Fall 2022

Deep Learning Prediction Models for Runway Configuration Selection and Taxi Times Based on Surface Weather

Shlok Misra
Embry-Riddle Aeronautical University

Deep Learning Prediction Models for Runway Configuration Selection and Taxi Times Based on Surface Weather

Shlok Misra

Thesis Submitted to the College of Aviation in Partial Fulfillment of the Requirements for the Degree of Master of Science in Aeronautics

Embry-Riddle Aeronautical University

Daytona Beach, Florida

October 2022

Deep Learning Prediction Models for Runway Configuration Selection and Taxi Times Based on Surface Weather

By

Shlok Misra

This thesis was prepared under the direction of the candidate's Thesis Committee Chair, Dahai Liu, Ph.D. , and has been approved by the members of the thesis committee. It was submitted to the College of Aviation and was accepted in partial fulfillment of the requirements for the Degree of Master of Science in Aeronautics.

Dahai Liu
Digitally signed by Dahai Liu
Date: 2022.11.21 14:38:33 -05'00'

Dahai Liu, Ph.D.
Committee Chair

Hong Liu
Digitally signed by Hong Liu
Date: 2022.11.21 16:10:38 -05'00'

Hong Liu, Ph.D.
Committee Member

Ahmed Abdelghany, Ph.D.
Digitally signed by Ahmed Abdelghany, Ph.D.
Date: 2022.11.21 16:16:33 -05'00'

Ahmed Abdelghany, Ph.D.
Committee Member

Donald S. Metscher
Digitally signed by Donald S. Metscher
Date: 2022.11.28 08:49:18 -05'00'

Donald S. Metscher, D.B.A.
Master of Science in Aeronautics
Program Coordinator

Steven Hampton
Digitally signed by Steven Hampton
Date: 2022.11.29 14:59:04 -05'00'

Steven Hampton, Ed.D.
Associate Dean, School of Graduate Studies, College of Aviation

Alan J. Stolzer
Digitally signed by Alan J. Stolzer
Date: 2022.11.29 15:32:29 -05'00'

Alan J. Stolzer, Ph.D.
Dean, College of Aviation

Christopher Grant
Digitally signed by Christopher Grant
Date: 2022.12.08 13:45:09 -05'00'

Christopher D. Grant, Ph.D.
Associate Provost of Academic Support

November 21, 2022

Signature Page Date

Abstract

Researcher: Shlok Misra

Title: Deep Learning Prediction Models for Runway Configuration Selection and Taxi Out Times Based on Surface Weather

Institution: Embry-Riddle Aeronautical University

Degree: Master of Science in Aeronautics

Year: 2022

Growth in air traffic demand in the United States has led to an increase in ground delays at major airports in the nation. Ground delays, including taxi time delays, directly impacts the block time and block fuel for flights which affects the airlines operationally and financially. Additionally, runway configuration selection at an airport significantly impacts the airport capacity, throughput, and delays as it is vital in directing the flow of air traffic in and out of an airport. Runway configuration selection is based on interrelated factors, including weather variables such as wind and visibility, airport facilities such as instrument approach procedures for runways, noise abatement procedures, arrival and departure demand, and coordination of ATC with neighboring airport facilities. The research problem of this study investigated whether runway configuration selection and taxi out times at airports can be predicted with hourly surface weather observations. This study utilized two sequence-to-sequence Deep Learning architectures, LSTM encoder-decoder and Transformer, to predict taxi out times and runway configuration selection for airports in MCO and JFK. An input sequence of 12 hours was used, which included surface weather data and hourly departures and arrivals. The output sequence was set to 6 hours, consisting of taxi out times for the regression models and runway configuration

selection for the classification models. For the taxi out times models, the LSTM encoder-decoder model performed better than the Transformer model with the best MSE for output Sequence 2 of 41.26 for MCO and 45.82 for JFK. The SHAP analysis demonstrated that the Departure and Arrival variables had the most significant contribution to the predictions of the model.

For the runway configuration prediction tasks, the LSTM encoder-decoder model performed better than the Transformer model for the binary classification task at MCO. The LSTM encoder-decoder and Transformer models demonstrated comparable performance for the multiclass classification task at JFK. Out of the six output sequences, Sequence 3 demonstrated the best performance with an accuracy of 80.24 and precision of 0.70 for MCO and an accuracy of 77.26 and precision of 0.76 for JFK. The SHAP analysis demonstrated that the Departure, Dew Point, and Wind Direction variables had the most significant contribution to the predictions of the model.

Keywords: taxi times, runway configuration, deep learning, sequence-to-sequence models, long short-term memory, transformer, aviation

Dedication

This thesis is dedicated to my family, friends, and professors who made my journey in graduate school one to cherish and remember for the rest of my life.

Acknowledgments

I would like to take this opportunity to recognize and express my gratitude to the people who assisted me in the successful completion of my thesis. Firstly, I would like to recognize my committee chair, Dr. Dahai Liu, for his guidance and leadership in the study. Additionally, I would like to acknowledge my committee members, Dr. Hong Liu, and Dr. Ahmed Abdelghany, for their support and guidance through the process.

Additionally, I would like to recognize the support and guidance from Dr. Donald Metscher and Ms. BeeBee Leong throughout my graduate degree. I also appreciate and am grateful for the assistance from the Graduate Teaching Assistant, Ms. Kayla Taylor, for her insightful comments and editorial contribution to this thesis.

Additionally, I would like to recognize my professors, colleagues, and friends in the Aeronautical Science Department, where I had the privilege of being the Graduate Teaching Assistant. Their encouragement and support helped me excel in my thesis and grow as a budding professional.

Finally, I would like to recognize my father, mother, and sister. Although we have been geographically distant for the past 6 years, they have always supported me at all times and never let the distance come between us. I would also specially recognize my friend Tanish Jain for his help in the coding and programming needs of this thesis.

Table of Contents

Page

List of Tables

List of Figures

Chapter I: Introduction

Growth in air traffic demand in the United States has led to increased air traffic and system delays in the National Airspace System (NAS; Federal Aviation Administration [FAA], 2018). The number of flights and passengers at 30 core airports in the United States for 2019 increased by 1.8% and 3.2%, respectively, compared to the yearly average of flight operations from 2015–2018 (FAA, 2020). With an increase in yearly flight operations and passengers traveled, flight delays for 2019 increased by 15% compared to the yearly average for 2015–2018. Weather-related events were the most significant cause of flight delays and accounted for 69.8% of delays in 2019. Flight delays lead to increased costs for airlines due to direct costs, such as passenger compensation, and indirect costs, such as passenger satisfaction and reputation loss (Gu et al., 2013). The cost of flight delays rose by 9.3% in 2019 ($8.3 billion) as compared to the yearly average for 2012–2018 due to an increase in expenses for fuel and crew compensation (FAA, 2020).

The increase in flight delays and delay costs significantly affects the operations of an airline. Airlines operate with constrained resources and schedule their flights in terms of fixed block times (Sohoni et al., 2017). Any disruption to block times can affect the overall operations of an airline. Additionally, block time calculations are used to calculate block fuel for each flight, which is the total fuel loaded into an aircraft before each flight (Skybrary, n.d.). Block time and block fuel for a flight are affected by delays occurring on the ground and in flight (Cirium, 2015; Sohoni et al., 2017). Delays in taxi out at the origin airport and taxi in at the destination airport directly affect the block fuel for a flight (Ramanjun & Balakrishnan, 2015). Moreover, taxi out and taxi in times are affected by

the runway configuration selected for take-off and landing operations at an airport (Diana, 2018). Additionally, delays during taxi operations at the origin and destination are not the only cause of flight delays for flights, taxi delays significantly affect fuel burn and emissions at major airports (Simaiakis & Balakrishnan, 2010).

Airlines invest considerable resources in forecasting such delays, which can aid in resource planning and allocation (Fan, 2019). Accurate delay forecasting based on available data has been demonstrated to be an effective delay mitigation tool for airlines. The research problem of this study investigated whether runway configuration selection and taxi out times at airports can be predicted utilizing hourly surface weather observations. This study utilized two Deep Learning architectures to predict taxi out times and runway configuration selection for two major airports in the United States.

Statement of the Problem

Ground delays affect the operating network of an airline and the operations of the entire NAS (FAA, 2018). Due to constrained ground resources, airlines invest considerable resources in optimizing ground operations at major airports (Kang & Hansen, 2018). The runway configuration at an airport is selected by the governing Air Traffic Control (ATC) facility. It can have a significant impact on the ground operations of an airline (Ramanjun & Balakrishnan, 2015). The runway configuration selection impacts the taxi out and taxi-in times at an airport, and the runway capacity, utilization, and throughput directly impact the overall capacity of an airport (Ramanjun & Balakrishnan, 2015). Although the direction of wind flow at an airport is considered a major factor in determining the runway configuration selection at an airport, interrelated factors such as predicted arrival and departure demand, noise abatement procedures, and

coordination with nearby airport ATC facilities influence the runway configuration selection as well (Ramanjun & Balakrishnan, 2015; Wang & Zhang, 2021).

There is a need for forecasting techniques that can model the relationship of these factors to predict runway configuration selection and taxi out times at airports. Although there is literature available on the viability of utilizing Artificial Feedforward Neural Networks for runway configuration prediction, there is a lack of literature on developing a robust Machine Learning algorithm for an airport with different runway configurations. Additionally, literature on predicting taxi times has not focused on the dependency of taxi times on time-related factors. Previous studies have only aimed to use weather factors to predict runway configuration selection and taxi times at a single point in time rather than for several periods or sequences. Consequentially, there is a lack of research on using sequence-to-sequence time series Deep Learning models to predict a sequence of runway configuration selection and taxi out times based on an input sequence of weather and operations-related input variables.

Purpose Statement

The purpose of this study was to develop two Deep Learning architectures to predict runway configuration selection and taxi out times at two major airports in the United States based on hourly surface weather observations. The models were developed for MCO International Airport (MCO) and JFK- John F. Kennedy International Airport (JFK). The study demonstrated the utilization of sequence-to-sequence time series models for predicting runway configuration selection and taxi out times for several periods (hours). Sequence-to-sequence models such as Recurrent Neural Network (RNN) encoder-decoder models, specifically Long Short Term Memory (LSTM) and Gated

Recurrent Units (GRUs), and Transformer models, were developed in the study. The models were used for a multi-class classification task (runway configuration selection) and a regression task (taxi out times).

The utilization of regularization and feature importance techniques determined the most significant variables affecting the runway configuration selection and taxi out times at an airport. Model performance for the regression task was evaluated on Mean Squared Error (MSE), Mean Absolute Squared Error (MASE), Mean Absolute Error (MAE), and R-Squared. The model performance for the classification task was evaluated on the accuracy, precision, recall, and Cohen's kappa scores.

Significance of the Study

Taxi times and runway configuration selection can directly affect ground delays at an airport (Wang & Zhang, 2021). Although the direction of wind flow at an airport is considered a major factor in determining the runway configuration selection at an airport, interrelated factors such as predicted arrival and departure demand, noise abatement procedures, and coordination with nearby airport ATC facilities influence the runway configuration selection as well (Ahmed et al., 2018). The complexity of interrelated factors that influence the runway selection configuration and the dynamic nature of weather-related factors make runway configuration challenging for airlines operating at major airports.

Time-series Deep Learning models, specifically sequence-to-sequence models, can be used to model the inter-relation of such variables and their temporal dependencies through time and create a prediction model. A runway configuration selection and taxi out times forecasting model, such as the models developed in this study, can aid an

airline and airport management in predicting taxi out and runway configurations at airports and determining the most significant weather-related predictors (Carvalho et al., 2020).

The runway configuration selection and taxi out times prediction models will allow airline management to make informed short-term operations decisions such as block fuel and contingency fuel planning along with resource and gate allocations. The theoretical significance of this study is the use of sequence-to-sequence for multi-class classification and regression. The application of sequence-to-sequence time series models in different domains is a subject of research for scientists. The effective utilization of the models in this study will be a significant contribution to the literature on RNN encoder-decoder models and Transformer models.

Research Questions

The study investigated the following research questions:

1. Can runway configuration selection be predicted based on hourly surface weather observations utilizing neural networks?
2. Can taxi out times be predicted based on hourly surface weather observations utilizing neural networks?
3. What variables are most significant in predicting runway configuration selection at the selected airports?
4. What variables are most significant in predicting taxi out times at the selected airports?

5. How does a Transformer model compare to an LSTM encoder-decoder model in sequence-to-sequence predictive performance for runway configuration selection and taxi out times at the selected airports?
6. What are the most effective model hyperparameters for sequence-to-sequence modeling?

Delimitations

The study was limited to data for two airports — MCO and JFK — to avoid the risk of developing underfitting models. The models developed in this study have low generalization power for predicting taxi out times and runway configuration selection at airports other than the two selected airports. The two airports selected have different runway layouts, with MCO having four parallel runways and JFK having four runways, which are parallel and intersecting (Horenjeff et al., 2010). Additionally, the Deep Learning models that developed for the study were limited to variants of RNNs such as LSTM and GRUs and Transformer models to preserve the temporal dependency of the data.

Limitations and Assumptions

The models developed in this study were based on the operations and weather at only MCO and JFK. Additionally, the models were developed utilizing RNNs and Transformers, which are based on certain assumptions. A central assumption of RNNs is that current data or information is dependent on previous time lags of data or information in the time series (Goodfellow et al., 2016). Multivariate autoregressive models such as Vector Autoregression (VAR) were not utilized based on the literature review on the

subject and the limitation of moving average and autoregressive models (Kirchgässner & Wolters, 2008).

The independent variables or predictors utilized to develop the prediction models were limited to the data variables available through the FAA Aviation System Performance Metrics (ASPM) and National Oceanic and Atmospheric Administration (NOAA) databases (FAA, n.d.; NOAA, n.d.). Significant predictors of runway configuration selection and taxi out times that might not be available through the FAA ASPM and NOAA databases were not used for model development. Finally, the study did not utilize any feature engineering technique for feature selection, as all the features selected for the model development were based on the availability of data and literature reviewed.

Summary

An increase in flight delays has a direct operational and financial impact on airlines and passengers. With an increasing number of flight operations and passengers traveling at major airports, there is an increase in flight delays and delay costs for airlines and passengers. Weather and traffic volumes continue to be the most significant causes of flight delays. Ground delays during the taxi phase significantly contribute to overall flight delays and fuel burn. Air traffic volumes, airport capacity, weather events, and runway configuration can impact ground delays at major airports. With the impact of weather on flight delays, airlines have explored delay forecasting techniques as possible mitigation rules.

Delay forecasting techniques, including Machine Learning models, have demonstrated proficiency in forecasting delays based on identified predictors. This study

used time series sequence-to-sequence Deep Learning models to predict runway configuration and taxi out times for two airports in the United States. A review of model architecture for similar use cases was conducted to develop a baseline model and choose effective hyperparameters and optimization methods.

Definitions of Terms

Attention Mechanism	Attention mechanism is used by Machine Learning models while processing sequential data where weights are assigned to the input sequence to decide which input steps are essential and should be retained for memory (Geron, 2019).
Convolutional Neural Network	Form of Neural Networks that utilizes convolution to concentrate on feature extraction from extensive multi-dimensional data (such as an image) to develop feature maps or kernels (Goodfellow et al., 2016).
Deep Learning	Form of Representation Learning that enhances Hierarchical Feature Learning where the models extract features and create representations in multiple levels of the training data with the use of high-level

	features that are defined in terms of lower-level features (Bengio, 2012).
Machine Learning	Computer programming that learns patterns from large data which can be used to create prediction models (Lee, 2019).
Recurrent Neural Network	Form of Neural Networks which utilize Recurrent Cells where an output connection from a cell feeds back into the cell as recurring input along with the next input to allow the processing of sequential data (Goodfellow et al., 2016).
Runway Configuration	Number and relative orientations of one or more runways on an airfield (Horenjeff et al., 2010).
Runway Configuration Selection	The runway(s) being used in an airport during a particular period (Wang & Zhang, 2021).
Shapley Value Imputation	A feature assessment technique which computes Shapley Values utilizing coalitional game theory by treating each feature as a player in the game (Molnar, 2021).

Transformer Model	The model architecture used for sequential data that uses modules such as Masked Multi-Head Attention and Feed Forward Layers that are stacked upon each other in the encoder and decoder section of the model along with Input Embedding for positional encoding of sequential data (Vaswani et al., 2017).

List of Acronyms

ASDE-X	Airport Surface Detection, Type X
ATC	Air Traffic Control
ASPM	Aviation System Performance Metrics
BPTT	Backpropagation Through Time
CNN	Convolutional Neural Network
ELU	Exponential Linear Unit
EWR	New York Liberty International Airport
FAA	Federal Aviation Administration
Google Colab	Google Colaboratory
GRU	Gated Recurrent Unit
IFA-SVR	Improved Firefly Algorithm-Support Vector Regressor
JFK	New York John F. Kennedy International Airport
LGA	New York LaGuardia International Airport
LSTM	Long Short Term Memory

MAE	Mean Absolute Error
MCO	Orlando International Airport
MSE	Mean Squared Error
NAS	National Airspace System
NOAA	National Oceanic and Atmospheric Administration
PSO-SVR	Particle Swan Optimization-Support Vector Regressor
RMSE	Root Mean Squared Error
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Network
SHAP	Shapley Value Imputation
XGBoost	Extra Gradient Boosting

Chapter II: Review of the Relevant Literature

The literature review was conducted to identify research on the impact of weather on ground delays and taxi times at airports. Additionally, literature on the effect of runway configurations on ground operations at major airports was reviewed to identify the problem and significance of the study. Finally, previous studies on the use of data-driven modeling techniques, including Machine Learning and Deep Learning techniques, to predict taxi times and runway configurations were reviewed to select significant independent variables to predict taxi time and runway configurations at major airports. The purpose of such a literature review was to study previous work, analyze research gaps, and build a foundation upon which the models were developed for this study.

Runway Configuration

Runway configuration is defined as the "number and relative orientations of one or more runways on an airfield" (Horenjeff et al., 2010, p. 177). The runway configuration is a critical design consideration for the development of an airport. Civil aviation authorities around the world publish advisories and guidelines on preferred runway configurations for airports based on factors including wind patterns, traffic volumes, noise abatement, and geographical location (FAA, 2014). There are five primary runway configurations: single runway, parallel runways, intersecting runways, Open-V runways, and hybrid runways (Horenjeff et al., 2010).

Runway Configuration Selection

Runway configuration selection refers to the runway(s) used in an airport during a particular period (Wang & Zhang, 2021). While an airport might contain multiple runways that can are used for take-off and landing operations, not all runways are used at

all points of time. Runway configuration selection is determined by weather, air traffic demand, noise abatement, coordination with neighboring ATC facilities, and the operating plan of the ATC facility at the airport (Ramanjun & Balakrishnan, 2015; Wang & Zhang, 2021).

Runway configuration selection has a direct impact on the ground operations of an airport as it affects the taxi times and determines the ground delays at an airport (Wang & Zhang, 2021). There is significant literature on different aspects of runway configurations, including research on optimizing and predicting runway configuration selection for an airport based on certain independent variables. A review of the literature on the topic provides insights into the theoretical foundations of runway configuration selection and the factors that influence the model development process.

Runway Configuration Selection Optimization

Literature on runway configuration selection optimization has focused on optimizing runway configuration selection for an airport as a queueing system problem. Generally, queueing system problems are mathematical problems that are used to explain congestion due to service demand, where the demand and service times are assumed random (Bertsimas et al., 2011; Jacquillat et al., 2016; Li & Clarke, 2010).

Stochastic Dynamic Programming. Li and Clarke (2010) evaluated the effect of runway configuration selection on the efficiency of an airport based on delays, fuel burn, and emissions. Li and Clarke utilized the principles of stochastic dynamic programming to develop a decision model for runway configuration selection. Utilizing stochastic wind information, runway configuration capacity curves, traffic demand for an airport, and

penalty terms for runway configuration switches, Li and Clarke developed a model to maximize the weighted arrival and demand capacity of an airport in a given time horizon.

The model proposed by the authors included Pareto-arrival-departure rate trade-off and configuration schedule optimization. Pareto-arrival-departure is a capacity curve for an airport for each runway configuration. Li and Clarke (2010) utilized the Pareto-arrival-departure rate trade-off in the model to maximize the weighted capacity of the airport while minimizing the number of unserved landings and take-offs. The configuration schedule optimizer utilized reward coefficients calculated from the solution of the Pareto-departure-arrival rate trade-off to optimize the runway configuration schedule. Li and Clarke developed the model for JFK and utilized weather and traffic data for the model development. Li and Clarke evaluated the model based on a decision simulation. They evaluated that the optimal decisions based on the model could reduce delays by as much as 80% depending on the weather conditions and the flight operations schedule when compared to historical runway configurations for the same conditions.

Utilizing a similar approach utilizing dynamic programming, Jacquillat et al. (2016) developed a model to optimize runway configurations and airport service rates. The model aimed to minimize the congestion costs within a stochastic queuing and operating system. The authors identified various endogenous and exogenous variables, such as weather variables and traffic volumes, which exhibited stochastic properties which were not known with certainty in advance. Some state variables used for the optimization problem included arrival queue length, departure queue length, runway configuration used in the preceding time horizon, and weather conditions. The decision variables for the optimization problem were the runway configuration selection for the

next time horizon and the rates of arrivals and departures that should have been served. Once the state variables and decision variables were specified, a Dynamic Programming model was developed and optimized.

Jacquillat et al. (2016) developed their model based on the operations and weather data for JFK. To improve model performance, the authors included an approximate one-step-look-ahead algorithm. The model indicated that optimal solutions to the decision variables were path-dependent and "on the stochastic evolution of arrival and departure queues during the day" (Jacquillat et al., 2016, p. 1). Jacquillat et al. evaluated that the deployment of the model could potentially reduce congestion costs by 20%–30% by optimizing runway configurations and arrival/departure rates at JFK. The model highlighted the stochasticity of endogenous and exogenous variables, such as arrival/departure flow and weather variables that influence congestion costs and the need to integrate operating stochasticity in optimization models.

Mixed Integer Programming. Bertsimas et al. (2011) developed a Mixed Integer Programming model to select an optimal runway configuration for an airport based on various independent conditions. The model was further extended to determine the optimal number of departures and arrivals served by an airport at different time horizons. Bertsimas et al. found that an optimal runway configuration selection is critical to reducing both in-flight and on-ground delays and their associated costs. Bertsimas et al. developed the Mixed Integer Programming model for the airports in the New York Metropolitan area to capture the relationship and interdependency of operations between the airports in the region. A novel contribution of the study was the development and application of a Mixed Integer Programming model for optimizing runway configuration.

Compared to traditional heuristic approaches to optimizing runway configuration selections, Bertsinas et al. assessed that the Mixed Integer Programming model could potentially reduce costs by up to 10% by optimizing runway configurations and arrival/departure demand.

Runway Configuration Selection Prediction

Data-driven techniques can be used to model the relationship between factors such as weather and runway configuration selection (Avery & Balakrishnan, 2016; Ramanjun & Balakrishnan, 2015; Wang & Zhang, 2021). Previous literature has focused on using different data-driven models, such as machine learning and discrete choice models.

Artificial Neural Network. Ahmed et al. (2018) utilized a Multi-Layered Artificial Neural Network approach to predict the runway configuration and the corresponding runway movements at Amsterdam Schiphol International Airport. The authors developed a Feedforward Neural Network and a Recurrent Backpropagation Neural Network. Ahmed et al. utilized data for two days that included 1,789 arrivals and departures at the selected airport. Additionally, data from hourly surface weather observations, including wind direction, wind speed, visibility, cloud ceiling, air temperature, dew point, and surface pressure, were used for the model development.

The authors only evaluated the models on MSE for the runway movements prediction model and not explicitly the runway configuration. For the Feedforward Neural Network, 13 neurons were used for the input layer, followed by 10 neurons in the single hidden layer. The sigmoid-tangent activation function was used for the output layer. The validation MSE for the Feedforward Neural Network was 0.00018. The

authors did not mention the hyperparameters used for the Recurrent Backpropagation Neural Network. However, the MSE was evaluated to be 0.00024, which was higher as compared to the Feedforward Neural Network. The authors did not conduct a feature or variable importance assessment to evaluate the most significant independent variables.

Convolutional Neural Network (CNN). Wang and Zhang (2021) utilized an assembled gridded weather forecast to predict the runway configuration selection at three major airports in the New York City area — JFK, New York LaGuardia International Airport (LGA), and Newark Liberty International Airport (EWR). Using a gridded weather forecast allowed the model to capture the interdependency of operational parameters of airports in the same geographical area rather than create isolated predictive models for each airport. Rather than using surface weather observations for airports, the authors utilized Rapid Refresh (RAP) data, which was a numerical weather model maintained by the National Center for Environmental Protection in the United States. The RAP weather predictions were generated for a 13 km horizontal grid rather than a geographical point. The authors utilized RAP data available for areas within 200 Nautical Miles (370.4 km) of the JFK City area. A significant contribution of the study was the development and utilization of a CNN model for the prediction task.

For the model development, Wang and Zhang (2021) used a CNN model due to its ability to extract features and process high-dimensional data. The authors used 63 independent variables for the model training, which included 23 surface weather variables. The CNN model architecture implemented sets of two 2-dimensional Convolutional layers followed by a Batch Normalization layer, Dropout layer, and Maximum Pooling layer. The Batch Normalization and Dropout layers were treated as

regularization features for the model. The final CNN architecture was built with six 2-dimensional convolutional layers, three Batch Normalization layers, three Dropout layers, and three Max Pooling layers. The final layer was a Dense layer with a Softmax activation function for the multi-label classification task. Each of the Convolutional layers utilized a Rectified Linear Unit (ReLU) activation function. The model predicted the runway configuration with an accuracy of 79.21%, 85.86%, and 87.25% for JFK, LGA, and EWR respectively. A limitation of the study identified by the authors was the lack of flight operations data in the model development, such as data on scheduled arrivals and departures at the airports.

Time Series Modeling. Rebollo et al. (2021) utilized a recursive multi-step (time-series) Machine Learning approach to predict runway configurations at the selected airport. Rebollo et al. utilized time of the day, surface weather data, future arrival and departure counts, and runway configuration at the previous time step as the independent variables for the prediction model. Surface weather data used for the model development included wind direction and speed, cloud ceiling, visibility, temperature, precipitation, and lightning probability. Rebollo et al. used 30-minute time steps and trained the model for 3-hour outlook and 6-hour outlook where the model could predict the runway configuration 3 hr and 6 hr in advance. A novel contribution of the study was the model development and evaluation strategy for the time series models.

