Artificial Intelligence-enhanced Predictive Modeling in Air Traffic Management

by

Yutian Pang

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Approved April 2023 by the
Graduate Supervisory Committee:

Yongming Liu, Chair
Hao Yan
Houlong Zhuang
Hamid Marvi
Yi Ren

ARIZONA STATE UNIVERSITY

May 2023

ABSTRACT

National Airspace Systems (NAS) are complex cyber-physical systems that require swift air traffic management (ATM) to ensure flight safety and efficiency. With the surging demand for air travel and the increasing intricacy of aviation systems, the need for advanced technologies to support air traffic management and air traffic control (ATC) service has become more crucial than ever. Data-driven models or artificial intelligence (AI) have been conceptually investigated by various parties and shown immense potential, especially when provided with a vast volume of real-world data. These data include traffic information, weather contours, operational reports, terrain information, flight procedures, and aviation regulations. Data-driven models learn from historical experiences and observations and provide expeditious recommendations and decision support for various operation tasks, directly contributing to the digital transformation in aviation.

This dissertation reports several research studies covering different aspects of air traffic management and ATC service utilizing data-driven modeling, which are validated using real-world big data (flight tracks, flight events, convective weather, workload probes). These studies encompass a range of topics, including trajectory recommendations, weather studies, landing operations, and aviation human factors. Specifically, the topics explored are (i) trajectory recommendations under weather conditions, which examine the impact of convective weather on last on-file flight plans and provide calibrated trajectories based on convective weather; (ii) multi-aircraft trajectory predictions, which study the intention of multiple mid-air aircraft in the near-terminal airspace and provide trajectory predictions; (iii) flight scheduling operations, which involve probabilistic machine learning-enhanced optimization algorithms for robust and efficient aircraft landing sequencing; (iv) aviation human factors, which predict air traffic controller workload level from flight traffic data with conformalized graph neural network. The uncertainties associated with these studies are given special attention and addressed through Bayesian/probabilistic machine learn-

ing. Finally, discussions on high-level AI-enabled ATM research directions are provided, hoping to extend the proposed studies in the future.

This dissertation demonstrates that data-driven modeling has great potential for aviation digital twins, revolutionizing the aviation decision-making process and enhancing the safety and efficiency of ATM. Moreover, these research directions are not merely add-ons to existing aviation practices but also contribute to the future of transportation, particularly in the development of autonomous systems.

To my beloved family.

ACKNOWLEDGEMENTS

It's taken a while to get here, and has been almost ten years since I embarked on my journey as a freshman. This degree is not an easy one, to start everything from scratch, hope the next ten years make it worth the effort. Nevertheless, I am grateful for the invaluable support received from many individuals to make this thesis a reality.

I want to extend my appreciation to my advisor, Dr. Yongming Liu, for providing me with the autonomy to grow and develop my cognitive abilities. Your support and guidance have been instrumental in helping me achieve this academic milestone. Thank you, Dr. Hao Yan, Dr. Sankaran Mahadevan, and Dr. Erin DeCarlo for the regular discussions on the aviation project for the past five years, allowing me to keep expanding my academic knowledge and boost my research aptitude. Thanks to the committee members, EAB members, and research professionals encountered during conferences for the helpful discussions. Thanks, to NASA and Ames Aviation Systems Division for providing the financial support and data resources that enabled the successful completion of this work.

Thanks go out to all of my collaborators, officemates, and friends, for allowing me to be a source of annoyance in your lives. Here is a list: YW, HY, XZ, QX, CL, NX, SC, HW, JH, RR, HI, SD, ZG, BZ, QC, YG, PZ, QZ, XY, YY, ZZ, QZ, XZ, YQ, BD... You are most likely all over the world at this point. If you ever read this, I wish for world peace and hope we still have a chance to meet and refresh our memories one day.

