUS	Flight: Amsterdam Airport Schiphol Weather: METAR	2015 - 2018
Y. Tang (Tang, 2022)	Models: DT, RF, KNN, SVM, Gradient Boosted Tree, Gaussian Naive Bayes, Logistic regression. Imbalance: Weighted performance metrics	Flight: BTS Weather: Weather Underground	Nov. 2019 - Dec. 2020
S. Choi et al. (Choi et al., 2016)	Models: DT, RF, AdaBoost, KNN Imbalance: SMOTE, RUS	Flight: BTS Weather: NOAA	2005 - 2015
S. Kim and E. Park (Kim & Park, 2024)	Models: DT, XGB, SVM, KNN, Logistic Regression, RF, LSTM Imbalance: RUS	Flight: Incheon Airport, BTS Weather: Korean Meteorological Administration, Weather Underground	2010 - 2021

Examining another vital aspect of flight delay prediction, it is important to analyse how data imbalance impacts forecast accuracy. We tested various methods to address data imbalance to determine if they enhance predictive performance. Choi et al. (Choi et al., 2016) developed the RF, AdaBoost, and KNN models. Additionally, to tackle data imbalances, they employed the Synthetic Minority Over-sampling Technique (SMOTE) and Random Under-Sampling (RUS). They found that applying these sampling techniques slightly lowered the overall accuracy of each model. However, the number of correctly predicted flight delays increased. Conversely, the count of accurately predicted on-time flights slightly decreased but remained acceptable. They concluded that using these techniques improves the detection of flight delays while still maintaining adequate performance for classifying on-time flights.

Therefore, the researchers used weather data and various techniques to address data imbalances. They combined these approaches to improve the accuracy of flight delay predictions. This emphasises the need for more advanced data processing methods to achieve more reliable results. Several researchers employed traditional methodologies. In fact, all these researchers encoded their categorical variables into numerical form, for example, using one-hot encoding (Kim & Park, 2024). They also normalised their variables. Missing values were handled either by linear interpolation (Choi et al., 2016) or by removing rows containing missing data (Tang, 2022), as these accounted for <1 % of the dataset. Features were selected based on their importance in predicting flight delays, using methods such as Pearson coefficients (Hendrickx et al., 2021). To address data imbalance, researchers used SMOTE

(Hendrickx et al., 2021; Choi et al., 2016), RUS (Kim & Park, 2024; Choi et al., 2016), or weighted metrics (Tang, 2022). For hyperparameter optimisation, grid search (Hendrickx et al., 2021; Kim & Park, 2024) was employed. Additionally, to improve model reliability, some researchers used cross-validation (Hendrickx et al., 2021; Choi et al., 2016). R. Hendrickx et al. (Hendrickx et al., 2021) implemented RF and MultiLayer Perceptron (MLP) models. They achieved relatively accurate results for departure and arrival delays with both models. However, the precision scores for both models were good, but the F1 scores were very poor. Furthermore, the authors demonstrated the potential impact of imbalance ratios on SMOTE or RUS. Y. Tang (Tang, 2022) developed models including DT, RF, KNN, SVM, Gradient Boosted Tree, Gaussian Naive Bayes, and Logistic Regression. He obtained excellent results, with the DT model achieving the best performance: 0.9778 in accuracy, 0.9777 in precision, 0.9778 in recall, and 97.78 % in F1-score. The RF and Gradient Boosted Tree models also performed well, but the KNN's performance was notably lower than that of the other models.

Furthermore, some researchers found it interesting to train and evaluate models by collecting weather data at different intervals relative to the flight departure time. Indeed, S. Kim and E. Park (Kim & Park, 2024) collected weather data at intervals of 2 h, 4 h, 8 h, 16 h, 24 h, and 48 h before flight departure. They used the DT, RF, SVM, KNN, Logistic Regression, Extreme Gradient Boosting (XGB), and LSTM models. They achieved acceptable results with the RF, DT, and LSTM models. In fact, these models outperformed the others. Additionally, they observed that the models' performance depends on the time interval of the weather data used before departure. As the interval increases, the models' performance decreases. Choi et al. (Choi et al., 2016) collected weather data at the following intervals before flight departure: 0 days, 1 day, and 5 days. They implemented the RF, AdaBoost, and KNN models. They achieved 80.36 % accuracy with the RF model and 71.43 % with the AdaBoost model. However, these two models struggled to predict the delay class. Furthermore, the KNN model achieved an accuracy of only 35.71 %. As in previous research, they found that the time interval between data collection and departure influences model performance. Indeed, when the time interval increases, overall accuracy decreases. For example, the accuracy for 0 days was 80.36 %, but it fell to 26.79 % for 5 days.

In this paper, we propose a DL method that has not been applied to flight delay prediction. We aim to conduct an in-depth study and analyse the reliability of this novel model in terms of prediction performance. Additionally, this model could address the gaps identified in previous research, as discussed in Table 4. Therefore, a comprehensive comparison is carried out between the developed model and several DL models used in earlier studies.

2.1. Liquid neural networks

In the field of artificial intelligence (AI), Neural Networks (NNs) are frequently used to solve very complex problems (Sajid, 2023). The principles of these ML algorithms are based on imitating the structure and operational capabilities of the human brain Sajid (2023). However, these networks have many drawbacks, including their inability to handle real-time data (Sajid, 2023). Consequently, over the past few years, researchers have been exploring innovative ways to enhance the performance of these NNs and overcome several limitations. Thus, a group of researchers at MIT has introduced a new approach called Liquid Neural Network (LNN) (Sajid, 2023).

2.1.1. Fundamental concepts

LNNs are a type of RNN characterised as continuous in time (G. Boesch, 2024; Keary, 2023; Sajid, 2023). These networks can process time-series data and adapt to changing environments. Inspired by the dynamic behaviour of liquids, LNNs are built with a dynamic architecture (Tyagi, 2023). Unlike traditional NNs, where weights and connections are fixed, LNNs have weights and connections that change based

Table 4
Summary of previous research on flight delay prediction results/gaps.

Author	Results	Gaps
N. Etani (Etani, 2019)	- Predictive models based on weather data proved superior to those without weather data - RF with weather data showed the best overall performance with high precision, recall and F1-score, especially for on-time arrival predictions - The learning period plays a major role in the performance of the models, with more extended learning periods producing better results	- Difficulties of models in capturing the complex factors influencing delays - Models tend to overfit - Global accuracy obtained by the models remains relatively low compared to other studies
V.Sujay et al. (Sujay et al., 2024)	- The proposed system achieves 95 % accuracy The proposed system correctly identified 5500 on-time flights and 4700 delayed flights, with just 500 misclassified on-time flights and 300 wrongly predicted delayed flights	- Adaptability of the proposed system to deal with real-time data and its deployment in different airline environments - Reliance on historical data and sensitivity to changes in the aviation environment.
D. Mavris et al. (Kim et al., 2016)	- Day-to-day delay status models: around 78 % accuracy for a delay threshold of 15 min and around 90 % accuracy for a delay threshold of 30min - Individual flight delay models: high accuracy, ranging from 85,32 % to 87,42 % - The accuracy of the day-to-day model for different airports ranges from 71.34 % to 91.81 %	- Increased complexity of deep RNNs, leading to convergence issues - Less accurate predictions for lower delay thresholds - Variability in model performance across different airports - Requires large amounts of data for optimal performance
R. Hendrickx et al. (Hendrickx et al., 2021)	- RF: precision of 66 % for departure delays and 71,3 % for arrival delays - MLP: precision of 61,4 % for departure delays and 69,2 % for arrival delays. - Sampling techniques improve the performance of the models in terms of F1-score, while reducing false positives.	- No validation of the model trained on one airport on others to ensure generalisability. - Need to explore other ML algorithms to validate the approach. - Difficulty integrating these predictive models into existing airport management systems due to accuracy below 80 %.
Y. Tang (Tang, 2022)	- DT model showed the best performance of the seven algorithms tested, with accuracy, precision, recall and F1-score close to 97 %. - Tree-based ensemble classifiers tend to perform better than other basic models.	- No use of sampling techniques such as SMOTE, which can improve model performance. - Data is limited to a single year, and extending the dataset to a longer period would make obtaining a more robust model possible.
S. Choi et al. (Choi et al., 2016)	- RF model showed the best performance with an accuracy of 80,36 % and an AUC of 0.68 - Using SMOTE improves the identification of delayed flights despite a drop in global accuracy - Integrating weather data considerably improves the accuracy of predictions compared to simply relying on flight data	- The uncertainty of the weather forecasts influences the performance of the models - Imbalanced data remains a challenge for accurate prediction even with the use of SMOTE/RUS - Classifiers tend to overfit, particularly for KNN and DT models - Models do not consider delays caused by non-weather factors, which impacts the accuracy of delay prediction

(continued on next page)

Table 4 (*continued*)

Author	Results	Gaps
S. Kim and E. Park (Kim & Park, 2024)	- High accuracy in short-term predictions for the optimal models of each airport: 74.9 % accuracy for the Incheon model, 85.2 % accuracy for the JFK model and 78.5 % accuracy for the MDW model in 2-hour forecasts. - LSTM models outperform the others in most of the evaluation measures for JFK and MDW - The Random Forest model showed the highest accuracy for ICN.	- Limited generalisation of results due to regional and national variability in weather conditions - Inferior performance related to the missing feature for the Incheon airport model

on current and previous input (Keary, 2023). This structure is inspired by the nervous system of a microscopic worm, which has only 302 neurons but exhibits dynamic and complex behaviour (G. Boesch, 2024; Sajid, 2023).

2.1.2. Advantages and limitations

LNNs offer many benefits. These neural networks are highly adaptable (Tyagi, 2023; Keary, 2023), capable of adjusting to changes in input data. This trait makes them particularly useful for real-time applications. Additionally, LNNs show significant improvements in robustness compared with classic NNs (Tyagi, 2023; Keary, 2023), as well as better handling of disturbances and noise. Their dynamic architecture allows these networks to focus solely on relevant information. Finally, LNNs require fewer computational resources than traditional NNs (Keary, 2023), which can be attributed to their smaller number of neurons and dynamic structure.

However, LNNs also have some limitations. LNNs struggle with static or fixed data, unlike classic NNs (Keary, 2023; G. Boesch, 2024). Furthermore, LNNs struggle to reach optimal weights during training due to the vanishing gradient problem (Keary, 2023; G. Boesch, 2024; Sajid, 2023). This makes it challenging to ensure their stability and convergence. As a result, their ability to learn long-term dependencies is limited. Additionally, tuning LNN parameters can be complex (Keary, 2023; G. Boesch, 2024; Sajid, 2023), often costly and time-consuming. If the settings are not correctly adjusted, LNNs are unlikely to perform as effectively as they could.

2.1.3. Applications

LNNs can be applied across many fields due to their ability to handle time series data. They are used in the medical sector (G. Boesch, 2024) for diagnoses based on medical images. In fact, LNNs are suitable for specific image processing tasks, such as object tracking and recognition. Additionally, LNNs can be utilised for autonomous drones (G. Boesch, 2024; Keary, 2023). A recent study by MIT researchers demonstrates that LNNs outperform other neural networks in this area. They enable drones to navigate more effectively in unfamiliar environments. LNNs are also applicable to self-driving cars (G. Boesch, 2024; Keary, 2023). Using LNNs instead of other AI models offers significant advantages: they can adapt to new scenarios beyond their training data, making these vehicles more reliable and reducing accident risk. Lastly, LNNs can be employed in natural language processing (NLP) (G. Boesch, 2024; Sajid, 2023). Traditional deep learning models generally require large training datasets for NLP tasks like sentiment analysis and machine translation, which can be very time-consuming. Therefore, incorporating LNNs into NLP can significantly reduce computational costs.

2.2. Long short-term memory

LSTMs, introduced by Hochreiter and Schmidhuber in 1997, are an advanced form of RNNs designed for processing sequential data (What is lstm – long short term memory?, 2024; Saxena, 2024). The core principle of these networks is their ability to store information over extended periods (What is lstm – long short term memory?, 2024). In fact, they can also handle long-term dependencies through their distinctive memory cell architecture (What is lstm – long short term memory?, 2024). This architecture includes three gates (What is lstm – long short term memory?, 2024; Saxena, 2024):

Input gate: controls the entry of new information into the memory cell.
Forget gate: identifies which information in the memory cell is removed.
Output gate: regulates the flow of information out of the memory cell.

All these gates control the flow of data into memory cells. As a result, LSTMs can selectively retain or forget information over long sequences.

2.2.1. Advantages and limitations

LSTMs offer several key advantages over traditional RNNs. In fact, LSTMs can handle long and complex data sequences by using their gate functions to retain useful information and discard irrelevant details (Srivatsavaya, 2023). Additionally, LSTMs significantly enhance both learning stability and performance. This is evident in the substantial reduction of the vanishing and exploding gradient problems faced by traditional RNNs (Srivatsavaya, 2023). Furthermore, LSTMs are adaptable, making them highly suitable for a wide range of applications that require processing sequential data.

However, LSTMs also face some significant limitations. Their architecture requires more computational power and memory than traditional RNNs (Srivatsavaya, 2023). Additionally, training LSTM models can be quite resource-intensive, often necessitating large datasets and extending training durations (Srivatsavaya, 2023). The complexity of LSTM models can make them very difficult to interpret and understand, leading to issues with model transparency (Srivatsavaya, 2023).

2.2.2. Applications

LSTMs can be used across many fields due to their ability to handle temporal data sequences. They are especially valuable for speech-related tasks (What is lstm – long short term memory?, 2024). In fact, LSTMs perform well in speech transcription and recognition (What is lstm – long short term memory?, 2024). They can recognise speech patterns and convert them into text. Additionally, they can be utilised for various video processing tasks such as object detection (What is lstm – long short term memory?, 2024). The architecture of LSTMs is well-suited for analysing video and extracting relevant information. LSTMs are also employed for time series forecasting, such as weather prediction, as they can recognise patterns in these types of data (What is lstm – long short term memory?, 2024).

2.3. MultiLayer perceptron

MLPs are a type of feedforward neural network developed in the 1960s. Such NNs were designed based on the structure of the human brain Shah (2024). Moreover, in this network, all the nodes are interconnected with nodes in other layers (What is multilayer perceptron?, 2023; Shah, 2024; What is multilayer perceptron?, 2024). The architecture of MLPs is composed of three types of layers:

Input layer (What is multilayer perceptron?, 2023; What is multilayer perceptron?, 2024): This layer receives the input signals and sends them to the hidden layers.
Hidden layers (What is multilayer perceptron?, 2023; What is multilayer perceptron?, 2024): One or many hidden layers process the information received from the input layer. Besides, the number of hidden layers depends on the specific task. In addition, each neuron

in the hidden layers employs a non-linear activation function, such as the Rectified Linear Unit (ReLU).
Output layer (What is multilayer perceptron?, 2023; What is multilayer perceptron?, 2024): This layer returns the result of the model. The number of nodes in this layer depends on the problem type.

The MLP principle is based on the backpropagation algorithm (What is multilayer perceptron?, 2023). The main aim of this algorithm is to adjust the weights of the connections between neurons (What is multilayer perceptron?, 2023; What is multilayer perceptron?, 2024). It significantly minimises errors in the model's predictions (What is multilayer perceptron?, 2023; What is multilayer perceptron?, 2024).

2.3.1. Advantages and limitations

MLPs offer some notable benefits. They are well-suited for handling non-linear problems (What is multilayer perceptron?, 2024). Additionally, these models are highly adaptable, as they can be used for a wide variety of tasks (What is multilayer perceptron?, 2023). MLPs can manage high-dimensional data (What is multilayer perceptron?, 2024), enabling them to solve complex problems. Because MLPs are robust against noise, they can efficiently handle missing values.

However, MLPs also have many limitations. Training MLPs can require substantial computational resources, especially for deep architectures and large datasets (What is multilayer perceptron?, 2024). Additionally, MLP models are highly susceptible to overfitting the training data (What is multilayer perceptron?, 2023). Indeed, if the model's structure is too complex, it may fail to generalise to unseen data. Finally, interpreting MLP predictions can be difficult due to their complex architecture (What is multilayer perceptron?, 2023).

2.3.2. Applications

MLPs are widely used across many fields because of their ability to identify complex relationships in data. They can notably be used in image processing tasks such as image recognition, although they are not the optimal model (Multilayer perceptron (mlp) 2024). They can also be applied to time series forecasting to predict financial trends or weather conditions by recognising complex patterns (Multilayer perceptron (mlp) 2024). Finally, they are also suitable for NLP tasks. Indeed, MLPs are effective for character recognition and can achieve promising results in tasks such as sentiment analysis and machine translation (Multilayer perceptron (mlp) 2024).

3. Methodology

As previously stated, the objective of this paper is to develop DL models capable of predicting weather-impacted flight delays. To address this problem, we followed a rigorous methodology illustrated in Fig. 5. This methodology includes several key steps, from problem definition, data collection and preprocessing to evaluating the prediction models.

3.1. Problem definition

Predicting flight delays due to weather conditions is a complex yet essential task in the aviation industry, especially for airlines. To solve this problem, we used a multiclass classification approach, aiming to predict three separate classes, as discussed when creating the target variable in Section 3.3.4.

Furthermore, it is crucial to distinguish delays caused by weather conditions from those caused by other factors. This can improve prediction accuracy. Additionally, this distinction enables tailored solutions to address each type of delay. However, this issue presents many challenges. One of these is data imbalance, as flight delays are much less common than on-time flights. Consequently, preprocessing and feature selection are essential to ensure the data is high-quality and relevant. Lastly, evaluating DL models requires strict criteria to ensure they are both accurate and robust.

By defining the problem this way, this paper focuses on developing accurate and robust models for predicting flight delays. We considered the inherent data complexities, such as delays and features.