For the model development, Rebollo et al. (2021) used a Random Forest classifier and an Extra Gradient Boost (XGBoost) classifier. The models were evaluated primarily on prediction the accuracy where the XGBoost classifier outperformed the Random Forest classifier on the 3-hour outlook and 6-hour outlook predictions. The models were

developed and tested for six airports, including Charlotte Douglas International Airport, Dallas Fort Worth International Airport, EWR, Dallas Love Field Airport, and LGA. The model exhibited the most robust prediction performance for Dallas Fort Worth International Airport with an accuracy of 89.3% for the 3-hour prediction and 82.8% for the 6-hour prediction.

Autoencoders. Dalmau and Herrema (2019) utilized a type of Machine Learning model called *Autoencoders* to develop a runway configuration prediction model. Utilizing two encoders and a decoder to develop the prediction model architecture, Dalmau and Herrema aimed to predict the runway configuration at Amsterdam Schiphol International Airport, Netherlands. The authors utilized a sequence-to-sequence Autoencoder model where each output at a time step was a class label prediction representing the predicted runway configuration. The independent variables utilized to develop the prediction model included surface weather data, departure and arrival demand, time of the day, day of the week, and runway configuration data for the previous time step. The input data was utilized in 15-minute time steps.

For the model development, Dalmau and Herrema (2019) utilized two encoders. The first encoder was used to input weather information and arrival and departure demand data for 6 hr prior to the predicted time step and 6-hour forecasts after the time step. The second encoder was used to input runway configuration predictions from the previous time steps. For both the encoders, bidirectional-RNNs were used, which allowed the encoder to receive information from both the past and future time step predictions. The final dense layer of the encoders was used at the first hidden layers of the decoder, which also utilized an RNN architecture. For the RNNs in the encoders and decoders,

LSTM cells were utilized. The model was developed with 16 LSTM cells in the encoders and 32 LSTM cells in the decoder. The model was trained on a batch size of 64, a learning rate of .001, and early stopping with a patience of five. The model was evaluated on precision, recall, and f1-scores. The model demonstrated precision of .86 for all runway configuration predictions for a 2-hour prediction outlook.

Discrete Choice Models for Runway Configuration Prediction

While Wang and Zhang (2021), Khater et al. (2021), Dalmau and Herrema (2019), and Ahmed et al. (2018) demonstrated the use of Neural Networks for runway configuration selection prediction, Avery and Balakrishnan (2016) analyzed the impact of factors such as wind speed and direction, visibility, air traffic demand, and ATC workload on runway configuration selection. The authors evaluated that runway configuration changes occur less often than what would be optimal or predicted due to such operational inertia. Avery and Balakrishnan explored the utilization of Discrete Choice Models to accommodate the possible operational inertia while making predictions and developing the model variables and weights. Discrete Choice models are a form of behavioral models that are used to explain or predict possible nominal decisions by an individual from a set of alternatives based on a utility function. Avery and Balakrishnan developed the utility function with random variables selection such as operational inertia, wind speed and direction, arrival and departure demand, cloud ceiling and visibility, and noise abatement procedures. The study's novelty was the modeling of human decision-making and operational inertia.

The authors added a positive contribution to the utility function if the prediction runway configuration was used in the previous time interval. Avery and Balakrishnan

(2016) developed the discrete choice model for San Francisco International Airport, LGA, and EWR. The model could predict the runway configurations with an accuracy of 81.2%, 81.3%, and 77.8% for a 3-hour prediction window for San Francisco International Airport, LGA, and EWR respectively. The authors found that the operational inertia variable was the most significant variable in predicting runway configuration, followed by the headwind component of the arrival runway. Additionally, configurations with high departure and arrival capacity were preferred during high-traffic demand periods.

Use of Empirical Observations for Model Development. Ramanjun and Balakrishnan (2015) utilized empirical observations to develop a statistical model to characterize the runway configuration selection by ATC personnel. Empirical observations were recorded and processed for EWR and LGA. The authors utilized the likelihood maximization utility function to estimate the parameters of the model, and the correlations between different choices were modeled using a multinomial nested logit model. Some of the models expected to influence the decision of runway configuration selection and added as parameters of the model were inertia, weather conditions (cloud ceilings and visibility), wind direction and speed, arrival and departure demand, and inter-airport coordination. Inertia was used as a parameter for the statistical Discrete Choice model developed in the study where the "utility function of the incumbent configuration is expected to be higher than those of the other candidate configurations due to the inertia factor" (Ramanjun & Balakrishnan, 2015, p. 4). Additionally, configuration proximity referred to the extent of change of the runway configuration needed and was measured in the difference in angle (measured in degrees) between the succeeding and preceding runway configurations. The model developed for EWR

comprised 57 parameters, and the model for LGA comprised 36 parameters. The runway configurations were predicted for a 3-hour forecast horizon. The models were evaluated on the accuracy, and the parameters were evaluated on the utility coefficients.

The models achieved an accuracy of 82% for EWR and 85% for LGA. The significance of each parameter on the runway configuration prediction was evaluated based on the utility coefficients of each parameter. For LGA and EWR, configuration from the previous time step (inertia) had the highest utility coefficient, followed departure/arrival demand, weather conditions, and configuration proximity.

Taxi Times Prediction

Taxi time prediction has been demonstrated to be an effective tool for developing solutions to mitigate the effects of delays. With increased fleet capacity and flight activities, airlines are increasingly interested in predicting taxi out and taxi in times to optimize ground operations, flight scheduling, and resource management at airports (Lian et al., 2018). Taxi time delays also directly impact the block times for a flight and affect the fuel burn and emissions for aircraft. Diana (2018) explained that taxi times prediction benefits airports, airlines, and regulatory analysts because it allows forecasting and assurance of on-time performance for aircraft operating in the airport. Taxi times prediction significantly affects block fuel and contingency fuel calculations and assists airline management in forecasting congestions at airports. Taxi times are influenced by several factors, including airport layout, ATC workload, runway configuration, weather, and air traffic demand. Different Machine Learning techniques can predict taxi times at airports (Diana, 2018; Lian et al., 2018).

Machine Learning Approach. Diana (2018) developed Machine Learning models to predict taxi out times at Seattle Tacoma International Airport by comparing the performance of different types of Machine Learning models. The author utilized Linear Regression, Penalized or Regularized Regression, Ridge Regression, Support Vector Regression, and Ensemble models for regression, such as Random Forest, Extra Trees, and Bagging. The author did not test the development of time series or Deep Learning models. Additionally, Diana utilized five independent variables for training the models, which were departure demand, departure throughput, percentage of total airport capacity utilized, approach condition, and runway configuration. Diana utilized data available from different datasets such as Aeronautical Radio Inc. (ARINC) Out-Off-On-In times, FAA Traffic Flow Management System, and U.S. Department of Transportation Aviation Service Quality Service. Diana used two samples of data with 1,380 observations in each sample. One sample consisted of data from June 2016 to August 2016, and the other sample consisted of data from June 2015 to August 2015.

The author utilized RMSE for cross-validation and Coefficient of Determination for the model evaluation and hyperparameter tuning. For the Coefficient of Determination evaluation, the Bagging Regression and Random Forest Regression scored 0.95, followed by Linear Regression and Ridge Regression, which scored 0.74 each. For the cross-validation evaluation, Linear Regression and Ridge Regression models had the lowest RMSE of 3.5443 and 3.5444 respectively. Diana evaluated that the Linear Regression and Ridge Regression models performed best when fewer instrument approach procedures were used, and runway configuration prediction was more stable. Support Vector Regression was evaluated to be the worst-performing model in all cases.

Utilizing Airport Surface Detection Equipment, Model X (ASDE-X) data, Lee et al. (2016) developed Machine Learning models to predict taxi out times at Charlotte Douglas International Airport. The ASDE-X data had multiple variables available for each flight, but the authors could not utilize all the available data variables due to data unavailability. Some of the variables utilized for the model development were terminal concourse, runway configuration, departure fix for the Standard Instrument Departure procedure, weight class of the aircraft, scheduled push-back time of the aircraft from the gate, number of departures for the runway, number of arrivals scheduled, the month of the year, and unimpeded taxi time. Lee et al. treated every data point as a separate input and did utilize any time series relationships for the model development. Utilizing the data collected, Lee et al. developed models using Linear Regression, Support Vector Machine, K-Nearest Neighbors, Random Forest, and Neural Network.

The developed models were evaluated based on RMSE and MAE. Lee et al. (2016) trained and tested the models on four different combinations of runway configurations and weather conditions. For four combinations, Linear Regression and Random Forest performed the best regarding RMSE and MAE evaluations, while Support Vector Machine and Neural Network had the highest errors. The Linear Regression model exhibited RMSE and MAE of as low as 4.83 and 3.79 respectively, while the Random Forest model exhibited RMSE and MAE of as low as 4.82 and 3.75 respectively.

Queuing Approach. Lian et al. (2018) explored the use of data-based predictions and queuing-based approaches with regard to causal factors to predict taxi times at airports. Lian et al. aimed to predict the taxi out times for Beijing Peking International

Airport, China, and used the data sourced from the Aviation System Performance database for the airport. Some of the parameters used to train the models for the prediction included the number of aircraft in the departure queue, the number of aircraft scheduled to land, the distance of the taxi route for the aircraft, airport delays in minutes, planned take-off time, and the actual pushback time. For the model development, the authors adopted a prediction method where causal factors were identified to select the input variables and model the dependencies. The authors tested the usability of Generalized Linear Models, Softmax Regression Models, Artificial Neural Networks, and Support Vector Regressions. Lian et al. utilized data for 13 days to train the models and used data for three days to test the model. A novel contribution of the study was the development and testing of two improved versions of the Support Vector Regressors.

Lian et al. (2018) utilized two versions of the Support Vector Regressors called Particle Swan Optimization-Support Vector Regressor (PSO-SVR) and Improved Firefly Algorithm-Support Vector Regressor (IFA-SVR), which demonstrated the highest accuracy rate and were the most effective in capturing non-standard taxi times. The IFA-SVR demonstrated an RMSE of 2.29 and a Mean Absolute Percentage Error (MAPE) of 13.2%. Regarding model performance, IFO-SVR was followed by the PSO-SVR model with an RMSE of 2.59 and MAPE of 13.6%. Lian et al. assessed the departure queue length, number of potential landing aircraft, and distance of taxi route to be the most significant independent variables for models. Additionally, utilizing the taxi delay time of the preceding hour as an input variable improved the prediction performance of the model.

Reinforcement Learning. Balakrishna et al. (2010) utilized another variation of Machine Learning called Reinforcement Learning and treated the taxi out prediction task as a sequential decision-making process and stochastic control problem. For a Reinforcement Learning model, a system state defined by various independent variables is used as an input for a model to train and predict a taxi out time. In the model, a utility function is updated based on the absolute difference between the predicted and actual value. Balakrishna et al. developed the model for Tampa International Airport. To define the system state as an input to the model, several state variables were identified which included number of aircraft in the queue at the departure runway, number of departure aircraft that will be taxiing, and number of arrival aircraft that will be taxiing. Additionally, the average taxi out time in the previous time interval was added as a state variable to introduce some temporal dependency in the system state. The Reinforcement Learning model was evaluated based on the mean error of the predictions.

The authors were able to predict the taxi out time for any given time period with a mean error of less than 1.5 minutes with an accuracy of 93.7%. In terms of predicting taxi out times for individual flights, the authors of the study were able to predict taxi out times with a mean error of less than two minutes with a probability of 81%. Balakrishna et al. evaluated that taxi out times prediction is a dynamically changing problem due to short-term changes in variables such as runway configuration and the number of departing and arriving traffic. For such a dynamically changing departure process, Reinforcement Learning is a more suited process rather than traditional statistical or parametric modeling processes that will not be able to capture short-term trends.

Fuzzy Rule-Based System. Unlike the Machine Learning approaches followed by Lian et al. (2018), Diana (2018), Balakrishna et al. (2010), and Ravizza et al. (2014) explored soft computing methods, particularly the Fuzzy Rule-Based System, to predict taxi out times. The independent variables used for the modeling process included taxi distance, number of departing aircraft, number of arriving aircraft, and number of aircraft in taxiing queue. The data was collected from Stockholm-Arlanda International Airport and Zurich International Airport to develop the Fuzzy Rule-Based Systems.

Ravizza et al. (2014) utilized two Fuzzy Rule-Based Systems named Mamdani Fuzzy Rule-Based System and Takagi and Sugeno Fuzzy Rule-Based System. The model performances were evaluated on RMSE, MAE, Root Relative-Squared Error, and Relative Absolute Error. Ravizza et al. explained that "TSK fuzzy Rule-Based Systems use fuzzy membership functions to subdivide the input space in the premise part and a weighted sum of multiple Linear Regression approaches in the consequent part" (p. 405). For Stockholm Arlanda International Airport and Zurich International Airport, the Takagi and Sugeno Fuzzy Rule-Based System had the lowest RMSE, MAE, Root Relative-Squared Error, and Relative Absolute Error. The Takagi and Sugeno Fuzzy Rule-Based System had an RMSE and MAE of 1.44 and 1.06 for Stockholm Arlanda International Airport and 1.30 and 0.96 for Zurich International Airport. The Takagi and Sugeno Fuzzy Rule-Based System was evaluated to be the best-performing model and was used to interpret the model and significance of the independent variables.

Theoretical Framework

The study utilized various concepts of Machine Learning and Deep Learning which were used as the theoretical framework guiding the model developed in the study.

Machine Learning

Advancements in the field of computing technology and statistical learning has allowed for increased utilization and deployment of Machine Learning models. Geron (2019) described Machine Learning as the "science and art of programming computers so they can learn from data" (p. 2). With extensive data available, Machine Learning models can leverage mathematical and statistical foundations to extract functional patterns and trends from large data. Machine Learning can also be compared to how humans learn, think, and make decisions or predictions.

Differences Between Traditional Programming and Machine Learning. Geron (2019) explained the difference between a traditional programming approach and Machine Learning approach towards problem-solving. In a traditional programming approach, a rule-driven program is developed based on domain knowledge and experience to process input data. The program is evaluated and modified before deployment. However, in a Machine Learning approach, a rule-driven program is not explicitly developed but instead learned by a computer with input data. Large input data can train and develop a Machine Learning algorithm.

A Machine Learning approach is useful when a rule-driven program cannot be developed. Additionally, Machine Learning models are more efficient, accurate, and adaptable than traditional programming models. Machine Learning models can also be used to analyze large amounts of data and capture relationships, trends, and patterns that

might not be understood otherwise. Machine Learning approaches have been advantageous in cases where existing solutions cannot be explicitly computed through a traditional rule-based approach. Applications of Machine Learning can be found in multiple industries, including aviation, banking, engineering, law enforcement, manufacturing, and automobiles industries. Some successful uses of Machine Learning include speech recognition, credit card fraud detection, image classification, document classification, and credit card fraud detection.

Applications of Machine Learning. Lee (2019) explains that while Machine Learning utilizes foundations from various disciplines, scientific computing, mathematics, and statistics form the core of Machine Learning models. Machine Learning models can be used for descriptive and predictive tasks. Descriptive tasks are used to understand and analyze large amounts of data, whereas predictive tasks are used to utilize historical data for making predictions in the future. Additionally, Machine Learning is a term that includes various types of models and algorithms that can be used for different types of tasks as objectives, including but not limited to supervised learning, unsupervised learning, representation learning, semi-supervised learning, and reinforcement learning.

Deep Learning

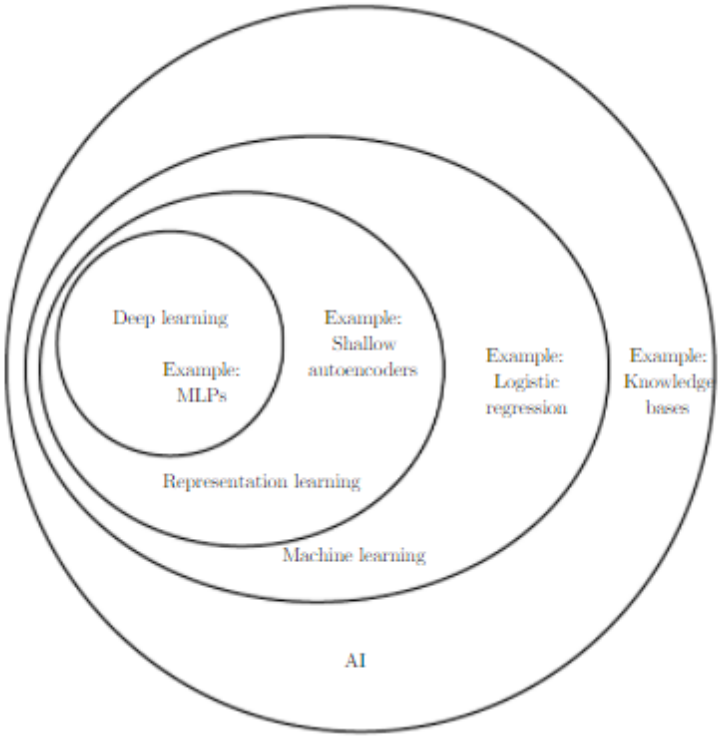
Deep Learning, just like Machine Learning, is a form of Artificial Intelligence that is developed with algorithms inspired by the neurons and functioning of a human brain (Geron, 2019). Deep Learning is considered a more modern and advanced form of Machine Learning that leverages more robust computing powers and larger datasets to solve more complex problems. Deep Learning models have demonstrated effectiveness

using more training data, large model parameters and weights, and increased computational power available (Goodfellow et al., 2016). Deep Learning has also benefited from backpropagation algorithms that use "a fast, greedy algorithm that can learn deep, directed belief networks one layer at a time, provided the top two layers form an undirected associative memory" (Hinton et al., 2006, p. 1527). A significant advantage of Deep Learning models is their ability to extract useful features from the training data and use those features to train the model resulting in performance that is more robust. Deep Learning models utilize Feature Learning by extracting significant features from large data and learning from significant representations of the data.

Differences Between Machine Learning and Deep Learning. Goodfellow et al. (2019) developed Figure 1 to describe the relationship between Artificial Intelligence, Machine Learning, Representation Learning, and Deep Learning. Deep Learning is considered a type of Representation Learning that uses feature extraction to enhance model training and performance. Deep Learning enhances the concepts of Representation Learning through Hierarchical Feature Learning, where the models extract features and create representations in multiple levels of the training data with the use of high-level features that are defined in terms of lower-level features (Bengio, 2012). Bengio (2009) described the concepts of Hierarchical Feature Learning as "automatically learning features at multiple levels of abstraction allow a system to learn complex functions mapping the input to the output directly from data, without depending completely on human-crafted features" (p. 2). The uses of Deep Learning models have significantly increased and are a significant domain of computation research.

Figure 1

Venn Diagram for Deep Learning



Note. Reprinted from "Deep Learning" by I. Goodfellow, Y. Bengio, and A. Courville, 2016 (https://www.deeplearningbook.org/). Copyright 2016 by Massachusetts Institute of Technology Press. Reprinted with permission (Appendix A1).

Deep Learning models are now used in applications such as face recognition, speech recognition, language translation, natural language processing, image recognition, voice recognition, and autonomous vehicles (Goodfellow et al., 2016). Some of the most common types of Deep Learning algorithms are Feedforward Neural Networks, CNN, Recurrent Neural Networks (RNN), and Autoencoders.

Neural Network Models

Neural Network models are a form of Deep Learning that were inspired by human brains and mimicked how neurological neurons transfer signals to each other. While research on Neural Networks can be dated back to 1944, the performance of Neural Network models developed in the 20th century was modest and restricted their advancement and utilization (Hardesty, 2017). However, the 21st century saw a resurgence of Neural Network research which was augmented by larger datasets and graphic chips or Graphic Power Units (GPU) which optimized computation. Due to the advancements in Deep Learning research, Neural Networks are commonly regarded as the best performing Artificial Intelligence system (Hardesty, 2017).

The simple Neural Network model is *The Perceptron* which was invented in 1957 by Frank Rosenblatt. A simple Perceptron consisted of linear threshold units and was originally used to compute simple binary classification problems (Geron, 2019). By utilizing linear threshold units, the Perceptron could compute linear combinations of the input data and define the binary classification problem based on threshold units. With further advancements, the Perceptron could also be used for multi-class classification problems. Like other linear classification problems, the Perceptron model demonstrated weakness is computation and performance and lost popularity in the 1970s. However, research on the Perceptron continued in the 1980s, which led to the development and utilization of the Multilayer Perceptron Model or Deep Neural Network models. Rather than a single linear threshold unit, Deep Neural Networks developed utilized several layers or hidden layers of neurons. Unlike the Perceptron model with the threshold linear unit, the neurons of Deep Neural Networks were capable of performing non-linear

operations that enhanced performance. Additionally, the introduction and advancement of backpropagation algorithms further improved the training and performance of Deep Neural Networks.

Neural Network Computation Overview

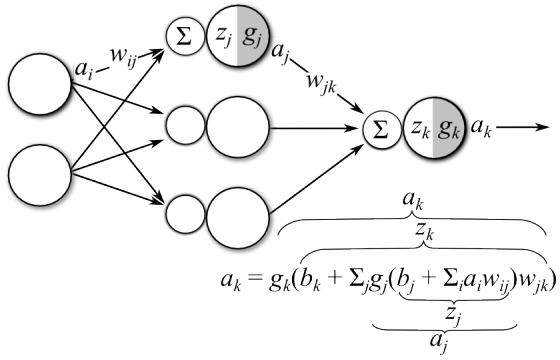
The Backpropagation training algorithm is a form of a Gradient Descent algorithm used in Deep Neural Networks to compute gradients through a forward pass and backward pass in the network. The Backpropagation computes the error gradients for each neuron in the network for every weight and bias term for the model. The Backpropagation algorithm repeats with every batch of training data to reduce the loss function of the model. After each batch (or mini-batch) is used to predict from the Neural Network (Forward Pass), the model output is compared to the desired output to compute the network error (loss function). The Backpropagation utilizes gradient calculation and calculates the contribution of each neuron in the network to the overall network error. By performing gradient descent through the hidden layers, it adjusts the weights and bias terms of the neurons in the Deep Neural Network to reduce the model error. Figure 2 depicts a Neural Network with two input neurons, one hidden layer of three neurons, and an output layer. The signal a_i transferred from the input layer to the hidden layers is multiplied by weight w_{ij} corresponding to that input-hidden layer along with the bias b_i to form a pre-activation signal z_j for the hidden layer. Equation 1 describes the formulation for the pre-activation signal z_j.

$$z_j = b_j + \sum_i a_i w_{ij} \tag{1}$$

The pre-activation signal is transformed to a_j due to the activation function g_j of the hidden layer. With a similar action of the output layer, the final output of the network is a_k as described in Figure 2.

Figure 2

Neural Network Formulation



Note. Reprinted From "Derivation: Error Backpropagation & Gradient Descent for Neural Networks" by D. Stansbury, 2020, The Clever Machine. (https://dustinstansbury.github.io/theclevermachine/derivation-backpropagation). Reprinted with permission (Appendix A2).

The model output a_k is compared to the desired output t_k to compute the model error E which is described in Equation 2.

$$E = \frac{1}{2} \sum_{k \in K} (a_k - t_k)^2 \tag{2}$$

The Backpropagation algorithm can be used to compute the error gradients for the model neurons using the error E value. The Backpropagation algorithm can be represented by the equation depicted in Equation 3.

$$\frac{\partial E}{\partial w_{jk}} = \frac{1}{2} \sum_{k \in K} (a_k - t_k)^2 = (a_k - t_k) \frac{\partial}{\partial w_{jk}} (a_k - t_k) = (a_k - t_k) \frac{\partial}{\partial w_{jk}} (a_k) \tag{3}$$

Utilizing Backpropagation, the gradient computation can be extended to the hidden layers gradients which is represented by Equation 4.

$$\frac{\partial E}{\partial w_{jk}} = (a_k - t_k)\frac{\partial}{\partial w_{jk}}(a_k) = (a_k - t_k)\frac{\partial E}{\partial w_{jk}} g_k(z_k) = (a_k - t_k)\acute{g_k}(z_k)a_j \quad (4)$$

The final error gradient computation is a product of the error term $(a_k - t_k)$, derivation of the output activation function $\acute{g_k}$, and activation output of node j in the hidden layer. The error gradient computation is used to update the weights and bias of the neurons. The output weight adjustments can be depicted in Equation 5 where μ is the learning rate.

$$w_{jk} - \mu\frac{\partial E}{\partial w_{jk}} \rightarrow w_{jk} \quad (5)$$

While Multilayer Perceptron or Deep Neural Networks demonstrated proficiency in various classification and regression tasks, they had limited performance in tasks such as image recognition, time series analysis, speech recognition, and signal processing. Advancements of Convolutional Neural Networks, Recurrent Neural Networks, and Autoencoders were attempts to improve and diversify the performance and applications of Neural Networks.

Convolutional Neural Networks

Convolutional Neural Networks (CNN) or Space Invariant Artificial Neural Networks are a form of Neural Networks that are commonly utilized for image recognition (Geron, 2019). Convolutional layers are regarded as critical components of CNNs. Convolution allows the CNN to concentrate on feature extraction from extensive multi-dimensional data (such as an image) to develop feature maps or kernels. Multiple convolutional layers could be stacked to assemble multiple low-level feature maps into large high-level feature maps. Some of the hyperparameters utilized for the convolutional

operation include the number of feature maps or kernels, the size of kernel, the number of strides, and the activation function. While CNNs are commonly associated with image recognition, CNNs have also been successfully used for time series analysis due to their ability to extract significant features from extensive multi-dimensional data and retain spatial context of input data. While spatial context due to local connectivity of convolutional operations is helpful for image recognition, spatial context can be used for retaining temporal relationships in 1-dimensional Convolutional layers. Although CNNs are be applied for time-series analysis, Recurrent Neural Networks are the most popular candidates for time series and sequential data analysis.

Recurrent Neural Network

RNNs are a form of Neural Networks that are best suited for sequential data. While regular Deep Neural Networks can handle sequential data, RNNs perform better while handling longer sequences. RNNs utilize Recurrent neurons or cells, which are similar to regular Neural Network cells, but also have an output connection feeding in a recurring input. An RNN cell receives an input, produces an output, and feeds that output back to itself. At a time frame t, an RNN cell receives an input $x_{(t)}$ and its own output from the previous time step $y_{(t-1)}$. Consequently, each RNN neuron has two sets of weights where w_x corresponds to input $x_{(t)}$ and w_y corresponds to input $y_{(t-1)}$. The output of a recurrent layer can be depicted by Equation 6 where b is the bias term.