To my parents, I express my greatest reverence for their encouragement and unconditional support. I owe everything to my parents, who worked tirelessly to raise me up and bring me to higher ground. Also, I want to give profound thanks to my labmate, collaborator, and wife, Jueming Hu, for the immeasurable source of inspiration and motivation. Moreover, I am indebted to her for the gift of our precious son Neo, whose arrival illuminated my path and imbued my work with greater purpose and significance. I am awestruck by their unwavering devotion and am honored to have all of them by my side.

TABLE OF CONTENTS

CHAPTER Page

CHAPTER Page

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

Figure Page

Figure Page

Figure Page

Figure Page

Chapter 1

INTRODUCTION

1.1 Overview and Motivations

Commercial aviation systems are complex cyber-physical systems that require efficient air traffic management (ATM) systems to ensure safe and efficient flight practices. It integrates cyberspace and physical facilities into a space-air-ground multi-dimensional networked intelligent system. With the increase in air travel demand and the growing operational complexity, the current ATM paper- and voice-based ATM systems developed for decades are facing substantially increased challenges. The decision-makers do not have access to a high level of automated operation recommendations or lack important real-time information feeds, resulting in the failure of situation awareness and aviation safety concerns. These drawbacks create a highly tactical basis environment, where operation practitioners have to make decisions based on incomplete or uncertain information. As a result, plans to optimize operations are often rendered obsolete as the situation evolves. To address these limitations, researchers from multiple universities, research institutes, and government authorities are undertaking advanced ATM research and digital transformation programs to push the steady evolution to a higher level of automated ATM systems. For instance, the European Union Aviation Safety Agency (EASA) published two versions of conceptual papers to provide guidelines on potential artificial intelligence (AI) applications for innovations in ATMs and their anticipated mean of compliance with existing industrial standards [5; 6]. Specifically, the MITRE Corporation provides the review article on aircraft trajectory prediction enabling the migration to trajectory-based operations (TBO), where the flights are represented as trajectories and shared across the ground, air, and space

air navigation service providers for the gradual evolution to TBO [7]. With an accurate trajectory prediction capability, aviation decision-makers can perform strategic planning and robust in-house management, leveraging system-wide information coordination.

AI is linked with most of the technological breakthroughs in the 21st century, and AI-enabled digital transformation is instrumental in shaping the fourth industrial revolution. Leading countries invested heavily in AI, with the goal of gaining strength in international competitions. The benefits of using AI in aviation are significant. For instance, data-driven models can digest vast volumes of data in real-time, providing decision-makers with accurate and timely information and operational recommendations, and leading to better-informed decisions to enhance aviation safety and efficiency. For airline operations, AI can also assist in predicting delays, which can help airlines and airports' resource allocation process, reducing financial losses and improving the passenger experience.

The concerns toward AI in aviation are obvious and twofold. The major challenge is the trustworthiness of AI solutions, including the explainability of AI models and regulatory compliance. Compared to a rule-based model, AI models receive the typical *black box* criticism as the complexity of the AI model brings a level of opaqueness that makes them look unverifiable, thus will not be able to meet regulatory compliance. However, the author would like to argue that there are some efforts into performing formal analysis types of verification and validation of simple structured AI models, providing a feasible route for AI trustworthiness analysis. Large deep-learning models are typical modular-based architecture, where each modular, or sub-network, accounts for different functionality, providing a level of explainability to the audience. Moreover, knowledge graphs also provide a way to include domain knowledge into the graph-structured deep learning model, resulting in knowledge-based learning. Lastly, risk assessment and control are also helpful directions to explore satisfying regulatory requirements and safety standards. The second major concern comes from the integrated cyber-physical security and privacy of the ATM data transmis-

sion process. The aviation communication security issue has been included in multiple development plans for future ATM systems, including the Next Generation Air Transportation System (NextGen), Single European Sky ATM Research (SESAR), and Civil Aviation ATM Moderation Strategy (CAAMS). Real-time AI-enabled decision-making substantially increased the communication and data transmission volume within the ATM system, and can digest masqueraded tempered signals and flight data for the AI training process resulting in misleading decision recommendations. The resolutions can either add cyber protection layers to the networked communications or adopt secure AI learning frameworks (i.e., federated learning).