3.2. Data collection

To carry out this paper on predicting flight delays due to weather conditions, we collected data from two primary sources: BTS and Weather Underground. The collection period covers 2010 to 2023, excluding 2019 and 2020. Indeed, these two years were excluded from

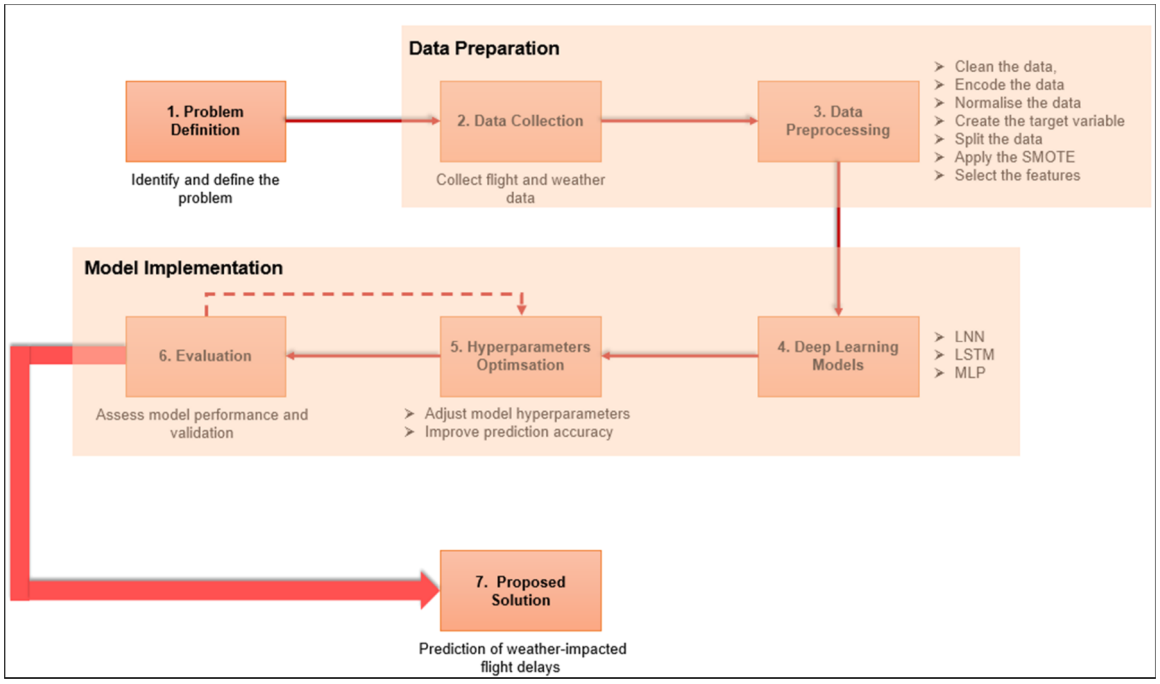


Fig. 5. Proposed methodology.

the analysis due to the exceptional disruptions caused by the COVID-19 pandemic. We utilised these two sources because of their reliability and relevance, as demonstrated by a previous study (Kim & Park, 2024). In addition, we limited our survey to data from a single airport, as delay conditions can vary by location. Below are the complete details of the data collection process.

We obtained flight data for JFK airport from the BTS website (Us bureau of transportation statistics, 2024). Indeed, the US BTS is a source renowned for the reliability and accuracy of its statistics on commercial flights. This data was extracted and stored in a Pandas data frame for further analysis. Table 5 lists all the features extracted.

We collected weather data from the Weather Underground website (Us weather underground, 2024). Indeed, Weather Underground is an online service that provides historical weather conditions. To obtain this data, we employed a web scraping technique to automate the retrieval process. We noted that the weather data is displayed on this website in a table generated dynamically by JavaScript. As a result, we used the Selenium (Pfalzgraf, 2020) package in Python to automate website navigation and data extraction. The Python script using Selenium was programmed to open the Weather Underground webpage for JFK Airport. Once the page loaded, the script extracted the most relevant weather information. We also converted the data during extraction into the appropriate units. For example, temperatures in Fahrenheit were converted to Celsius, wind speeds in miles per hour to kilometres per hour, and so on. Moreover, we stored the weather data in a Pandas Dataframe. This data is crucial as it offers significant insights into the potential influence of weather on flight delays. Note that we collected weather data at different time intervals before flight departure, namely 0, 2, 4, 8, 16, 24, and 48 h, as mentioned in Section 2.1 of the literature review. Table 6 summarises all the features extracted.

With these two datasets, we obtain a solid database rich in information. Consequently, it will be helpful for us to develop reliable flight delay prediction models successfully.

3.3. Data preprocessing

Data preprocessing is a vital step because it guarantees that the data is correctly prepared for both model training and evaluation. In fact, a well-prepared dataset can greatly improve the overall performance of the models by delivering more accurate predictions. The next section explores the various stages of preprocessing, including handling missing values, encoding categorical data, normalising variables, and creating the target variable. It also covers splitting data into training, validation, and test datasets, using a sampling technique, and selecting the most important features.

3.3.1. Handling missing values

Managing missing values in datasets is a crucial step in data preprocessing. Indeed, missing values can introduce bias and errors into prediction models if not handled correctly. From the literature review, we see that authors have either replaced or removed missing values to keep data integrity. Below is the method adopted to deal with missing values in this paper:

1. We determined the columns and rows containing missing values. In this way, we found that the flight and weather data contained missing values, with the majority of these being present in the flight data.
2. We removed some columns responsible for missing values in the flight data. Indeed, the decision to delete these columns was justified because they were unnecessary for predicting flight delays. As a result, the columns 'DIVERTED', 'CANCELLED' and 'CANCELLATION CODE', all taken from the flight data, have been removed.
3. We also replaced specific missing values with values that we believed to be consistent. For example, we observed that when the flight was on-time, the columns associated with delays, such as weather or

Table 5
List of flight data features.

Features	Descriptions	Attribute Type	Examples
FL DATE	The flight date	DateTime	1/1/2010 12:00:00 AM
OP UNIQUE_CARRIER	The unique code of the carrier	Categorical (nominal)	AA
TAIL_NUM	The tail number	Categorical (nominal)	N319AA
OP_CARRIER_FL_NUM	The flight number	Numerical (discrete)	185
ORIGIN	The airport of origin	Categorical (nominal)	JFK
DEST	The airport of destination	Categorical (nominal)	LAX
CRS_DEP_TIME	The scheduled departure time	Numerical (continuous)	2000
DEP_TIME	The actual departure time	Numerical (continuous)	1958
DEP_DELAY	The difference in minutes between the scheduled departure time and the actual departure time	Numerical (nominal)	−2.0
DEP_DEL15	An indicator of whether the flight is delayed on departure	Binary (categorical)	0.0
TAXI_OUT	The time elapsed between the departure from the boarding gate at the airport of origin and the take-off	Numerical (continuous)	5.0
WHEELS_OFF	The time at which the aircraft's wheels lift off the ground	Numerical (continuous)	2003
WHEELS_ON	The time at which the aircraft wheels touch down on the ground	Numerical (continuous)	2252
TAXI_IN	The time elapsed between the wheels landing and arrival at the boarding gate at the airport of destination	Numerical (continuous)	8.0
CRS_ARR_TIME	The scheduled arrival time	Numerical (continuous)	2240
ARR_TIME	The actual arrival time	Numerical (continuous)	2300
ARR_DELAY	The difference in minutes between the scheduled arrival time and the actual arrival time	Numerical (continuous)	20.0
ARR_DEL15	An indicator of whether the flight is delayed on arrival	Binary (categorical)	1.0
CANCELLED	An indicator of whether the flight has been cancelled	Binary (categorical)	0.0
CANCELLATION_CODE	An indicator to indicate the reason for cancellation	Categorical (nominal)	B
DIVERTED	An indicator of whether the flight has been diverted	Binary (categorical)	0.0
CRS_ELAPSED_TIME	The expected duration of the flight in minutes	Numerical (continuous)	160.0
ACTUAL_ELAPSED_TIME	The actual duration of the flight in minutes	Numerical (continuous)	182.0
AIR_TIME	The duration of the actual flight, from take-off (WHEELS	Numerical (continuous)	169.0

(continued on next page)

Table 5 (*continued*)

Features	Descriptions	Attribute Type	Examples
	OFF) to landing (WHEELS ON)		
DISTANCE	The distance between the two airports in miles	Numerical (continuous)	2475.0
CARRIER_DELAY	The carrier delay in minutes	Binary (categorical)	5.0
WEATHER_DELAY	The weather delay in minutes	Binary (categorical)	3.0
NAS DELAY	The National Air System (NAS) delay in minutes	Binary (categorical)	6.0
SECURITY_DELAY	The security delay in minutes	Binary (categorical)	2.0
LATE_ARRIVAL_DELAY	The Late Aircraft delay in minutes	Binary (categorical)	4.0

Table 6
List of weather data features.

Features	Descriptions	Attribute Type	Examples
DateTime	The flight date	DateTime	2010–01–01 00:51:00
Temperature (°C)	The temperature in Celsius	Numerical (continuous)	1
Dew Point (°C)	Dew point temperature in Celsius	Numerical (continuous)	−1
Humidity (%)	The humidity	Numerical (continuous)	92
Wind	The wind direction	Categorical (nominal)	SW
Wind Speed (km/h)	The wind speed in km/h	Numerical (continuous)	14
Wind Gust (km/h)	The wind gust in km/h	Numerical (continuous)	6
Pressure (hPa)	The precipitation in mm	Numerical (continuous)	1015.58
Precip. (mm)	The precipitation in mm	Numerical (continuous)	0
Condition	The weather conditions	Numerical (continuous)	Cloudy

security delays, included missing values. As a result, we replaced these missing values with zeros, purely as a matter of logic. Similarly, in other situations where coherent values could replace missing values, we applied this approach.

4. Finally, we also deleted some rows containing missing values, as they represented only a small proportion of the total dataset.

This proposed method for handling missing values enabled us to retain as much data as possible without introducing bias.

3.3.2. Data encoding

An essential step in data preprocessing is the encoding of categorical variables. This step is necessary for the correct use of DL models. Indeed, these models require numerical inputs during the training and prediction phases. In this way, the categorical variables must be transformed into a format that the models can interpret (Chaudhary, 2023). From the literature review, we found that authors often utilised encoding techniques such as one-hot encoding. Below is the method adopted to encode categorical variables:

1. We identified all the categorical variables contained in the dataset.
2. We adopted the following technique for encoding each categorical variable: the label encoding method (Kumar 2024). This approach involves attributing a unique numerical value to each categorical value. This technique is illustrated in Fig. 6.

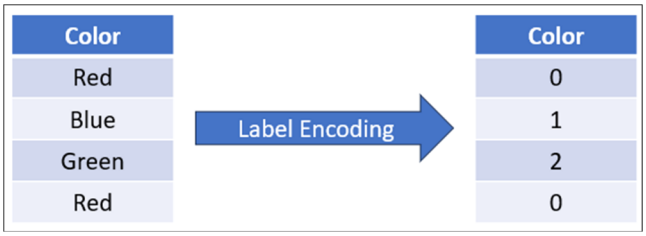


Fig. 6. Label encoding method used on a categorical variable.

3. We checked that each categorical variable had been correctly encoded.

It should be noted that we prefer label encoding over one-hot encoding. Indeed, label encoding does not require creating new columns. As a result, it helps to maintain the data's dimension and significantly reduces the complexity of calculations (Kumar 2024). However, one of the main limitations of label encoding is the arbitrary order it imposes on the data (Kumar 2024). Therefore, it can cause the model to assume false relationships.

3.3.3. Data normalisation

A crucial step in data preprocessing is normalising numerical variables. This step involves scaling the numerical variables to ensure they all have the same range, 0 to 1 (What is normalization in machine learning? a comprehensive guide to data rescaling, 2024). Normalisation is essential because it enables DL models to perform optimally (What is normalization in machine learning? a comprehensive guide to data rescaling, 2024). Indeed, these models perform best when the input data are on comparable scales. Consequently, given the recurrent presence of data with very different value ranges, we adopted a normalisation technique (What is normalization in machine learning? a comprehensive guide to data rescaling, 2024). From the literature review, we found that authors often use normalisation techniques such as Min-Max Scaling. Below is the method used to normalise the numerical variables:

1. We identified all the numerical variables contained in the dataset.
2. We applied the Min-Max (Gupta, 2022) Scaling technique for each numerical variable. This method uses Eq. (1) (39):

$$X_{scaled} = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{1}$$

where x is the value of the variable, X_{min} the minimum value of the variable, X_{max} the maximum value of the variable and X_{scaled} the scaled value of the variable.

In this way, each variable value is normalised using Eq. (1) to the range [0, 1].

3. We checked that each numerical variable was correctly normalised.

This proposed data normalisation method ensured that each variable contributed equally to model training. As a result, it prevented certain variables from dominating the learning process of DL models. Additionally, this method helped to maintain stability in the optimisation process, enabling faster convergence during gradient-based learning. Consequently, the risks linked to vanishing or exploding gradients were minimised.

3.3.4. Target variable creation

Another crucial step in data preprocessing is creating the target

variable. Indeed, making the target variable is vital for clearly defining the objective of the model prediction (Sharma, 2023). We identified that it is more pertinent to focus on arrival delays, given the aim of predicting flight delays due to weather conditions. In this way, we can directly capture the possible influence of weather conditions on the ability of aircraft to land on time. We created a target variable that categorises flights into three classes:

On-time flights (0): Flights that arrive on time or with a slight delay of <15 min, as established by BTS.
Flights delayed due to non-weather conditions (1): Delays caused by factors other than extreme weather conditions, such as technical or operational problems.
Flights delayed due to weather conditions (2): Delays uniquely caused by extreme weather conditions.

This classification guarantees the models are trained consistently, leading to accurate predictions aligned with the expected objectives.

3.3.5. Data split

Another key step in data preprocessing is data splitting. This involves dividing the data into training, validation, and test sets for model training and evaluation. Furthermore, this approach ensures that prediction models are optimised and perform accurately. It also helps us develop models that can generalise unseen data efficiently. From the literature review, we found that authors systematically use this process of dividing data into several sets. The data-splitting method is described below.

- We divided the data according to the following proportions: 56 % for training, 24 % for validation, and 20 % for testing.

This proposed method of splitting the data helps considerably reduce the risk of overfitting. It also allows the model's hyperparameters to be fine-tuned on the validation dataset.

3.3.6. Data resampling technique – smote

In this paper, sampling techniques are essential for addressing class imbalance in flight data. As discussed in Section 3.2 during data collection, we observed that the flight datasets are highly imbalanced, with most flights arriving on time rather than being delayed. From the literature review, we noted that this imbalance often leads to bias in prediction models, reducing their accuracy in forecasting delayed flights. To address this issue, the authors considered resampling techniques such as SMOTE and RUS. SMOTE generates synthetic samples for the minority class (delayed flights) to balance the dataset. Conversely, RUS reduces the number of samples in the majority class, which includes on-time flights. Our primary focus will be on SMOTE, as it generates synthetic data that preserves valuable information from the original dataset. This information could be lost with subsampling techniques such as RUS. Therefore, by creating additional data points for the minority class, SMOTE can enhance the model's ability to predict delayed flights without discarding potentially useful data from the majority class. The SMOTE pseudocode is shown in Algorithm 1 (Smote implementation in pyspark 2020; Smote 2002).

To implement the SMOTE algorithm, first, we create an instance of SMOTE and set the oversampling ratio. As we already know, the default value for this ratio is 100 %. This means that by the end of the process, the number of samples from the minority classes will match that of the majority class. Furthermore, after creating the SMOTE object, we can apply it directly by providing the features and their corresponding targets. In return, we obtain the resampled features and targets. Additionally, it is essential to note that SMOTE was applied only to the training data, not to the validation or test data, for several reasons. Firstly, most DL models require a large amount of balanced data to prevent bias towards any class during training. Secondly, for validation and testing, it is necessary to work exclusively with real data. This ensures accurate and reliable evaluations, free from the influence of generated data.

Algorithm 1
Synthetic minority oversampling technique (SMOTE).

Inputs: Number of minority instances (T), SMOTE percentage (N), number of nearest neighbours (k)
Output: Synthetic data S

if $N < 100$ **then**
  $T = \left\lfloor \frac{N}{100} T \right\rfloor$
  $N = 100$
end if
for $i = 1, 2, \ldots, T$ **do**
  Find the k nearest (minority class) neighbours of x_i
  $\widehat{N} = \left\lfloor \frac{N}{100} \right\rfloor$
  while $\widehat{N} \neq 0$ **do**
    Select one of the k nearest neighbours, call this $\overline{x}$
    Select a random number $\alpha \in [0, 1]$
    $\widehat{x} = x_i + \alpha(\overline{x} - x_i)$
    Append $\widehat{x}$ to S
  $\widehat{N} = \widehat{N}$ - 1
  end while
end for
return S

3.3.7. Feature selection

An essential step in data preprocessing is selecting relevant features from the dataset. This step is vital for pinpointing the variables that most significantly influence flight delays. Moreover, with the merging of the flight and weather datasets, we have 37 features available, which can demand substantial computing resources. Therefore, it is highly necessary to utilise feature selection methods to reduce the dataset's dimensionality. From the literature review, we found that authors often use Pearson's correlation coefficients to identify the most pertinent features. The method we adopted for selecting essential features is described below:

1. We removed the columns relating to flight arrivals. This decision focused exclusively on the factors available at the time of flight departure.
2. We applied the mutual information score (Chang, 2024; Mutual information score 2024) approach, which determines the most informative features in relation to the target variable. At the end of the algorithm, we obtained the information gain for each variable. We decided to keep only those variables with a score greater than 0.15.
3. We calculated the Pearson coefficients using a heatmap of the variables selected in the previous step. This method allows us to identify correlations between different variables. We used it to find pairs of variables with high correlation (coefficient> 0.8). After identifying these pairs, we kept only those with the highest mutual information score. This helps prevent information redundancy during model training.

The proposed method preserves only the most informative and non-redundant features. In addition, it not only optimises model performance but also considerably minimises computational costs.

3.4. Deep learning models

In this paper, we aim to explore the application of various deep learning (DL) approaches to predict flight delays caused by weather conditions. Specifically, we focus on developing and evaluating three models to achieve the objectives of this study: the LNN, LSTM, and MLP. Using these three models for prediction offers a valuable approach, enabling us to conduct a thorough and careful analysis of the many

factors that can influence these types of delays. Finally, we will compare the performance of these DL models to identify the most accurate method for predicting flight delays.

3.4.1. Liquid neural networks

In this section, we will describe the development process of the LNN model. This model was created to predict whether a flight is on time or delayed due to weather or non-weather factors.

3.4.1.1. Model architecture. Compared with classical NNs, the LNN model's architecture is slightly more complex and elaborate due to its dynamic nature. The LNN model comprises three principal components: the input layer, the liquid layer and the output layer, as illustrated in Fig. 7.

- **Input layer:** This layer is required to deal with the reception of the input data (Basic understanding of neural network structure 2023). These data are represented as vectors containing features associated with flights and weather conditions. These features have been carefully and correctly identified during the feature selection process. Moreover, the size of this layer equals the number of features in the input data. This initial layer is also utilised to prepare the data sent to the liquid layer.
- **Liquid layer:** This layer is the principal component of the LNN model. It is intended to capture any complex temporal relationships within the data (Basic understanding of neural network structure 2023). We defined an ODE function to model such dynamics over time (Prakash, Aug 2023; Neural ode 2023). It consists of a network with linear layers and a Tanh activation function to introduce nonlinearity. Moreover, a solver, namely 'dopri5', is used to integrate this ODE over a specified time interval (scipy.integrer.ode 2023). Such integration is vital for modelling the continuous-time dynamics of the data (Prakash, Aug 2023; Neural ode 2023). As most LNNs have a single liquid layer, we have followed this trend and used only one layer (G. Boesch, 2024). The size of this layer will be a hyperparameter to be adjusted during model optimisation.
- **Output layer:** This layer receives the final hidden states of the liquid layer as input (Basic understanding of neural network structure 2023). Using the received data, this linear-type layer produces a distribution of probabilities over the three target classes (Basic understanding of neural network structure 2023): on-time flight, non-weather delay, and weather delay. The class with the highest probability is identified as the predicted class.