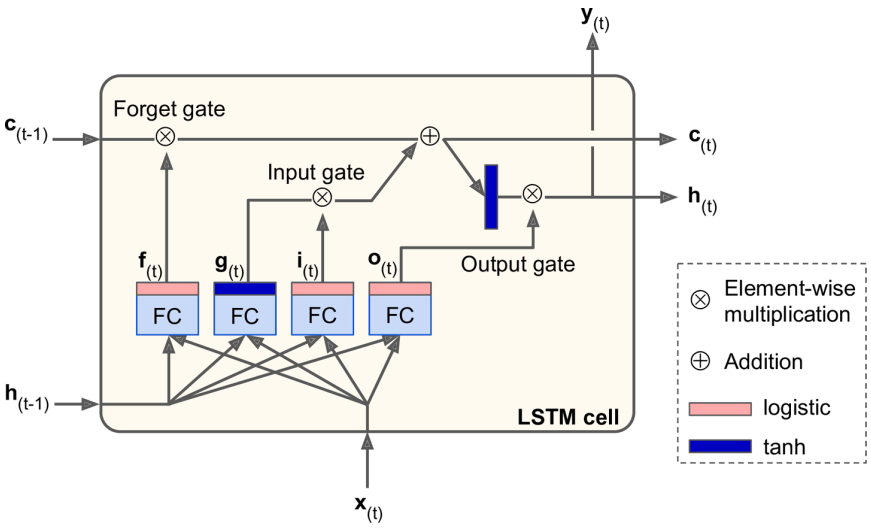
$$y_{(t)} = \phi(w_x^t x_t + w_y^t y_{(t-1)}) + b \quad (6)$$

RNNs have demonstrated better performance than other time series forecasting techniques due to their ability to compute trends and seasonality (Geron, 2019). RNNs utilize a variation of Backpropagation called Backpropagation Through Time (BPTT).

BPTT, just like the Backpropagation algorithm, is used to update model parameters (weights and biases) to minimize the model error. However, BPTT rolls through all input time steps where the errors are calculated and accumulated for each time step. With the total model error, BPTT updates model parameters through the input time steps. While utilizing RNN models is advantageous for time series data, RNNs have demonstrated weakness capturing relationships in long sequential data due to the exploding and vanishing gradient problem. Variants of the RNN architecture, such as LSTM and GRU, have demonstrated better performance while handling longer sequences due to the presence of gates in the cells. The presence of gates allows LSTM cells to combat the vanishing and exploding gradient problem such as preserving long-term dependencies in the data. LSTM cells contain Input Gates, Output Gates, and Forget Gates and utilize Tangent and Sigmoidal operations on the data to preserve significant long term data in sequences. Figure 3 depicts the architecture of an LSTM cell.

Figure 3

Long Short Term Memory Cell

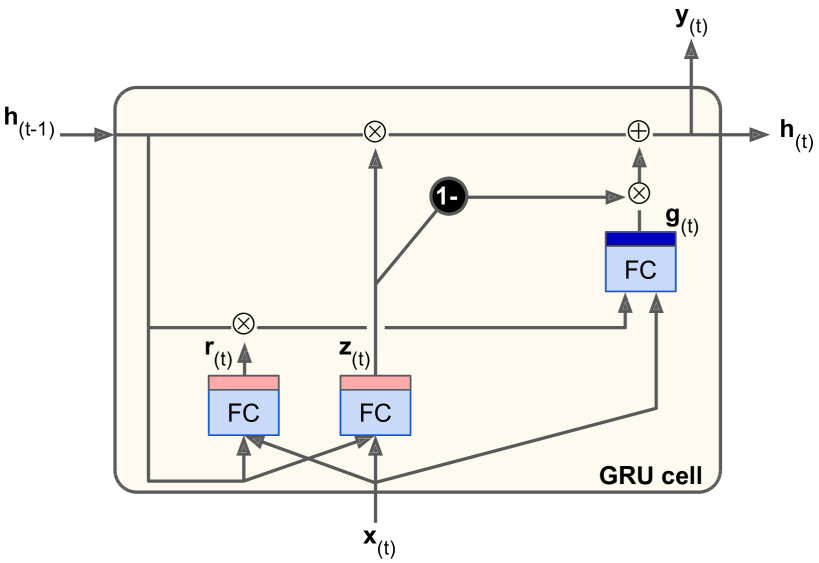


Note. Reprinted from "Hands-On Machine Learning with Sci-Kit Learn, Keras, and Tensorflow: Concepts, Tools, and Techniques to Build Intelligent Systems" by A. Geron, 2019, p. 517. Copyright 2019 by O'Reilly. Reprinted with permission (Appendix A3).

GRU cells are simpler versions of LSTM cells while demonstrating comparable performance. Rather than three different types of gates, GRU cells do not contain an Output Gate and a single gate controller controls the Input and Forget Gates. Figure 4 depicts the architecture of a GRU cell.

Figure 4

Gated Recurrent Unit Architecture



Note. Reprinted from "Hands-On Machine Learning with Sci-Kit Learn, Keras, and Tensorflow: Concepts, Tools, and Techniques to Build Intelligent Systems" by A. Geron, 2019, p. 519. Copyright 2019 by O'Reilly. Reprinted with permission (Appendix A3).

The Input Gate used in LSTM and GRU cells determines whether information from the input data must be modified or retained by the model. The Input Gate decides the operation based on the input x_t and previous cell state h_{t-1}, and the computation is depicted in Equation 7.

$$i_t = \sigma(W_{xi}^T x_{(t)} + W_{hi}^T h_{(t-1)} + b_i) \tag{7}$$

The Forget Gate used in LSTM and GRU cells determines the information from the input data that should be erased from memory. The computation is depicted in Equation 8.

$$f_t = \sigma(W_{xf}^T x_{(t)} + W_{hf}^T h_{(t-1)} + b_f) \tag{8}$$

The Output Gate used in LSTM cells determines the information from the LSTM that should be transferred to the next cell state h_{t+1}. The computation is depicted in Equation 9.

$$O_t = \sigma(W_{xo}^T x_{(t)} + W_{ho}^T h_{(t-1)} + b_o) \quad (9)$$

While RNN and its variation have been considered suitable for time series forecasting, literature suggests that utilizing RNNs in combination with other Neural Network architectures such as CNNs and Autoencoders can further improve results. Qu et al. (2020) demonstrated the use of CNNs for flight delay prediction utilizing operations and meteorological data. Qu et al. utilized a Dual-Channel CNN and Squeeze and Excitation-Densely Connected CNN to conduct a time series flight delay prediction task.

Sequence-to-Sequence Forecasting Using Recurrent Neural Networks

Due to their architectural features, RNNs are compelling candidates for time series forecasting. LSTM models used for sequence-to-sequence modeling utilize an encoder-decoder architecture. LSTM and GRUs can take in a sequence of input data to produce a single output, as in the case of sequence-to-vector models, or an output sequence of variable length, as in the case of sequence-to-sequence models. Sequence-to-sequence models are useful for applications such as language translation and are compelling candidates for the temporal analysis of data. The encoder component of a sequence-to-sequence model takes the sequence input and maps the input into a high-dimensional vector. The high dimensional vector is used by the decoder component to transform it into an output sequence. The Attention mechanism included in encoder-decoder models has further improved sequence-to-sequence model performance for processing sequential data. A Transformer is a type of attention encoder-decoder model

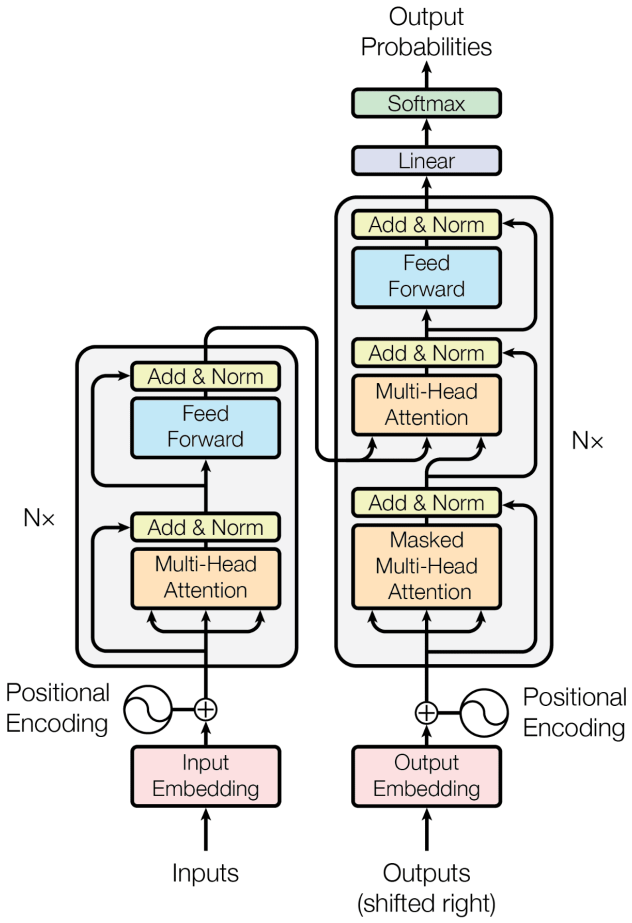
that has gained prominence as a robust sequence-to-sequence model for time series forecasting and has been considered a viable alternative to RNNs or time series forecasting.

Transformer Models

Transformer models, introduced in 2017, have gained popularity for sequence-to-sequence modeling in Natural Language Processing and time series forecasting (Vaswani et al., 2017). Similar to the LSTM sequence-to-sequence models, Transformer models utilize an encoder-decoder architecture. However, Transformers utilize an Attention mechanism which, in addition to the encoded vector, uses the input sequence and decides which steps of the input sequence are essential and should be retained for memory. Along with the encoded vector from the encoder, the model uses weights assigned to different inputs from the sequence by the attention mechanism. The decoder will utilize the encoded vector along with the weights information for the output sequence. Unlike RNNs, Transformer models do not use Recurrent cells and modules that are stacked upon each other in the encoder and decoder section of the model. The modules are composed of Masked Multi-Head Attention and Feed Forward Layers for the attention mechanism. Additionally, Input Embedding is used for the positional encoding of the input sequence. Figure 5 depicts the Transformer model proposed by Vaswani et al. (2017).

Figure 5

Transformer Model Architecture



Note. From "Attention Is All You Need" by A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. Gomez, L. Kaiser, and I. Polosukhin, 2019, *Advances in Neural Information Processing Systems*, *30*, p. 3 (https://papers.nips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf). Copyright 2017 by Curran Associates. Reprinted with permission (Appendix A4).

The Multi-Head Attention layers utilize linear and scaled dot-product attention mechanisms for their functioning. The attention mechanism used in the Multi-Head Attention layers is depicted in Equation 10, where Q is a matrix that contains the query

for the vector representation of the input sequence, K is a matrix that contains the key for the vector representation of the input sequence, and V are the values of vector representation of the input sequence.

$$Attention\ (Q, K, V) = Softmax\ \left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{10}$$

The Attention mechanism is used to assign weights a to the vector representation of the input sequence, which defines how each input of the sequence is influenced by the other inputs. The Softmax function assigns weights a to the input vectors between 0 and 1 as depicted in Equation 11.

$$a = softmax\ \left(\frac{QK^T}{\sqrt{d_k}}\right) \tag{11}$$

While Transformer models are commonly used for Natural Language Processing applications such as language translation, modifications to the architecture can lead to their effective utilization for time-series tasks as well (Zeng et al., 2022). Firstly, the Input Embedding is removed since the input sequence will contain numerical data. A linear transformation can be used instead to transform the input sequence to a high-dimensional vector representation. Additionally, the Softmax layer from the decoder is removed because the output will be real values and not probabilities.

Summary

The section reviewed the literature on topics deemed necessary for the study. Some of the topics reviewed included runway configurations, the impact of runway configuration on airport operations, runway configuration optimization and prediction, the impact of taxi out times on flight delays, taxi out times prediction, and Deep Learning concepts such as RNNs and sequence-to-sequence models such as Transformer models. The literature on runway configuration and taxi out prediction substantiated the need for

further research while exposing the research gaps in past literature. While Machine Learning techniques have successfully been utilized for runway configuration and taxi out time prediction, there is a significant gap in utilizing time series prediction models for analysis. Literature suggested that the use of any time series models and Deep Learning models is vastly underutilized despite of their demonstrated success for sequence modeling. The most significant research gap is the lack of sequence-to-sequence modeling for taxi out times and runway configuration prediction tasks.

The section also reviewed several Machine Learning models that have demonstrated success in time series analysis and focused on the increased success and utilization of sequence-to-sequence models. The demonstrated success of RNNs and Transformer models warrants their application and testing for predicting runway configuration and taxi out times. Additionally, while LSTM models have been used for time series prediction tasks in the reviewed literature, Transformer models have not been utilized despite their demonstrated success. This study aimed to close the research gap and add to the literature on sequence-to-sequence modeling and its application for runway configuration and taxi out times prediction.

Chapter III: Methodology

The study aimed to develop prediction models for runway configuration and taxi out times based on surface weather observations for two airports in the United States. The two airports used for the model development were MCO and JFK. For the model development, data for hourly surface weather observations, traffic volumes, runway configuration, and taxi out times per hour were collected from different databases and processed to develop a corpus of data. This chapter reviews the methodology adopted for the study, including the data collection, data preprocessing, model development, programming, hyperparameter selection, feature assessment, and model evaluation procedures. The details provided in this chapter are consequential for replicating the model development strategy or using the results for further research and development on the subject.

Research Method Selection

The study utilized a data-driven exploratory approach to analyze and capture the temporal relationship of the variables of interest to fill the research gaps within Machine Learning and Aviation. While advances in Machine Learning and Deep Learning algorithms have resulted in a variety of modeling choices, the researcher adopted Deep Learning models that have demonstrated proficiency in time series prediction especially sequence-to-sequence modeling. LSTM encoder-decoder and Transformer models were developed to predict runway configuration and taxi out times based on surface weather observations for two airports in the United States. Once the models were developed, the LSTM encoder-decoder and Transformer models were evaluated and compared for the regression task (taxi out times) and classification task (runway configuration). For the

taxi out times prediction, the models were evaluated on RMSE, MAE, R-Squared, and MSE. For the runway configuration prediction, the models were evaluated on the accuracy, precision, recall, and kappa score. To assess the impact of each variable on the prediction models developed, a feature assessment evaluation was conducted. Shapley Value Imputation (SHAP) analysis was utilized to determine the effect of each variable on the predictions made by the model.

Apparatus and Materials

Deep Learning models require a large corpus of data to capture dependencies and relationships among variables. There is no single database that could be used to develop a model to answer the research questions. The hourly surface weather observations for the two airports were downloaded from an open-source Local Climatological Data repository managed by the NOAA (NOAA, n.d.). The hourly traffic data (number of scheduled departures and arrivals), taxi out times, and runway configurations for the two airports were downloaded from an open-source ASPM database managed by the FAA. The datasets were downloaded in Comma Separated Values (CSV) format and were pre-processed using Microsoft Excel and the Pandas library on the Python programming language.

For the model development, popular Machine Learning and Deep Learning libraries on Python, such as NumPy, Sci-Kit Learn, Keras, and Tensorflow 1.0, were used. All steps of the model development and evaluation were conducted utilizing Python operations and libraries.

Population/Sample

The sample for this study is flights that departed from JFK and MCO from January 2012 to December 2021 and the corresponding hourly surface weather observations, hourly traffic demand, runway configuration selection, and taxi out times. While the study utilized data from only two airports in the United States, the population for the study can be considered all the flights to and from airports with similar weather patterns, traffic operations, and runway configurations. To develop Machine Learning models, especially models that utilize a Deep Learning architecture, many data points were required to ensure adequate model fit and lower the potential for model underfitting or model overfitting. The two airports chosen for this study were based on traffic demand and runway configurations. MCO comprised of four parallel runways, and JFK comprised four intersecting and parallel runways. Utilizing two airports with different runway configurations allowed the researcher to develop and validate Deep Learning models with varying runway configurations and weather patterns.

Sources of the Data

The data to build the Deep Learning models was sourced from two separate databases. The researcher utilized a database for hourly surface weather observations for the two airports from January 2012 to December 2021 from an open-access data repository managed by the NOAA (NOAA, n.d.). The hourly traffic demand, runway configuration selection, and taxi out times data were sourced from the FAA ASPM database (FAA, n.d.). All the data sources utilized for this study were developed and

maintained by reputed governmental agencies in the United States. Hence, the data sources were expected to provide unbiased and accurate data.

Treatment of the Data

Initial data cleaning was conducted utilizing Microsoft Excel. The ASPM and NOAA datasets were joined to create a new dataset using the Pandas library on Python. Finally, there was a dataset for each airport which consisted of data on surface weather observations, traffic, taxi out times, and runway configurations for each airport. Once the datasets were joined, further data pre-processing was required to ensure there were no missing values. The Pandas library was used to ensure every hour in the specified period had a separate row in the dataset. Considering the dataset for the study was a time series dataset, cells with missing values could not be deleted so as to preserve the temporal nature of the data. For continuous variables, linear interpolation was used to estimate the values of the missing cells. For discrete variables, forward filling was used to estimate the values of the missing cells. Linear interpolation and forward filling are standard methods used in time series analysis to compensate for missing values. Once the data was cleaned and pre-processed, the two datasets each consisted of 87,671 instances or rows.

Python libraries such as Sci-Kit Learn, Tensorflow, and Keras cannot process discrete input values. Discrete variables require Binarization to convert discrete values to continuous numerical values. Binarization, commonly called dummy values, can be conducted utilizing the Pandas library, where the discrete variables are transformed into vectors of binary numbers. Additionally, the classification task (runway configuration prediction) task for JFK was treated as a multi-class classification problem due to the number of dependent variables to be predicted. The runway configuration variable

needed to be one-hot encoded utilizing the OneHotEncoder feature on Sci-Kit Learn before the models could be developed. Table 1 describes the continuous and discrete variables in the dataset. Out of the 15 variables defined in Table 1, Runway Configuration and Taxi out Times were treated as the dependent variables for the study.

Table 1

Variables Used for the Classification Models

Variable	Source	Data Type	Notes
Day of the week	MS Excel	Discrete	7 days
Departures	ASPM	Continuous	
Arrivals	ASPM	Continuous	
Surface pressure	NOAA	Continuous	
Temperature	NOAA	Continuous	Unit: Celsius
Dew point temperature	NOAA	Continuous	Unit: Celsius
Precipitation	NOAA	Continuous	Unit: Inches
Weather type	NOAA	Discrete	No Significant Weather, Fog, Rain, and Thunderstorm
Pressure change	NOAA	Continuous	Unit: Inches of Mercury (Hg)
Relative humidity	NOAA	Continuous	
Visibility	NOAA	Continuous	Unit: Statute Miles
Wind direction	NOAA	Discrete	North-East, North-West, South-East, South-West, Calm, Variable
Wind Speed	NOAA	Continuous	Unit: Knots
Taxi out time	ASPM	Continuous	Unit: Minutes
Runway configuration	ASPM	Discrete	

Before the data was used for the model development, the data needed to be split for training, validation, and testing. Due to the temporal nature of the data, cross-validation or random shuffling strategy could not be adopted, so the data was split into

80% for training, 10% for validation, and 10% for testing. The data splitting was conducted utilizing the Train-Test Split function in Sci-Kit Learn, where Shuffle was set to False.

Python Programming Language

Python is a high-level programming language that is used for multi-disciplinary Purposes, including web development, scientific computing, web scraping, automation, data analytics, and Machine Learning (Van Rossum & Drake, 2009). Python has become an increasingly popular choice as a programming language for data analytics and Machine Learning over other programming languages such as R, Julia, and Java. Python requires the use of a development environment that acts as an interface for programmers to input commands and view and access outputs. For the analysis, all Python operations were conducted on Jupyter Notebook and Google Colaboratory (Google Colab). Several Python libraries were used for the study.

Pandas

Pandas is an open-source data analysis and manipulation library used in Python (McKinney, 2010). Pandas can be used for data analysis tasks, including indexing, reading data files, writing and re-writing data in a file, reshaping data structures, data range generation, data shifting, filtration, data slicing, data merging, and joining. Pandas was used for the initial data preprocessing of the study, which included joining the NOAA and ASPM datasets, compensating for hourly intervals, and linearly interpolating and forward filling missing values. Pandas library was used to index the datasets utilizing the Date/Time column to structure the data for time series analysis.

NumPy

NumPy is an open-source Python library used for mathematical operations with data arrays (Harris et al., 2010). NumPy can perform advanced data structuring and mathematical operations utilizing data arrays. NumPy was used for restructuring the input data for model development, standardization of input data, and transforming the output data structures. NumPy is a powerful data analysis tool when used in conjunction with other Python Libraries such as Pandas and Tensorflow.

Sci-Kit Learn

Sci-Kit Learn is a popular open-source Machine Learning Library on Python that is primarily used for preprocessing and model development tasks (Pedregosa et al., 2011). Sci-Kit Learn is used for creating Machine Learning models, including Regression, Random Forest, Decision Tree, Support Vector Machine, Naïve Bayes Classifier, and Extreme Gradient Boost. For the study, Sci-Kit Learn was primarily used for the Train-Test split to form the training, validation, and testing data sets and the Standard Scalar operation to standardize the training, validation, and testing datasets. Sci-Kit Learn was also used for model evaluation tasks such as developing the confusion matrix and computing the accuracy, MAE, MSE, RMSE, precision, kappa score, and recall.

Tensorflow

Tensorflow is an open-source Machine Learning and Deep Learning library developed by Google's Brain Team in 2019 (Abadi et al., 2016). Tensorflow is a popular library for Deep Learning due to its data structure utilization of 3-dimensional tensors for training which optimizes performance, improves flexibility, and reduces training time.

The study utilized Tensorflow to develop all the models in the study. Features included in the Tensorflow library such as TimeSeriesGenerator were used to easily reshape the training, validation, and testing data to the tensor format for developing the models.

Model Development and Architecture

A Deep Learning model involves the selection and validation of various hyperparameters, including number of cells per layer, the number of hidden layers, activation functions, loss function, and optimization algorithm. The hyperparameters for this study were selected based on available literature and baselines model and a trial-and-error approach.

Activation Functions

Unlike the Perceptron model, modern Deep Neural Network models can utilize non-linear operators as activation functions. Enhancement and discovery of activation functions remain a focus of modern Deep Learning. However, some of the most common activation functions include Linear, Sigmoid, TanH, ReLU, Softmax, and Radial Based Functions. The ReLU activation functions, depicted in Equation 12, consist of various variants, which include Leaky ReLU, Random ReLU, Parametric ReLU, and Exponential Linear Unit (ELU). ELU, depicted in Equation 14, is a popular choice as an activation function in the hidden layers, while the Softmax activation function, expressed in Equation 13, is commonly used as an activation function for output layers of the multi-class classification models. The Sigmoid activation function, expressed in Equation 15, was used as an activation function in the output layer of binary classification models.

$$\mathrm{ReLU}(X) = \max\{x, 0\} \tag{12}$$

$$\mathrm{Softmax}(x_i) = \exp(x_i) / \sum \exp(x_j) \tag{13}$$

$$\mathrm{ELU}_{\alpha} = \begin{cases} \alpha(\exp(z) - 1) \text{if } z < 0 \\ z \text{ if } z > 0 \end{cases} \quad (14)$$

$$\text{Sigmoid } \phi = \frac{1}{1 + e^{-z}} \quad (15)$$

Loss Function

Deep Learning models like the ones developed in this study need a loss function to minimize during the training stage of model development. Based on the output and expected probability distribution and domain need of the modeling problem, various loss functions can be used in conjunction with an optimization algorithm to minimize the loss function. Some standard loss functions used in Deep Learning are Hinge Loss, Kullback-Leibler Divergence, and Cross-Entropy for classification problems and MSE and MAE for regression problems.

The models were developed for two airports which were MCO and JFK. For MCO, there were two possible runway configurations resulting in two possible outputs to the binary classification model. For the binary classification model, Binary Cross Entropy/Log Loss, depicted in Equation 16, was used as the loss function.

$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} y_i . \log(p(y_i)) + (1 - y_i) . \log(1 - p(y_i)) \quad (16)$$

While the classification model developed to predict the runway configuration for MCO could be treated as a binary classification model, the classification model to predict the runway configuration at JFK was developed as a multi-class classification model. For JFK, four possible runway configurations or outputs were possible. For the multi-label classification model, Categorical Cross Entropy, depicted in Equation 17, was used as the loss function, and the output class was one-hot encoded.

$$H_p(q) = -\sum_{i=1}^{N} y_i . log\hat{y}_i \quad (17)$$

The regression models developed to predict the taxi out times at the two airports utilized MSE as the loss function. MSE, depicted in Equation 18, measures the difference between the predicted output value and the actual output value.

$$\mathbf{MSE} = \frac{\mathbf{1}}{\mathbf{N}}\sum_{\mathbf{i=1}}^{\mathbf{n}}(\mathbf{y_i} - \widehat{\mathbf{y_1}})^{\mathbf{2}} \quad (18)$$

Optimization Algorithm

An optimization algorithm is a crucial hyperparameter in developing Deep Learning models that minimize the loss function and the generalization error. Some of the standard optimization algorithms or optimizers used in Deep Learning are Gradient Descent, Stochastic Gradient Descent, and Adam. Based on published literature and understanding of the subject, Adam was chosen as the optimizer for developing all the models in the study. Adam has been regarded as the most effective and robust optimizers for developing Deep Learning models. Adam, which is derived from adaptive moment estimation, utilizes exponential weighted moving averages, which is also known as leaky averages, to gain an estimate of the momentum and second moment of the gradient (Brownlee, 2017; Kingma & Ba, 2017). Adam is an extension of Stochastic Gradient Descent, which maintains the same learning rate for all weight updates. However, Adam maintains a dynamic learning rate with a per-parameter learning rate, which is adapted based on the average first moment and the average second moment of the gradient.

Adam requires a few configuration parameters before it can be used. Alpha is considered the learning or step size, Beta1 is considered the exponential decay rate for the first-moment estimate, Beta2 is considered the exponential decay rate for the second-moment estimate, and Epsilon is a small number to prevent any division by zero. Considering the models were developed utilizing the Tensorflow Python Library, default

values of the parameter configurations preset by Tensorflow were used where Alpha was set to .001, Beta1 was set to .9, Beta2 was set to .999, and Epsilon was set to e^{-8} (Brownlee, 2017).