Real-world data is another core component of data-driven studies. The typical machine learning model takes real-world collected data as input and makes predictions based on the historically recorded data, to ensure that they can be applied in the real world with confidence. Although there are redundant open-source aviation traffic data, meteorological data, and operational data available for research, in this thesis, we focus on investigating the Sherlock aviation data warehouse maintained by NASA ARC. Sherlock provides an extensive data repository for aviation big data research. Notably, this thesis emphasizes the importance of data engineering, as the aviation data used can be in the petabyte range.

Lastly, applying AI techniques in aviation is still in the exploration stage. The potential benefits of AI in aviation are enormous. The use of data-driven models and AI technologies in aviation is an exciting development that has the potential to transform air traffic management and advanced avionics.

1.2 Research Objectives

Based on the above discussions, the author conducts broad reading and literature review to identify engineering research objectives. In this thesis, the author investigated three major research directions in the general field of ATM. Here is a list of engineering research

objectives,

- Develop a reliable probabilistic aircraft trajectory prediction model considering convective weather impacts. In real-world practices, the aircraft trajectories are guided by the last on-file flight plan, but are impacted by various factors, considered as various sources of uncertainties. One of the biggest uncertainties comes from the convective weather and meteorological conditions. Thus, the first study focuses on reducing trajectory prediction errors with convective weather, such that the predicted trajectory gets closer to the ground truth.

- In the terminal maneuvering area (TMA), the major focus of trajectory prediction changes to behavior/intention modeling in multi-agent scenarios. Thus, the major objective of this stage is to develop a reliable probabilistic multi-aircraft trajectory prediction model in the TMA. Besides predicting the probabilistic trajectories for multiple aircraft, data-driven model robustness and intention modeling are two additional interests. The most recent advancements in computer vision research are investigated and integrated into the proposed model.

- Develop a probabilistic machine learning-enhanced aircraft landing scheduling model to improve the landing runway throughput during busy operation hours. In this study, the motivation comes from the inefficiency of the current first-come-first-served landing scheduling policy. An optimization algorithm is adopted to search for the best landing sequences of several aircraft, and probabilistic machine learning is used to predict the minimum landing time difference between successive landing aircraft pairs with a reliability tolerance. Integrating these two parts together results in a machine-learning algorithm for optimal aircraft landing scheduling, to address aviation safety and efficiency.

- Develop air traffic controller (ATCo) workload level prediction model with in-house

human-in-the-loop (HITL) experiment data. The objective of this aviation human factor research is to propose a better non-intrusive ATCo workload prediction algorithm, taking advantage of the graph-structured spatiotemporal air traffic data. In this way, proper workload management and resource allocation can be performed, to reduce ATCo fatigue level, avoid ATCo workload overhead, and reduce aviation safety concerns.

- Technical exploration for aviation data engineering and aviation monitoring and simulation software platforms are two other engineering focuses. The fundamental interests in data engineering are the exploration of agile geospatial data processing tools to perform ultra-efficient big queries on air traffic data, and digital transformation tools for encoded aviation data strings (i.e., the digital translators of flight plan strings/landing and departure procedures/ARINC air navigation records to geospatial coordinates). On the other hand, the development of aviation software can reduce the research effort on aviation simulations, flight visualizations, and data analytics.

1.3 Organization

The rest of this thesis is divided into five chapters, grouped into three aviation research topics followed by a conclusion.

Chapter 2 proposes an advanced Bayesian Deep Learning method for aircraft trajectory prediction considering weather impacts. A brief review of both deterministic and probabilistic trajectory prediction methods is given, with particular emphasis on learning-based methods. Next, a deterministic trajectory prediction model with classical deep learning methods is proposed to handle both spatial and temporal information using a nested convolution neural network, recurrent neural network, and fully-connected neural network. Following this, the deterministic neural network model is extended to be a Bayesian deep

learning model to consider uncertainties where the posterior distributions of parameters are estimated with variational inference for enhanced efficiency. Both mean prediction and confidence intervals are obtained by giving the last on-file flight plans and weather data in the region. The proposed methodology is validated using air traffic and weather data from the Sherlock data warehouse. Data pre-processing procedures for big data analytics are discussed in detail. Demonstration and metrics-based validation are performed during severe convective weather conditions for several air traffic control centers. The results show a significant reduction in prediction variance. A comparison with existing methods is also performed. Several conclusions and future works are given based on the proposed study.