3.4.2. Long short-term memory

In this section, we explain the LSTM model developed. This model has been built to predict whether a flight is on time, delayed by weather conditions, or delayed by non-weather conditions.

3.4.2.1. Model architecture. There are three main components in the architecture of the LSTM model: the input layer, the LSTM layers and the output layer, as shown in Fig. 8.

- **Input layer:** The input layer oversees receiving the input data (Basic understanding of neural network structure 2023). These data inputs consist of vectors representing the flight and weather features, previously selected during the preprocessing and feature selection phases. The dimension of these feature vectors equals the number of input features. Besides, this layer does not involve any complex transformation. Indeed, it only serves as a starting point for the data sequences introduced into the model.
- **LSTM layers:** These layers can be considered the principal component of the model. We utilised several stacked LSTM layers during the development of this DL model. As a result, this allows the capture of any complex temporal relationships present in the input data (Basic understanding of neural network structure 2023). In addition, the LSTM layers receive the outputs of the previous layer (Basic understanding of neural network structure 2023) and generate hidden states initialised to zero for each data sequence. Moreover, the output of the last LSTM layer at the earlier time step is passed as the input to the output layer. Both the number and the size of these layers are hyperparameters that will be adjusted when optimising this model's performance.
- **Output layer:** The output layer is a linear layer which generates a probability distribution over the three classes of the target variable, namely on-time flight, non-weather delay and weather delay (Basic understanding of neural network structure 2023). To create these class probabilities, this layer uses the hidden states of the last LSTM layer at the previous time step and predicts the most likely class based on these probabilities.

3.4.3. MultiLayer perceptron

In this section, we will detail the MLP model we developed by utilising the PyTorch library. This model is intended to predict whether a flight is on time or delayed due to weather or non-weather conditions.

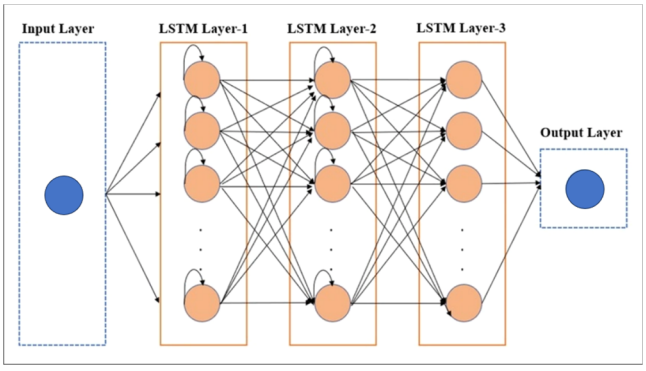


Fig. 8. Diagram of an example LSTM model.

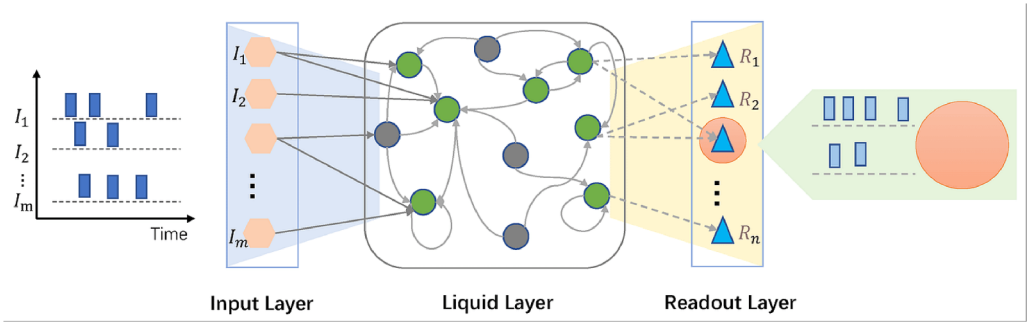


Fig. 7. Diagram of an example LNN model.

3.4.3.1. Model architecture. As we know, the MLP architecture is quite conventional compared with the two previous models. Indeed, the MLP model has three main components in its architecture, found in most NN models: the input layer, the hidden layers, and the output layer, as shown in Fig. 9.

- **Input layer:** This layer accepts the input data (Basic understanding of neural network structure 2023). These data are in vector form, representing the flight and weather-related features selected during feature selection. Moreover, the size of this layer is fixed by the total number of features included in these vectors. This layer does not modify the data; it passes it to the hidden layers for further processing.
- **Hidden layers:** These hidden layers are a series of fully connected layers. These hidden layers follow a particular structure: they consist of a linear layer, a Leaky ReLU activation function and a dropout layer. The Leaky ReLU function is utilised to introduce non-linearity into the model (Basic understanding of neural network structure 2023). In this way, the model can capture complex relationships in the data (Basic understanding of neural network structure 2023). Moreover, the dropout layer helps reduce the risk of overfitting. Besides, each hidden layer takes as input the outputs of the previous layer (Basic understanding of neural network structure 2023). The number of hidden layers and their sizes are some hyperparameters that will be adjusted when optimising the model's performance.
- **Output layer:** This layer is a classical output layer very similar to those of the other NNs. It is linear and is used to produce a set of probabilities for the three target classes defined when creating the target variable (Basic understanding of neural network structure 2023), as explained in section 3.3.3: on-time flight, delays due to non-weather conditions, and delays due to weather conditions.

3.4.4. Loss function

For the LNN, LSTM and MLP models, we decided to employ the CrossEntropyLoss function. Indeed, this function is well-suited for multiclass classification tasks (What is cross-entropy loss function? 2023). The following Eq. (2) sets this function (Koech, 2020):

$$CrossEntropyLoss(y,p) = \sum_{c} y_c \log(p_c) \tag{2}$$

where y_c is the actual label and p_c is the class model with a probability c. This loss function combines the softmax function with the loss computation. In other words, this function first converts the logits produced by the output layer into class probabilities (What is cross-entropy loss function? 2023). Subsequently, it calculates the loss between these probabilities obtained and the actual class labels by using Eq. (2).

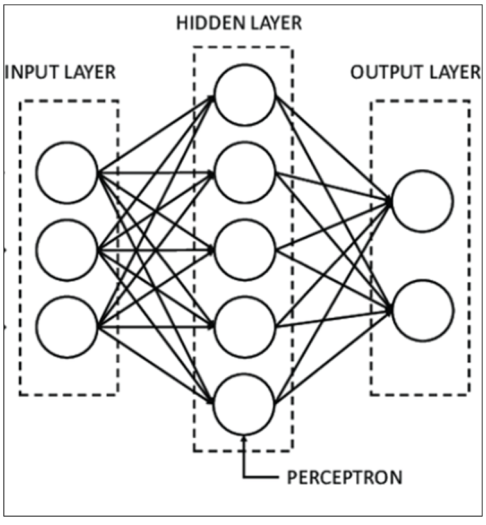


Fig. 9. Diagram of an example MLP model.

3.4.5. Optimiser

We decided to use the Adaptive Moment Estimation (Adam) optimiser for training the LNN, LSTM, and MLP models. We chose this optimiser because it is essential for tuning the weights of DL models (Demystifying the adam optimizer in machine learning 2024). In fact, Adam is well-suited for NNs like LNNs, LSTMs, and MLPs because it combines the advantages of gradient descent methods with momentum and Adaptive learning rate techniques (What is adam optimizer? 2024). One of the main benefits of this optimiser is its ability to deliver very fast convergence towards the optimal solution (Demystifying the adam optimizer in machine learning 2024). This is crucial for optimising model performance within a reasonable timeframe (Demystifying the adam optimizer in machine learning 2024). Furthermore, Adam is recognised for its robustness, which is vital when addressing complex problems such as flight delay prediction. The Adam optimiser primarily relies on the learning rate as a key parameter. This parameter is adjusted during the hyperparameter optimisation process of the DL models.

3.4.6. Activation function

During DL model training, activation functions transmit gradients, enabling effective learning. Additionally, they affect how information is processed and spread throughout the network (Activation functions: Relu vs. leaky relu 2023).

We chose to use the Leaky ReLU activation function in the MLP model. This activation function allows the neural networks to capture all nonlinear relationships in the input data and process complex phenomena (Activation functions: Relu vs. leaky relu 2023). This function is defined by Eq. (3) (Pytorch - leakyrelu 2024):

$$f(x) = \begin{cases} \alpha x, & si\ x < 0 \\ x, & si\ x \geq 0 \end{cases} \tag{3}$$

where α is a positive coefficient, set to around 0.01 by default.

As Eq. (3) demonstrates, this function introduces a slope for negative input values (What is leaky relu? 2024). This activation function offers many significant advantages. It effectively resolves the problem of neurons stopping learning when input values are negative or gradients are zeroed out by including the slope in the equation (What is leaky relu? 2024). It also improves overall network learning. Lastly, it is easy to implement and significantly enhances the model's robustness.

Moreover, we chose to use the tanh activation function in the LNN model. As with the Leaky ReLU function, the tanh function introduces non-linearity into the NN. This allows the model to learn more complex non-linear relationships between the input and the output data (Chidalu, 2023). This function is defined by Eq. (4) (Tanh activation 2024; Chidalu, 2023):

$$f(x)\frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{4}$$

where x represents the input value and e represents the exponential, a mathematical term.

As shown in Eq. (4), it converts the input values to produce output values within the range of −1 to 1 (Tanh activation 2024; Chidalu, 2023). Furthermore, the function is symmetrical around the origin, meaning it produces negative outputs for negative input values and positive outputs for positive input values (Chidalu, 2023). This function also allows models to achieve good performance. The main limitation of this tanh function is that it can encounter the vanishing gradient problem (Tanh activation 2024; Chidalu, 2023).

3.5. Hyperparameter optimisation

We optimised the hyperparameters, a crucial step in developing and

training DL models. In fact, selecting the correct hyperparameter values is a complex task that can significantly affect the performance of the LNN, MLP, and LSTM models. This optimisation aims to identify the best combination of hyperparameters from those tested to produce the most accurate models. Table 7 displays the hyperparameters tested for each model.

Below is a detailed description of each hyperparameter utilised in this paper:

- **Learning rate:** The learning rate is a crucial hyperparameter when training DL models. It controls the size of the network weights updates during backpropagation (Understanding learning rate 2019). Therefore, selecting the appropriate learning rate is challenging. A learning rate that is too high can cause the model to diverge (Brownlee, 2020), while a rate that is too low can significantly slow convergence. Consequently, the model may become trapped in local minima (Brownlee, 2020).
- **Number of epochs:** The number of epochs indicates how many complete passes the training data makes through the algorithm. (Epoch: An essential notion in real-time programming 2023; Epoch in machine learning 2024) One epoch means each sample in the training dataset gets the opportunity to update the model weights (Epoch: An essential notion in real-time programming 2023; Epoch in machine learning 2024). Therefore, the number of epochs is a crucial hyperparameter. It directly affects the speed and quality of training. Hence, it is necessary to optimise it while avoiding excessive adjustments. Indeed, too few epochs may lead to undertraining (Epoch in machine learning 2024). This often results in underfitting and poor performance. Conversely, too many epochs can cause the model to overtrain Epoch in machine learning (2024), leading to overfitting. Additionally, an excessive number of epochs can prevent the model from achieving overall good performance during predictions.
- **Batch size:** The batch size refers to the number of samples used before the model's weights are updated during training (Epoch: An essential notion in real-time programming 2023; Epoch in machine learning 2024). Like the number of epochs, it is a key hyperparameter. It affects the speed and stability of the learning process. A smaller batch size allows for more frequent updates to the model weights, which can lead to more precise convergence (Epochs, batch size, iterations - how are they important to training ai & deep learning models 2023). However, it may also introduce noise and instability during updates. Conversely, a larger batch size provides a more stable gradient estimate but requires more memory resources and can significantly accelerate training (Epochs, batch size, iterations - how are they important to training ai & deep learning models 2023).
- **Number of hidden nodes:** The number of hidden nodes relates to the size of the hidden layers in an NN. This hyperparameter is sensitive because it can influence training speed and complexity. A smaller size may significantly limit the model's learning capacity, while a larger size may enable the model to capture more complex relationships in the data (Sivakumar 2023). However, it can also increase the risk of overfitting. Additionally, it may slow down training and demand more computational power and memory (Sivakumar 2023).
- **Number of layers:** The number of layers indicates the layer count between the input and output of the neural network. This hyperparameter directly affects the network's depth and complexity. Therefore, choosing the correct number of layers is essential for optimising the model. As a neural network has more layers, it can learn more complex relationships (Setup neural networks hyperparameters for best results 2022). It can also enhance performance on intricate tasks. However, it may sometimes increase the risk of overfitting (Sivakumar 2023) and lead to gradient issues such as vanishing or exploding gradients (Jain, 2024).

Table 7
List of Models and Their Hyperparameters.

S/No	Models	Hyperparameters
1	LNN	Learning rate, Number of Epochs, Batch size, Number of hidden nodes
2	LSTM	Learning rate, Number of Epochs, Number of layers, Number of hidden nodes
3	MLP	Learning rate, Number of Epochs, Number of layers, Number of hidden nodes

For the LSTM and MLP models, we used a grid search to optimise hyperparameters. Indeed, this method has been frequently used in previous research on flight delay prediction. It enables the exploration of various hyperparameter combinations to identify the one that delivers the best performance. This approach was essential because it allowed us to determine the optimal configurations for these two models.

Conversely, we had to manually optimise the hyperparameters for the LNN model for several reasons. In fact, applying grid search to the LNN was unfeasible due to the extremely long execution time per combination. Additionally, significant computing resources were required to perform this optimisation method. Ultimately, we tested each hyperparameter combination individually to identify the best one for the LNN model.

3.6. Evaluation metrics

Evaluating different models is crucial for predicting weather-related flight delays. In a multiclass classification problem, it is essential to select suitable performance metrics to ensure that the models accurately meet the problem's requirements. For this research, the metrics chosen are accuracy, precision, recall, and F1-score. These metrics are selected for their relevance in assessing classification problems where class imbalances can influence overall performance. Such performance metrics are vital for a comprehensive evaluation of a DL model in a 3-class classification task, especially when performance varies across classes. Using these metrics, we can identify which classes the model manages well and pinpoint problematic areas. They also aid in understanding the nature of errors encountered for each class.

The accuracy (Klu 2024) indicates the proportion of correct predictions out of the total number of forecasts made. Furthermore, the precision (Klu 2024) is measured as the ratio of positive predictions that are correct out of all positive predictions. Additionally, the recall (Klu 2024) value reflects the ratio of truly correct positive predictions out of the total number of actual positive cases. Lastly, the F1-score (Klu 2024) is the harmonic mean of precision and recall, offering a balanced measure between the two.

These performance metrics can be computed by using the following Eqs. (5)–8 (Harikrishnan, 2019):

$$Accuracy = \frac{TN + TP}{TN + FP + TP + FN} \tag{5}$$

$$Precision = \frac{TP}{TP + FP} \tag{6}$$

$$Recall = \frac{TP}{TP + FN} \tag{7}$$

$$F1 - Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{8}$$

Where TP is the number of true positives, TN the number of true negatives, FP the number of false positives and FN the number of false negatives.

We also utilised the confusion matrix to evaluate our DL models. Indeed, this matrix is a crucial tool, especially for multiclass classification problems. Moreover, it is designed to compare the actual and predicted values for each class (How to interpret a confusion matrix for a machine learning model 2024). Through this comparison, the confusion matrix helps us visualise and better understand the models' predictions (Understanding the confusion matrix in machine learning 2024).

As discussed in Section 3.1, this problem involves a three-class classification. Consequently, the confusion matrix used to evaluate our models is a 3×3 matrix. Each element of the matrix indicates the number of predictions made for each possible combination of actual and predicted classes. An example of the confusion matrix for three classes is shown in Fig. 10.

Using this matrix, we could more easily visualise the distribution of correct and incorrect predictions for each class. As a result, we obtained a more overall view of the model's performance (Understanding the confusion matrix in machine learning 2024). In fact, we directly identified the classes where the model faced difficulties by analysing the matrix (How to interpret a confusion matrix for a machine learning model 2024). Once identified, we attempted to adjust the optimisation methods to enhance accuracy for those classes. Additionally, the confusion matrix provides a clearer understanding of class imbalances, which is crucial for correctly interpreting the performance metrics mentioned earlier (Understanding the confusion matrix in machine learning 2024; How to interpret a confusion matrix for a machine learning model 2024).

Furthermore, we tested the models for overfitting to evaluate their performance. This test compares the loss function on the training data with that on the validation data obtained during training (Muralidhar, 2021). As mentioned in Section 3.4.4, we used the CrossEntropyLoss function. This comparison is essential for accurately assessing how the models perform on both the training and validation datasets. Indeed, if the model fits the training data too closely, it may significantly reduce its ability to generalise effectively to unseen data, namely the validation data. Therefore, by monitoring the loss function values for both datasets, we can detect and attempt to mitigate overfitting issues.

4. Results and discussions

We carried out all our experiments on the Cranfield University High-Performance Computing (HPC) machine called the Crescent 2. This advanced system allowed us to utilise an Nvidia T4 16GB GPU with 8 CPUs. The selection of this GPU was motivated by its performance and energy efficiency, which made model training quicker and more efficient. These two advantages were crucial for conducting the experiments and obtaining results as swiftly as possible.

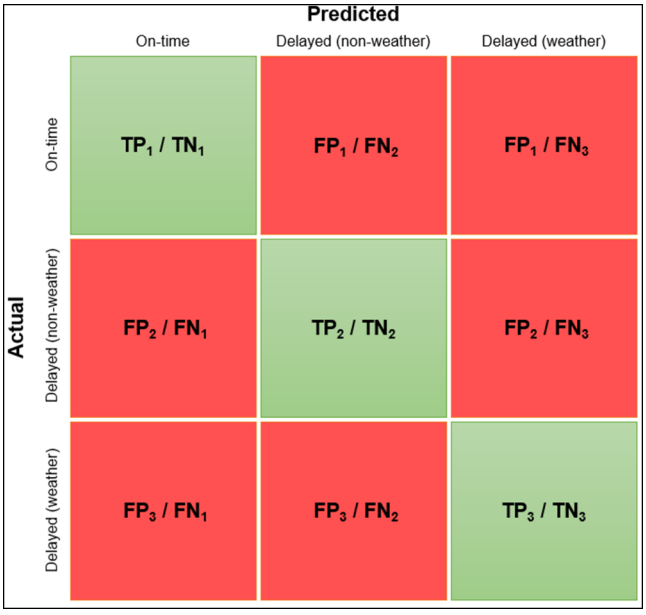


Fig. 10. Structure of a confusion matrix for a three-class classification task.