Regularization

Regularization is an effective strategy used in Machine Learning to reduce overfitting or testing errors. Based on the bias-variance tradeoff concept, regularization techniques are used to improve the simplicity, computation efficiency, and generalization of the model (Goodfellow et al., 2016). Some standard regularization techniques used for Deep Learning are Lasso, Ridge, Early Stopping, and Dropout. For the study, Early Stopping and Dropout were used as regularizers.

Dropout was used as a regularizer and was set to 50%. At every iteration of training, 50% of the cells in the dropout layer were removed along with their inbound and outbound connections. Training different iterations with different cells in the layer resembled an ensemble training method where the final model would not be dependent on any particular cell or connection, which would improve the generalization power of the model. Early Stopping is another regularization technique similar to a cross-validation technique where the model was trained on a training and validation set. The loss function of the validation set was monitored, and when the validation loss function stopped reducing or improving, the training was stopped. An essential parameter for Early Stopping was *Patience*, which is the number of epochs with no improvement or reduction of the validation loss function after which training was stopped. For the study, Patience was set to 20.

Repeat Vectors

Repeat Vectors are used for sequence-to-sequence modeling and can be added to the model architecture using the TensorFlow library (Geron, 2019). The Repeat Vector layer is added to repeat the input from the previous layer and repeat the input *n* times where *n* is an adjustable hyperparameter. For sequence-to-sequence modeling, *n* is set to the length of the output sequence so that the input can be repeated *n* number of times for generating *n* different predictions to complete the output sequence.

Time Distributed

For a sequence-to-sequence problem, we require the LSTM encoder to layers to be capable of generating an output sequence rather than just an output vector. Along with Repeat Vectors, Time Distributed layers are used as wrapper layers to every temporal step in the input sequence (Geron, 2019). Additionally, the Time Distributed layer requires the input layers to be 3-dimensional tensors to produce 3-dimensional time distributed output tensors for the output sequence. Applying a Time Distributed layer allows a simpler and fully connected dense layer to each temporal step in the input sequence. The advantage of this addition is that the model requires few weights and parameters for improved computational speed and capacity.

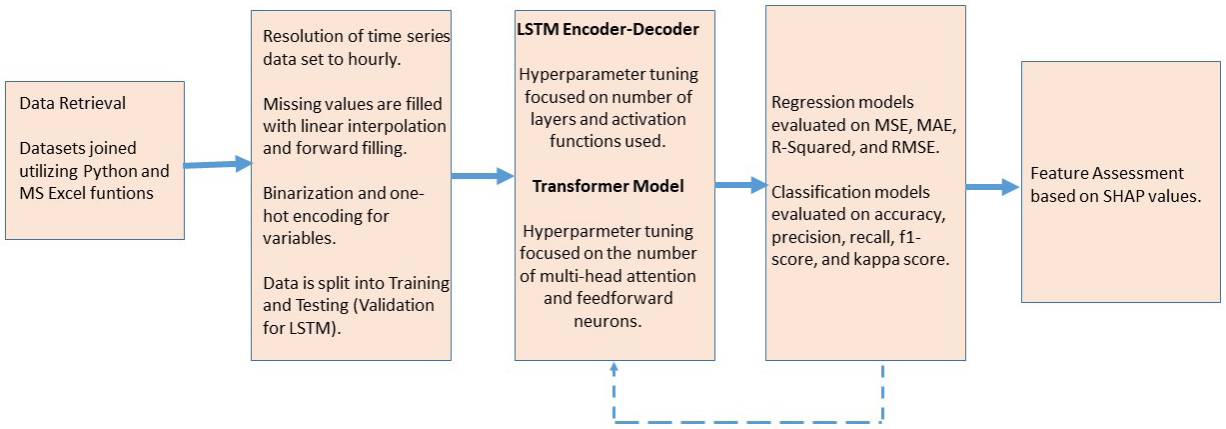
Model Development

To predict the taxi out times and runway configurations, two Deep Learning model architectures were developed, validated, and tested to ensure the best model fit. An LSTM encoder-decoders model was compared with a Transformer model for sequence-to-sequence modeling. For sequence-to-sequence modeling, various parameters must be set to shape the input and output sequences. Considering every row in the datasets used

for model development corresponded to an hour, the dataset was treated as a time series dataset with an hourly resolution. Appendix C depicts the coding utilized to transform the data from a Dataframe format to create the input and output sequences for the supervised learning task. Figure 6 illustrates the model development pipeline deployed for this study.

Figure 6

Model Development Pipeline



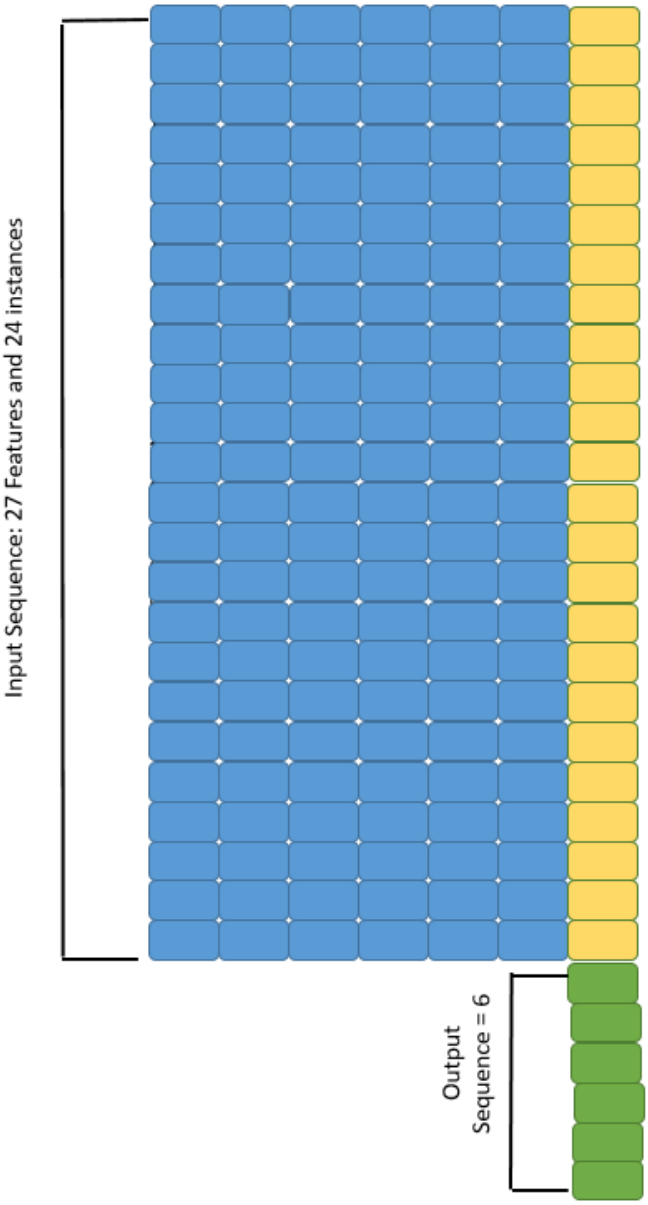
For both the runway configuration and taxi out times prediction tasks, the window length for the input sequence was set to 24, implying 24 hours of data was used for the prediction. Additionally, the models were trained on mini-batches of 16 instances to ensure computation efficiency. With the inclusion of additional features due to the Binarization of the discrete variables, 27 features were used for the input data. The input tensors for training the Deep Learning models must be shaped as [batch size, window size, features].

Considering the models developed were sequence-to-sequence, the output needed to be a sequence too. The output sequence was set to four time steps which implied the model predicted taxi out times and runway configurations for 6 hours. Multiple output

values for each prediction indicated that the classification and regression tasks could be treated as multi-label supervised learning models. Figure 6 illustrates the input sequence used to predict an output sequence for the class label.

Figure 7

Input and Output Sequencing



Runway Configuration Class Labels

The Deep Learning models for the study were developed, validated, and tested on two different airports in the United States, each with two different runway layouts and weather patterns. The two airports selected have different runway layouts, with MCO having four parallel runways and JFK having four runways which are parallel and intersecting runways. Depending on the different runway configurations possible at the two airports, the class labels were coded for the multi-class classification tasks. The class labels were one-hot encoded for the classification problem and were based on the historical runway configuration data gathered from the ASPM database. Table 2 describes the different class labels used to train the classification models.

MCO is a major Class B airport located in MCO, Florida. The airport has four parallel runways, which are Runway 36L/18R, Runway 36R/18L, Runway 17L/35R, and Runway 17R/35L. The departure and arrival flows are conducted in two possible configurations, which are North or South. Runway configuration prediction for MCO was treated as a binary classification problem with 1 indicating a North flow and 0 denoting a South flow.

JFK is a major airport located in Queens, JFK. The airport has four parallel and intersecting runways, which are Runway 13L/31R, Runway 13R/31L, Runway 22L/04R, and Runway 22R/04L. The departure and arrival flows are conducted in four possible configurations. The runway configuration prediction for JFK was treated as a multi-class classification task with four class labels.

Table 2

Description of Class Labels

Airport	Departure Runway Configurations	Class Labels Code
MCO	RWY36R/RWY35L	N
	RWY18L/RWY17R	S
JFK	RWY13R	E
	RWY22R	W
	RWY31L	N
	RWY31R	S

Note. N, S, E, and W denote North, South, East, and West configurations.

Model Evaluation

The models were evaluated for the classification and regression tasks separately. The taxi out time prediction was treated as a regression task and the models were evaluated on MSE, MAE, R-Squared, and RMSE. The runway configuration selection prediction task was treated as a classification task and the models were evaluated on the accuracy, precision, recall, and Cohen's kappa score.

Feature Assessment

Model interpretability is a critical aspect of building any Machine Learning model. Understanding and assessing the impact of each feature or variable on the prediction of the model allowed for the acquisition of insights into the prediction mechanisms of Deep Learning models rather than treating the models as *black boxes*. While Feature Importance and Variable Importance are common feature assessment techniques, they have been demonstrated to perform poorly in capturing attribute dependency among the attributes or features used for model development. This weakness of Feature Important and Variable Importance might over-emphasize or under-emphasize some features depending on how those features correlate with other features. This

weakness is commonly referred to as the high-correlation variable problem (Hooker et al., 2019). A relatively modern and advanced technique called Shapley Additive Explanations (SHAP) can be used to assess the features utilized to develop a Machine Learning model, especially a neural network model (Molnar, 2021). Derived from a game theory approach to explain the output of models, SHAP computes Shapley Values utilizing coalitional game theory by treating each feature as a *player* in the game. The SHAP computation can be illustrated by Equation 19, where g is the explanation model, $\acute{z} \in \{0,1\}^M$ is the coalition vector, M is the maximum coalition size, and $\emptyset_j$ is the feature attribution of a feature j.

$$g(\acute{z}) = \emptyset_0 \acute{+} \sum_{J=1}^{M} \emptyset_J \acute{z}_j \tag{19}$$

A significant advantage of utilizing SHAP to interpret a model is the robustness of SHAP to attribute dependency. Using Shapley Value Imputation, SHAP is robust to the multicollinearity among the features (Lipovetsky & Conklin, 2001; Lundberg & Lee, 2017). The mean magnitude of SHAP values will be derived utilizing the SHAP library in Python.

Summary

The primary focus of the study was the development of the LSTM encoder-decoder and Transformer models for the sequence-to-sequence tasks for predicting runway configuration selection and taxi out times. The chapter focused on the data collection and pre-processing techniques that included the joining of the databases, Binarization for discrete variables, and processing cells with missing values. Additionally, the class labels or dependent variables for the classification and regression tasks were described. The section introduced the programming language Python and the

associated libraries used for data pre-processing, model development, model evaluation, and feature assessment tasks. Pandas, NumPy, Sci-Kit Learn, TensorFlow, and SHAP libraries on Python were used in this study.

Chapter IV: Results

This chapter presents the results of the study. The chapter includes results for the exploratory data analysis for the data utilized for the study, final model architectures for the LSTM Encoder-Decoder and Transformer models, model evaluation metrics, and feature assessment used for the models.

Exploratory Data Analysis

Exploratory Data Analysis (EDA) is an essential stage of any Machine Learning task and is used to understand the trends and patterns in the data used for the model development. Through various descriptive statistical techniques and data visualization methods, EDA can be conducted to gain deeper insights into the data used for the model development.

The initial EDA was focused on the dependent variables used for the regression task (taxi out times) and classification task (runway configuration selection). Figure 8 is a histogram of the taxi out times at MCO ($M = 13.93$, $SD = 7.34$) and JFK ($M = 27.47$, $SD = 14.58$).

Figure 8

Histogram of Taxi Out Times at MCO and JFK

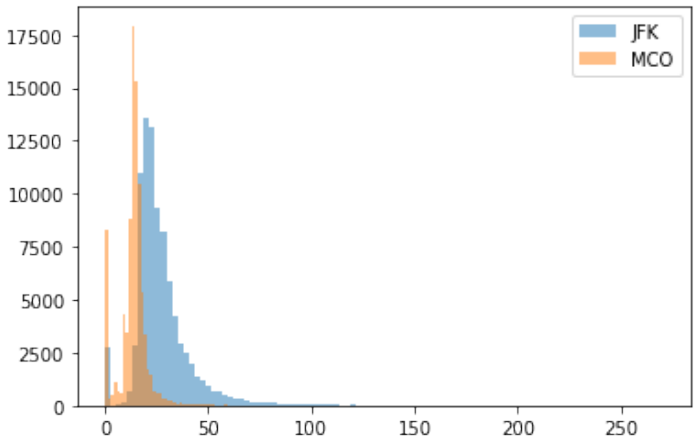


Figure 9 depicts the average taxi out times for MCO and JFK by Month. The average taxi out time at JFK was higher every month than the average taxi out time at MCO. Additionally, the average taxi out time for JFK was the highest in January (34.68 min), and the average taxi out time for MCO was the highest in July (15.24 min).

Figure 9

Average Taxi Out Time Per Month

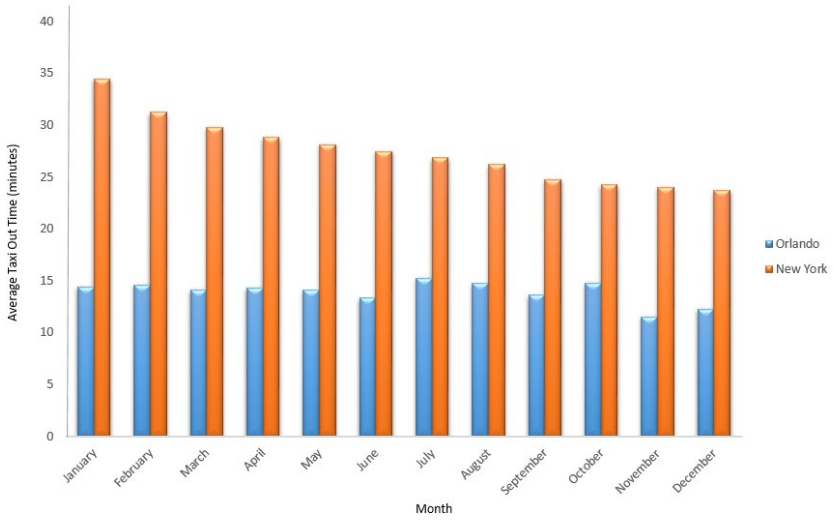


Figure 9 depicts the average taxi out time at the MCO and JFK per year. JFK has seen a decrease in the average taxi out time through the years. Similarly, MCO has seen a decrease in the average taxi out time per year from 2012 to 2021.

Figure 10

Average Taxi Out Time Per Year

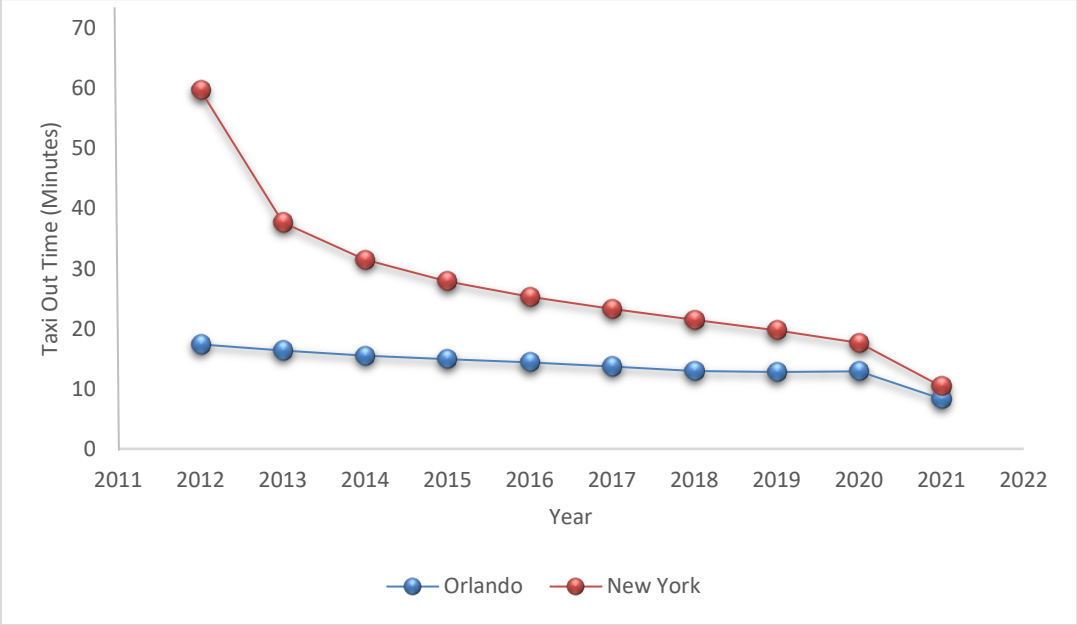


Figure 11 depicts the runway configuration selections at JFK. Configuration C was the most popular configuration, followed by Configuration B.

Figure 11

Runway Configuration Selection for JFK

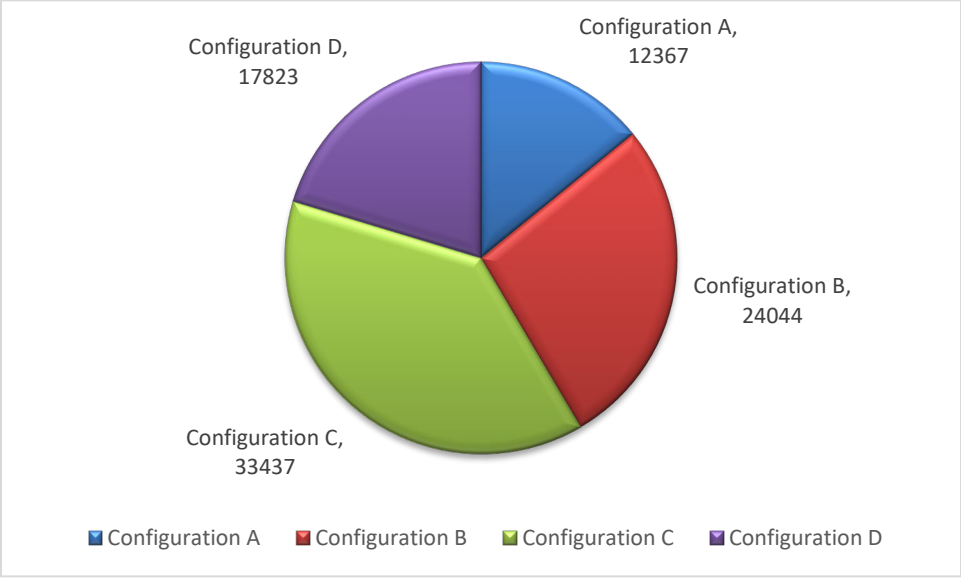


Figure 12 depicts the runway configuration selections at MCO. The North configuration was more widely used as compared to the South configuration.

Figure 12

Runway Configuration Selection for MCO

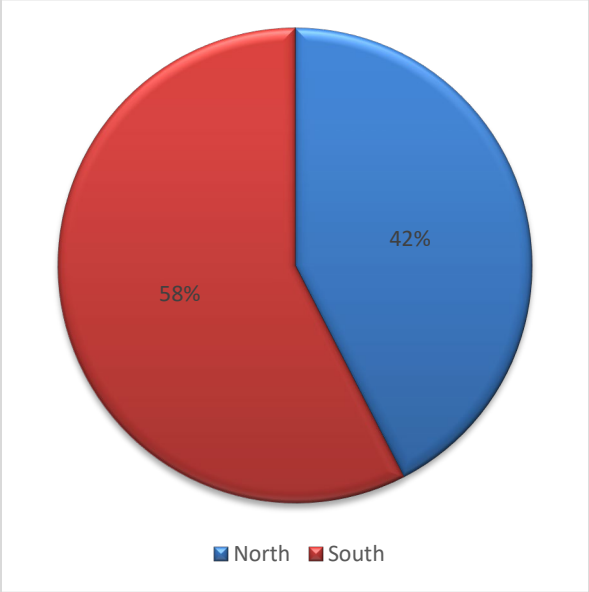


Figure 13 is a wind rose diagram for the historical wind direction and speed data for MCO. As depicted, westerly winds were the most prominent winds in terms of frequency and wind speed. Wind direction and wind speed are essential aspects for the study to predict runway configuration, as the literature suggested that wind direction was a significant variable for prediction models developed in previous studies.

Figure 13

Wind Rose Diagram for MCO

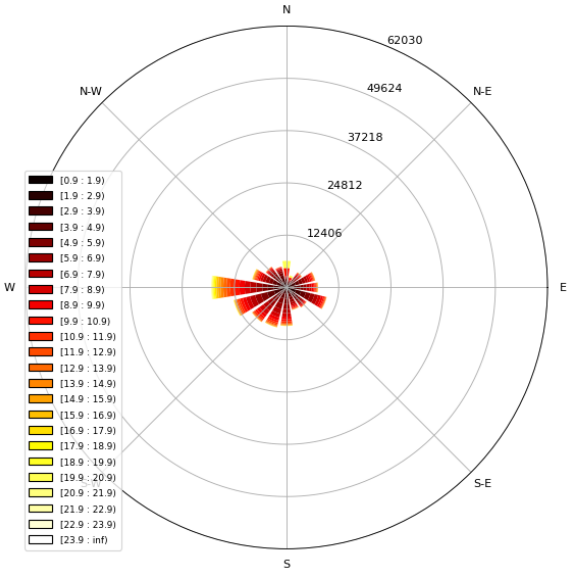
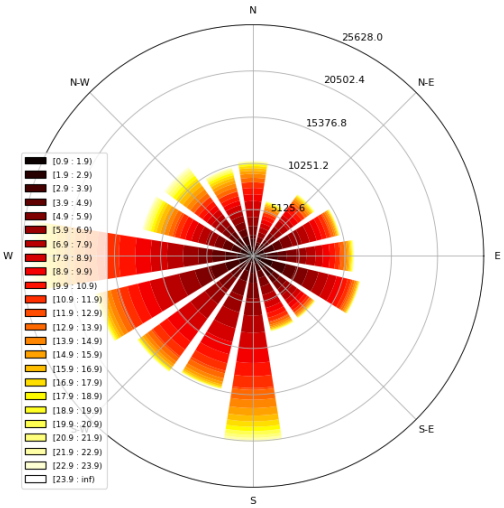


Figure 14 is a wind rose diagram for the historical wind direction and speed data for JFK. As depicted, southerly and westerly winds were the most prominent winds in terms of frequency and wind speed.

Figure 14

Wind Rose Diagram for JFK



Time Series Analysis

Due to the temporal nature of the data used in the study, a significant focus of the EDA was to evaluate some data characteristics for time series analysis. The analysis included stationarity testing, Autocorrelation Function (ACF), and Partial Autocorrelation Function (PACF). Stationarity Testing is conducted on time series datasets to ensure that properties such as mean, variance, and autocorrelation structure do not change over the time covered by the data. While not a significant assumption for Deep Learning models, stationarity allows researchers to understand the trends and seasonality in time series data and accommodate any changes in periodic fluctuations in the time series over time. The distribution of a stationary time series dataset would be similar irrespective of the point of time the sampling is conducted, as parameters such as variance and mean would remain relatively constant over time. While the stationarity of a dataset can be evaluated visually through various visualization techniques, various statistical tests such as Augmented Dickey-Fuller Test, Kwiatkowski–Phillips–Schmidt–Shin (KPSS) Test and Philip-Perron Test exist to test stationarity (Kulaksizoglu, 2015).

An Augmented Dickey-Fuller Test was conducted at a significance level of .05. The Augmented-Dickey Fuller Test tests the null hypothesis that a unit root is not present in the time series analyzed. Based on the test statistic, which is a negative number, the null hypothesis can be rejected and determined that the unit root is not present in the time series and that the time series is stationary. Table 3 depicts the results of the Augmented Dickey-Fuller Test for MCO. All the variables used for the airport with continuous values were evaluated with the Augmented Dickey-Fuller Test and were determined to be

stationary. The properties of the variables, such as mean, variance, and autocorrelation functions did not change with time.

Table 3

Augmented Dickey-Fuller Test for MCO

Variable	Test Statistic	Number of Lags Chosen
Hourly departures	-8.3071	66
Hourly arrivals	-9.2989	66
Altimeter	-21.5164	66
Dew point	-14.9095	66
Temperature	-12.8554	66
Precipitation	-36.7054	50
Pressure changes	-49.198	65
Relative humidity	-21.494	66
Visibility	-27.897	66
Taxi out time	-27.5861	66

Note. The critical value for all variables was -2.862. All variables were determined to be stationary.

Table 4 depicts the results of the Augmented Dickey-Fuller Test for JFK. All the variables used for the airport with continuous values were evaluated with the Augmented Dickey-Fuller Test and were determined to be stationary. The properties of the variables, such as mean, variance, and autocorrelation functions did not change with time.

Table 4

Augmented Dickey-Fuller Test for JFK

Variable	Test Statistic	Number of Lags Chosen
Hourly departures	-11.9618	65
Hourly arrivals	-11.6401	66
Altimeter	-25.7672	66
Dew point	-10.2717	66
Temperature	-7.7673	66
Precipitation	-49.7126	50
Pressure changes	-51.4636	65
Relative humidity	-22.6754	66
Visibility	-31.5319	66
Taxi Out Time	-25.5431	66

Note. The critical value for all variables was -2.862. All variables were determined to be stationary

The ACF is used to define how the time series data points of a variable are related to previous data points of the same variable. ACF can be used to understand the self-similarity of a data point of a variable to the previous data points in the same variable. While ACF is a significant assessment for univariate time series analysis, utilizing ACF for the dependent variable for a multi-variate time series analysis is helpful to understand how much of the variance in the dependent variable can be captured by lagged versions of the same dependent variable itself. An ACF at a significance of .05 was conducted for taxi out times data for MCO and JFK. Figure 15 depicts the ACF plot with 40 lags for MCO. It is evident that the taxi out times variable experiences a significant

autocorrelation effect from its previous data points. However, the autocorrelation effects periodically shift between positive and negative values.