Chapter 3 proposes the Bayesian Spatio-Temporal grAph tRansformer (B-STAR) architecture to model the spatial and temporal relationship of multiple aircraft under uncertainties. The design of the B-STAR structure takes advantage of conclusions from the previous deep learning robustness study. The previous work shows that the stochastic classifier after the deterministic CNN extractor has sufficient robustness enhancement rather than a stochastic feature extractor before the stochastic classifier. This advises on utilizing stochastic layers in building decision-making pipelines within a safety-critical domain. The performance of the proposed B-STAR model is first validated on the standard ETH/UCY pedestrian dataset, with UQ competence. Then, the multi-aircraft near-terminal interactive trajectory prediction model is trained and validated with real-world flight recording data. The sensitivity study on the prediction/observation horizon and the graph neighboring distance threshold are performed.

Chapter 4 develops a novel machine learning-enhanced methodology for aircraft landing scheduling. Data-driven machine learning (ML) models are proposed to enhance automation and safety. ML enhancement is adopted for both prediction and optimization. First, the flight arrival delay scenarios are analyzed to identify the delay-related factors, where strong multi-model distributions and arrival flight time duration clusters are ob-

served. A multi-stage conditional ML predictor is proposed for improved prediction performance of separation time conditioned on flight events. Next, this chapter proposes incorporating the ML predictions as safety constraints of the time-constrained traveling salesman problem formulation. The scheduling problem is then solved with mixed-integer linear programming (MILP). Additionally, uncertainties between successive flights from historical flight recordings and model predictions are included to ensure reliability. This chapter demonstrates the real-world applicability of this proposed method using the flight track and event data from the Sherlock database of the Atlanta Air Route Traffic Control Center (ARTCC ZTL). The case studies demonstrate that the proposed method can reduce the total landing time compared with the First-Come-First-Served (FCFS) rule during the busy time period. Unlike the deterministic heuristic FCFS rule, the proposed methodology also considers the uncertainties between aircraft and ensures confidence in the scheduling. Finally, several concluding remarks and future research directions are given.

Chapter 5 first performs a review of research on ATCo cognitive workload, mostly from the air traffic perspective. Then, this chapter briefly introduces the setup of the human-in-the-loop (HITL) simulations with retired ATCos, where the air traffic data and cognitive workload labels are obtained. The simulations are conducted under three Phoenix approach scenarios while the human ATCos are requested to self-evaluate their workload ratings (i.e., low-1 to high-7). Preliminary data analysis is conducted. Next, this chapter proposes a graph-based deep-learning framework with conformal prediction to identify the ATCo workload levels. The number of aircraft under the controller's control varies both spatially and temporally, resulting in dynamically evolving graphs. The experiment results suggest that (a) besides the traffic density feature, the traffic conflict feature contributes to the workload prediction capabilities (i.e., minimum horizontal/vertical separation distance); (b) directly learning from the spatiotemporal graph layout of airspace with graph neural network can achieve higher prediction accuracy, compare to hand-crafted traffic

complexity features; (c) conformal prediction is a valuable tool to further boost model prediction accuracy, resulting a range of predicted workload labels.

Chapter 6 is the concluding chapter of this dissertation. It begins with a summary of aviation research topics, followed by a list of major contributions. The chapter then offers multi-level insights and future research directions, ranging from task-level to conceptual-level suggestions for using AI in aviation decision-making. The task-level insights are problem-specific, and the motivations behind them can be any of the following: (i) incrementally improving model performance; (ii) expanding the research scope; (iii) redefining experimental setups; or (iv) providing new functionalities to an existing model. On the strategic or conceptual level, this thesis discusses critical future topics, such as security, safety, explainability, and compliance, whenever AI applications are involved in the general ATM domain.