4.1. Exploratory data analysis

In this section, we will examine in depth the data used for predicting flight delays caused by weather conditions. For this purpose, we conducted an EDA, which is an essential phase in this study. Note that the EDA was performed after addressing missing values and creating the target variable to ensure high-quality and reliable data. This approach allows us to understand the dataset's key features and identify relevant patterns (What is exploratory data analysis (eda)? 2023). It also helps us to detect anomalies in the data (What is exploratory data analysis (eda)? 2023). Additionally, this analysis provides valuable insights into the distribution of variables and the relationships between features (What is exploratory data analysis (eda)? 2023). It also highlights factors that influence delays. This EDA will serve as a solid foundation for implementing DL models.

Firstly, as part of the EDA, we plotted several graphs based on the flight data to help us better understand the dataset and identify any potential trends.

Fig. 11 illustrates the distribution of flights. Most flights are on time, suggesting that the majority arrive without significant delay. This category is by far the most common among the three. We observe that a notable number of flights are delayed due to reasons unrelated to weather. Although this category represents a considerable proportion of flights, it is much less frequent than on-time flights. Lastly, the least common category involves flights delayed because of weather-related reasons. Therefore, there is a significant imbalance in the flight data across these three categories. This finding aligns with other research studies indicating the imbalanced data used to predict flight delays.

Fig. 12 shows the distribution of arrival delays. As expected, most flights experience little or no delay. This is reflected by the high peak on the left of the graph, near zero minutes of delay. We also observe that the frequency of flights drops sharply as delays increase. However, there remains a significant proportion of flights with serious delays, some of which are caused by weather conditions.

Fig. 13 illustrates the distribution of flight delay causes. We observe that weather delays are less common than other causes, such as air carrier and NAS delays. Although they occur less frequently, weather delays can be particularly unpredictable. This small proportion highlights the need for more precise weather data to enhance the prediction of delays caused by weather. For this purpose, we utilised weather data obtained from the Weather Underground website to identify weather delay patterns.

Secondly, during this EDA, we created numerous graphs using weather data. These visualisations offered valuable insights into weather patterns and their potential influence on flight delays.

Fig. 14 depicts the dispersion of weather delays relative to temperature. There is no clear linear trend. This suggests that temperature is not strongly linked to weather delays. Additionally, at very low or very high temperatures, the density of points diminishes, indicating fewer flights in such conditions. However, weather delays in these extreme temperatures do not show a significant rise. All these observations lead us to conclude that weather delays are not solely dependent on temperature.

Fig. 15 displays the dispersion of weather delays relative to wind speed. We notice that most weather delays are primarily concentrated between 0 and 40 km/h wind speed. Additionally, the points representing major weather delays are spread across various wind speeds. This suggests that extreme weather delays can occur regardless of wind speed. While there is no clear linear trend, there is a slight decline in the occurrence of long weather delays at very high wind speeds. From these

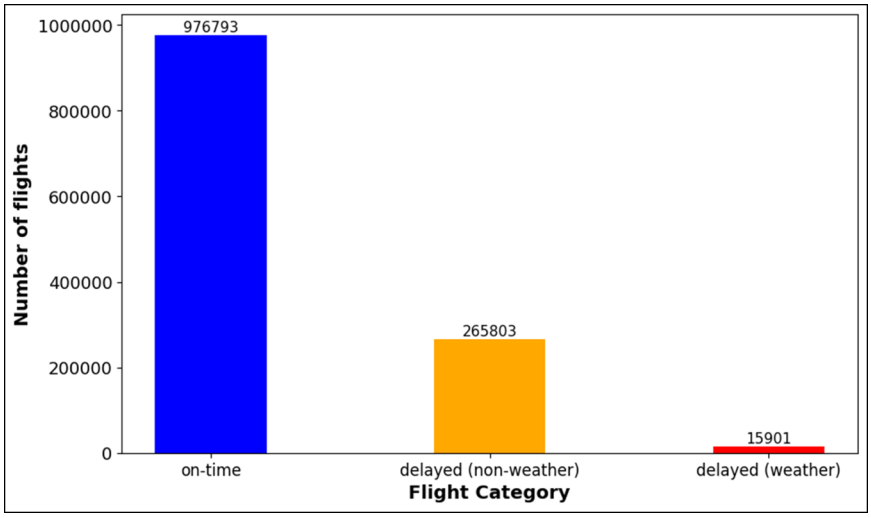


Fig. 11. Distribution of flights.

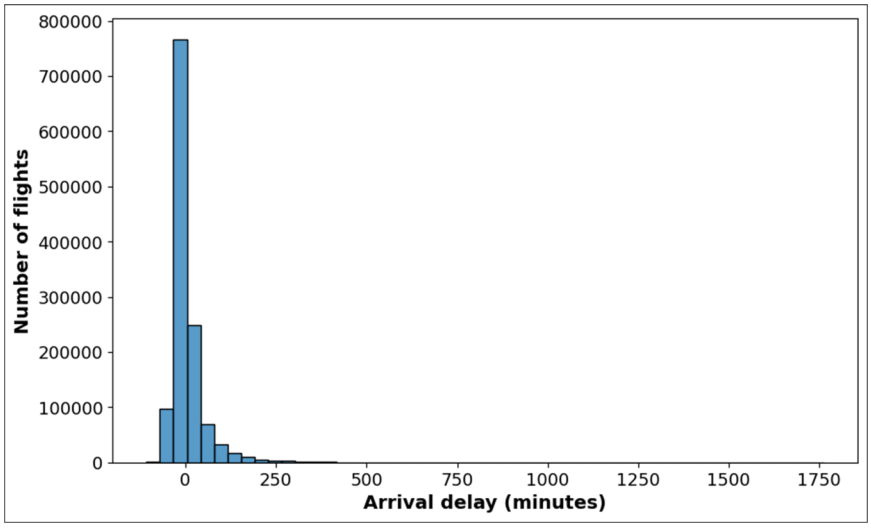


Fig. 12. Distribution of arrival flight delays.

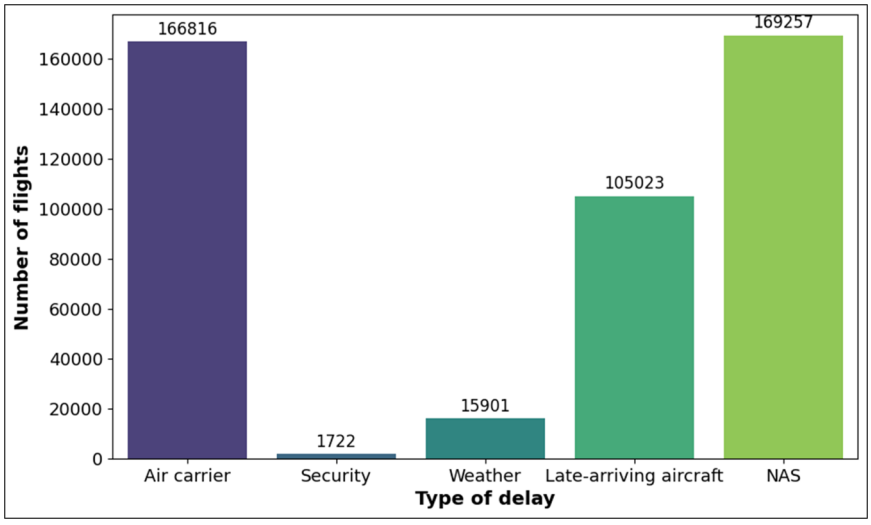


Fig. 13. Distribution of causes of delay.

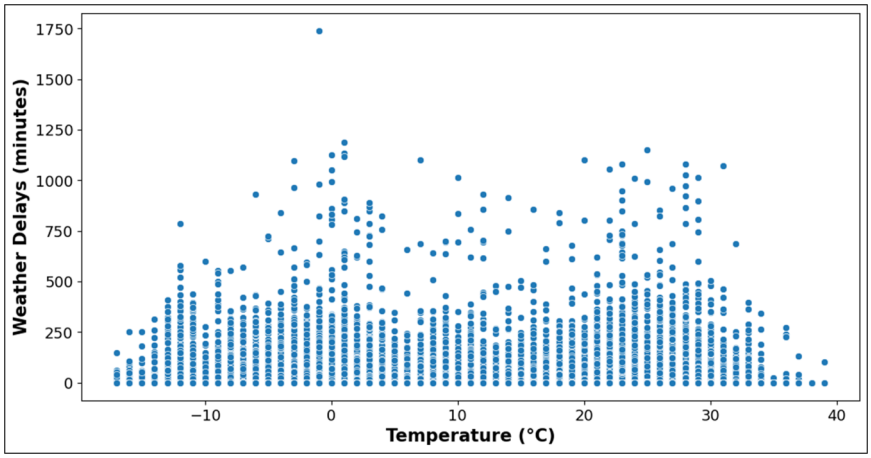


Fig. 14. Distribution of weather delays as a function of temperature.

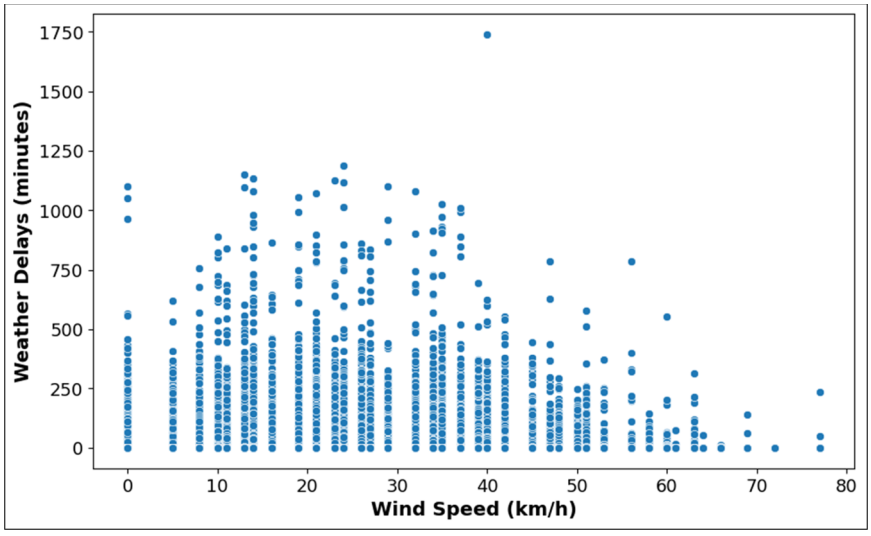


Fig. 15. Distribution of weather delays as a function of wind speed.

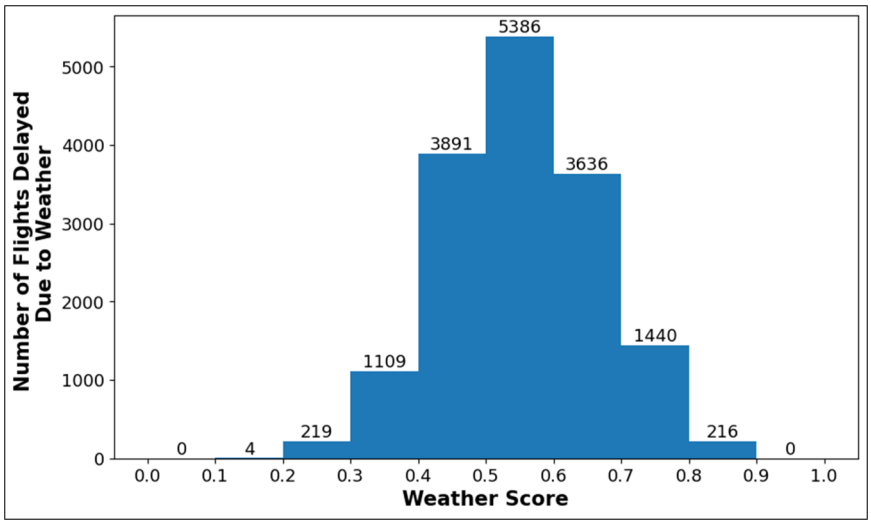


Fig. 16. Distribution of flights delayed due to weather conditions by weather score.

observations, we can conclude that although wind speed influences weather delays, it is not the sole decisive factor.

Moreover, during the EDA, we conducted an extensive analysis of the weather data to examine its influence on flight delays in detail. In this way, we calculated a weather score for each flight. This score is based on several weather factors, namely temperature, dew point, humidity, wind speed, wind gusts, pressure, precipitation, and general weather conditions. This idea was inspired by previous research carried out by Schultz et al. (Schultz et al., Oct 2021). They calculated the weather score from weather data collected via METAR using the ATMAP algorithm. We adopted the concept of the ATMAP algorithm by classifying different weather features according to their severity. The classifications for the severity of weather variables were determined through research and arbitrarily. These classifications are then combined to compute an overall score, weighted by the importance of each variable. This score enables a consistent way to quantify the influence of weather conditions on flight delays. It is represented as a ratio between 0 and 1. The closer the score is to 1, the more severe the weather conditions.

Fig. 16 illustrates the distribution of flights delayed due to weather conditions by weather score. We notice that most flights delayed because of weather have a weather score between 0.4 and 0.6. This concentration suggests that moderate weather conditions significantly influence weather-related flight delays. Conversely, we see that both very low and very high weather scores account for a smaller number of delayed flights. Therefore, we conclude that extreme weather conditions are not the sole cause of flight delays. Additionally, moderate conditions can also cause severe disruption when combined. This finding underscores the importance of incorporating a comprehensive range of weather features into flight delay prediction models. From all these observations, we can state that the weather score approach is valuable and can assist in accurately forecasting weather-impacted flight delays.

4.2. Performance and model analysis

In this section, we will examine in detail the performance of three DL models, namely the LNN, the LSTM, and the MLP. The main aim of this analysis is to assess how well each model predicts weather-related flight delays. To do this, we will evaluate each model using the methods described in Section 3.6. These methods include various metrics, such as precision, recall, F1-score, overall accuracy, and the confusion matrix. We will also review the loss function curves for each model to understand their behaviour during training and validation.

The features used to train and evaluate the three models are described in Table 8. This table lists the selected features, including weather and flight information.

Table 8
List of features utilised for training and evaluation.

S/ No.	Features	Descriptions
1	FL_DATE	The flight date
2	CRS_DEP_TIME	The scheduled departure time
3	DEP_DELAY	The difference in minutes between the scheduled departure time and the actual departure time
4	DEP_DEL15	An indicator of whether the flight is delayed on departure
5	TAXI_OUT	The time elapsed between the departure from the boarding gate at the airport of origin and the take-off
6	WHEELS_OFF	The time at which the aircraft's wheels lift off the ground
7	CRS_ARR_TIME	The scheduled arrival time
8	CRS_ELAPSED_TIME	The expected duration of the flight in minutes
9	Temperature (°C)	The temperature in Celsius
10	Humidity (%)	The humidity
11	Wind	The wind direction
12	Wind Speed (km/h)	The wind speed in km/h
13	Pressure (hPa)	The pressure in hPa
14	Condition	The weather condition

The 14 features listed in Table 8 were carefully chosen for their potential impact on prediction accuracy, in accordance with the approach outlined in Section 3.3.7. Selecting these features is crucial to ensuring that the models can capture some of the fundamental dynamics of weather-related flight delays. This method was based on the analyses presented in Fig. 17 and Table 9.

As explained in Section 3.3.7, Fig. 17 allowed us to retain all features with a score above the threshold of 0.15, marked by the red line. We considered that any features with a score below this threshold had less influence on the prediction model. Subsequently, Table 9 enabled us to eliminate the strong redundancies between pairs of variables with correlation coefficients greater than 0.8. Therefore, based on Fig. 17 and Table 9, we removed the 'DISTANCE', 'Dew Point (°C)' and 'DEP TIME' columns.

4.2.1. Liquid neural network

In this section, we assess the performance of the LNN model. As outlined in Section 3.5, we manually tested numerous hyperparameter configurations due to the extensive execution time involved. These configurations varied in terms of the number of hidden nodes, batch size, learning rate, and dropout rate. For each setup, we carried out a series of thorough tests to evaluate the model's performance. The results were analysed and compared to understand the influence of each hyperparameter on prediction accuracy. Specifically, we investigated how the number of hidden nodes and batch size affect the model's performance. Furthermore, we examined the impact of using a regularisation technique such as dropout on the model. We also observed how the learning rate influences model convergence. The findings of these analyses are discussed below and offer a detailed evaluation of the performance for each tested hyperparameter configuration. This comparative analysis facilitates the identification of the most effective configurations.

Since the LNN model is relatively new, there is no published reference for its hyperparameters. Therefore, we had to make some logical and well-considered decisions when selecting these values. For the learning rate, values of 0.001 and 0.0001 were chosen because they are commonly used in ML and DL models (Bengio, 2012; Bisandu & Moulitsas, 2024; Bisandu et al., 2021; Bisandu et al., 2022; D.B. Bisandu & Moulitsas, 2023; Bisandu et al., 2024; DB Bisandu & Moulitsas, 2023). Regarding the number of epochs, we tested with 20 and 50 to evaluate the model with a small and larger number of iterations, which helps prevent overfitting. We did not exceed 50 epochs because the LNN model demands significant computational resources. Concerning the batch size, since the model's runtime is high, we opted for larger values compared to usual parameters (Bengio, 2012), testing three options: small with 512, medium with 1024, and large with 2048. For the number of hidden nodes, we followed a similar approach by testing 50, 100, and 350. This helps in finding the best balance between complexity and performance. Finally, to improve the model's robustness, we used dropout, applying the default value of 0.5 as shown in Table 10.

Configuration: Number of hidden nodes = 350, Number of epochs = 50, Batch size = 1024, Learning rate = 0.0001, Dropout rate = 0.5.

The number of hidden nodes is a crucial hyperparameter that directly influences the prediction capacity of the LNN model. We observe that in the tested configurations, values of 50, 100, and 350 hidden nodes were analysed. Configurations with 350 hidden nodes demonstrate better overall performance. Concerning class 0, all configurations show satisfactory results for this class, although those with 350 nodes perform slightly better than the others. Notably, Configuration 1 with 350 hidden nodes achieves a precision of 0.96 and a recall of 0.91 when using weather data collected 2 h before flight departure. These results surpass those of Configuration 2 with 100 hidden nodes, which has a precision of 0.96 and a recall of 0.89 for the same weather data timing. Similarly, the performance of Configuration 1 exceeds that of Configuration 7 with 50 nodes, which records a precision of 0.96 and

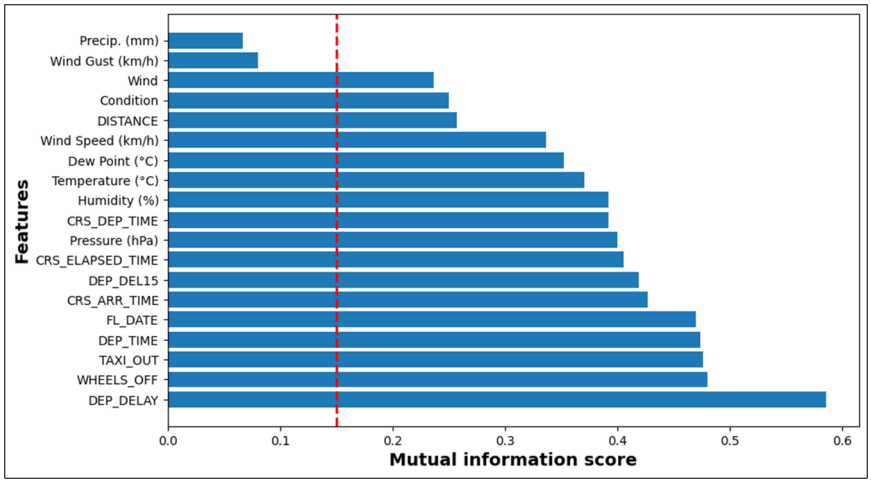


Fig. 17. Mutual information score for features.