Figure 15

ACF Plot for Taxi Out Time at MCO

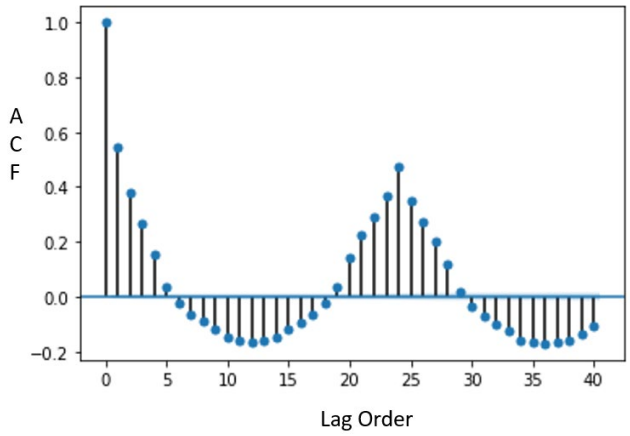
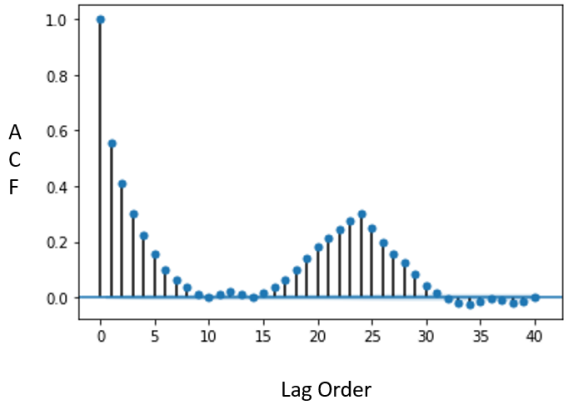


Figure 16 depicts the ACF plot with 40 lags for JFK. It is evident that taxi out times experiences a significant autocorrelation effect from its previous data points. However, the autocorrelation effects changes and remain positive through the lag period.

Figure 16

ACF Plot for Taxi Out Time at JFK



The autocorrelation for time series data points can also be evaluated through PACF. PACF is similar to ACF but only captures the autocorrelation effect of a lagged version of a variable that is not captured by other succeeding lagged versions. PACF is used to understand the unique autocorrelation effect rather than the cumulative autocorrelation effect of a lagged version of the variable. The PACF for the taxi out times variable was computed at a significance of .05 for both airports. Figure 17 depicts the PACF for the taxi out times at MCO with 40 lags. It is evident that the partial correlation of the lagged versions of taxi out times is significant up to 16 data points. However, the partial correlation effect switches from positive to negative as the lag order increases.

Figure 17

PACF Plot for Taxi Out Time at MCO

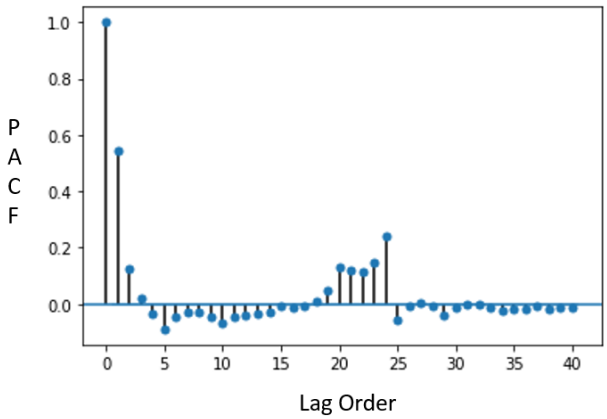
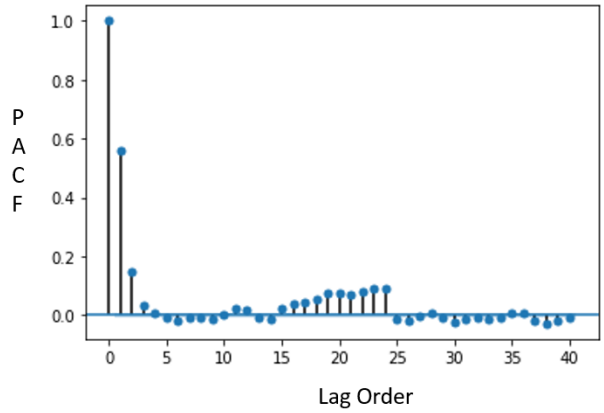


Figure 18 depicts the PACF for taxi out times at JFK with 40 lags. It is evident that the partial correlation of the lagged versions of taxi out times is significant up to a lag order of 16. However, the partial correlation effect switches from positive to negative as the lag order increases. Unlike MCO, the partial autocorrelation for JFK is only significant up to a lag order of 3, and the partial autocorrelation remains positive.

Figure 18

PACF Plot for Taxi Out Time at JFK



A Granger's Causality Test was conducted to analyze the relationship between the time series variables in the data. A Granger's Causality Test can be used to statistically evaluate the causality effect of a time series on another time series. The Granger's Causality Test is used as a statistical hypothesis test to analyze if one time series can be used to predict another time series. Granger's Causality is tested in the context of linear autoregressive models. Granger's Causality for two variables or two-time series X_1 and X_2 can be represented by Equations 20 and 21, where p is the maximum number of lagged observations, A is the matrix containing the coefficients of the model, and E_1 and E_2 are the residuals or prediction errors for each time series.

$$X_1(t) = \sum_{j=1}^{p} A_{11j} X_1(t-j) + \sum_{j=1}^{p} A_{12j} X_2(t-j) + E_1(t) \quad (20)$$

$$X_2(t) = \sum_{j=1}^{p} A_{21j} X_1(t-j) + \sum_{j=1}^{p} A_{22j} X_2(t-j) + E_2(t) \quad (21)$$

Appendix B1 depicts the Python code utilized to create the causality matrix. A Lag of 12 was used for the test. Figure 19 and Figure 20 depict the causality matrix for the data for MCO and JFK respectively. Significant causality between Wind Direction

variables and taxi out times was observed. Additionally, the Weather Type variables demonstrated significant causality with taxi out times as well.

Figure 19

Granger's Causality Test for MCO

	Departures	Arrivals	Altimeter	Dew_Point	Temperature	Precipiaion	PressureChang	HourlyRelativeHumidit	HourlyVisibility	FG	NOSIG	RA	TS	CLM	NE	NW	SE	SW	Taxi_Out
Departures	1	0	0	0	0	0	0	0	0	0	0	0.0017	0	0.0765	0	0.0147	0	0	0
Arrivals	0	1	0	0	0	0	0	0	0	0	0.0145	0	0	0.0006	0	0.0009	0	0.0002	0
Altimeter	0	0	1	0	0	0.0001	0	0	0.0032	0	0	0	0.0067	0.0423	0	0	0.0191	0	0
Dew_Point	0	0	0.0002	1	0	0.018	0.0895	0	0.0008	0	0	0.0007	0	0.1837	0	0	0	0	0
Temperature	0	0	0	0	1	0.0608	0	0	0	0	0.0002	0	0	0.3846	0	0	0	0	0
Precipiaion	0	0	0	0	0	1	0	0.6601	0	0.0008	0	0	0	0.72	0	0.0735	0	0.0005	0
PressureChange	0	0	0	0	0	0	1	0	0	0	0	0.1081	0	0.8681	0	0	0	0	0
HourlyRelativeHumidity	0	0	0	0	0	0	0	1	0.6863	0	0	0	0	0.9788	0	0	0	0	0
HourlyVisibility	0	0.1617	0	0.001	0	0	0	0	1	0	0	0	0	0.1091	0	0.0132	0.0058	0.7802	0
FG	0	0	0.0203	0.0137	0	0.024	0.0001	0	0	1	0.1336	0.3177	0.0013	0.7781	0	0.0043	0	0.2518	0
NOSIG	0.3	0.9634	0	0	0	0	0	0	0	0	1	0	0	0.8231	0.1494	0	0.3181	0	0.5532
RA	0	0	0	0	0.0936	0	0.7501	0	0	0.0011	0	1	0	0.6617	0.0518	0.0174	0.4208	0	0
TS	0	0	0	0	0	0	0	0	0.0195	0	0	0	1	0.9257	0	0.0001	0	0	0
CLM	0.0149	0.3971	0.2533	0.2358	0.0055	0.4837	0.2726	0	0.4027	0.5295	0.2445	0.4632	0.5688	1	0.0549	0.308	0.6012	0.3143	0.7178
NE	0	0.0001	0	0.0032	0	0.2853	0	0	0.0023	0.0069	0.1749	0.7183	0.1258	0.8012	1	0	0	0	0
NW	0.0263	0.9783	0	0	0	0	0.0013	0	0.5098	0.0323	0.0001	0.0367	0.0078	0.075	0	1	0	0	0.0123
SE	0	0	0	0	0	0.0006	0	0	0.0012	0.0048	0.1021	0.1651	0	0.0004	0	0	1	0	0
SW	0.044	0	0	0	0	0.6196	0.0012	0	0.017	0.0008	0.0005	0.0483	0.3374	0.4147	0	0.0094	0	1	0
Taxi_Out	0	0	0.5092	0	0	0	0	0	0.348	0	0	0	0	0.6222	0	0.2197	0	0	1

Figure 20

Granger's Causality Test for JFK

	Departures	Arrivals	Altimeter	Dew_Point	Temperature	Precipiaion	PressureChange	HourlyRelativeHumidity	HourlyVisibility	FG	NOSIG	RA	TS	CLM	NE	NW	SE	SW	Taxi_Out
Departures	1	0	0.4146	0	0	0.1478	0.5153	0	0	0	0	0.5193	0.6825	0	0	0	0.0209	0.003	0
Arrivals	0	1	0.8853	0	0	0.7578	0	0	0	0	0	0.7251	0.9797	0	0	0	0	0.0004	0
Altimeter	0	0	1	0	0	0.4189	0.7218	0	0.3783	0.1593	0.0003	0.0078	0.0321	0	0.3112	0	0	0.9654	0
Dew_Point	0.0087	0.0112	0	1	0	0	0.0006	0	0	0	0	0	0.0921	0	0.8916	0	0	0	0
Temperature	0	0	0	0	1	0.33	0	0	0.0257	0	0	0	0	0	0	0	0	0	0
Precipiaion	0.0475	0.0231	0	0	0	1	0	0	0	0.78	0	0	0	0.0004	0	0	0	0	0
PressureChange	0	0	0.0667	0	0	0	1	0	0	0.0277	0	0	0.1951	0.281	0.815	0	0	0	0
HourlyRelativeHum	0	0	0	0.0029	0	0.2031	0	1	0.0917	0.2245	0	0	0	0.1732	0	0	0.0014	0.6159	0
HourlyVisibility	0.11	0.0981	0	0	0	0	0	0	1	0	0	0.3228	0	0.0064	0	0	0	0	0.0028
FG	0.0001	0.0018	0	0.9422	0	0.0001	0.2633	0	0	1	0	0	0.4626	0	0	0	0.0124	0	0.129
NOSIG	0.0499	0.0618	0	0	0	0.1277	0	0	0	0	1	0	0	0.7955	0	0	0	0	0
RA	0.0612	0.0967	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0.1476
TS	0	0	0	0	0	0.0021	0	0	0.0104	0.5978	0	0	1	0.4883	0.0859	0	0	0.4284	0
CLM	0.2937	0.0156	0	0	0.0003	0.0419	0.0906	0	0.7694	0	0.0008	0	0.0855	1	0.5175	0.8108	0	0	0.0005
NE	0.0006	0.2357	0	0	0	0.1202	0.1835	0	0	0.0335	0.0002	0	0.9354	0	1	0	0	0	0.2646
NW	0.0062	0.0005	0.7355	0	0	0	0	0.0001	0.0673	0.0558	0.0003	0.0642	0	0	0	1	0	0	0.0291
SE	0.1249	0.0208	0	0	0	0	0	0.0027	0	0.3047	0	0	0	0	0	0	1	0.017	0.6601
SW	0	0	0	0.3163	0.0149	0.0138	0	0	0	0.0364	0	0.0001	0.0729	0	0	0.0027	0	1	0
Taxi_Out	0	0	0	0	0	0	0.4002	0	0	0.0117	0	0	0	0	0	0.2255	0.0001	0	1

Model Architecture

A vital component of the study was to develop robust Deep Learning models to predict runway configuration selection and taxi out times. LSTM Encoder-Decoder and Transformer models were used as baseline models, and their associated hyperparameters were tuned to develop the final models. Domain knowledge and trial-and-error were used

to identify the best hyperparameters concerning the training loss and validation loss. The EDA and time series analysis were also used to select and initialize specific hyperparameters such as input and output length and variable section. The EDA was essentially used to analyze the data imbalance. An imbalanced data, either for the independent or dependent variables, can induce bias or lead to overfitting models. The EDA demonstrated that the data was moderately balanced and could be used for model development without any data transformation. The time series analysis was used to understand various time series characteristics, such as the autocorrelation and causality of the multi-variate time series data. The Granger's Causality Test for MCO and JFK demonstrated a significant causality associated with the variables that could be used to model the relationships. The PACF and ACF plots were used to select the input sequence length of 24 for the model development.

LSTM Encoder-Decoder Model

The LSTM encoder-decoder architecture was used for the classification and regression tasks. The initial model was built based on the literature available on the subject and proven techniques in the field of time series forecasting. Figure 15 depicts the LSTM encoder-decoder model architecture used for the classification and regression tasks. Appendix D depicts the coding utilized to develop and evaluate the model. A 1-dimensional convolution layer was used as the input layer with 12 filters, and the kernel size was set to 6. The input layer was followed by a layer of 32 LSTM units. Considering the temporal nature of the task, each layer was set to return the output sequence. After the LSTM layer, a Leaky ReLU layer was added with an alpha of .01, followed by a dropout layer set at 0.3. The dropout layer was added as a regularizer to prevent overfitting.

The encoder section of the model continued with another LSTM layer of 64 LSTM units and another Leaky ReLU and dropout layer. An additional LSTM layer of 64 LSTM units was added. However, the last layer acted as the last layer of the encoder section of the model, so the layer did not return a sequence. To connect the encoder and decoder sections, a repeat vector layer for six time steps was added. The repeat vector for six time steps was added because the desired output sequence consisted of six output values. The decoder layer started with an LSTM layer of 32 LSTM units with ReLU activation, followed by another LSTM layer of 64 LSTM units. For computation efficiency and sequence-to-sequence modeling effectiveness, a TimeDistributed Dense layer of 32 units and ReLU activation function was added.

The LSTM encoder-decoder model ended with the output layer. The output layer was adjusted based on the task specifications. For the regression task, the output layer was a Dense layer of a single unit. For the binary classification task in the case of MCO, a Dense layer of a single unit was added. However, a sigmoid activation function was added to the ouput binary values. For the multi-class classification task in the case of JFK, a Dense layer of four units was added with the softmax activation function.

An LSTM model requires a loss function and optimizer for the training of the model. For the regression task, MSE was used as the loss function and Adam was used as the optimizer. For the binary classification model, Binary Cross Entropy was used as the loss function while Categorical Cross Entropy was used as the loss function for the multi-class classification task. Adam was used as the optimizer for both the classification tasks. Table 3 depicts the model summary for the multi-class classification problem utilizing the

LSTM encoder-decoder model. The LSTM encoder-decoder model consisted of 104,501 parameters. Figure 21 depicts the model plot for the LSTM encoder-decoder model.

Table 3

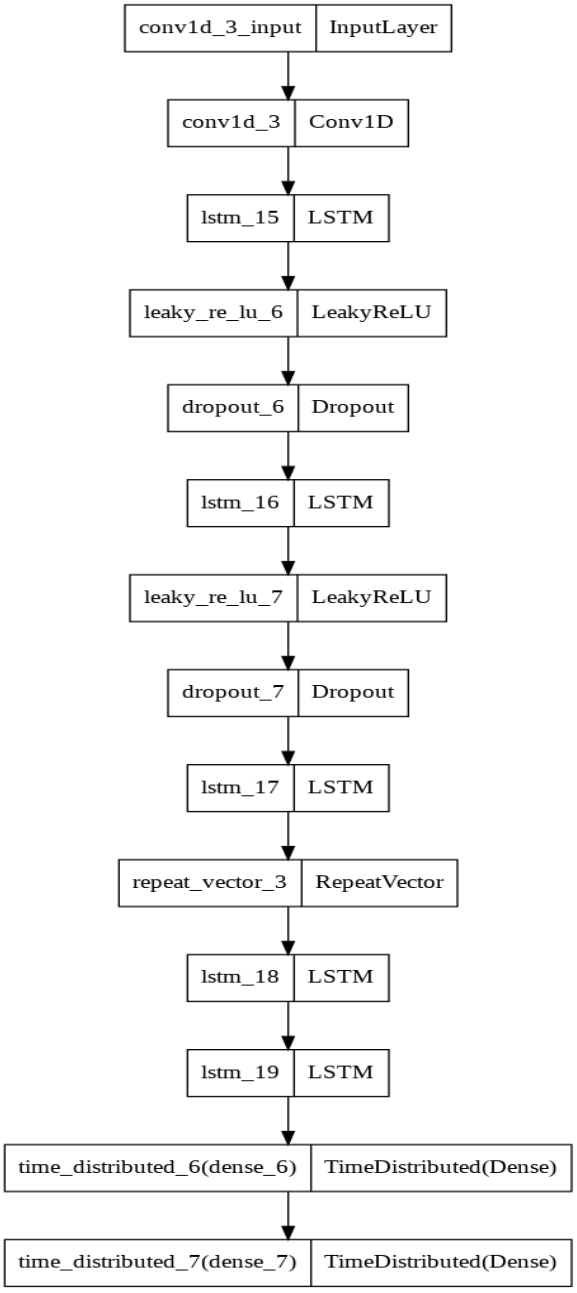
Model Summary for the LSTM Encoder-Decoder Model

Layer	Output Shape	Number of Parameters
1-D Convolutional	(None, 19, 12)	1,524
LSTM-1	(None, 19, 32)	5,760
Leaky ReLU	(None, 19, 32)	0
Dropout	(None, 19, 32)	0
LSTM-II	(None, 19, 64)	24,832
Leaky ReLU	(None, 19, 32)	0
Dropout	(None, 19, 32)	0
LSTM-III	(None, 64)	33,024
Repeat Vectors	(None, 6, 64)	0
LSTM-IV	(None, 6, 32)	12,416
LSTM-V	(None, 6, 32)	24,832
Time Distributed	(None, 6, 32)	2,080
Time Distributed	(None, 6, 32)	33

Note. Total trainable parameters in the model was 104,501

Figure 21

LSTM Encoder-Decoder Model Plot



Transformer Model

The Transformer model was used for classification and regression tasks. To use the Transformer for the time series task rather than a Natural Language Processing task, several aspects of the traditional Transformer architecture needed to be adjusted to fit the requirements of the task. The input and output sequence data was identical to that used for the LSTM encoder-decoder model.

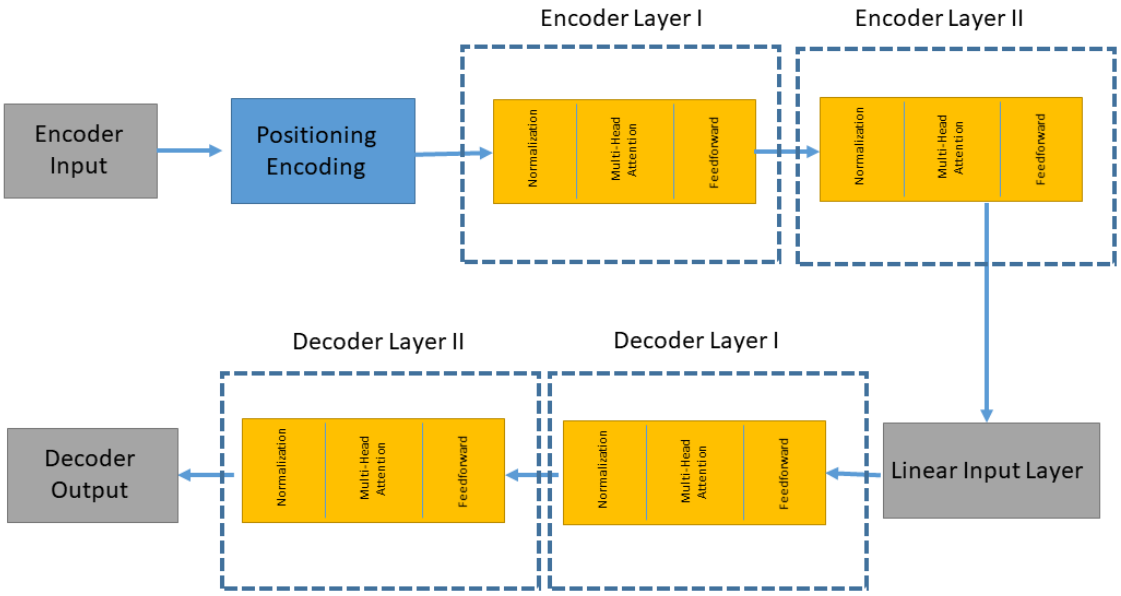
Unlike the LSTM encoder-decoder model, the Transformer model cannot be directly created as a function of Tensorflow. Several functions needed to be coded to develop the final Transformer model. The encoder section of the model was first developed with the addition of normalization, multi-head attention, and dropout layers. Epsilon is a hyperparameter added to the normalization layer, which is a variance that is added to avoid the division of any term by 0. The epsilon was set to 1×10^{-6}. The feedforward section of the encoder consisted of a normalization layer, a 1-dimensional convolutional layer, and a dropout layer. The convolutional layer consisted of 12 filters with a kernel size of 3 and ReLU as the activation function. A final dense layer was added to complete the encoder sequence.

The decoder layer was used to generate the desired output sequence. The decoder layer was initiated using a simple linear dense layer followed by multi-head attention and feedforward layers. For simplicity, the hyperparameters for the encoder and decoder layers were kept similar, if not the same. The encoder and decoder consisted of two multi-head attention layers and two feedforward layers. The head size for the multi-head attention layers was set to 32 for all the layers. The dropout was set to 0.3, and the number of transformer blocks was set to 4.

The transformer model utilized Adam as the optimizer and MSE for the loss function for the regression tasks. For the binary classification task and multi-class classification task, binary cross-entropy and categorical cross-entropy were used as loss functions respectively. Early stopping with a patience of 10 was set as a regularizer, and the model was trained on batch sizes of 64. Figure 22 is a pictorial representation of the Transformer model developed for the study. Appendix E depicts the coding used to develop the models.

Figure 22

Transformer Model Architecture



Model Evaluation

The LSTM encoder-decoder and Transformer models were evaluated separately for the regression and classification tasks. The taxi out time prediction task was treated as regression tasks, and the models were evaluated on MSE, RMSE, R-squared, and MAE.

Taxi Out Time Prediction

The LSTM encoder-decoder and Transformer models were developed to predict the taxi out times for MCO and JFK. Additionally, using Repeat Vectors and Time Distributed layers, an output sequence of six continuous values was generated that represented taxi out times prediction for a six-hour outlook. Utilizing a customized query, each sequence component could be used as a variable to be evaluated against the actual value of the sequence to compute the MSE, RMSE, R-Squared, and MAE. Table 4 depicts the model evaluation parameters for the taxi out time prediction for MCO and JFK. Figure 23 and Figure 24 depict the loss curve measured by MSE for the training and validation sets for MCO and JFK respectively.

Based on the evaluation results, the LSTM encoder-decoder models performed slightly better than the Transformer model for MCO and JFK. There was an improvement in model performance from Sequence 1 to Sequence 2 and then a degradation in performance for the LSTM encoder-decoder and Transformer models. The best prediction performance was exhibited by the LSTM encoder-decoder model for MCO to predict the Sequence 2 taxi out time. Based on the Loss Curves for MCO and JFK, the training was stopped once the validation loss stopped reducing due to the early stopping used as a regularizer. Due to the difference in the training and validation loss, there is an observable scope for overfitting in both models.