Chapter 2

STRATEGIC AIRCRAFT TRAJECTORY PREDICTION UNDER CONVECTIVE WEATHER

2.1 Introduction

In the context of air traffic management (ATM), the development of the next-generation national air transportation system (NextGen) [8] is of critical importance. NextGen aims to efficiently and safely accommodate the increasing air traffic flows within the United States. A key element of NextGen is the capability to predict and share air traffic trajectories and conflicts with all involved stakeholders. In NextGen, information sharing between aircraft will be greatly enhanced so that each aircraft receives and transmits the cooperative surveillance information. Thus, aircraft can take over a certain amount of ATM tasks from ground air traffic controllers. These decision-support tools (DSTs) [9] include flight plan (FP) changes, dynamic weather reroute (DWR), trajectory prediction (TP), and conflict detection and resolution (CDR). With the advancement of these tools, NextGen can alleviate the mental workloads of ground air traffic controllers [10; 11; 12], especially when infrequent but critical events happened [13].

It has been shown that there are huge trajectory uncertainties during aviation operations. The uncertainty of trajectory predictions (TP) comes from multiple sources. A typical TP tool makes decisions based on the aircraft's performance, pilot's intent, and Terminal Radar Approach Control (TRACON) regulations [14]. The unawareness of the pilot's intent and assumptions of the aircraft's conditions contribute to the prediction uncertainty. Also, when multiple aircraft are heading to the same region, communication and avoiding maneuver between each of the aircraft will cause the uncertainty of the trajectory prediction [15].

Among all sources of uncertainties, the environmental factor is one of the most significant contributors to the TP uncertainty [16; 17; 18]. The convective weather condition usually generates rapidly and randomly, which represents aleatoric uncertainties. The accuracy of the weather prediction, such as resolution, accuracy, and forecasting intervals, will also affect the TP uncertainties and represents epistemic uncertainty sources. In addition, the pilot and controller's decision regarding weather conditions brings in additional uncertainties from human factors. Due to the nature of uncertainties that contribute to the TP errors, a deterministic TP model will not be sufficient for the efficient and reliable TP models, especially when dealing with increasingly congested airspace [19].

The availability of different massive aviation databases [20; 21] has enabled the possibility of changing the conventional TP tool into a multi-source data-driven approach [22], such as machine learning (ML) methods. Deep learning (DL), as a special type of ML technique, draws significant interest in academia and industries and has achieved great success in object detection [23], natural language processing [24; 25], dimension reduction [26], generative learning [27], and reinforcement learning [28]. Numerous ML studies have been performed in ATM during the last decades, although multiple challenges (i.e., data collection, data privacy, data storage, data cleansing, and data opening) still exist in real-world applications [29]. Compared to the traditional ML approaches, the DL model has the following advantages: 1) DL model has an architecture by stacking multiple back-propagation compatible layers to discover the inherent information within the data, such as the convolutional layers and recurrent layers to consider the complex spatial structures and temporal structures presented in the TP problem; 2) DL model can take advantage of the big data as the DL algorithm is easy to incorporate multiple factors into the TP model. However, the traditional DL model has the following limitations. 1) The inherent characteristics of DL models are prone to overfitting, especially for recurrent neural networks (RNNs) [30]. To solve this, significant fine-tuning of network parameters combined with stochastic regular-

ization techniques (SRTs) such as dropout [31] are often required. Researchers typically choose the parameters of the DL model subjectively. 2) Traditional DL models cannot be used to quantify the uncertainties of the trajectories.

Bayesian techniques are known as a remedy to address the issues of overfitting and uncertainty quantification. More specifically, the DL model that incorporates Bayesian variational inference, known as Bayesian deep learning (BDL), is shown to be robust for overfitting problems [32]. Furthermore, instead of using a single point estimate of model parameters in classical DL, the Bayesian methods yield distribution for each of the parameters, which provides an inherent estimate of prediction uncertainty.