Table 9
Pairs of features with high correlation coefficients.

S/No.	Feature 1	Feature 2	Correlation Coefficient
1	WHEELS_OFF	DEP_TIME	0.832192
2	CRS ELAPSED TIME	DISTANCE	0.993380
3	Temperature (°C)	Dew Point (°C)	0.905055

a recall of 0.88 at the same interval. For classes 1 and 2, configurations with 350 hidden nodes tend to perform marginally better overall. Given the difficulty of predicting these classes and the influence of other hyperparameters, some configurations may occasionally produce slightly superior outcomes compared to those with 350 hidden nodes. It is worth noting that Configuration 1 achieves an F1-score of 0.64 for class 1 and 0.23 for class 2, compared to an F1-score of 0.62 for class 1 and 0.22 for class 2 using weather data at a 2-hour interval before flight departure. Overall, there appears to be a trend towards better performance with 350 hidden nodes. This can be attributed to the enhanced capacity of the LNN model to capture complex features as the number of hidden nodes increases (Sivakumar 2023). This is especially important for minority classes, where data can be more diverse and less frequent. However, having too many nodes may also lead to overfitting (Sivakumar 2023).

Additionally, batch size is another key hyperparameter that influences the prediction of the LNN model. We tested configurations with batch sizes of 512, 1024, and 2048. We observe that too-large batch sizes tend to produce less efficient results compared to moderate batch sizes. While the results for class 0 remain excellent regardless of batch size, there are noticeable differences in performance for the other two classes concerning this hyperparameter. For example, Configuration 11 with a batch size of 1024 achieves a precision of 0.96 and a recall of 0.90 for class 0 using weather data collected 0 h before flight departure. In contrast, Configuration 10 with a batch size of 2048 achieves a precision of 0.96 but a recall of 0.89 for the same period. For classes 1 and 2, Configuration 11 records an F1 score of 0.68 and 0.27, respectively, compared to 0.67 for class 1 and 0.26 for class 2 for Configuration 10. Furthermore, we note that very small batch sizes yield results quite similar to those with moderate batch sizes, with the main difference being increased instability in performance related to weather data collection timing. Therefore, it appears that moderate batch sizes are the most appropriate choice for the LNN model. This likely occurs because larger batch sizes reduce the frequency of weight updates, potentially slowing the learning process for classes 1 and 2. Additionally, too-small batch sizes may introduce more variance in the updates (Epochs, batch size, iterations - how are they important to training ai & deep learning models 2023), which can negatively impact learning.

Moreover, we find that the learning rate is crucial for the stability of the LNN model's convergence. We observe that configurations with a learning rate of 0.0001 show more stable performance. For example, configuration 14 with a learning rate of 0.0001 achieves a global accuracy of 86.45 % for weather data collected 0 h before flight departure. Configuration 18, with a learning rate of 0.001, reaches a global accuracy of 85.61 % for the same weather data timing. In some cases, configurations using a learning rate of 0.001 may perform better than those with 0.0001 because of the influence of other hyperparameters. Additionally, we see that Configuration 14 with a learning rate of 0.0001 achieves a global accuracy between 85.52 % and 86.45 %. In contrast, Configuration 16 with a learning rate of 0.001 shows a global accuracy ranging from 82.91 % to 86.24 %. These results highlight the stability gained with configurations using a learning rate of 0.0001. Therefore, a very high learning rate can cause oscillations and unstable convergence. Using a learning rate of 0.0001 is the most optimal for this hyperparameter, ensuring good performance. This rate allows the LNN model to converge reliably while preventing rapid overfitting.

Furthermore, the number of epochs significantly influences the prediction performance of the LNN model. We observe that

Table 10
Performance metrics of the LNN model for configuration 20.

Time	Class 0			Class 1			Class 2			Global
	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1	Acc.
0h	0.96	0.92	0.94	0.71	0.70	0.71	0.18	0.67	0.29	86.82 %
2h	0.95	0.92	0.94	0.72	0.67	0.70	0.16	0.69	0.25	86.51 %
4h	0.96	0.91	0.94	0.70	0.70	0.70	0.17	0.65	0.27	86.57 %
8h	0.96	0.92	0.94	0.70	0.67	0.68	0.16	0.69	0.26	86.16 %
16h	0.96	0.91	0.93	0.68	0.67	0.68	0.16	0.68	0.25	85.86 %
24h	0.96	0.91	0.93	0.70	0.67	0.68	0.15	0.68	0.24	85.90 %
48h	0.96	0.92	0.94	0.71	0.67	0.69	0.14	0.65	0.23	86.08 %

configurations with 50 epochs yield better results in terms of F1-score for classes 1 and 2. For example, Configuration 11 with 50 epochs achieves an F1-score of 0.70 for class 1 and 0.27 for class 2 when using weather data collected 2 h before flight departure. In contrast, Configuration 1 with 20 epochs has an F1-score of 0.64 for class 1 and 0.23 for class 2 with the same weather data interval. Additionally, configurations with 50 epochs offer higher overall accuracy compared to those with 20 epochs. Specifically, Configuration 11 attains a maximum global accuracy of 87.84 %, while Configuration 1 reaches 85.71 %. More epochs allow the LNN model to better learn the feature patterns of classes 1 and 2. However, it is also important to be cautious of overfitting if the model becomes too complex. Therefore, regularisation techniques such as dropout are essential to maintain the model's ability to generalise model.

Furthermore, it is noted that the use of dropout techniques yields quite interesting performance. Indeed, configurations with a dropout rate of 0.5 show effective regularisation and better performance on classes 1 and 2. We observe that Configuration 20 with a dropout rate of 0.5 achieves an F1-score of 0.71 for class 1 and 0.29 for class 2 for weather data collected 0 h before flight departure. In comparison, Configuration 11, without dropout, obtains an F1-score of 0.68 for class 1 and 0.27 for class 2. Additionally, configurations with dropout result in more stable performance. All these observations indicate that dropout helps prevent overfitting and improves the model's generalisation. Applying dropout introduces a certain degree of robustness to the model. Thus, the LNN model is forced to learn more general representations that are less specific to individual samples. This is especially beneficial for classes 1 and 2, where overfitting can be a significant problem.

Finally, we note that the timing of weather data collection significantly affects the model's performance. We observe that configurations exhibit some fluctuations depending on when the weather data is collected.

- **0 h before departure:** We notice that performance is generally good with very high precision for class 0, but frequently lower F1-scores for class 2. For instance, Configuration 1 shows a precision of 0.96 and a recall of 0.91 for class 0. However, for the same configuration, we observe an F1-score of 0.68 for class 1 and 0.26 for class 2. One possible indication is that weather conditions very close to the departure time capture the most accurate data, thus reducing overall uncertainty.
- **2 to 8 h before departure:** We find that the results in terms of performance fluctuate slightly, with a small reduction for the two classes 1 and 2. For instance, we can see that configuration 1 at 2 h has a precision of 0.96 and a recall of 0.91 for class 0. Moreover, we observe a drop in the F1-score for class 1 to 0.64 and for class 2 to 0.23. We can explain this perhaps by a reduction in the accuracy of weather forecasts as the time horizon extends.
- **16 to 48 h before departure:** We can see a tendency for performance to decline. For instance, Configuration 1 with weather data of 48 h shows a precision of 0.96 and a recall of 0.90 for class 0, but lower F1-scores for classes 1 and 2 with 0.63 and 0.19, respectively. All this seems to indicate that long-range weather forecasts are less reliable, thereby introducing more noise into the model's input data.

Based on this descriptive and critical analysis of the results of the LNN model for 20 different hyperparameter configurations, we identified the best value for each hyperparameter: Number of hidden nodes = 350, Number of epochs = 50, Batch size = 1024, Learning rate = 0.0001, Dropout rate = 0.5.

These hyperparameter values correspond to Configuration 20. Furthermore, as shown in Table 10, we observe that performance remains stable for this configuration, regardless of the timing of weather data collection. Finally, we chose Configuration 20, using weather data collected 0 h before flight departure, as the final LNN model since it exhibits the best performance. Below, we will analyse the final model selected for the LNN by examining the loss function for training and validation as well as the confusion matrix.

Fig. 18 displays the confusion matrix from the optimal configuration of the LNN model. Class 0, the dominant class within the validation data, is predicted with high accuracy. The LNN model achieved a large number of true positive predictions, with 178,687 instances, compared to false predictions for the other classes: 14,703 for class 1 and 1579 for class 2. For class 1, we see a significant number of correct predictions, totalling 37,735 instances. Additionally, there are notable false predictions: 7882 as class 0 and 7975 as class 2. This indicates that, despite using SMOTE, the model still struggles to accurately classify this class from the others. This may be due to the persistent imbalance in the test data. Regarding class 2, it records fewer true positive predictions, with 2116 instances. This is understandable, given that class 2 is the smallest class relative to the others. Moreover, class 2 shows a relatively high number of false predictions, with 135 as class 0 and 888 as class 1. These results suggest that the LNN model has difficulty predicting class 2 with high accuracy. Similar to class 1, this can be attributed to the class imbalance in the test data and the inherent complexity of class 2.

Fig. 19 shows how the training and validation loss change over the number of epochs for the best configuration of the LNN model identified earlier. The training loss curve consistently decreases, indicating that the model is still improving and fine-tuning its weights throughout the epochs. This downward trend suggests effective learning from the training data. Additionally, there is no evidence of overfitting on the graph. The validation loss curve, shown in orange, also generally declines, although with some slight fluctuations. These variations may reflect temporary changes within the validation data, which could momentarily impact performance. Overall, the decreasing trend in validation loss indicates that the LNN model can generalise well to unseen data. Furthermore, the convergence of both curves towards very low and similar values demonstrates the robustness of the LNN model when applied to new data.

4.2.2. Long short-term memory

In this subsection, we will focus mainly on assessing the performance of the LSTM model in predicting weather-affected flight delays. As outlined in Section 3.5, we employed a grid search approach. This method allows us to test the following hyperparameters: learning rate, number of epochs, number of layers, and number of hidden nodes. From this comprehensive analysis, we identified the optimal combinations of these hyperparameters to maximise the accuracy of flight delay predictions. Below is a detailed and critical evaluation of the results obtained with different LSTM configurations. The results also provided significant insights into the ability of the LSTM model to forecast weather-related flight delays.

Table 11 shows the hyperparameters tested in the LSTM model. The values for each hyperparameter are listed to illustrate the different configurations explored during the optimisation of the LSTM model. We selected the hyperparameter values during the grid search for several specific reasons. For the learning rate, we used values of 0.001 and 0.0001, similar to those in the LNN model. This is because, in a previous study using the LSTM model, these two values were chosen to train and evaluate the model. Regarding the number of epochs and hidden nodes, we used the same values as those used for the LNN model for these two hyperparameters, for similar reasons. We limited the epochs to 50 because a grid search could require significant computational resources in this case. For the number of layers, we aimed to test the LSTM model with a small number of 2, a medium number of 5, and a larger number of 10 layers. We did not adopt the same values used in the previous research by S. Kim and E. Park, which employed 1, 2, and 3 layers. This decision was motivated by the desire to explore a broader and more comprehensive range of values. Additionally, we used a fixed batch size of 1024 and a fixed dropout rate of 0.5 for all configurations. Fixing these two hyperparameters was essential because otherwise, the grid

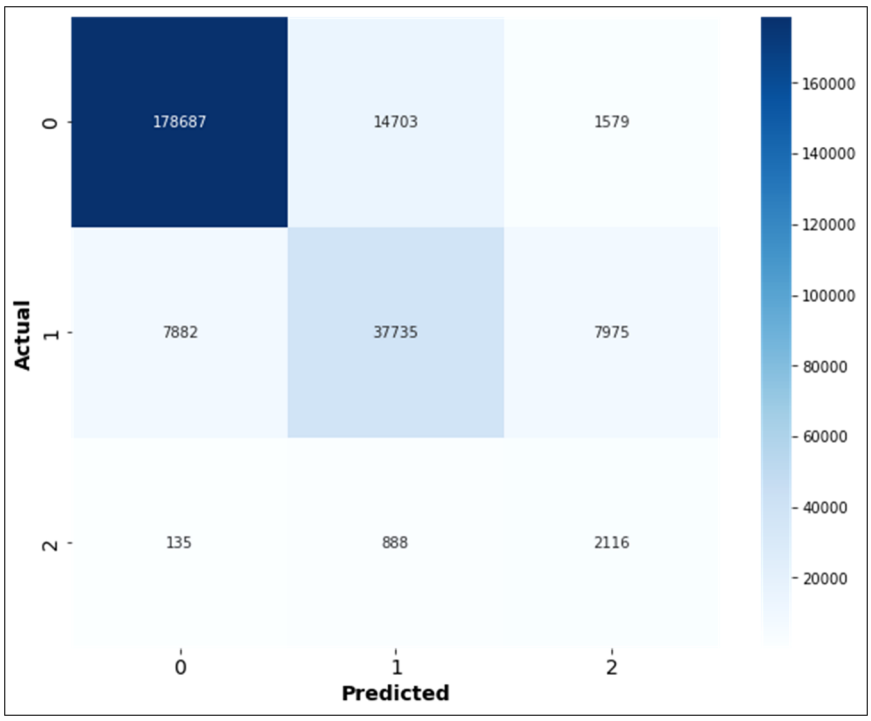


Fig. 18. Confusion matrix for the final LNN model.

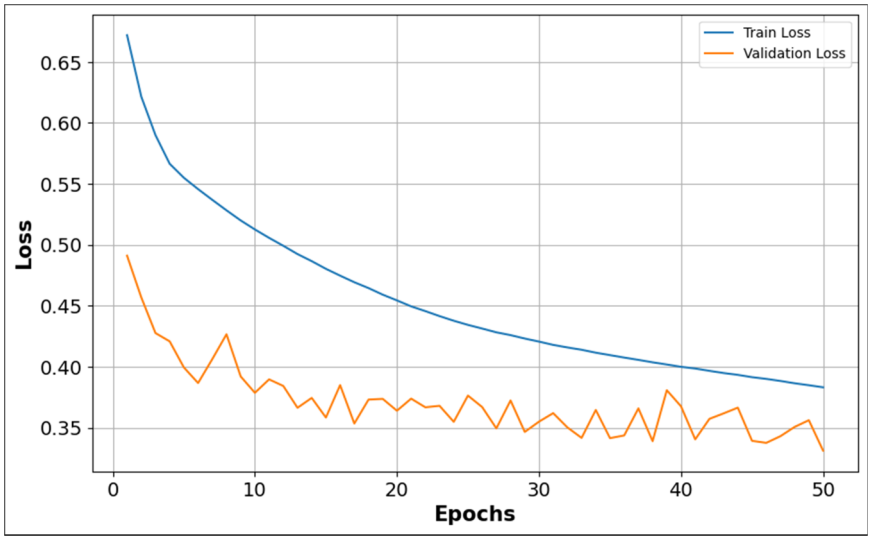


Fig. 19. Training and validation loss over epochs for the final LNN model.

Table 11
Tested hyperparameters in LSTM model.

Hyperparameters	Value
Learning rate	0.001, 0.0001
Number of Epochs	20, 50
Number of layers	2, 5, 10
Number of hidden nodes	50, 100, 150

search would require considerable resources and time.

For each weather data collection interval, we will display the results of the five best hyperparameter configurations for the LSTM model below, as shown in Table 12. These configurations were chosen based on their performance. By presenting these results, we can compare and

Table 12
5 Best hyperparameter configurations for the LSTM model with weather data collected at 0 h before flight departure.

S/No	Configurations
1	batch size=1024, number of epochs=50, number of layers=5, learning rate=0.001, dropout rate=0.5, number of hidden nodes=150
2	batch size=1024, number of epochs=50, number of layers=10, learning rate=0.001, dropout rate=0.5, number of hidden nodes=150
3	batch size=1024, number of epochs=50, number of layers=10, learning rate=0.001, dropout rate=0.5, number of hidden nodes=5
4	batch size=1024, number of epochs=20, number of layers=5, learning rate=0.001, dropout rate=0.5, number of hidden nodes=100
5	batch size=1024, number of epochs=50, number of layers=2, learning rate=0.0001, dropout rate=0.5, number of hidden nodes=150

analyse the performance of this DL model depending on the time before flight departure of the weather data collection. This approach allows us to provide a general overview of the model's ability to predict various weather timing scenarios.

Table 13 results demonstrate the influence of the learning rate on this LSTM model's performance. We observe that configurations with a learning rate of 0.001 generally perform slightly better than those with 0.0001. In fact, nearly all configurations in each top 5 adopt a learning rate of 0.001. For instance, with weather data collected 0 h before flight departure, configurations 1 and 2 with a learning rate of 0.001 achieve global accuracies of 86.14 % and 86.48 %, respectively. Configuration 5, which uses a learning rate of 0.0001, achieves an overall accuracy of 85.81 %. Furthermore, this trend also appears for other weather data collection timings. Specifically, with weather data collected 24 h before flight departure, results show that configurations with a learning rate of 0.001 reach a maximum accuracy of 86.65 %, compared to 85.69 % for those with 0.0001. This finding can be explained by the fact that a higher learning rate enables the model to converge more rapidly towards optimal solutions. However, caution is necessary because an excessively high learning rate can also increase the risk of overshooting the optimal solutions.

Additionally, the number of epochs influences the prediction performance of the LSTM model. In fact, configurations with 50 epochs generally perform better. This is highlighted by the fact that over 60 % of the top 5 configurations include 50 epochs. For example, using weather data collected 2 h before flight departure, configurations 1 and 2 with 50 epochs achieve overall accuracies of 87.13 % and 86.33 %, respectively. Configuration 5 with 20 epochs reaches an overall accuracy of 85.67 %. Furthermore, Configuration 1 has an F1-score of 0.66 for class 1 and 0.26 for class 2, compared to 0.66 for class 1 and 0.25 for class 2 in Configuration 5. Similar trends are observed regardless of the timing of weather data collection. For instance, with data collected 8 h before departure, the configuration with 50 epochs reaches a maximum global accuracy of 86.60 %, versus 85.77 % for the configuration with 20 epochs. This confirms that increasing epochs enables the model to better capture patterns in the data, thereby reducing prediction errors.