Table 4

Model Evaluation Parameters for the Taxi Out Time Prediction

		LSTM Encoder-Decoder		Transformer	
Parameter	Sequence	MCO	JFK	MCO	JFK
MSE	1	43.12	46.52	47.23	49.52
	2	41.26	45.82	47.19	48.81
	3	43.26	45.99	49.27	49.58
	4	44.89	47.25	51.34	49.57
	5	47.84	50.15	54.29	50.02
	6	51.29	52.52	58.92	51.03
R-Squared	1	0.57	0.49	0.52	0.48
	2	0.61	0.51	0.52	0.49
	3	0.59	0.50	0.48	0.48
	4	0.56	0.46	0.47	0.48
	5	0.47	0.45	0.47	0.47
	6	0.43	0.39	0.45	0.47
RMSE	1	6.57	6.82	6.87	7.04
	2	6.42	6.76	6.86	6.98
	3	6.57	6.78	7.02	7.04
	4	6.78	6.87	7.16	7.05
	5	6.92	7.08	7.36	7.07
	6	7.16	7.24	7.67	7.14
MAE	1	3.92	4.21	4.31	4.38
	2	3.84	4.13	4.29	4.31
	3	3.97	4.52	4.38	4.42
	4	4.12	4.84	4.41	4.51
	5	4.22	5.26	4.43	4.50
	6	4.56	5.46	4.52	4.57

Figure 23

Taxi Out Time Loss Curve for MCO

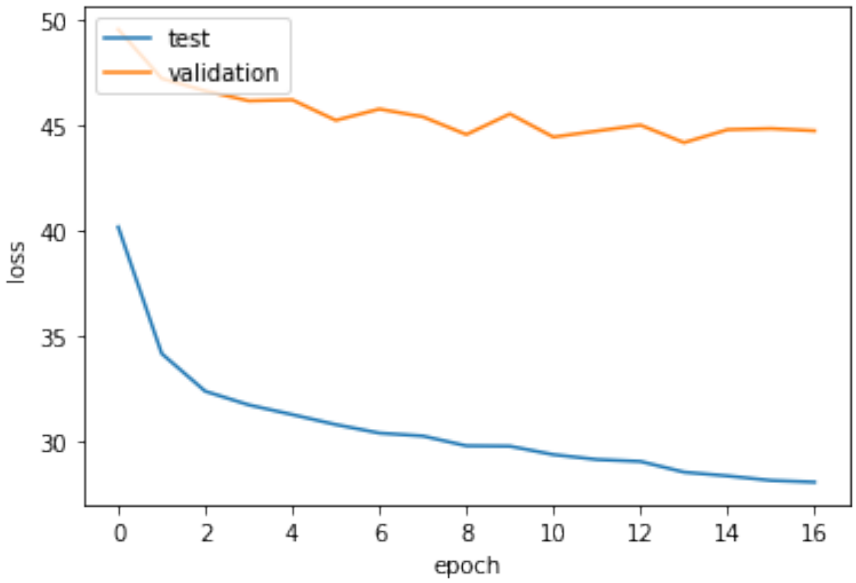
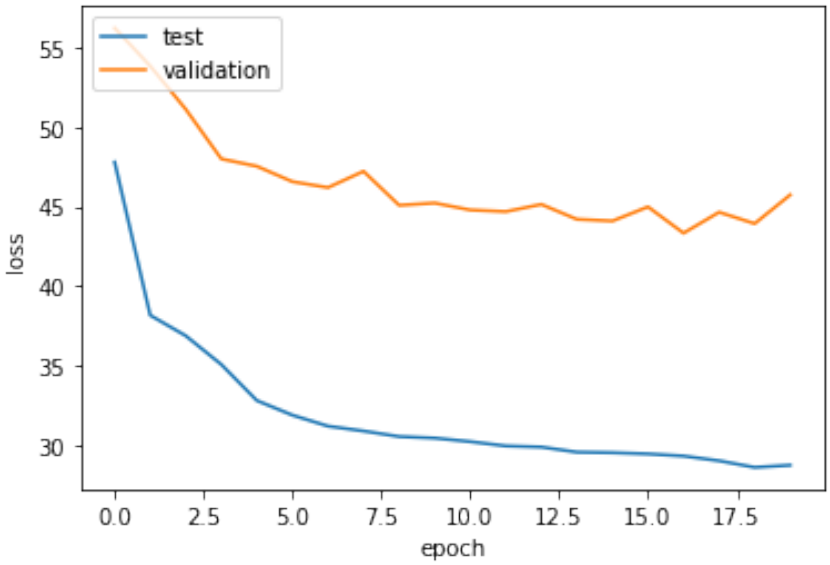


Figure 24

Taxi Out Time Loss Curve for JFK



Runway Configuration Selection Prediction

The LSTM encoder-decoder and Transformer models were developed to predict the runway configuration selection for MCO and JFK. Considering the runway configuration selection prediction task was a classification task, the output layer needed to be modified for the LSTM encoder-decoder and transformer models.

For the binary classification task for MCO, a single cell in the output layer with the sigmoid activation function was used. For the multi-class classification task for JFK, four cells in the output layer with Softmax activation function were used. Additionally, for the multi-class classification task, the labels needed to be one-hot encoded before the model development. Using Repeat Vectors and Time Distributed layers, an output sequence of six values was generated that represented runway configuration selection prediction for a 6-hour outlook. Utilizing a customized query, each component of the sequence could be used as a variable to be evaluated against the actual value of the sequence to computer the accuracy, precision, recall, and kappa score. Table 5 depicts the model evaluation parameters for the runway configuration prediction for MCO and JFK. Figure 25 and Figure 26 depict the loss curve measured by cross entropy for the training and validation sets for MCO and JFK respectively.

Unlike the taxi out time models, the LSTM encoder-decoder and Transformer models have comparable performance for the runway configuration selection task. For the LSTM encoder-decoder model, the best prediction performance is observed in Sequence 3, while the best prediction performance for the Transformer model is observed in Sequence 3 and Sequence 4. The training loss decreased with increasing epochs for MCO and JFK. However, due to early stopping used as a regularizer, the training was

stopped due to a lack of improvement in validation set performance. Despite utilizing early stopping as a regularizer, the loss curves indicate overfitting for both models.

Table 5

Model Evaluation Parameters for the Runway Configuration Selection Prediction

		LSTM Encoder-Decoder		Transformer	
Parameter	Sequence	MCO	JFK	MCO	JFK
Accuracy	1	78.26	76.54	72.32	73.24
	2	79.25	77.12	72.18	71.25
	3	80.24	77.24	71.26	73.54
	4	78.56	74.26	70.15	69.54
	5	73.45	71.54	66.54	68.24
	6	70.15	66.25	62.38	63.58
Precision	1	0.78	0.77	0.76	0.76
	2	0.79	0.78	0.76	0.76
	3	0.70	0.78	0.78	0.80
	4	0.70	0.80	0.82	0.79
	5	0.66	0.78	0.80	0.78
	6	0.64	0.76	0.80	0.78
Recall	1	0.83	0.82	0.78	0.77
	2	0.84	0.80	0.78	0.78
	3	0.72	0.78	0.80	0.78
	4	0.72	0.84	0.84	0.80
	5	0.72	0.78	0.82	0.78
	6	0.60	0.76	0.80	0.76
Kappa	1	0.54	0.54	0.56	0.58
	2	0.56	0.52	0.54	0.53
	3	0.52	0.51	0.56	0.58
	4	0.50	0.56	0.60	0.56
	5	0.51	0.52	0.58	0.56
	6	0.46	0.50	0.53	0.54

Figure 25

Runway Configuration Selection Loss Curve for MCO

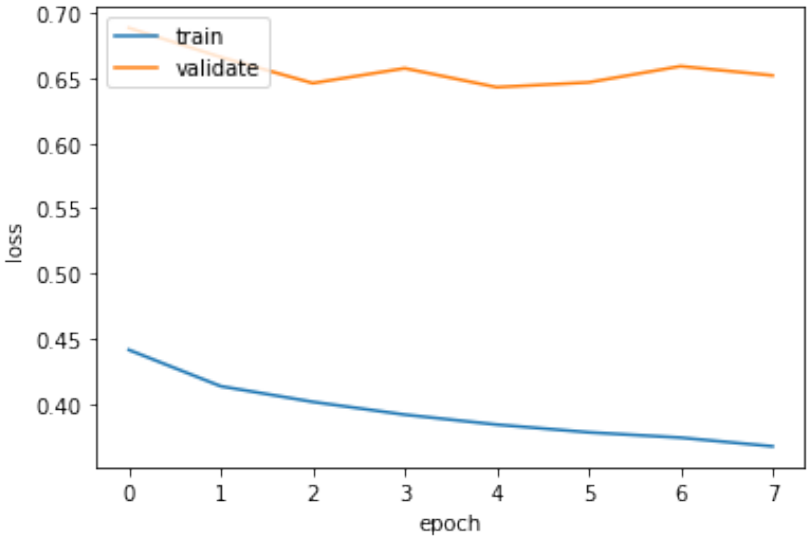
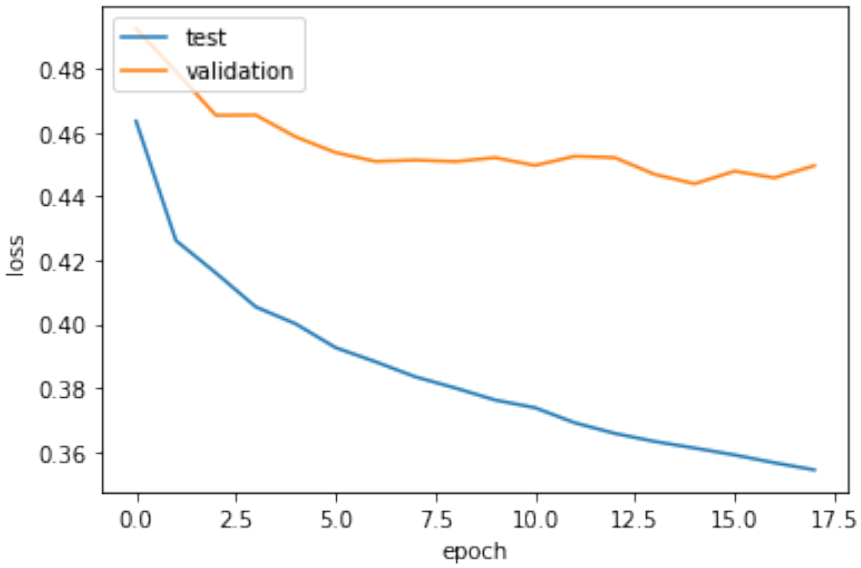


Figure 26

Runway Configuration Selection Loss Curve for JFK



To analyze the classification model performance, a confusion matrix was developed for the testing set. Considering the LSTM encoder-decoder and Transformer models predicted six sequences for two airports, a total of 24 confusion matrices could be

created. However, only the best-performing sequence for MCO and JFK was chosen to develop the confusion matrix. Figure 27 depicts the confusion matrix for Sequence 2 for MCO. The runway configuration selection prediction task for MCO was a binary classification task with two possible outcomes (Table 2). There was a better predictive performance observed for predicting the South runway configuration. Figure 28 depicts the confusion matrix for Sequence 3 for JFK. The runway configuration selection prediction task for MCO was a multi-class classification task with four possible outcomes (Table 2). The best predictive performance was observed for predicting the North runway configuration followed by predicting the West runway configuration.

Figure 27

Confusion Matrix for MCO

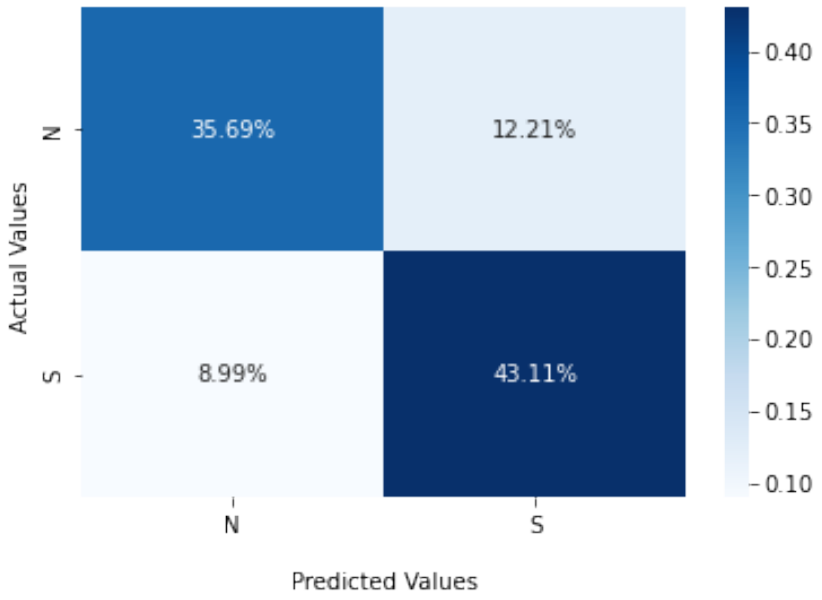
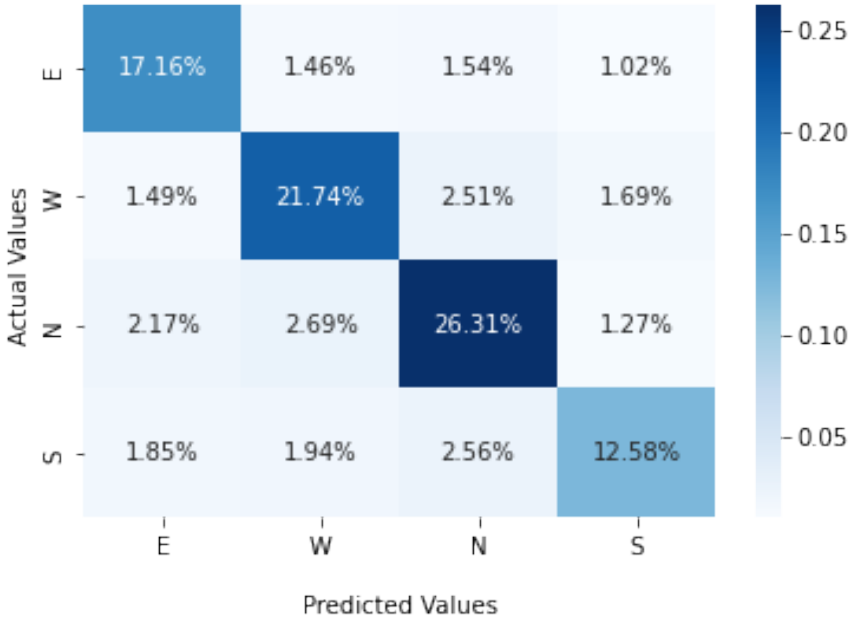


Figure 28

Confusion Matrix for JFK

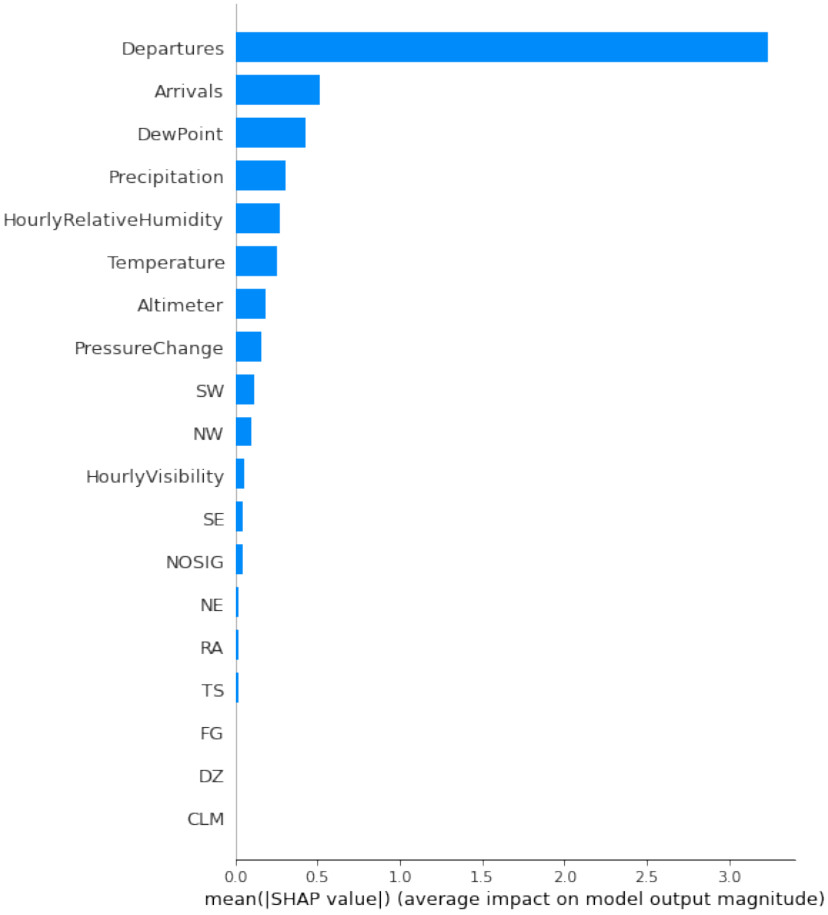


Model Interpretation

SHAP was used to interpret the models to get a better insight regarding the predictors or factors that significantly influenced the predictions of the Deep Learning models developed in this study. The SHAP library on Python was used to conduct the SHAP analysis. Due to computation feasibility, only the best-performing models were used for the SHAP analysis. Additionally, like the loss functions used in the model development, all output sequences were evaluated with the same weights. Figure 29 depicts the mean absolute SHAP values for the taxi out time prediction model at MCO. The number of departures had the highest impact on the model predictions, followed by the number of arrivals.

Figure 29

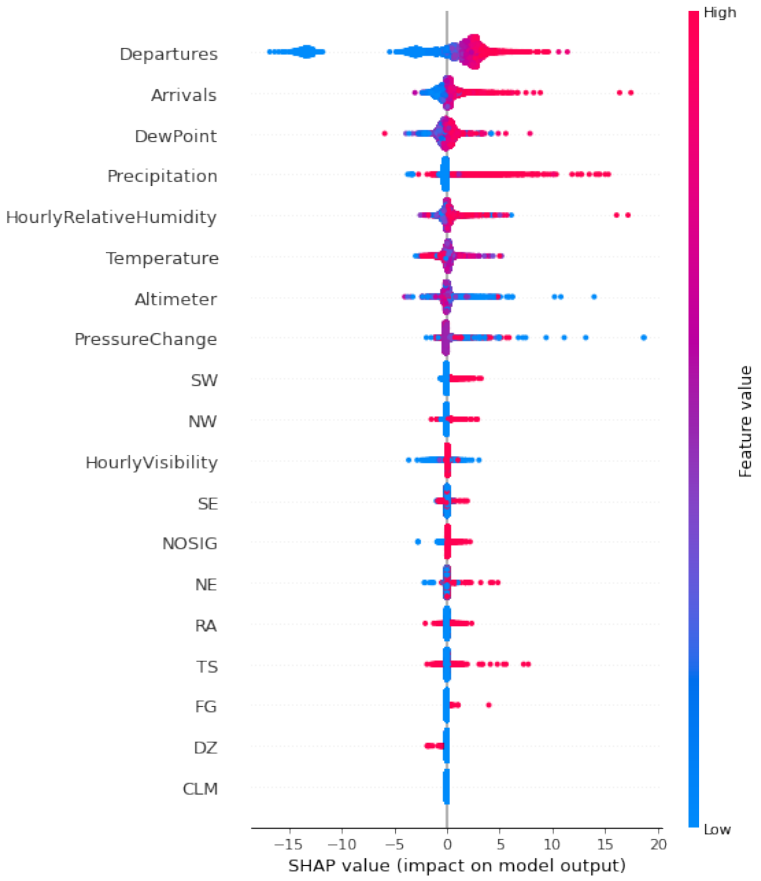
Mean Absolute SHAP Values for Taxi Out Times at MCO



SHAP values can be positive or negative depending on their impact on the model prediction. Figure 30 depicts the magnitude of the SHAP values of the different Independent Variables for the taxi out time prediction model at MCO and their impact on the model predictions. The number of departures had significantly high positive and negative SHAP values, while number of arrivals has a high positive SHAP value, but a low negative SHAP value.

Figure 30

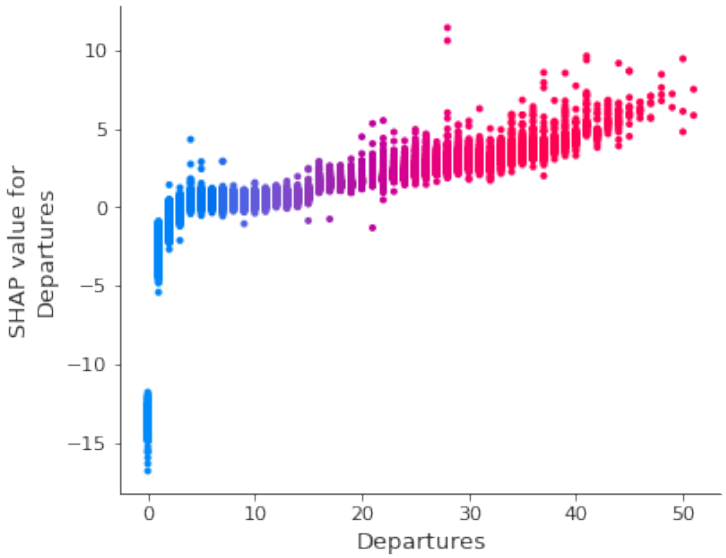
SHAP Values for Taxi Out Times at MCO



A SHAP Dependence plot can be used to evaluate the effect of a variable value on the SHAP value over the entire dataset. Figure 31 depicts a SHAP Dependence plot for the Departures variable for the taxi out times prediction model at MCO. A positive relationship between the number of departures and SHAP value is observed, which implies that as the number of departures increases, the impact of departures compared to other variables in the dataset increases on the model predictions.

Figure 31

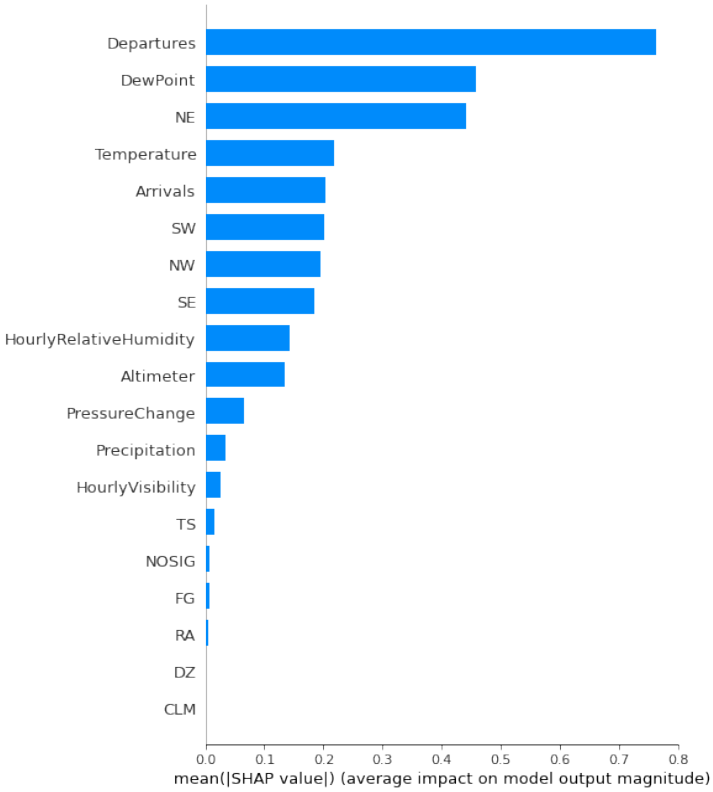
SHAP Dependence Plot for Taxi Out Times at MCO.



A SHAP analysis could be conducted for the runway configuration selection model at MCO too. Figure 32 depicts the mean absolute SHAP values for the binary classification model. Just like the taxi out times model, Departures has the highest mean absolute SHAP value, but the magnitude of the SHAP value is lower. There is a more significant impact of other variables, such as Dew Point and Wind Direction (NE, NW, SE, and SW) on the predictions made by the model.

Figure 32

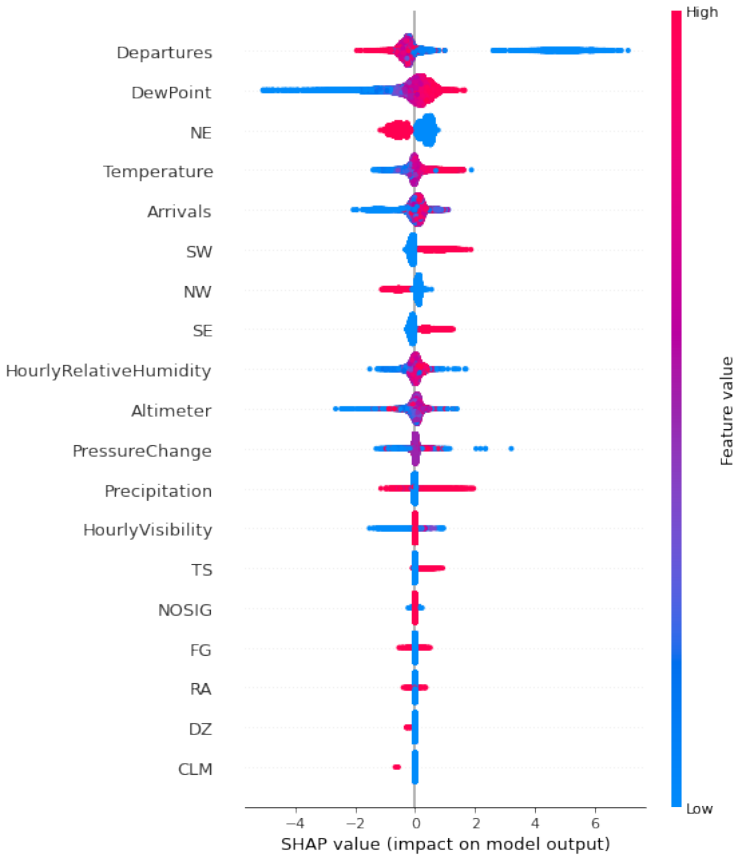
Mean Absolute SHAP Values for Runway Configuration Selection at MCO



The SHAP values for the runway configuration selection prediction models demonstrate a higher balance in terms of the impact of different variables on the model predictions as compared to the SHAP values for the taxi out times prediction model. Figure 33 depicts the SHAP values for the different independent variables for the runway configuration selection prediction model at MCO.

Figure 33

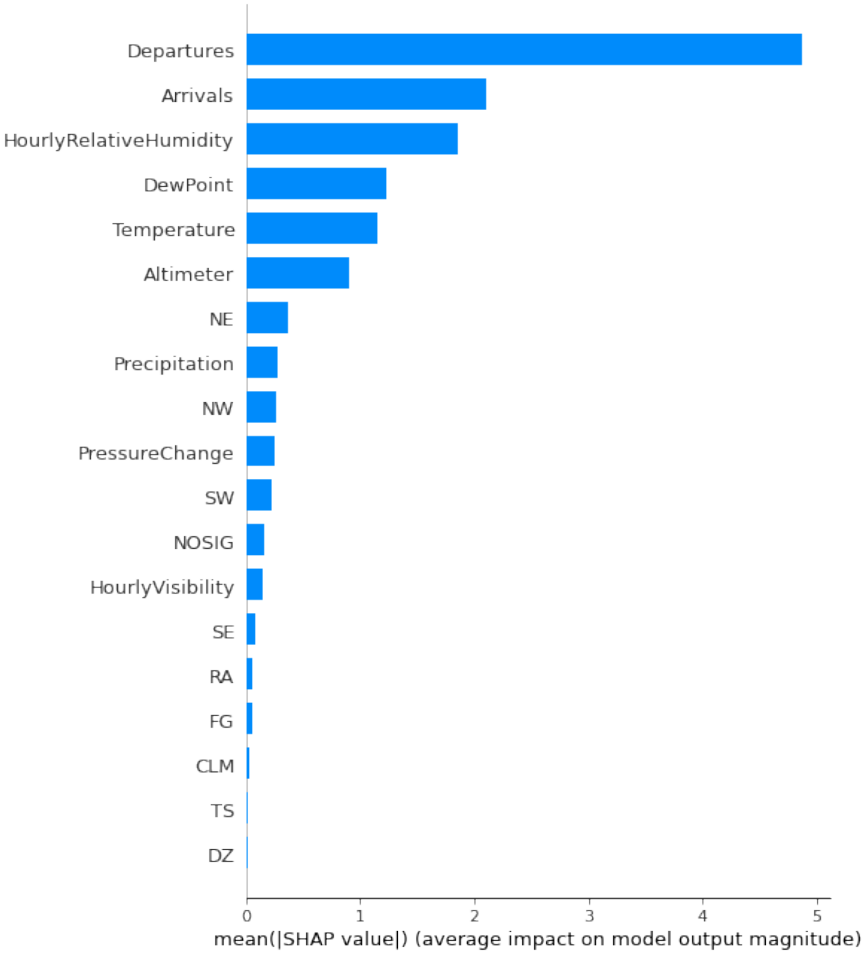
SHAP Values for Runway Configuration Selection at MCO



The SHAP analysis was conducted for the taxi out times and runway configuration selection prediction models at JFK too. Just like the taxi out times prediction model for MCO, Departures had the highest impact on the predictions of the model, followed by Arrivals. Figure 34 depicts the mean absolute SHAP values for the taxi out times prediction at JFK.

Figure 34

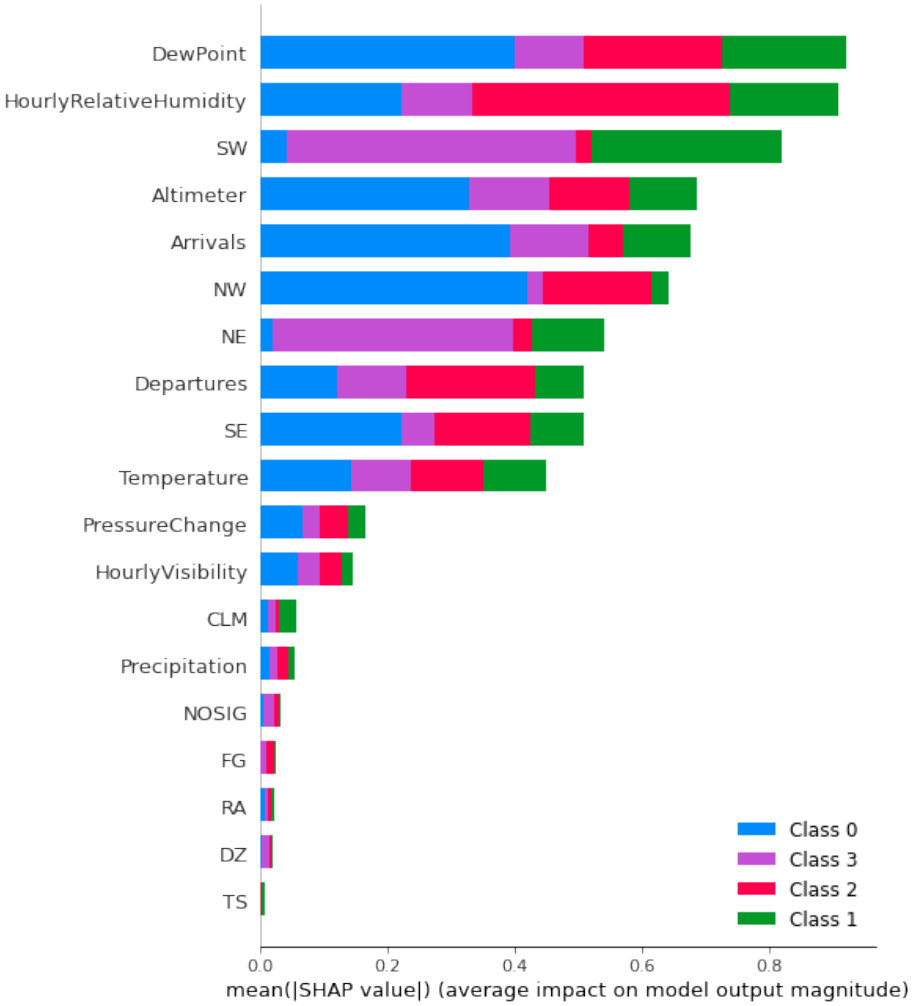
Mean Absolute SHAP Values for Taxi Out Times at JFK



The runway configuration selection task for JFK was a multi-class classification task with four possible outputs. A SHAP analysis was conducted for the multi-class classification task as well. Figure 35 depicts the mean absolute SHAP values for the runway configuration selection model at JFK. The mean absolute SHAP values were also classified by the four one-hot encoded classes. Dew Point had the highest mean absolute SHAP values, followed by Relative Humidity. The runway configuration selection SHAP values for the multi-class classification values differed from the runway configuration selection SHAP values for the binary classification problem.

Figure 35

Mean Absolute SHAP Values for Runway Configuration Selection at JFK



Note. For the SHAP analysis, E, N, S, and W were treated as Class 0, Class 1, Class 2, and Class 3 respectively.

Considering the Dew Point and Relative Humidity had the highest SHAP values, the SHAP Dependence plots were analyzed for the Dew Point and Relative Humidity variables. Figure 36 and Figure 37 depict the SHAP Dependence plot for Dew Point and Relative Humidity at JFK respectively. The SHAP Dependence plot can be developed for each class by treating it as a binary classification task. As evident in Figure 36, the SHAP

score for Dew Point is negatively skewed around 0 to -2 when the Dew Point temperature is less than 20 °C. Such a SHAP analysis can be extended to other classes presented in the multi-class classification task. Similarly, the SHAP scores for Relative Humidity are negatively skewed as well, but the curve flattens after the Relative Humidity rises over 50. Extremely low SHAP values are observable at low Relative Humidity values.

Figure 36

SHAP Dependence Plot for Dew Point at JFK

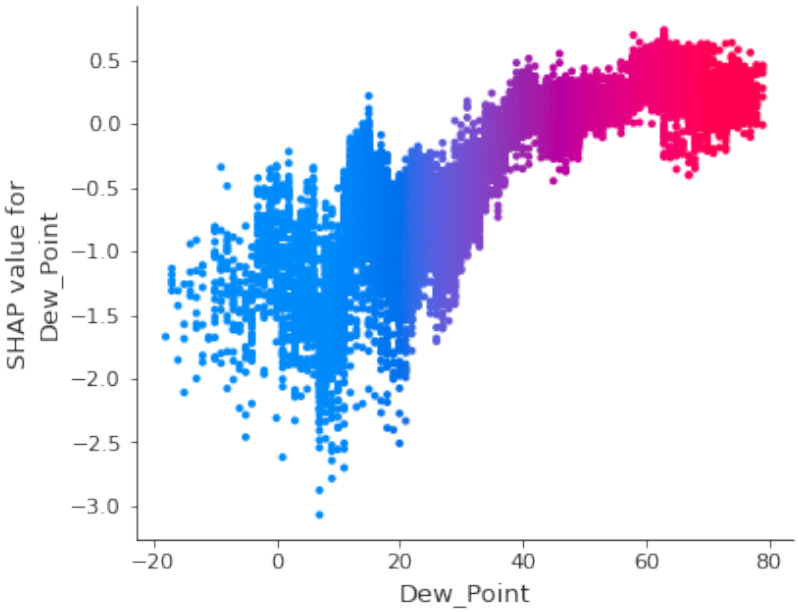
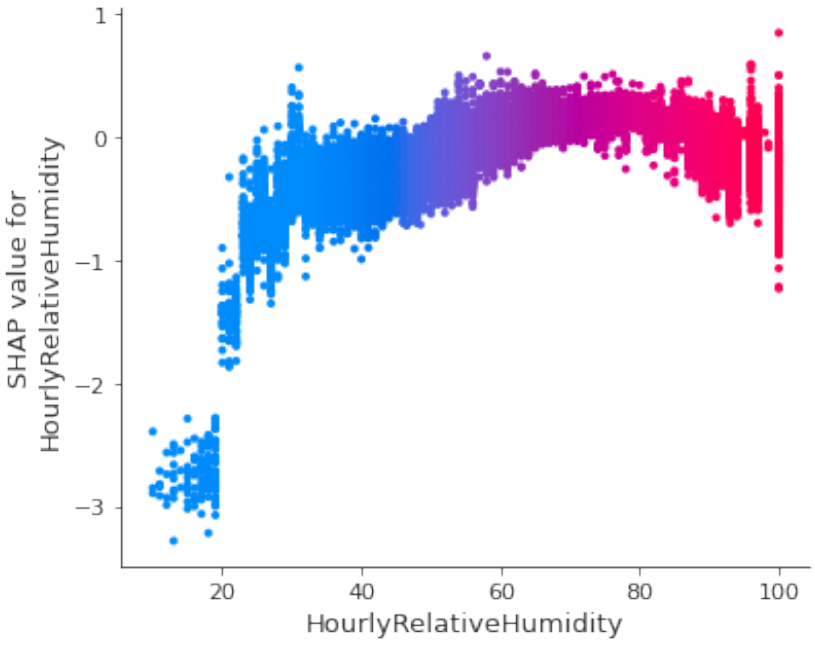


Figure 37

SHAP Dependence Plot for Relative Humidity at JFK



Summary

Chapter IV presented the results of the study. Firstly, an EDA was conducted to analyze and understand the data that was used for the study. Aspects of the data that were analyzed included the data imbalance of the dependent variable, distribution of the dependent variable, autocorrelation effect for the dependent variable, and the causality and correlation between the variables. The EDA focused on not only understanding the relationship between the variables used in the study, but also understanding the autocorrelation of variables due to the temporal nature of the data. The Granger Causality Test demonstrated the causality between the different time-series variables used in the study.

Based on the data collected, the LSTM encoder-decoder and Transformer models were developed. The final model architectures for the LSTM encoder-decoder and

Transformer models were described, including the hyperparameters, loss function, and optimization algorithms. The taxi out times prediction models were evaluated on MSE, MAE, RMSE, and R-Squared values and the runway configuration selection prediction models were evaluated on the accuracy, precision, recall, and Kappa score values.

While the LSTM encoder-decoder and Transformer models demonstrated comparable performance, the LSTM encoder-decoder models demonstrated better model evaluation scores for the taxi out times prediction task. The best model evaluation scores were observed in output Sequence 2 for the taxi out times prediction model. The LSTM encoder-decoder models also outperformed the Transformer model on most of the model evaluation scores for the runway configuration selection task. The best model evaluation scores were observed in output Sequence 3 for the runway configuration selection prediction model.

The SHAP method was used to interpret the models. Based on the model performance, the LSTM encoder-decoder models were interpreted utilizing the SHAP plots. For the taxi out times prediction model, Departures and Arrivals were the most significant variables for the prediction models. For the runway configuration selection prediction model, Departures, Dew Point, and Wind Direction-related variables were determined to be the most significant variables for the prediction models.

Chapter V: Discussion, Conclusions, and Recommendations

The purpose of the study is to develop Deep Learning sequence-to-sequence models to predict taxi out times and runway configuration selection based on hourly surface weather variables. This chapter discusses the key takeaways of the study based on the literature reviewed and background, the methodology used for the study, and the results of the study. Additionally, several conclusions from the study will be discussed and will be used to state the theoretical and practical contribution of the study. Finally, the results will be used for recommendations and discuss areas for further research.

Discussion

A significant focus of the study was to develop Deep Learning models to predict taxi out times and runway configuration selection. A review of related literature indicated that models to predict taxi out times and runway configuration selection had been developed before. However, models developed before have varied in different aspects, such as the model algorithms used, variables used for model development, and Machine Learning architectures used for model development. However, a gap in the literature that this study aimed to focus on was the utilization of sequence-to-sequence models to predict a sequence of taxi out times and runway configuration selection values rather than just a singular vector or data point. Considering the data was structured as a time series dataset with an hourly resolution, each output sequence data point would correspond to an hourly prediction. Based on the literature reviewed, such a model would be helpful for users as it would allow the users to not just predict values for the next time period but also predict values for the following six periods.

Deep Learning models that have demonstrated success in sequence-to-sequence learning were used. Two novel architectures, LSTM encoder-decoder and Transformers, were used. The model development required several hyperparameter tuning, which was done primarily on trial and error. A baseline model was initially used for training and testing. Several hyperparameters such as the number of layers, number of neurons in each layer, and activation functions were tuned to develop the final model. A significant focus of further research could be on further hyperparameter tuning utilizing other methods such as Keras Tuner, GridSearch, and Random Search. The lack of optimal hyperparameters used is a significant challenge for Deep Learning model development and is a limitation while assessing and comparing model performance.

The predictions were evaluated based on the testing data for MCO and JFK. For the taxi out times prediction task, the LSTM encoder-decoder model performed better than the Transformer model. However, the regression performance for both models can be considered modest, considering the highest R-Squared value observed for Sequence 2, which implies that the model was only able to capture approximately 61% of the variance in the data. Such a medium R-Squared value in a multi-variate task can be attributed to either poor model performance, high out-of-sample error for the model, or poor selection of features. Based on the Loss Curve plotted for the models (figures 22–25), it is apparent that the models were overfitting due to the divergence of the training and validation curves and the difference between the training and validation loss. While multiple regularizers, such as dropout layers and early stopping, were used, the effects of overfitting were observed in the model performance. The feature assessment technique used for this study, SHAP values, indicated that the number of hourly departures and

number of hourly arrivals were the most significant factors that influenced the predictions. The results are intuitive as a higher number of departures and arrivals per hour in an airport would lead to higher congestion which could impact the taxi out times.

Similar to the approach adopted for the taxi out times prediction, LSTM encoder-decoder and Transformer models were used to develop models to predict the runway configuration selection. The baseline models had to be adjusted to output binary and categorical values rather than continuous values. For the binary classification task for MCO, the LSTM encoder-decoder model performed better than the Transformer model, with the best performance observed for Sequence 3. However, the performance of the Transformer and LSTM encoder-decoder models was almost the same for the multi-class classification task for JFK. The SHAP analysis demonstrated the significant influence of Dew Point, Departures, and wind direction-related variables on the runway configuration selection predictions. It is important to note that due to the binarization process of the Wind Direction variable, the effect or influence of each wind direction was individually assessed rather than the cumulative effect of all the wind directions.

Based on the domain understanding, the possible role of the Departures and Wind Direction variables can be understood. However, the effect of Dew Point on runway configuration selection would be a case for further analysis and research on feature engineering. The main focus of the study was the development of the LSTM encoder-decoder and Transformer models for the use cases in this study which warrants a more detailed discussion on the architecture and performance of those models.

LSTM Encoder-Decoder Model

The LSTM encoder-decoder models developed in this study were based on baseline models published in related literature. Hyperparameter tuning was conducted primarily through monitoring the validation loss and trial-and-error. The final LSTM encoder-decoder model developed can be considered a deep model with 13 layers that included convolutional, LSTM, dropout, repeat vector, and time-distributed layers. The final model contained 104,501 trainable parameters and will require robust computational power during deployment. The computational efficiency of the LSTM encoder-decoder model was not a focus of the study and can be considered a limitation and area of future research. The performance of the LSTM encoder-decoder model can be attributed to a large number of baseline models available in the literature and the different types of algorithms that were used based on recommended practices.

Transformer Model

A large amount of available literature for Transformer models focused on the utilization of the model for Natural Language Processing tasks such as language translation. The utilization of Transformers for time-series tasks can be considered relatively recent, which highlights the potential for further research and development to create baseline models. Due to the scope of this study, only baseline models were utilized for hyperparameter tuning. With the limited literature on the subject, it is expected that the performance of the Transformer model was affected by the lack of optimal hyperparameters and the lack of baseline models used in this study. The poor performance of the Transformer models cannot be attributed to the inability of the Transformer models to model time-series or sequential data, as such models have

demonstrated better performance than LSTM encoder-decoder models in similar sequential data use cases.

The Transformer model developed in this study can be considered rudimentary compared to the LSTM encoder-decoder model, with only two layers each in the encoder and decoder sections. Standard and recommended practices of utilizing positional encoding followed normalization, multi-head attention, and feedforward neural layers were used for developing the model. The output layers were adjusted based on the continuous, binary, or categorical outputs desired for the taxi out times and runway configuration selection models. Further research on the hyperparameter tuning and model development for the Transformer models is expected to add valuable literature on the subject.

Conclusions

Deep Learning models, specifically LSTM encoder-decoder and Transformer models, were developed, validated, and tested to predict taxi out times and runway configuration selection for MCO and JFK. The significance of the study was highlighted by the research gap, and domain needs to be identified from the literature. Based on statistical tests such as the Augmented Dickey Fuller Test to test stationarity, ACF and PACF to identify autocorrelation, and the Grangers Causality Test to identify causality between the time series variables, the dataset used for the model development was assessed to be appropriate to capture and model temporal relationships.

For the taxi out time prediction task, the LSTM encoder-decoder model performed better than the Transformer model for MCO and JFK. Out of the six output sequences, Sequence 2 demonstrated the best performance for JFK. The SHAP analysis

demonstrated that the Departure and Arrival variables had the most significant influence on the predictions.

For the runway configuration prediction tasks, the LSTM encoder-decoder model performed better than the Transformer model for the binary classification task at MCO. The LSTM encoder-decoder model and the Transformer model had a comparable performance for the multiclass classification task at JFK. Out of the six output sequences, Sequence 3 demonstrated the best performance for JFK. The SHAP analysis demonstrated that the Departure, Dew Point, and Wind Direction variables had the most significant influence on the predictions.

Theoretical Contributions

The study contributed to the literature in several ways. Firstly, the study contributed to the literature on the impact of weather variables on taxi out times and runway configuration selection. Utilizing Deep Learning models, the relationship between different weather variables and taxi out times could be modeled and quantified through a deployable model. Additionally, utilizing a SHAP analysis, the models could be interpreted to understand the modeled relationships better. Secondly, the study contributed to the literature on the use of time series modeling applications in aviation. The literature reviewed for this study suggested that time series modeling techniques such as linear autoregressive and moving average have been utilized for different aviation prediction tasks. However, this study added to the body of literature by highlighting additional use cases of time series Deep Learning models that can be used for various aviation applications.

The study also contributed to the literature on sequence-to-sequence modeling. Previous literature on predicting taxi out times and runway configuration selection treated the dependent variable as a single data point or vector. The advancement of sequence-to-sequence techniques in Deep Learning has allowed users to predict a sequence of outputs with some sequential relationship between them. The methodology and results of this study are novel because it demonstrates an additional use case of sequence-to-sequence modeling in aviation. Finally, the study also contributed to the literature on the different applications of LSTM and Transformer models. As reviewed in the literature, the advancement of Deep Learning is rapidly progressing, with scholars researching and developing different model architectures and techniques to improve feasibility and performance. While this study did not contribute to developing any new or improved model architecture or technique, the study did contribute towards the applications research aspect of Deep Learning research by utilizing and testing the existing models in an aviation use case.

Practical Contributions

The results of the study have various practical contributions for different stakeholders in the aviation industry, including airlines, ATC, airports, and regulatory bodies. The literature review suggested the significant effect of different runway configurations on the operations at an airport including the effects on taxi times, capacity restrictions, and ground delays. Previous literature on utilizing predictive modeling has focused on accurately forecasting runway configurations as a tool for resource allocation and forecasting. Similarly, forecasting taxi out times at an airport can help airlines and

airports accurately predict airport capacity restraints and ground delays to help mitigate such delays in advance and efficiently utilize airport resources.

A runway configuration selection and taxi out times forecasting model, such as the models developed in this study, can aid an airline and airport management in predicting taxi out and runway configurations at airports and determining the most significant weather-related predictors. The runway configuration selection and taxi out times prediction models will allow airline managers to make better informed short-term operations decisions such as block fuel and contingency fuel planning along with resource and gate allocations.

Limitations of the Findings

The findings of this study have several limitations that need to be considered. The performance of Deep Learning models is significantly affected by the feature engineering or selection process used for their development. While the literature reviewed suggested significant features or variables that demonstrated success in predicting taxi out times and runway configuration selection in previous studies, the features used in this study for the development of the models was restricted by the availability and access to data. The data used in this study was limited to two public access databases managed by the FAA and NOAA. Previous studies reviewed utilized data from different sources, such as ARINC and ASDE-X, that were not accessible to the researcher for this study (Diana, 2018; Lee et al., 2016). Features that might have improved the performance of the models and have significant relationships to the dependent variables used in the study might not be included due to the lack of availability of data. Additionally, the model development utilized a large number of features. Utilizing a large number of features can lead to a

compromise on the performance and the generalization of the model predictions. Dimension reduction techniques are utilized to cope with, such issues, which are commonly known as the Curse of Dimensionality. This study did not utilize any dimensionality reduction techniques due to the scope and objective of the study. Techniques commonly used for dimension reduction, such as Principal Component Analysis, Truncated Singular Value Decomposition, and Latent Discriminant Analysis, were not used for the study and can be considered a limitation of the study.

The critical component of the model development for Deep Learning models is the loss function used to train the model. The models developed in this study utilized a time series multivariate input to predict an output sequence. The loss function considered every sequence output with equal weightage or importance which could lead to sub-optimal optimization while training the model. Similarly, the loss curve also considered each output of the sequence with equal weightage and did not consider any sequential dependency. Hence, the loss curve cannot be directly interpreted to evaluate the model performance. There is a need to develop a more domain-specific loss function or adjust the currently utilized loss function to improve the model performance interpretation and the optimization of the model training.

The development and training of a Deep Learning model significantly depends on the hyperparameter tuning method or strategy used. The LSTM encoder-decoder and Transformer models were developed based on baseline models that were available in published literature for similar use cases. Additionally, the hyperparameters were manually tuned based on previous knowledge and trial and error. The manual process of hyperparameter tuning can be considered a limitation of the study as this could have led

to sub-optimal hyperparameters that could have affected the performance of the models. Finally, the study did not consider the computational power and efficiency of the models in the study. If the models are deployed for commercial use, the efficiency, data flow, and computational feasibility of model deployment need to be considered as well.

Recommendations

While the results can be considered novel and have theoretical and practical contributions, there are several recommendations proposed for further research on the topic. The recommendations are targeted towards regulatory authorities including the FAA, airlines, and researchers in the Machine Learning community.

Recommendations for Regulatory Authorities

While the model development and testing focused on just two airports in the United States, the methodology and results of the study are expected to benefit stakeholders in the aviation industry around the world. The development of the models in this study was possible due to the availability and quality of data available to the researcher. Data utilized in this study included surface weather observations and airport traffic data. The models could be expanded to include data related to gate allocations, gate delays, passenger demand, departure and arrival queues, airspace capacity, and departure queuing sequence. Such variables have been demonstrated to be significant predictors of runway configuration selection and taxi times in previous studies and were not utilized in this study due to restricted access. Previous studies utilized data from different sources, such as ARINC and ASDE-X, that were not accessible to the researcher for this study (Diana, 2018; Lee et al., 2016). Regulatory authorities such as the FAA can

work to expand the amount of data collected and attempt to release such data to the public, which can encourage research and development efforts.

The utilization of predictive models, such as the ones developed in this study, will also require regulatory approval. As long as Machine Learning models are not deployed and utilized, their usability will not be tested and will remain an area of uncertainty for Machine Learning researchers. Authorities such as the FAA need to encourage the utilization of Machine Learning models in airports and air traffic operations while maintaining the safety standards in the NAS.

Recommendations for Airlines

The models developed in the study were trained on public access data from the FAA ASPM and NOAA databases. The FAA ASPM database did not contain data variables specific to airlines, such as terminal or gate allocation information. Gate allocation can affect the taxi out times for an aircraft and has been demonstrated to be a significant predictor of taxi out times. Additionally, certain data variables are critical for an airline operation at a particular airport, such as the utilization of intersection departures or the departure fixes after departure from a particular runway. Airlines should invest on collecting and maintaining data that might be unique to their airline operations and might improve the performance of such prediction models. The results and model pipeline of this study should be used as a theoretical foundation for airlines to collect and maintain their data and develop predictive models that will be suited to their unique operations.

Recommendations to the Machine Learning Community

The study utilized two Deep Learning architectures. While there is significant literature on utilizing LSTM encoder-decoder models for time series sequence-to-sequence learning, the literature on Transformers for multivariate time series sequence-to-sequence problems and the availability of baselines models is minimal. Significant literature on the utilization of Transformer models is limited to use cases in Natural Language Processing. Further research and development efforts are required in developing Transformer models for time series models due to their demonstrated success in processing sequential data in Natural Language Processing. The development and success of models will encourage the deployment of predictive models in airport and airline operations and expand research and development opportunities in the aviation industry.