Furthermore, we observe that the impact of the number of layers on the model's performance varies. There is no specific number of layers that consistently optimises the performance of the LSTM model. When using weather data collected 0 h before departure, configurations with 5 and 10 layers slightly outperform those with only 2 layers. However, this trend is not consistent across all weather data collection times. For instance, with weather data gathered 16 h before departure, Configuration 1 with 2 layers achieves a global accuracy of 86.77 %, compared to 85.48 % and 85.15 % for Configuration 3 with 10 layers and 5 layers, respectively. Similarly, for weather data collected 24 h prior to departure, Configuration 1 with 2 layers exhibits an F1-score of 0.68 for class 1 and 0.24 for class 2, marginally better than Configuration 2 with 5 layers, which records an F1-score of 0.64 for class 1 and 0.23 for class 2. All these findings suggest that increasing the number of layers does not invariably enhance the model's performance. This indicates that increasing the complexity of the LSTM model is not always beneficial and may even diminish its effectiveness. Therefore, the optimal number of layers appears to depend on other factors, particularly additional hyperparameters and perhaps the timing of weather data collection.

Furthermore, we observe that the number of hidden nodes in each LSTM layer influences the model's ability to predict accurately. We note that using a higher number of hidden nodes, such as 150, often results in better model performance. In fact, configurations with this number of hidden nodes are the most common among the top 5 configurations, compared to those with fewer hidden nodes, namely 50 and 100. For example, when using weather data collected 0 h before departure, configurations with 150 hidden nodes achieve a maximum global accuracy of 86.48 %, compared to 86.24 % for the configuration with 100 hidden nodes. This trend varies across different weather data collection timings. Specifically, with weather data collected 2 h before departure, Configuration 1 with 50 nodes shows an F1-score of 0.95 for class 0, 0.66 for class 1, and 0.26 for class 2. In contrast, Configuration 5 with 150 nodes displays slightly lower F1-scores for classes 0 and 2, with 0.94 and 0.25 respectively. Moreover, with weather data collected 24 h before departure, configurations with 100 nodes predominantly feature in the top 5. Additionally, for weather data collected 24 or 48 h prior to departure, configurations with 150 nodes are highly present in both top 5 sets. This may indicate that weather data collected only a few days before flight departure might require a more complex LSTM model with more nodes. Based on these observations, the effectiveness of the number of nodes heavily depends on the timing of weather data collection and other hyperparameters. Although higher node counts may increase overfitting risk, configurations with 150 nodes perform slightly better and demonstrate greater consistency for classes 1 and 2 than those with 100 or 50 hidden nodes.

Finally, we observe that the timing of weather data collection significantly impacts the performance of the LSTM model. This effect is not necessarily linear, with important nuances depending on the timing. Moreover, we identify a general trend where weather data gathered closer to flight departure yields better scores across various performance metrics, although with some limitations. For instance, with weather data collected 2 h before departure, configurations achieve a maximum global accuracy of 87.13 %. In contrast, for weather data collected at 0 h before departure, the highest global accuracy attained by different configurations is slightly lower at 86.48 %. When examining the metric results for each class in detail, we observe that the outcomes are marginally better for weather data collected at 0 h rather than 2 h prior to departure. At 0 h, configurations show a maximum F1-score of 0.68 for class 1 and 0.26 for class 2, compared to 0.66 for class 1 and 0.26 for class 2 at 2 h. Performance remains relatively consistent even for weather data collected well in advance of departure. For example, configurations using weather data gathered 24 h before departure reach a maximum accuracy of 86.65 %. This suggests that the LSTM model can capture long-term weather trends relevant to predicting weather delays. There is a slight performance decline when weather data is collected 48 h before departure, with a maximum accuracy of 86.31 %. Additionally, we note a decrease in prediction performance for class 2 as the time between weather data collection and departure increases. Specifically, the minimum F1-score for class 2 drops from 0.25 with data collected 0 h before departure to 0.22 with data from 48 h prior. This minor decline seems reasonable, as it is unlikely that weather conditions 48 h in advance will closely resemble those when the flight approaches departure.

As a result, this critical and descriptive analysis of the results enabled

Table 13
Performance metrics of the LSTM model for the top 5 configurations with weather data collected at 0 h before flight departure.

Config.	Class 0			Class 1			Class 2			Global
	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1	Acc.
1	0.96	0.92	0.94	0.71	0.66	0.68	0.16	0.74	0.26	86.14 %
2	0.95	0.93	0.94	0.73	0.62	0.67	0.16	0.74	0.26	86.48 %
3	0.94	0.94	0.94	0.73	0.60	0.66	0.16	0.71	0.26	86.47 %
4	0.95	0.94	0.94	0.73	0.60	0.66	0.15	0.75	0.25	86.24 %
5	0.95	0.93	0.94	0.71	0.62	0.66	0.15	0.72	0.25	85.81 %

us to identify the best configuration for achieving optimal outcomes with the LSTM model. Although Configuration 1, with weather data collected 2 h before departure, shows slightly better overall accuracy, we opted for Configuration 1 using weather data collected 0 h before departure, as illustrated in Fig. 13, for several reasons. First, the two overall accuracies are relatively similar. Second, the selected configuration achieves better performance metrics for class 1. Since the paper focuses on flight delay prediction, particularly on weather delays, it is essential to obtain consistent and reliable results for both classes 1 and 2. Additionally, the chosen configuration offers a better ability to differentiate between the two delay classes. The hyperparameter values for the final LSTM model are as follows: learning rate = 0.001, number of epochs = 50, number of layers = 5, number of hidden nodes = 150, weather data collection timing: 0 h, dropout rate = 0.5, and batch size = 1024.

To gain a broader understanding of the final LSTM model's performance, we will primarily focus on analysing the training and validation loss functions and the confusion matrix.

Fig. 20 presents the confusion matrix for the final LSTM model. We observe that the performance for class 0 is excellent. In fact, this matrix shows that for on-time flights, the model made 179,350 correct predictions, with only 15,619 predictions misclassified as either class 1 or class 2. These results clearly illustrate the strong performance of the LSTM model in predicting class 0. For flights delayed due to non-weather reasons, the LSTM performs quite well, with 35,131 correct predictions. The confusion matrix shows some confusion when predicting class 1, with 8164 instances wrongly classified as class 0 and 10,297 instances misclassified as class 2. Regarding class 2, the model correctly predicted 2336 out of a total of 3139 instances. Consequently, it misclassified 135 instances as class 0 and 668 instances as class 1. From these observations, we conclude that while the LSTM model can effectively distinguish between on-time and delayed flights, it faces challenges in accurately identifying patterns in non-weather and weather delay data.

Fig. 21 depicts the training and validation loss across epochs. This graph demonstrates that the LSTM model converges effectively over the training process. Specifically, the blue training loss curve decreases steadily until it begins to level off as epochs increase, indicating good learning. The orange validation loss curve also trends downward, but less markedly than the training curve. Therefore, there is no clear evidence of overfitting. Additionally, the validation loss exhibits some fluctuations, likely reflecting a certain instability in the validation data.

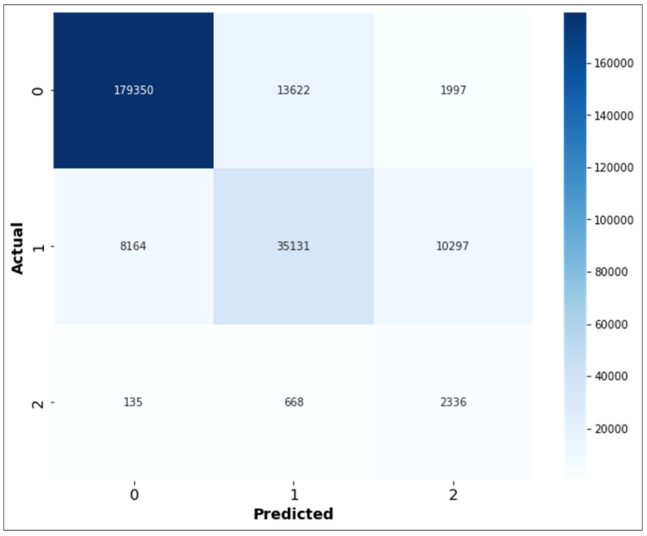


Fig. 20. Confusion Matrix for the LSTM model.

4.2.3. MultiLayer perceptron

In this subsection, we will discuss the evaluation of the MLP model's performance. As explained in Section 3.5, we adopted the same method as for the LSTM model, namely grid search. Using this approach, we were able to thoroughly test the following hyperparameters: learning rate, batch size, number of epochs, and the number of hidden nodes. Building on the results of all configurations tested after the grid search, we will present a descriptive and critical analysis of the MLP model below. This analysis will also include a discussion of the potential impact of weather data collection timing on the performance metrics used for the MLP model.

Table 14 presents the hyperparameter values tested in the grid search for the MLP model. Each value listed was selected for a specific reason. For the learning rate, we followed the same approach as for the previous two models, using values of 0.001 and 0.0001 as they are commonly adopted in optimising learning rates for ML/DL models (Bengio, 2012). Regarding the number of epochs, unlike the two earlier models, we expanded the tested range: 10, 20, 30, and 50 epochs. Similar to the LNN and LSTM models, we did not exceed 50 epochs to keep the grid search manageable. For batch size, unlike the LNN model, we chose only two values: 512 and 1024. This decision aimed to prevent the grid search from becoming too computationally demanding. We selected 1024 because it is close to the batch size of 1000 used in a previous study with the MLP model. Additionally, we chose 512, which, although increases training time, allows for more precise convergence. Concerning the number of hidden nodes, we adopted a different approach compared to the LNN and LSTM models. We used a variable number of nodes per layer, decreasing towards the output layer to reduce model complexity and computational resource needs. Furthermore, we fixed the number of layers at 3 and used a dropout rate of 0.5 across all configurations. This was done to avoid making the grid search too resource-intensive and time-consuming.

Similar to the LSTM model, we will present below the results obtained by the five best hyperparameter configurations for each timing of weather data collection in Table 15. To define these different tops 5, we selected them by considering all the performance metrics used in Table 16.

Based on the results obtained with the MLP model, we observe that the learning rate can influence its performance. Specifically, configurations with a learning rate of 0.001 generally outperform those with a learning rate of 0.0001. This is supported by the fact that most of the optimal configurations have a learning rate of 0.001. For example, in the case of weather data collected 0 h before flight departure, all configurations in the top 5 use a learning rate of 0.001. These configurations reach a maximum global accuracy of 86.96 %. Additionally, we see that, for weather data from 4 h before departure, Configuration 1 with a learning rate of 0.001 achieves an F1-score of 0.69 for class 1 and 0.26 for class 2. In comparison, Configuration 5 with a learning rate of 0.0001 has an F1-score of 0.66 for class 1 and 0.24 for class 2. Although the difference is slight, it indicates a trend where a higher learning rate helps the model converge faster and identify patterns in the data more effectively. Moreover, we notice that, in some cases, configurations with a learning rate of 0.0001 can perform better than those with a learning rate of 0.001. For instance, using data collected 2 h before departure, the best configuration has a learning rate of 0.0001, yielding a global accuracy of 86.09 %. Overall, however, a learning rate of 0.001 tends to offer marginally better performance for the MLP model. This trend can be explained by faster convergence and improved ability to adjust weights during training, leading to a better capture of weather data patterns, especially for classes 1 and 2.

Additionally, we observe that the number of epochs influences the performance of the MLP model. Among various tests with four different epoch counts, the configurations most frequently appearing in the top 5 are those with a high number of epochs, especially 50 epochs. Generally, increasing the number of epochs enhances the overall accuracy, although this improvement is not always linear. For instance, for

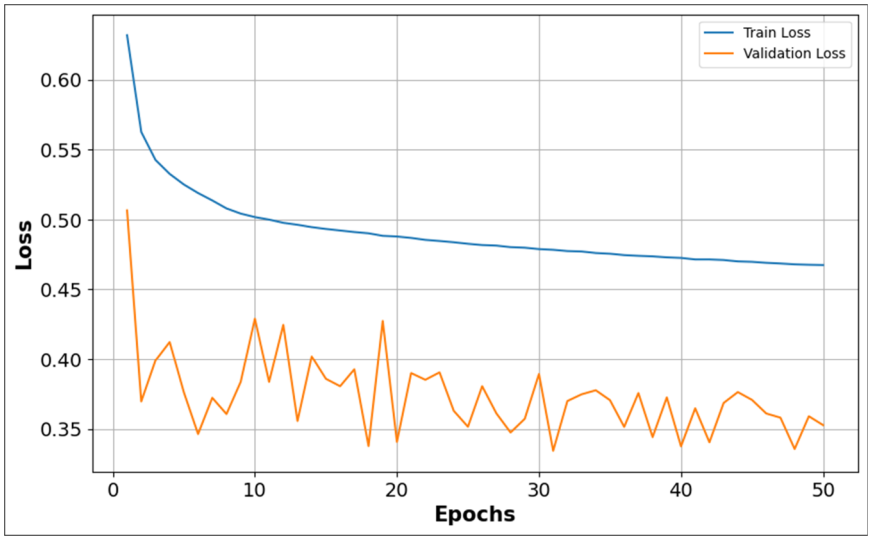


Fig. 21. Training and validation loss over epochs for the LSTM model.

Table 14
Tested hyperparameters in MLP model.

Hyperparameters	Value
Learning rate	0.001, 0.0001
Number of Epochs	10, 20, 30, 50
Batch Size	512, 1024
Number of hidden nodes	[128, 64, 32], [256, 128, 64], [512, 256, 128]

Table 15
5 Best hyperparameter configurations for the MLP model with weather data collected at 0 h before flight departure.

S/No	Configurations
1	batch size=1024, number of epochs=50, number of layers=3, learning rate=0.001, dropout rate=0.5, number of hidden nodes=[512,256,218]
2	batch size=1024, number of epochs=20, number of layers=3, learning rate=0.001, dropout rate=0.5, number of hidden nodes=[128, 64, 32]
3	batch size=512, number of epochs=50, number of layers=3, learning rate=0.001, dropout rate=0.5, number of hidden nodes=[512, 256, 128]
4	batch size=512, number of epochs=20, number of layers=3, learning rate=0.001, dropout rate=0.5, number of hidden nodes=[512, 256, 128]
5	batch size=512, number of epochs=30, number of layers=3, learning rate=0.001, dropout rate=0.5, number of hidden nodes=[512, 256, 128]

weather data collected 2 h prior to departure, configurations with 50 epochs dominate the top 5, achieving a maximum global accuracy of 86.09 %. Furthermore, in the case of weather data from 0 h before departure, Configuration 1 with 50 epochs attains an F1-score of 0.69 for class 1 and 0.27 for class 2. Conversely, Configuration 2 on the same data reaches an F1-score of 0.67 for class 1 and 0.26 for class 2. Nonetheless, valuable results can be obtained with fewer epochs. For example, with weather data gathered 16 h before departure, the best configuration uses 10 epochs, with F1-scores of 0.68 and 0.25 for classes 1 and 2, respectively. This can be attributed to the other hyperparameters being well optimised. Therefore, while configurations with more epochs generally offer better performance and higher metric scores, some cases demonstrate that fewer epochs can be equally effective.

Furthermore, the results regarding the performance of the MLP model reveal some intriguing trends related to batch size. Overall, we observe that the total performance varies only slightly across the tested configurations, regardless of batch size. However, some nuances are notable. Generally, configurations with a batch size of 512 tend to have a slight advantage over those with a batch size of 1024. For instance, among all the top-performing configurations evaluated at different data collection times, most use a batch size of 512. Taking weather data from 2 h before departure as an example, all top 5 configurations rely on a batch size of 512, reaching a maximum global accuracy of 86.09 %. Similar patterns are visible for other timings, though less pronounced. For weather data from 4 h prior, Configuration 1 with a batch size of 512 achieves a recall of 0.66 for class 1 and a global accuracy of 86.97 %, compared to Configuration 4 with a batch size of 1024, which records a recall of 0.63 for class 1 and a global accuracy of 86.62 %. These results suggest that a batch size of 512 may offer better generalisation due to more frequent weight updates. However, this performance difference is very subtle and not consistently observed. Sometimes, batch sizes of 1024 can perform similarly or even slightly better. For example, with weather data from 8 h before departure, Configuration 1 using a batch size of 1024 achieves an F1-score of 0.68 for class 1 and 0.26 for class 2. In comparison, Configuration 5 with a batch size of 512 records an F1-score of 0.67 for class 1 and 0.24 for class 2. Although batch sizes of 512 generally exhibit a small tendency towards better overall performance, the impact of this hyperparameter on the MLP model's results remains moderate. These implications may vary based on other hyperparameters and the timing of weather data collection.

Table 16
Performance metrics of the MLP model for the top 5 configurations with weather data collected at 0 h before flight departure.

Config.	Class 0			Class 1			Class 2			Global
	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1	Acc.
1	0.96	0.92	0.94	0.71	0.67	0.69	0.17	0.74	0.27	86.31 %
2	0.95	0.94	0.95	0.75	0.61	0.67	0.16	0.74	0.26	86.96 %
3	0.96	0.91	0.94	0.71	0.68	0.69	0.16	0.74	0.26	86.18 %
4	0.96	0.92	0.94	0.72	0.67	0.69	0.16	0.74	0.26	86.11 %
5	0.96	0.92	0.94	0.72	0.66	0.69	0.15	0.76	0.25	85.94 %

Furthermore, we find that the number of hidden nodes impacts the performance of the MLP model. A general trend shows that configurations with a hidden node structure of [512, 256, 128] tend to perform better than those with simpler structures. In fact, these more complex configurations frequently rank among the best choices for different collection timings, appearing in over 60 % of cases. For instance, using data gathered 4 h before flight departure, Configuration 1 with a node structure of [512, 256, 128] achieves an overall accuracy of 86.97 %. In contrast, Configuration 5 with a simpler node structure of [256, 128, 64] reaches an accuracy of 86.00 %. Additionally, such complex structures are effective across other timings. For example, at 24 h, Configuration 1 with a node structure of [512, 256, 128] attains an F1-score of 0.69 for class 1 and 0.25 for class 2, along with an overall accuracy of 86.68 %. However, the influence of the number of nodes is not consistent. In some cases, configurations with fewer nodes perform equally well or better. For example, with weather data from 48 h prior, Configuration 1 with [256, 128, 64] nodes achieves 86.37 % accuracy, compared to 85.88 % for Configuration 2 with [512, 256, 128] nodes. This variability indicates a complex interaction between the number of nodes, other hyperparameters, and the timing of the weather data. Therefore, we observe that configurations with more nodes generally perform better with more recent data. This can be attributed to the capacity of complex configurations to capture more intricate patterns. Nevertheless, for weather data collected well in advance of flight departure, simpler configurations can sometimes deliver comparable or even superior results.