References

Abadi, M., Agarwal, A., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M, Jia, Y., Jozefowicz, R., Kaiser, L., Kudler, M., Levenberg, J., Man, D,... Zheng, X. (2016). Tensorflow: A system for large-scale Machine Learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)* (pp. 265–283). https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/45166.pdf

Ahmed, M., Alam, S., & Barlow, M. (2018). A multi-layer artificial neural network approach for runway configuration prediction. *International Conference on Research in Air Transportation*. https://drive.google.com/file/d/1_mwuIWmi4BKXk1kYtHjPPXL3dS5Y4q_A/view

Avery, J., & Balakrishnan, H. (2016). Data-driven modeling and prediction of the process for selecting runway configurations. *Transportation Research Record*, *2600*(1), 1–11. https://doi.org/10.3141/2600-01

Balakrishna, P., Ganesan, R., Sherry, L., & Levy, B. (2010). Estimating taxi out times with a reinforcement learning algorithm. *2008 IEEE/AIAA 27th Digital Avionics Systems Conference*. https://doi.og/ 10.1109/DASC.2008.4702812

Bengio, Y. (2009). Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, *2*(1), 1–127. https://doi.org/10.1561/2200000006

Bengio, Y. (2012). Deep Learning of representations for unsupervised and transfer learning. *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*. https://proceedings.mlr.press/v27/bengio12a/bengio12a.pdf

Bertsimas, D., Frankovich, M., & Odoni, A. (2011). Optimal selection of airport runway configurations. *Operations Research*, *59*(6). https://doi.org/10.1287/opre.1110.0956

Brownlee, J. (2017, July 3). Gentle introduction to the Adam optimization algorithm for deep learning. *Machine Learning Mastery*. https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/#:~:text=Adam%20is%20a%20replacement%20optimization,sparse%20gradients%20on%20noisy%20problems

Carvalho, L., Sternberg, A., Gonçalves, L., Cruz, A., Soares, J., & Brandao, D. (2020). On the relevance of data science for flight delay research: A systematic review. *Transport Reviews, 41*(4). https://doi.org/10.1080/01441647.2020.1861123

Cirium. (2015, November 9). *What is "Block Time" in airline schedules? Why does it matter?* https://www.cirium.com/thoughtcloud/block-time-airline-schedules/#:~:text=Block%20time%20includes%20the%20time,t%20break%20these%20elements%20apart

Dalmau, R., & Herrema, F. (2019). Encoder-decoder approach to predict airport operational runway configuration. *SESAR Innovation Days*. ISSN 0770-1268. https://www.sesarju.eu/sites/default/files/documents/sid/2019/papers/SIDs_2019_paper_37.pdf

Diana, T. (2018). Can machines learn how to forecast taxi out time? A comparison of predictive models applied to the case of Seattle/Tacoma International Airport. *Transportation Research Part E: Logistics and Transportation Review*, *119*, 149–164. https://doi.org/10.1016/j.tre.2018.10.003

Fan, T. (2019). Schedule creep – In search of an uncongested baseline block time by examining scheduled flight block times worldwide 1986–2016. *Transportation Research Part A: Policy and Practice*, *121*, 191–217. https://doi.org/10.1016/j.tra.2019.01.006

Federal Aviation Administration. (2014). Airport design. *Advisory Circular AS 150/5300-13A*. https://www.faa.gov/documentLibrary/media/Advisory_Circular/150-5300-13A-chg1-interactive-201907.pdf

Federal Aviation Administration. (2018). NextGen implementation plan 2018-19. *Office of NextGen*. https://www.faa.gov/nextgen/media/ NextGen_Implementation_Plan-2018-19.pdf

Federal Aviation Administration. (2020). *Air traffic by the numbers*. https://www.faa.gov/air_traffic/by_the_numbers/media/Air_Traffic_by_the_Numbers_2020.pdf

Federal Aviation Administration. (n.d.). *FAA operations & performance data.* https://aspm.faa.gov/

Geron, A. (2019). *Hands-on Machine Learning with Scikit-Learn, Keras and TensorFlow: Concepts, tools, and techniques to build intelligent systems* (2nd ed.). O'Reilly.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning* (1st ed.). MIT Press.

Gu, X., Li, M., & Xu, S. (2013). Research on indirect cost of irregular flight. *Procedia Computer Science*, *19*, 1053–1058. https://doi.org/10.1016/j.procs.2013.06.148

Hardesty, L. (2017). *Explained: Neural networks*. MIT News Office. https://news.mit.edu/2017/explained-neural-networks-deep-learning-0414

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N.J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M.H., Brett, M., Haldane, A., del Rio, J.F., Wiebe, M., Peterson, P.,… Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*, 357–362. https://www.cs.toronto.edu/~fritz/absps/ncfast.pdf

Hinton, G., Osindero, S., & The, Y. (2006). A fast learning algorithm for deep belief nets. *Neural Computing*, *18*(7), 1527–1554. https://www.cs.toronto.edu/~fritz/absps/ncfast.pdf

Horonjeff, R., McKelvey, F., Sproule, W., & Young, S. (2010). *Planning and design of airports* (5th ed.). McGraw-Hill Education.

Jacquillat, A., Odoni, A, & Webster, M. (2016). Dynamic control of runway configurations and of arrival and departure service rates at JFK airport under stochastic queue conditions. *Transportation Science*, *51*(1). https://doi.org/10.1287/trsc.2015.0644

Kang, & Hansen, M. (2018). Assessing the impact of tactical airport surface operations on airline schedule block time setting. *Transportation Research. Part C, Emerging Technologies*, *89*, 133–147. https://doi.org/10.1016/j.trc.2018.01.018

Kingma, D., & Ba, J. (2017). Adam: A method for stochastic optimization. *3rd International Conference on Learning Representations*. https://doi.org/10.48550/arXiv.1412.6980

Kirchgässner, G., & Wolters, J. (2008). *Introduction to modern time series analysis*. Springer. https://doi.org/10.1007/978-3-540-73291-4

Kulaksizoglu, T. (2015). Lag order and critical values of the augmented dickey-fuller test: A replication. *Journal of Applied Econometrics*, *30*(6), 1010- 683 1010. https://doi.org/10.1002/jae.2458

Lee, W. (2019). Python Machine Learning (1st ed.). Wiley.

Lee, H., Malik, W., & Jung, Y. (2016). Taxi out times prediction for departure at Charlotte airport using Machine Learning techniques. 16th AIAA Aviation Technology, Integration, and Operations Conference. https://doi.org/10.2514/6.2016-3910

Li, L., & Clarke, J. (2010). A stochastic model of runway configuration planning. AIAA Guidance, Navigation, and Control Conference. https://doi.org/10.2514/6.2010-7697

Lian, G., Zhang, Y., Desai, J., Xing, Z., & Luo, X. (2018). Predicting taxi out time at congested airports with optimization-based support vector regression methods. *Mathematical Problems in Engineering*, *2018*, 11. http://doi.org/10.1155/2018/7509508

Lipovetsky, S., & Conklin, M. (2001). Analysis of regression in game theory approach. *Applied Stochastic Models in Business and Industry*, *17*, 319–330. https://doi.org/10.1002/asmb.446

Lundberg, S., & Lee, S. (2017). A unified approach to interpreting model predictions. *Advances in Neural Information Processing Systems 30 (NIPS 2017).* https://papers.nips.cc/paper/2017/file/8a20a8621978632d76c43dfd28b67767-Paper.pdf

McKinney, W. (2010). Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference*, *445*, 51–56. https://conference.scipy.org/proceedings/scipy2010/mckinney.html

Molnar, C. (2021). *Interpretable Machine Learning: A guide for making black box models explainable.* https://christophm.github.io/interpretable-ml-book/

National Oceanic and Atmospheric Administration. (n.d.). *Data tools: Local climatological data (LCD).* https://www.ncdc.noaa.gov/cdo-web/datatools/lcd

Pedregosa, F., Varoquaux, Ga"el, Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., & Perrot, M. (2011). Scikit-Learn: Machine Learning in Python. *Journal of Machine Learning Research*, *12*(Oct), 2825–2830.

Qu, J., Zhao, T., Ye, M., Li, J., & Liu, C. (2020). Flight delay prediction using deep Convolutional Neural Network based on fusion of meteorological data. *Neural Processing Letters*, *52*, 1461–1484. https://doi.org/10.1007/s11063-020-10318-4

Ramajun, V., & Balakrishnan, H. (2015). Data-driven modeling of the airport configuration selection process. *IEEE Transactions on Human-Machine Systems*, *45*(4), 490–499. http://doi.org./ 10.1109/thms.2015.2411743

Ravizza, S., Chen, J., Atkin, J., Stewart, P., & Burke, E. (2014). Aircraft taxi time prediction: Comparisons and insights. *Applied Soft Computing*, *14*, 397–406. https://doi.org/10.1016/j.asoc.2013.10.004

Rebollo, J., Khater, S. & Coupe, W. (2021). A recursive multi-step Machine Learning approach for airport configuration prediction. AIAA Aviation 2021 Forum. https://doi.org/10.2514/6.2021-2406

Simaiakis, I., & Balakrishnan, H. (2010). Impact of congestion on taxi times, fuel burn, and emissions at major airports. Transportation Research Record, 2184(1), 22–30. http//doi.org/10.3141/2184-03

Skybrary. (n.d.). Fuel-Flight planning definitions. https://skybrary.aero/articles/fuel-flight-planning-definitions

Sohoni, M., Lee, Y., & Klabjan, D. (2011). Robust airline scheduling under block time uncertainty. Transportation Science, 45(4), 451– 464. http://doi.org/10.1287/trsc.1100.0361

Vaswani, A., Shazer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. 31st Conference on Neural Information Processing Systems. https://doi.org/10.48550/arXiv.1706.03762

Van Rossum, G., & Drake, F. L. (2009). *Python 3 reference manual*. CreateSpace.

Wang, Y., & Zhang, Y. (2021). Prediction of runway configurations and airport acceptance rates for multi-airport system using gridded weather forecast. *Transportation Research Part C: Emerging Technologies*, *125*. https://doi.org/10.1016/j.trc.2021.103049

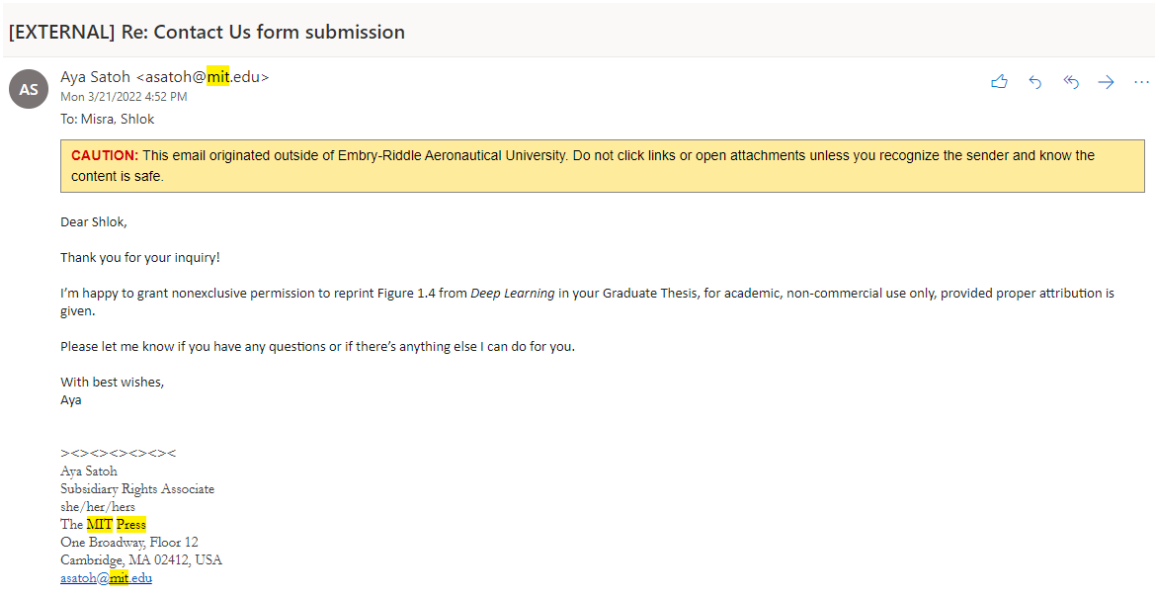
Zeng, A., Chen, M., & Zhang, L., & Xu, Z. (2022). *Are Transformers effective for time series forecasting?* ArXiv. https://doi.org/10.48550/arXiv.2205.13504

Appendix A

Permission to Publish Images

Figure A1

Permission for Figure 1



[EXTERNAL] Re: Contact Us form submission

AS

Aya Satoh <asatoh@mit.edu>
Mon 3/21/2022 4:52 PM
To: Misra, Shlok

CAUTION: This email originated outside of Embry-Riddle Aeronautical University. Do not click links or open attachments unless you recognize the sender and know the content is safe.

Dear Shlok,

Thank you for your inquiry!

I'm happy to grant nonexclusive permission to reprint Figure 1.4 from *Deep Learning* in your Graduate Thesis, for academic, non-commercial use only, provided proper attribution is given.

Please let me know if you have any questions or if there's anything else I can do for you.

With best wishes,
Aya

><><><><><><
Aya Satoh
Subsidiary Rights Associate
she/her/hers
The MIT Press
One Broadway, Floor 12
Cambridge, MA 02412, USA
asatoh@mit.edu

Figure A2

Permission for Figure 2

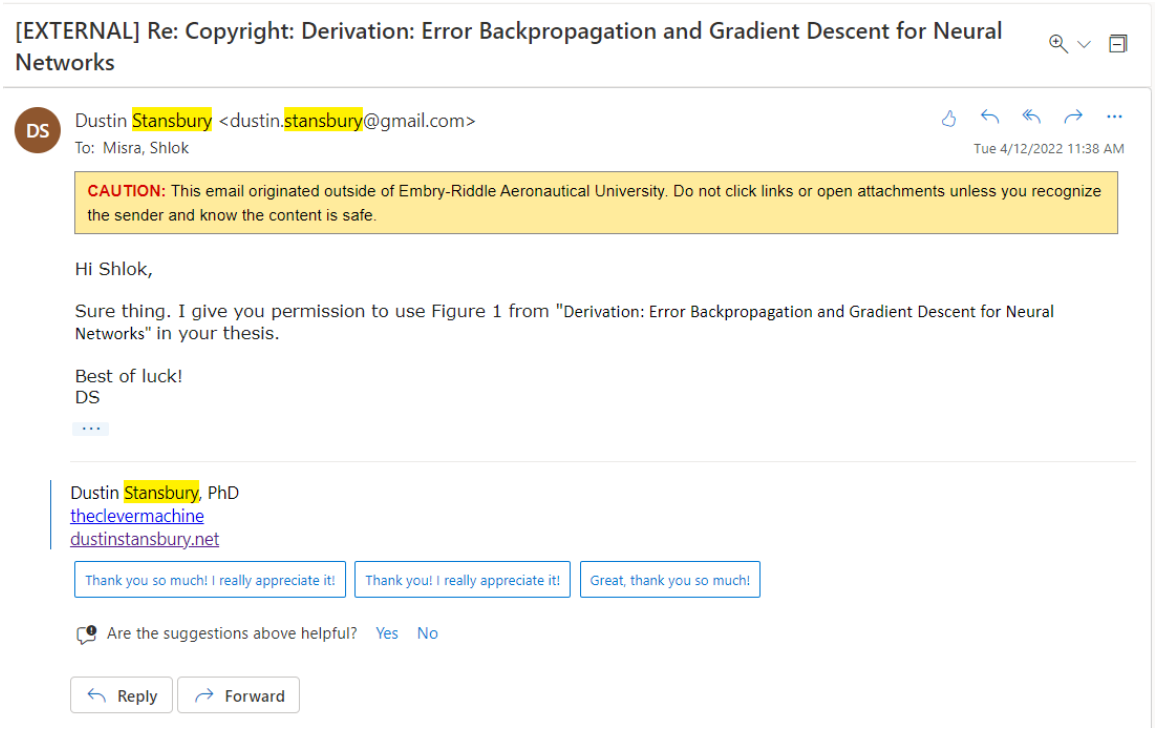


Figure A3

Permission for Figure 3 and Figure 4

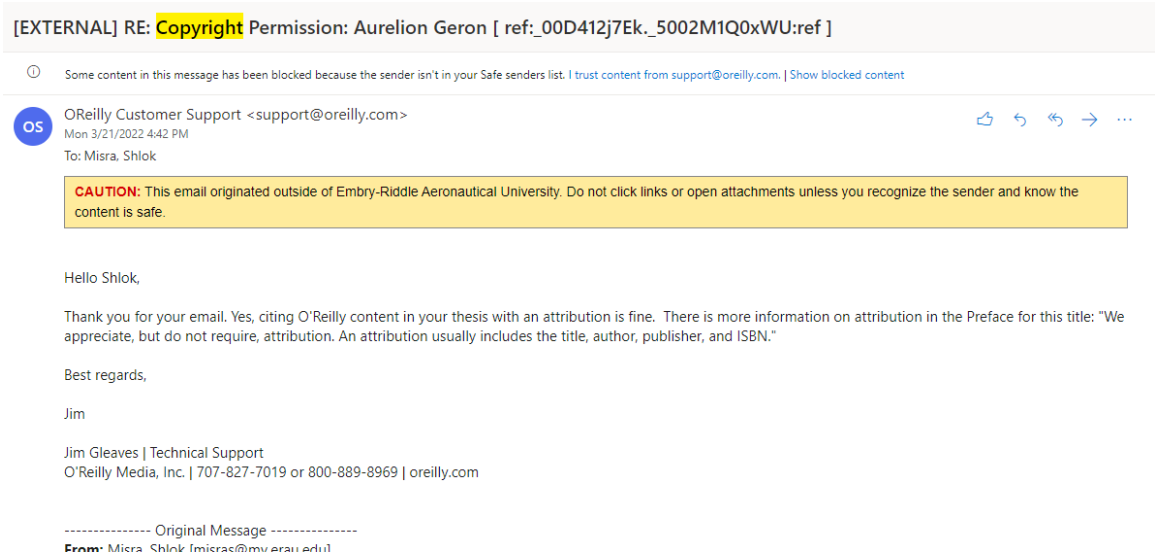
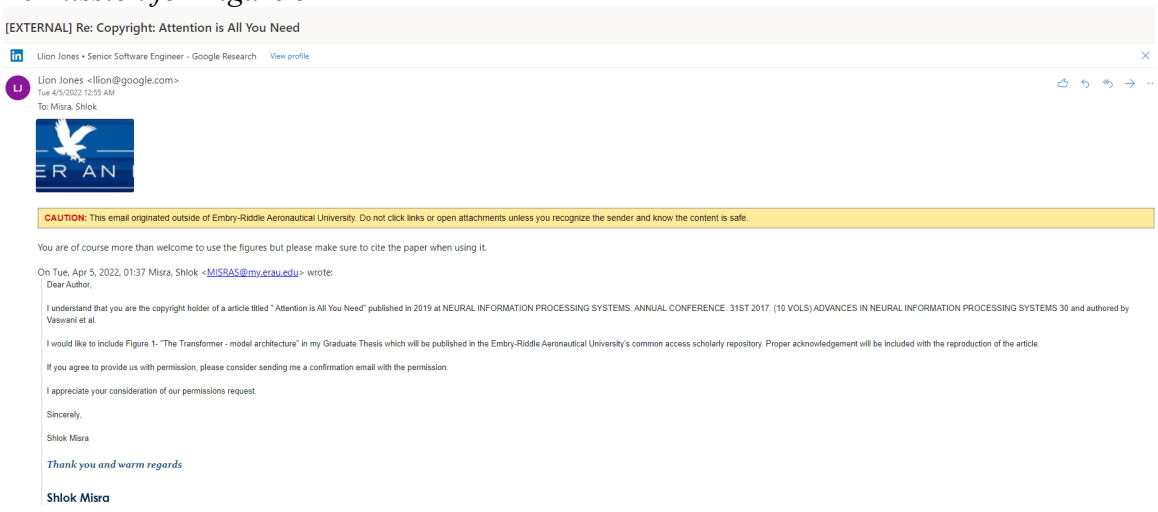


Figure A4

Permission for Figure 5



[EXTERNAL] Re: Copyright: Attention is All You Need

Llion Jones • Senior Software Engineer - Google Research View profile

Llion Jones <llion@google.com>
Tue 4/5/2022 12:55 AM
To: Misra, Shlok

CAUTION: This email originated outside of Embry-Riddle Aeronautical University. Do not click links or open attachments unless you recognize the sender and know the content is safe.

You are of course more than welcome to use the figures but please make sure to cite the paper when using it.

On Tue, Apr 5, 2022, 01:37 Misra, Shlok <MISRAS@my.erau.edu> wrote:

Dear Author,

I understand that you are the copyright holder of a article titled " Attention is All You Need" published in 2019 at NEURAL INFORMATION PROCESSING SYSTEMS. ANNUAL CONFERENCE. 31ST 2017. (10 VOLS) ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS 30 and authored by Vaswani et al.

I would like to include Figure 1- "The Transformer - model architecture" in my Graduate Thesis which will be published in the Embry-Riddle Aeronautical University's common access scholarly repository. Proper acknowledgement will be included with the reproduction of the article.

If you agree to provide us with permission, please consider sending me a confirmation email with the permission.

I appreciate your consideration of our permissions request.

Sincerely,

Shlok Misra

Thank you and warm regards

Shlok Misra

Appendix B

Granger's Causality Test

B1 Python Code for Granger's Causality Test

Figure B1

Python Code for Granger's Causality Test

```
In [25]: from statsmodels.tsa.api import VAR
         from statsmodels.tsa.stattools import adfuller
         from statsmodels.tools.eval_measures import rmse, aic

In [26]: variables = df.columns

In [27]: from statsmodels.tsa.stattools import grangercausalitytests
         maxlag=12
         test = 'ssr_chi2test'
         def grangers_causation_matrix(data, variables, test='ssr_chi2test', verbose=False):
             df = pd.DataFrame(np.zeros((len(variables), len(variables))), columns=variables, index=variables)
             for c in df.columns:
                 for r in df.index:
                     test_result = grangercausalitytests(data[[r, c]], maxlag=maxlag, verbose=False)
                     p_values = [round(test_result[i+1][0][test][1],4) for i in range(maxlag)]
                     if verbose: print(f'Y = {r}, X = {c}, P Values = {p_values}')
                     min_p_value = np.min(p_values)
                     df.loc[r, c] = min_p_value
             df.columns = [var + '_x' for var in variables]
             df.index = [var + '_y' for var in variables]
             return df
         Table=grangers_causation_matrix(df, variables=df.columns)
```

Appendix C

Python Code to Create Input and Out Sequence

```
import pandas as pd
import numpy as np
```

```
n_timesteps_in=24
n_features=21
n_timesteps_out=6
```

```
from numpy.ma.core import array


X = []
Y = []

list_of_x_cols = ['Departures','Arrivals','Altimeter','Dew_Point','Temperature',
                  'Precipiaion','PressureChange','HourlyRelatieHumidity','HourlyVisibility',
                  'DZ','FG','NOSIG','RA','TS','CLM','NE','NW','RB','SE','SW',
                  'Ru0way_Co0figuratio0_Bi0ary']
for i in range(len(df) - n_timesteps_in-n_timesteps_out):
  x = df[list_of_x_cols][i:i+n_timesteps_in]
  y = df["Taxi_Out"][i+n_timesteps_in:i+n_timesteps_in+n_timesteps_out]
  print(y)
  X += [np.array(x)]
  Y += [np.array(y)]

X = np.array(X)
Y = np.array(Y)
print(np.shape(X), np.shape(Y))
```

Appendix D

Python Code to Develop the LSTM Encoder-Decoder Model

```
#@title Import Libraries
from random import randint
from numpy import array
from numpy import argmax
import keras.backend as K
from tensorflow.keras import models
from numpy import array_equal
import tensorflow as tf
import numpy as np
from tensorflow.keras.models import Sequential, Model
from tensorflow.keras.layers import LSTM, Bidirectional
from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras import Input
from tensorflow.keras.layers import TimeDistributed
from tensorflow.keras.layers import RepeatVector
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.utils import plot_model
import matplotlib.pyplot as plt
```

```
[ ] model_Single_LSTM_default_output = Sequential(name='model_Single_LSTM_default_output')
    #model_Single_LSTM_default_output.add(Input(shape=(n_timesteps_in, n_features)))
    model_Single_LSTM_default_output.add(tf.keras.layers.Conv1D(filters=12, kernel_size=6,
                                                    strides=1, padding="valid", input_shape=(n_timesteps_in, n_features)))
    model_Single_LSTM_default_output.add(tf.keras.layers.LSTM(32, return_sequences=True,
    stateful=False))
    model_Single_LSTM_default_output.add(tf.keras.layers.LeakyReLU(alpha=0.01))
    model_Single_LSTM_default_output.add(tf.keras.layers.Dropout(0.3))
    model_Single_LSTM_default_output.add(tf.keras.layers.LSTM(64, return_sequences=True,
    stateful=False))
    model_Single_LSTM_default_output.add(tf.keras.layers.LeakyReLU(alpha=0.01))
    model_Single_LSTM_default_output.add(tf.keras.layers.Dropout(0.3))
    model_Single_LSTM_default_output.add(tf.keras.layers.LSTM(64, return_sequences=False,stateful=False))
    # Single LSTM Layer with default output
    #model_Single_LSTM_default_output.add(LSTM(32))
    #Repeat the output of LSTM n_timesteps (4 in our example)
    model_Single_LSTM_default_output.add(RepeatVector(n_timesteps_out))
    model_Single_LSTM_default_output.add(LSTM(32, activation='relu', return_sequences=True))
    model_Single_LSTM_default_output.add(LSTM(64, activation='relu', return_sequences=True))
    model_Single_LSTM_default_output.add(TimeDistributed(Dense(32, activation='relu')))
    model_Single_LSTM_default_output.add(TimeDistributed(Dense(1)))
    model_Single_LSTM_default_output.compile(loss='mse', optimizer='adam')
```

```
early_stopping= tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=5)
history=model.fit(X_train,y_train,validation_split=0.1, shuffle=False, epochs=200, callbacks=[early_stopping])
```

Appendix E

Python Code to Develop the Transformer Model

```
from tensorflow import keras
from tensorflow.keras import layers

def transformer_encoder(inputs, head_size, num_heads, ff_dim, dropout=0):
    # Normalization and Attention
    x = layers.LayerNormalization(epsilon=1e-6)(inputs)
    x = layers.MultiHeadAttention(
        key_dim=head_size, num_heads=num_heads, dropout=dropout
    )(x, x)
    x = layers.Dropout(dropout)(x)
    res = x + inputs

    # Feed Forward Part
    x = layers.LayerNormalization(epsilon=1e-6)(res)
    x = layers.Conv1D(filters=ff_dim, kernel_size=1, activation="relu")(x)
    x = layers.Dropout(dropout)(x)
    x = layers.Conv1D(filters=inputs.shape[-1], kernel_size=1)(x)
    return x + res
```

```
def build_model(
    input_shape,
    head_size,
    num_heads,
    ff_dim,
    num_transformer_blocks,
    mlp_units,
    dropout=0,
    mlp_dropout=0,
):
    inputs = keras.Input(shape=input_shape)
    x = inputs
    for _ in range(num_transformer_blocks):
        x = transformer_encoder(x, head_size, num_heads, ff_dim, dropout)

    x = layers.GlobalAveragePooling1D(data_format="channels_first")(x)
    for dim in mlp_units:
        x = layers.Dense(dim, activation="relu")(x)
        x = layers.Dropout(mlp_dropout)(x)
    outputs = layers.Dense(6)(x)
    return keras.Model(inputs, outputs)
```

```
input_shape = X_train.shape[1:]

model = build_model(
    input_shape,
    head_size=256,
    num_heads=4,
    ff_dim=4,
    num_transformer_blocks=4,
    mlp_units=[128],
    mlp_dropout=0.4,
    dropout=0.25,
)

model.compile(
    loss="mean_squared_error",
    optimizer=keras.optimizers.Adam(learning_rate=1e-4)
)
```

```
#model.summary()

callbacks = [keras.callbacks.EarlyStopping(patience=10, \
    restore_best_weights=True)]

model.fit(
    X_train,
    y_train,
    validation_split=0.2,
    epochs=200,
    batch_size=64,
    callbacks=callbacks,
)

model.evaluate(X_test, y_test, verbose=1)
```