Finally, various tests conducted to evaluate the performance of the MLP model using different configurations reveal the significant impact of the timing of weather data collection on its effectiveness. We observe a general trend where data gathered closer to the time of flight departure results in better performance. For example, with weather data collected 0 h before departure, Configuration 1 attains F1-scores of 0.69 and 0.27 for classes 1 and 2 respectively. For weather data collected 2 h prior to departure, Configuration 1 achieves an F1-score of 0.68 for class 1 and 0.26 for class 2. We find that the optimal configurations for each timing of weather data collection have fairly similar scores on these performance metrics, except for data collected 48 h before departure. This decline strongly indicates that using weather data close to the flight departure significantly enhances predictive accuracy. Moreover, we note that for class 2, the F1-score decreases from 0.27 for weather data gathered 0 h before departure to 0.23 for data collected 48 h prior. This demonstrates that the model struggles to predict flight delays caused by weather conditions with older data. This makes sense, as weather conditions recorded 48 h before a flight are unlikely to remain the same at departure time. Nonetheless, the stability in relative performance up to 24 h before departure suggests that the MLP model can identify relevant meteorological patterns even with data collected a day in advance.

Based on the thorough analysis of the results, we identified the optimal hyperparameter settings for the MLP model: Learning rate = 0.001, Number of epochs = 50, Number of layers = 3, Number of hidden nodes = [512, 256, 128], Timing of weather data collection: 0 h, Dropout rate = 0.5, Batch size = 1024.

This represents Configuration 1 with weather data collected 0 h before departure. This choice was made to the detriment of other interesting configurations, such as Configuration 1 with weather data collected 4 h before departure. Indeed, the selected configuration showed a fairly high overall accuracy, with the best F1-scores obtained for classes 1 and 2 among all the tested configurations. Additionally, the precision achieved for class 2 is particularly notable for this optimal configuration. Below, we will focus on the performance of the best MLP model chosen, aiming to analyse its performance from various perspectives. To do this, we will primarily examine the training and validation loss, as well as the confusion matrix.

Fig. 22 displays the confusion matrix of the final MLP model. This matrix illustrates an overall strong performance in classifying class 0, namely, on-time flights. The final MLP model correctly predicts 178,853 instances of class 0 and exhibits a relatively low error rate, with only 13,987 instances misclassified as class 1 and 2129 as class 2. For class 1, the model demonstrates a fairly good classification ability, with 36,046 correct predictions. However, it misclassified 7909 instances as class 0 and 9637 as class 2. Lastly, class 2 performed less well, with only 2336 correct predictions, compared to 677 misclassified as class 1 and 126 as class 0. These results indicate that the model has some difficulty in clearly distinguishing whether a flight is delayed due to non-weather

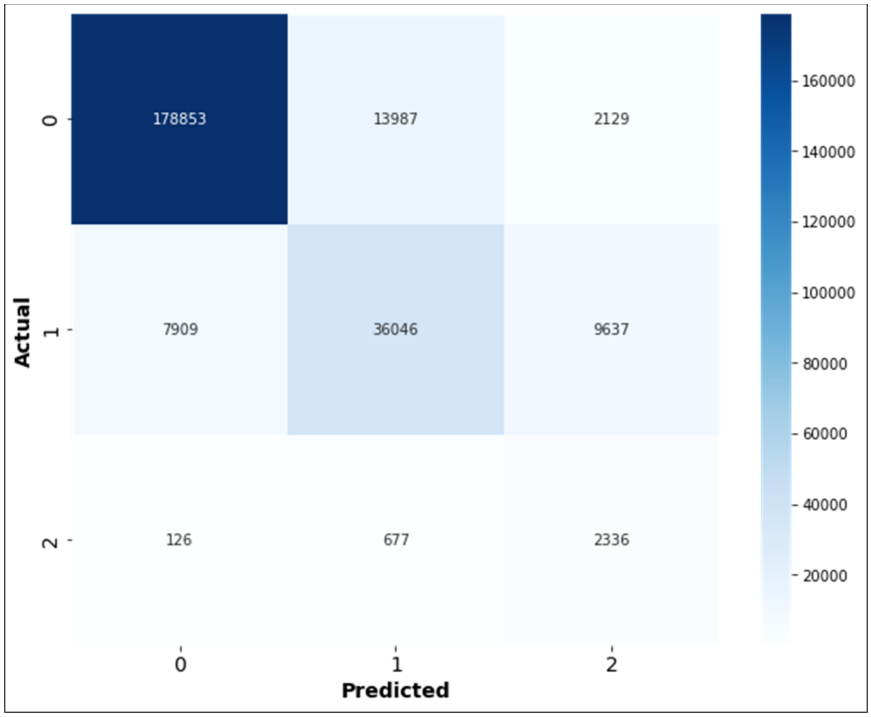


Fig. 22. Confusion Matrix for the MLP model.

conditions or weather conditions. Indeed, too many flights are incorrectly classified as delayed due to weather conditions. Nonetheless, it is preferable to predict class 2 rather than under-predict this class.

Fig. 23 illustrates the training and validation loss for the MLP model. The graph shows that the training loss curve, in blue, generally decreases across epochs. It also appears to level off as the number of epochs increases. This suggests that the model continues to learn effectively and adjusts through weight updates. Regarding the validation loss, in orange, it exhibits much higher variability. While it appears to decrease slightly, it mostly fluctuates around a certain value after 10 epochs. This stability may indicate that the MLP model is reaching a point of overfitting, as it keeps learning from training data without any clear improvement in validation data. The consistent validation loss also suggests that the model strikes a good balance between bias and variance. These fluctuations could result from imbalances in the validation data.

4.2.4. Model comparative analysis

Below, we will compare the performance of the best LNN, LSTM, and MLP models. The main aim of this comparison is to identify the most effective DL model developed among these three by using several performance metrics. This comparative, constructive, and critical analysis will allow us to highlight the strengths and weaknesses of each approach from different perspectives.

Table 17 summarises the performance of the top LNN, LSTM, and MLP models. It is important to note that all three models utilise weather data collected 0 h before flight departure. This indicates that using the most recent weather data before departure is more effective for accurately predicting flight delay weather outcomes.

The analysis of these models' performance reveals similarities and subtle differences. We observe that these DL models achieve similar overall accuracy. The LNN model scores 86.82 %, slightly surpassing the MLP and LSTM models, which reach 86.31 % and 86.14 %, respectively. This minor variation suggests that these models achieve comparable overall performance. Additionally, we observe similar performance among the three models for class 0, with all achieving identical scores across the three performance metrics. Such results demonstrate that all three models possess an excellent ability to identify on-time flights. Regardless of the chosen model, performance for class 0 remains consistently high.

Furthermore, we notice that the performance differences are more significant for the other two classes. For class 1, the LNN outperforms the other two models with an F1-score of 0.71, compared to 0.69 for the MLP and 0.68 for the LSTM. Likewise, we see that the LNN achieves a higher recall for class 1 than the other models. This suggests that the LNN model is more effective at correctly identifying non-weather delays. The results are quite interesting for class 2, namely delays caused by weather conditions. Although the LNN model has the highest F1-score, the other two models demonstrate better recall, with 0.74 compared to 0.67 for the LNN. This indicates that these models are more likely to predict a flight as class 2, even if this leads to slightly lower precision.

Table 18 offers intriguing insights into the computational efficiency of the three DL models. Regarding model training, we observe that the MLP model achieves the best result among these models. In fact, its training time is approximately 1015 s. Its training duration is relatively short compared with the other two models. This finding highlights the MLP model's strong computational efficiency. In contrast, the LSTM model requires more than twice as long, with a training time of 2853 s. The LNN model records the worst result, taking 21,093 s. This LNN model's performance reflects its slow training.

Regarding model evaluation, we observe that all these models yield quite promising results with relatively brief testing durations. As with training times, the same pattern emerges in the models' performance regarding computational efficiency. Indeed, the MLP model performs very well again, with a testing time of 3.1419 s. In comparison, the LSTM model takes 8.3534 s, while the LNN model requires 29.4773 s.

It is helpful to question the relevance of training execution time as a metric for assessing model performance. Indeed, in many real-time prediction scenarios, the use of pretrained models is often considered. This approach is valuable because it can significantly reduce the computational costs associated with training. As a result, this performance metric becomes less significant for performance evaluation. Employing a pretrained model is particularly beneficial for the LNN since its training time is notably long. By adopting a pretrained model approach, the LNN model can overcome this disadvantage and achieve performance closer to that of the other two models while maintaining computational efficiency.

From all these observations of the performance of the models, we can draw the following conclusions:

- The LNN model shows the best performance for prediction accuracy. It also delivers an interesting balance between classes 1 and 2. In addition, its F1-score for class 2 is very beneficial for predicting flight weather delays. Although acceptable, its testing time remains the longest of the three models.

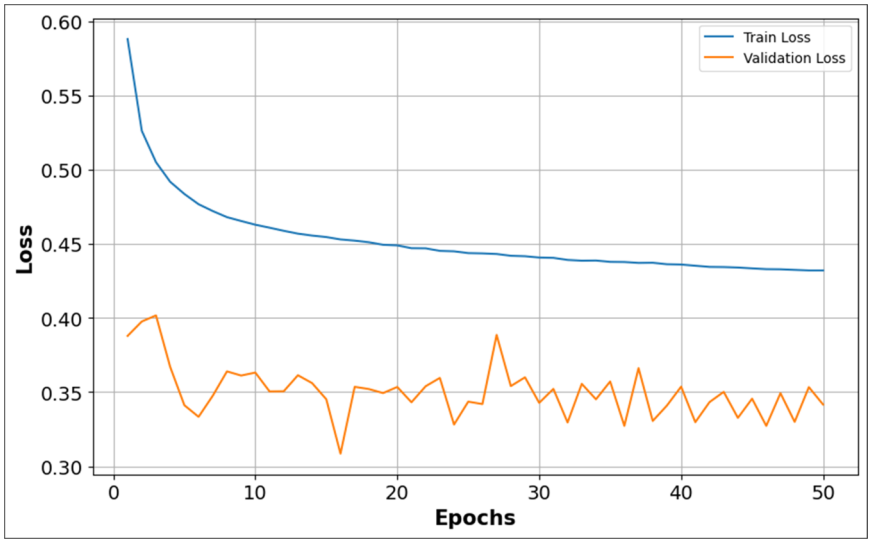


Fig. 23. Training and validation loss over epochs for the MLP model.

Table 17
Summary table of the performance of the best LNN, LSTM and MLP models.

Model	Class 0			Class 1			Class 2			Global
	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1	Acc.
LNN	0.96	0.92	0.94	0.71	0.70	0.71	0.18	0.67	0.29	86.82 %
LSTM	0.96	0.92	0.94	0.71	0.66	0.68	0.16	0.74	0.26	86.14 %
MLP	0.96	0.92	0.94	0.71	0.67	0.69	0.17	0.74	0.27	86.31 %

Table 18
Execution time in training and testing for LNN, LSTM and MLP models.

Model	Execution Time	
	Training Time (s)	Testing Time (s)
LNN	21,093.2756	29.4773
LSTM	2853.6685	8.3534
MLP	1014.9677	3.1419

- The MLP model delivers an acceptable compromise between the prediction performance and the execution speed. Its predictive performance is slightly lower than that of the LNN model, especially for classes 1 and 2. Its testing time is the shortest among the three models. This may be a decisive factor.
- The LSTM model shows the poorest performance in terms of prediction. Indeed, the results achieved for classes 1 and 2 are slightly lower than those of the MLP model. Its testing time, while longer in comparison with the MLP model, is still within an acceptable range.

Given the LNN model's suitability for real-time applications, we selected it as the best option. After training, this deep learning model can be saved and deployed directly for predictions. This addresses the primary issue related to the LNN model's training time. We prioritised prediction accuracy over computational efficiency. This decision was motivated by the goal of maximising the model's accuracy for class 2, which is particularly difficult to predict. Additionally, although the MLP model offers notable advantages, the difference in testing time is not sufficient to justify choosing it over the LNN model. By saving the LNN after training, we ensure that future predictions can be made relatively quickly while maintaining good accuracy.

To broaden this analysis, we will now compare the performance of the best model identified earlier with two other models from previous research in this field. This comparison is essential for thoroughly evaluating the potential improvements offered by the LNN model developed in this paper relative to prior work. The two models from previous research include a DL model and a machine learning (ML) model. Using them, we can assess the performance of the best model from various perspectives. This comparative evaluation will help us determine whether the proposed model signifies a genuine advance over existing techniques. In this way, we will be able to conclude the contributions of the work presented in this paper to the current state of the art in area.

Table 19 displays a performance comparison between the LNN model developed in this paper and two models from the study by S. Kim and E. Park (Kim & Park, 2024): an LSTM model and an RF model. We selected these two models for several reasons. First, the previous study was conducted in 2024, allowing comparison with recent results. Additionally, the study uses data very similar to those used here, sourced from the exact origin and related to JFK airport. The only difference is that the LNN model was trained with two additional years of data, while the models in the study also incorporated data from the COVID-19 period. That previous study also tested their models with weather datasets collected at different times before the flight. It is essential to highlight a key difference in the structure of the results. The LNN model distinguishes three classes, whereas the LSTM and RF models identify only two. The LNN model separates delays into two categories: delays caused by non-weather conditions (class 1) and those caused by weather conditions (class 2). In contrast, the LSTM and RF models uniformly predict flight delays as class 1, regardless of cause. This structural difference in classification complicates direct comparison but provides some interesting insights. Moreover, we have not considered the metrics related to training and testing times, as the decision was made because the LSTM and RF models did not use the same computational resources as the LNN model.

Regarding overall accuracy, we find that the LNN model developed in this paper achieves a higher score than the model reported by S. Kim and E. Park (Kim & Park, 2024). In fact, the LNN model achieves 86.82 % accuracy, slightly surpassing the LSTM model at 85.2 % and the RF model at 84.3 %. Although this difference is minimal, it indicates a certain improvement in the overall performance of the LNN model relative to these previous approaches.

For the on-time flight class, we observe that the LNN model delivers superior results compared with the other two models, LSTM and RF. The LNN model achieves very high precision, recall, and F1-score of 0.96, 0.92, and 0.94, respectively. In contrast, the LSTM and RF models show lower scores, ranging from 0.83 to 0.88, for the same performance metrics. This demonstrates that the LNN model is more effective at predicting on-time flights than the two models from the previous study. This improvement highlights one of the advantages of adopting the LNN model.

Furthermore, comparing the other classes is more difficult because the result structures differ between the LNN model and the RF and LSTM models. We note that both the LSTM and RF models perform well for predicting flight delays, with F1-scores of 0.85 and 0.84, respectively. The LNN model achieves an F1-score of 0.71 for delays caused by non-weather conditions and 0.29 for those caused by weather conditions. This highlights a significant difference between the models' ability in the previous study and the LNN model in this paper to predict delays. It is hard to determine whether the LNN model underperforms compared to the other two, as they do not explicitly distinguish between weather and non-weather delays. We can attribute this difference to the fact that differentiating weather and non-weather delays is likely a more complex task than simply classifying delays.

Table 19
Comparison of the performance of the best model with two models from previous research.

Model	Class 0			Class 1			Class 2			Global
	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1	Acc.
LNN	0.96	0.92	0.94	0.71	0.70	0.71	0.18	0.67	0.29	86.82 %
	Class 0			**Class 1 (Combined)**						**Global**
Model	**Prec.**	**Rec.**	**F1**	**Prec.**		**Rec.**		**F1**		**Acc.**
LSTM (Kim & Park, 2024)	0.83	0.88	0.86	0.88		0.82		0.85		85.02 %
MLP (Kim & Park, 2024)	0.84	0.86	0.85	0.85		0.85		0.84		84.03 %

To enable a direct comparison for delay prediction, we followed the subsequent approach. Using the confusion matrix in Fig. 12, we combined predictions from classes 1 and 2 into a single class. Then, we recalculated each performance metric. With this method, the LNN model achieves 0.86 for precision, 0.75 for recall, and 0.80 for F1-score. These results show that when classes 1 and 2 are merged to assess the overall performance of the LNN model on flight delays, the scores are slightly lower than those of the other two models for some metrics. Despite this decline in performance on the delay classes, the LNN model offers a notable advantage in terms of interpretability and usefulness. Indeed, the ability to differentiate between types of delay may be essential.

From these observations, it is evident that the LNN model developed in this paper offers additional value in the field of flight delay prediction. The LNN model is characterised by increased overall accuracy and improved performance in predicting class 0, while also allowing for the differentiation of delays into two types. With the two delay classes of the LNN model, more detailed predictions are possible, providing better insights into the factors influencing flight delays. However, the LNN model has some limitations, including difficulties in predicting flight delays compared to previous models. Additionally, these limitations highlight a performance imbalance between predicting delay classes and class 0. Several factors may account for these limitations. Firstly, weather data is often uncertain, which can affect the accuracy of weather-related delay predictions. Furthermore, many other factors beyond weather conditions are likely to cause flight delays, and these may not be accurately reflected in the data used for training.

5. Conclusions and future works

This paper focuses on implementing and evaluating DL models for predicting flight delays, with particular emphasis on those caused by weather conditions. The main goal was to introduce a novel DL approach, previously unseen in this field, through the development of the LNN model. Additionally, this thesis is distinguished by its detailed analysis of delays, which enables the distinction between delays caused by weather and other factors.

We examined various neural network (NN) architectures, including LNN, LSTM, and MLP models. By implementing a comprehensive data processing approach, from addressing missing data to managing data imbalance with the SMOTE algorithm, we ensured each model received high-quality data for training. The models were optimised using established methods, such as grid search and trial-and-error, to identify the best-performing configurations.

The results showed that the LNN model performs slightly better overall than the other two models tested in this thesis, namely the LSTM and MLP models. Additionally, the LNN model demonstrated superior performance in specific metrics compared to the LSTM and RF models used in a previous study. We were able to enhance certain aspects of flight delay prediction, particularly global accuracy and precision, recall, and F1-score for on-time flights. These improvements highlight the effectiveness of the LNN model and its capacity to outperform previous approaches in this field. This thesis also demonstrated that using weather data collected near the departure time improves the overall accuracy of arrival delay predictions. By differentiating delays by cause, this thesis provides additional insights into delay prediction.

However, this paper has some limitations. Although the LNN model performed well overall, predicting weather delays remains challenging. The results for these delays produced by the LNN model showed significantly lower scores compared to other categories. Therefore, there is a clear need for improvement in this specific area. Additionally, the LNN model is very demanding on computational resources, leading to longer training times than other methods. Finally, since weather data is integrated into the model, some level of uncertainty may be present in the data, which can negatively affect the robustness of the model. Overall, the LNN model provides a strong foundation for flight delay prediction, opening new avenues to improve prediction accuracy and optimise flight operations.

5.1. Future works

Potential directions for future work and improvements have been identified based on the observations, results, and limitations outlined in this thesis. Firstly, the optimisation of hyperparameters can be further enhanced. This could involve testing additional configurations for the LNN model by adjusting the dropout rate or expanding the range of hyperparameter values explored during the grid search for the LSTM and MLP models. We might also include a hyperparameter representing the SMOTE imbalance ratio and vary it to determine whether this improves prediction accuracy. Furthermore, simplifying the optimisation process of the LNN model by transitioning from a trial-and-error approach to a more systematic grid search could be considered. To facilitate this, we will need to identify ways to reduce the resource demands of the model.

Secondly, to improve the predictions, especially for class 2, it could be helpful to collect weather data from the itineraries of each flight. Indeed, arrival delays can also be influenced by weather conditions encountered during or shortly before the flight's arrival. Additionally, to better evaluate the models and their generalisation, it may be relevant to assess the models using flight and weather data from another airport. This approach would help determine whether the developed models can be applied at other airports while maintaining consistent predictions. Furthermore, to broaden the scope of this thesis, it would be beneficial to consider cancelled flights as a form of indefinite delay. Consequently, when predicting delays, the models should also be able to predict whether a flight is cancelled, regardless of weather conditions. To facilitate this, a second output layer would need to be added to the models, specifically for predicting cancellations. Lastly, it would be valuable to examine the prediction of flight delays caused by other factors. This would involve analysing and integrating additional data related to operational or technical issues, which can contribute to airline delays. By including other causes alongside weather delays, the predictive utility of the model can be enhanced, resulting in a more robust tool.

Data statement

The source code and datasets used in the experiments are released under the MIT License and are publicly available through https://doi.org/10.57996/cran.ceres-2797.

Funding

We gratefully acknowledge Cranfield University for providing access to High Performance Computing (HPC) facilities, which were instrumental to this work.

CRediT authorship contribution statement

Desmond B. Bisandu: Conceptualization, Methodology, Investigation, Data curation, Writing – original draft. **Majuran Chandrakumar:** Conceptualization, Methodology, Investigation, Data curation, Writing – original draft. **Irene Moulitsas:** Conceptualization, Supervision, Project administration, Funding acquisition, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Supplementary materials

Supplementary material associated with this article can be found, in

the online version, at doi:10.1016/j.iswa.2026.200656.

Data availability

Data and software will be made available

References

Air transport, passengers carried. https://data.worldbank.org/indicator/IS.AIR. PSGR, 2022. [Online; Last accessed 28/07/2024].

Flight tracking statistics. https://www.flightradar24.com/data/statistics, 2024. [Online; Last accessed 28/07/2024].

Airline on-time statistics and delay causes. https://www.transtats.bts.gov/OT_Delay/ot_delaycause1.asp?qv52ynB=qn6n&20=E, 2024. [Online; Last accessed 28/07/2024].

Cost of disrupted flights to the economy. https://img.airhelp.com/Documents/AH_disruption_economic_cost.pdf?updatedAt=1695047330127, 2022. [Online; Last accessed 28/07/2024].

Airline on-time performance and causes of flight delays. https://www.bts.gov/explore-topics-and-geography/topics/airline-time-performance-andcauses-flight-delays, Mar 2021. [Online; Last accessed 28/07/2024].

Understanding the reporting of causes of flight delays and cancellations. https://www.bts.gov/topics/airlines-and-airports/understandingreporting-causes-flight-delays-and-cancellations, Apr 2024. [Online; Last accessed 28/07/2024].

How accurate are weather forecasts? 5 reasons for inaccuracy. https://tempest.earth/resources/how-accurate-are-weather-forecasts/#:~:Text=The%20accuracy%20of%20weather%20predictions,that%20the%20weather%20will%20change., 2024. [Online; Last accessed 28/07/2024].

A. Anupkumar. Investigating the costs and economic impact of flight delays in the aviation industry and the potential strategies for reduction. https://scholarworks.lib.csusb.edu/cgi/viewcontent.cgi?article=2885&context=etd, May 2023. [Online; Last accessed 23/06/2024].

U.s. passenger carrier delay costs. https://www.airlines.org/dataset/us-passenger-carrier-delay-costs/#:~:Text=Assuming%20%2447%20per%20hour*%20as,2019%20to%20be%20%2433%20billion., Jul 2024. [Online; Last accessed 27/07/2024].

What exactly is the environmental impact of flight disruptions? https://www.airhelp.com/en-int/blog/what-exactly-is-the-environmentalimpact-of-flight-disruptions/, 2023. [Online; Last accessed 28/07/2024].

Mamdouh, M., Ezzat, M., & Hefny, H. A. (2023). A novel intelligent approach for flight delay prediction. *Journal of Big Data, 10*(12).

Hatıpoglu, I., & Tosun, O. (2024). Predictive modeling of flight delays at an airport using machine'' learning methods. *Applied sciences*.

Song, C., Guo, J., & Zhuang, J. (2020). Analyzing passengers' emotions following flight delays—A 2011–2019 case study on skytrax comments. *Journal of Air Transport Management, 89*. Oct.

Efthymiou, M., Njoya, E. T., Lo, P. L., Papatheodorou, A., & Randall, D. (Dec 2018). The impact of delays on customers' satisfaction: An empirical analysis of the british airways on-time performance at heathrow airport. *Journal of Aerospace Technology and Management, 11*.

Choi, S., Kim, Y. J., Briceno, S., & Mavris, D. (2016). Prediction of weather-induced airline delays based on machine learning algorithms. In *2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC)* (pp. 1–6). pages.

Etani, N. (2019). Development of a predictive model for on-time arrival flight of airliner by discovering correlation between flight and weather data. *Journal of Big Data*.

Sujay, V., Lalitha, S., Sreedhar, S. S., Omotunde, H., Vijaya, S., & Mery, K. P. (2024). Enhancing hyperparameters for improved flight delay prediction using machine learning algorithms. In *2024 International Conference on Integrated Circuits and Communication Systems (ICICACS)* (pp. 1–5). pages.

Kim, Y. J., Choi, S., Briceno, S., & Mavris, D. (2016). A deep learning approach to flight delay prediction. In *2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC)* (pp. 1–6). pages.

Kim, S., & Park, E. (2024). Prediction of flight departure delays caused by weather conditions adopting data-driven approaches. *Journal of Big Data, 11*.

Tang, Y. (2022). Airline flight delay prediction using machine learning models. In *2021 5th International Conference on E-Business and Internet (ICEBI 2021)* (pp. 151–154). pages.

Hendrickx, R., Zoutendijk, M., Mitici, M., & Schafer, J. (2021). Considering airport planners' ¨preferences and imbalanced datasets when predicting flight delays and cancellations. In *2021 IEEE/AIAA 40th Digital Avionics Systems Conference (DASC)* (pp. 1–10). IEEE. pages.

Sajid, H. (2023). *Liquid neural networks: Definition, applications, & challenges - unite.ai.* https://www.unite.ai/liquid-neural-networks-definition-applicationschallenges/. May[Online; Last accessed 03/06/2024].

Boesch, G. (2024a). *What are liquid neural networks?*. https://viso.ai/deep-learning/what-are-liquid-neural-networks/. May.

Keary, T. (2023). *Liquid neural network (lnn)*. https://www.techopedia.com/definition/liquid-neural-network. SepOnline; Last accessed 03/06/2024].

B. Tyagi. Liquid neural networks: Revolutionizing ai with dynamic information flow. https://tyagi-bhaumik.medium.com/liquid-neural-networksrevolutionizing-ai-with-dynamic-information-flow-30e27f1cc912, Jul 2023. [Online; Last accessed 03/06/2024].

What is lstm – long short term memory? https://www.geeksforgeeks.org/deeplearning-introduction-to-long-short-term-memory/, Jun 2024. [Online; Last accessed 12/08/2024].

Saxena, S. (2024). *What is lstm? introduction to long short-term memory*. https://www.analyticsvidhya.com/blog/2021/03/introduction-to-long-shortterm-memory-lstm/. AugOnline; Last accessed 12/08/2024].

P. Srivatsavaya. Lstm — Implementation, advantages and diadvantages. https://medium.com/prudhviraju.srivatsavaya/lstm-implementationadvantages-and-diadvantages-914a96fa0acb, Oct 2023. [Online; Last accessed 12/08/2024].

Shah, E. (2024). *Multilayer perceptron in machine learning*. https://www.scaler.com/topics/multilayer-perceptron/. AprOnline; Last accessed 12/08/2024].

What is multilayer perceptron? https://deepchecks.com/glossary/multilayerperceptron/, 2023. [Online; Last accessed 12/08/2024].

What is a multilayer perceptron (mlp) neural network? https://www.shiksha.com/online-courses/articles/understanding-multilayer-perceptron-mlpneural-networks/#art, Jan 2024. [Online; Last accessed 12/08/2024].

Multilayer perceptron (mlp). https://blog.marketmuse.com/glossary/multilayer-perceptron-definition/, 2024. [Online; Last accessed 12/08/2024].

Us bureau of transportation statistics. https://transtats.bts.gov/DL_SelectFields.aspx?gnoyr_VQ=FGJ&QO_fu146_anzr=b0-gvzr, 2024. [Online; Last accessed 03/06/2024].

Us weather underground. https://www.wunderground.com/history/daily/us/ny/new-york-city/KJFK, 2024. [Online; Last accessed 03/06/2024].

Pfalzgraf, B. (2020). *How to use selenium to web-scrape with example*. https://towardsdatascience.com/how-to-use-selenium-to-web-scrape-with-example-80f9b23a843a. Mar[Online; Last accessed 04/08/2024].

R.A. Chaudhary. An introduction to data encoding and decoding in data science. https://www.sitepoint.com/data-encoding-decoding-data-scienceintroduction/, Jul 2023. [Online; Last accessed 04/08/2024].

S. Kumar. What is label encoding? Application of label encoder in machine learning and deep learning models. https://www.analyticsvidhya.com/blog/2020/03/one-hot-encoding-vs-label-encoding-using-scikit-learn/, Jan 2024. [Online; Last accessed 04/08/2024].

What is normalization in machine learning? a comprehensive guide to data rescaling. https://www.datacamp.com/tutorial/normalization-in-machine-learning, Jan 2024. [Online; Last accessed 04/08/2024].

Gupta, S. (2022). *How to apply min-max normalization to your data*. https://sourabharsh.medium.com/how-to-apply-min-max-normalization-to-your-data-f976d1633d2b. JanOnline; Last accessed 04/08/2024].

Sharma, J. (2023). *Multilayer perceptron (mlp)*. https://medium.com/jkumarsharma998/features-and-target-variable-e8bf6a716c5e. Nov[Online; Last accessed 12/08/2024].

Smote implementation in pyspark. https://medium.com/hwangdb/smote-implementation-in-pyspark-76ec4ffa2f1d, Aug 2020. [Online; Last accessed 22/07/2024].

Smote. https://www.cs.cmu.edu/afs/cs/project/jair/pub/volume16/chawla02a-html/node6.html, Feb 2002. [Online; Last accessed 22/07/2024].

Chang, M. (2024). *An intuitive view on mutual information*. https://towardsdatascience.com/an-intuitive-view-on-mutual-information-db0655535f84. Mar[Online; Last accessed 04/08/2024].

Mutual information score. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.mutual_info_score.html, 2024. [Online; Last accessed 04/08/2024].

Basic understanding of neural network structure. https://medium.com/sarita_68521/basic-understanding-of-neural-network-structure-eecc8f149a23, Oct 2023. [Online; Last access 23/06/2024].

Prakash, V. (2023). *Liquid neural network: A adaptive way to train ml model*. https://ved933409.medium.com/liquid-neural-network-a-adaptive-way-to-train-ml-model-b6922d755066. AugOnline; Last accessed 21/07/2024].

Neural ode (part-1). https://medium.com/theneuralduck/neural-ode-part-1.f9a9f0e49385, Jun 2023. [Online; Last accessed 21/07/2024].

scipy.integrer.ode. https://docs.scipy.org/doc/scipy/reference/generated/scipy.integrate.ode.html, Jun 2023. [Online; Last accessed 21/07/2024].

Boesch, G. (2024b). *What are liquid neural networks?*. https://viso.ai/deep-learning/what-are-liquid-neural-networks/. May[Online; Last accessed 21/07/2024].

What is cross-entropy loss function? https://365datascience.com/tutorials/machine-learning-tutorials/cross-entropy-loss/#1, Jun 2023. [Online; Last accessed 23/06/2024].

Koech, K. E. (2020). *Cross-entropy loss function*. https://towardsdatascience.com/cross-entropy-loss-function-f38c4ec8643e. Oct[Online; Last accessed 23/06/2024].

Demystifying the adam optimizer in machine learning. https://medium.com/weidagang/demystifying-the-adam-optimizer-in-machine-learning-4401d162cb9e, Jan 2024. [Online; Last accessed 23/06/2024].

What is adam optimizer? https://www.geeksforgeeks.org/adam-optimizer/, Mar 2024. [Online; Last accessed 23/06/2024].

Activation functions: Relu vs. leaky relu. https://medium.com/sreeku.ralla/activation-functions-relu-vs-leaky-relu-b8272dc0b1be, Mar 2023. [Online; Last access 23/06/2024].

Pytorch - leakyrelu. https://pytorch.org/docs/stable/generated/torch.nn.LeakyReLU.html, 2024. [Online; Last accessed 23/06/2024].

What is leaky relu? https://www.educative.io/answers/what-is-leaky-relu, 2024. [Online; Last accessed 23/06/2024].

O.T. Chidalu. What is the tanh activation function? https://www.educative.io/answers/what-is-the-tanh-activation-function, 2023. [Online; Last accessed 25/07/2024].

Tanh activation. https://paperswithcode.com/method/tanh-activation#:~:Text=Tanh%20Activation%20is%20an%20activation,for%20multi%2Dlayer%20neural%20networks., 2024. [Online; Last accessed 25/07/2024].

Understanding learning rate. https://towardsdatascience.com/https-medium-com-dashingaditya-rakhecha-understanding-learning-rate-dd5da26bb6de, Jun 2019. [Online; Last accessed 23/06/2024].

J. Brownlee. Understand the impact of learning rate on neural network performance. https://machinelearningmastery.com/understand-the-dynamics-of-learning-rate-on-deep-learning-neural-networks/, 2020. [Online; Last accessed 29/07/2024].
Epoch: An essential notion in real-time programming. https://datascientest.com/en/epoch-an-essential-notion, 2023. [Online; Last accessed 29/07/2024].
Epoch in machine learning. https://www.geeksforgeeks.org/epoch-in-machine-learning/, 2024. [Online; Last accessed 29/07/2024].
Epochs, batch size, iterations - how are they important to training ai and deep learning models. https://www.sabrepc.com/blog/Deep-Learning-and-AI/Epochs-Batch-Size-Iterations, 2023. [Online; Last accessed 29/07/2024].
Sivakumar, D. (2023). *Hyperparameters tuning in neural networks*. https://www.scaler.com/topics/deep-learning/neural-network-hyperparameters-tuning/. OnlineLast accessed 29/07/2024.
Setup neural networks hyperparameters for best results. https://www.linkedin.com/pulse/setup-neural-networks-hyperparameters-best-results-takbiri, 2022. [Online; Last accessed 29/07/2024].
A. Jain. All about gradient descent, vanishing gradient descent and exploding gradient descent. https://medium.com/abhishekjainindore24/all-about-gradient-descent-vanishing-gradient-descent-and-exploding-gradient-descent4bd112c9a4e4, 2024. [Online; Last accessed 29/07/2024].
Klu. F-score: What are accuracy, precision, recall, and f1 score? https://klu.ai/glossary/accuracy-precision-recall-f1, 2024.
N.B. Harikrishnan. Confusion matrix, accuracy, precision, recall, f1 score. https://medium.com/analytics-vidhya/confusion-matrix-accuracyprecision-recall-f1-score-ade299cf63cd, Dec 2019.
How to interpret a confusion matrix for a machine learning model. https://www.evidentlyai.com/classification-metrics/confusion-matrix, 2024. [Online; Last accessed 29/07/2024].
Understanding the confusion matrix in machine learning. https://www.geeksforgeeks.org/confusion-matrix-machine-learning/, 2024. [Online; Last accessed 29/07/2024].
K. Muralidhar. Learning curve to identify overfitting and underfitting in machine learning. https://towardsdatascience.com/learning-curve-to-identify-overfitting-underfitting-problems-133177f38df5, Feb 2021. [Online; Last accessed 29/07/2024].
What is exploratory data analysis (eda)? https://www.ibm.com/topics/exploratory-data-analysis, 2023. [Online; Last accessed 13/08/2024].
Schultz, M., Reitmann, S., & Alam, S. (Oct 2021). Predictive classification and understanding of weather impact on airport performance through machine learning. *Transportation Research Part C: Emerging Technologies, 131*.
Bengio, Y. (2012). Practical recommendations for gradient-based training of deep architectures. *Neural Networks: Tricks of the Trade* (2nd Edition, pp. 437–478). Berlin Heidelberg: Springer. pages.
Bisandu, D. B., & Moulitsas, I. (2024). Prediction of flight delay using deep operator network with gradient-mayfly optimisation algorithm. *Expert Systems with Applications, 247*, Article 123306. https://doi.org/10.1016/j.eswa.2024.123306
Bisandu, D. B., Salih Homaid, M., Moulitsas, I., & Filippone, S. (2021). A deep feedforward neural network and shallow architectures effectiveness comparison: Flight delays classification perspective. In *Proceedings of the 5th International Conference on Advances in Artificial Intelligence* (pp. 1–10). https://doi.org/10.1145/3505711.3505712
Bisandu, D. B., Moulitsas, I., & Filippone, S. (2022). Social ski driver conditional autoregressive-based deep learning classifier for flight delay prediction. *Neural Computing and Applications, 34*(11), 8777–8802. https://doi.org/10.1007/s00521-022-06898-y
Bisandu, D. B., & Moulitsas, I. (2023a). A hybrid ensemble machine learning approach for arrival flight delay classification prediction using voting aggregation technique. *AIAA Aviation 2023 Forum* (p. 4326). https://doi.org/10.2514/6.2023-4326
Bisandu, D. B., Soviani-Sitoiu, D. A., & Moulitsas, I. (2024). An enhanced deep autoencoder for flight delay prediction. *Journal of Aviation/Aerospace Education and Research, 33*(4), 1–15. https://doi.org/10.58940/2329-258X.2056
Bisandu, D. B., & Moulitsas, I. (2023b). A deep BiLSTM machine learning method for flight delay prediction classification. *Journal of Aviation/Aerospace Education & Research (JAEER), 32*(2). https://doi.org/10.58940/2329-258X.1992

https://dspace.org/

Deep liquid neural network for prediction of weather-impacted flight delay

Bisandu, Desmond B.; Chandrakumar, Majuran; Moulitsas, Irene

2026-05
Elsevier
https://doi.org/10.1016/j.iswa.2026.200656
Bisandu DB, Chandrakumar M, Moulitsas I. (2026) Deep liquid neural network for prediction of weather-impacted flight delay. Intelligent Systems with Applications, Volume 30, May 2026, Article number 200656

Downloaded from a DSpace repository. https://dspace.org/