Therefore, this chapter proposes a Bayesian deep learning-based probabilistic aircraft trajectory prediction method under weather impact. Multiple convolutional, recurrent, and fully-connect layers are constructed to discover the complex spatial-temporal relationship from historical data. Bayesian theory is then applied to form a probabilistic learning framework. This work performs experiments with real flight trajectories, flight plans, and convective weather measurement data from the Sherlock database. To the best knowledge of the authors, this is the first paper that the BDL is used to perform the TP task considering weather impacts. Another new contribution of the proposed study is the proposed unique probabilistic CNN+RNN+FCNN architecture for big data analytics involving dynamically changing weather and trajectory information in ATM.

The rest of the paper is organized as follows. Section 5.2 reviews the studies performed in the general field of trajectory prediction. Section 2.3 discusses the evolution of deep learning combining Bayesian theory as well as our proposed trajectory prediction framework. Section 5.4 introduce the proposed TP framework as well as the processing pipeline on each of the three building-blocks. The test results of the proposed framework and the comparison with other NN models are described in Section 2.5. A study on the contributions of explicitly introducing weather features into the TP framework is also conducted.

Concluding remarks and future directions are described in Section 5.6.

2.2 Literature Review

Trajectory prediction has been long regarded as a key functional component in different research areas such as ground vehicle trajectory prediction [33; 34; 35; 36], and pedestrian trajectory prediction [37; 38; 39; 40] in both deterministic and probabilistic sense. Here, this chapter focuses on the aircraft TP in the field of ATM. Existing methodologies can be divided into deterministic and probabilistic methods. Details are shown below.

2.2.1 Deterministic Methods

The deterministic approach gives a point estimate of the aircraft location without associated uncertainties to the predicted trajectory. Most approaches focus on the modeling of aircraft's intent, motions, and dynamics. The nominal approach to do the prediction is to propagate an estimation of the state space and find the future predictions given a sequence of past observations.

Data-based TP

Most data-driven methods include state-space model [41], Kalman and particle filtering [42], classical neural networks (NNs) [43], and physical-based learning [44]. A state-space model is proposed for the short-time trajectory prediction near the terminal area by using a selection of flight plan or dead reckoning method instead of a flight plan (FP) based guidance [41]. The key idea is that, if an aircraft is not following its route, the best estimation is using the dead reckoning. The experiment is conducted with four selected trajectories and evaluated by the distance between the predicted tracks and the radar recorded tracks. Some of the developed TP tools have been bringing into practice. A TP function for tactical flight management is developed based on aircraft motions [45]. These aircraft motions are sim-

ply an extension of aircraft states. This function continuously generates four-dimensional (4D) coordinates and updates the flight management system. The model has been deployed in the Advanced Concepts Flight Simulator at NASA Ames Research Center. Similar work on TP evaluates the future positions of the aircraft based on the aircraft maneuver state [46]. Kalman filtering is used for state estimation and noise smoothing. The experiment is demonstrated by a computer simulation of reverse turn maneuvers of an aircraft model. Neural network has also been used in ATM. A vertical plan TP model is proposed using a simple neural network [43]. The prediction is made with the knowledge of previous aircraft locations. The experimental result shows that NNs are more efficient and have lower average error comparing to non-parametric methods. Another group of deterministic approaches focuses on the selection of multiple trajectory candidates. These methods utilize the first part of the radar measurements of the flight tracks as the indicator for selection among trajectory candidates for single aircraft [47] or multi-aircrafts TP [42].

Human Factors on TP

There is another research direction focusing on the human factors on TP. The pilot's intent is inferred and combined with aircraft motions and states for the TP task with a hybrid estimation algorithm [48]. This work first acquires the air traffic control (ATC) regulations, the flight plan, and the environmental factors to infer the pilot's intent. Then the prediction is modeled as a function of aircraft state and pilot's intent. The experiment is conducted with a 6-DOF (degree of freedom) simulation of Boeing 747-200 aircraft. Another interesting work is trying to model the aircraft trajectory into a set of intent-related languages [49]. These languages are used for describing aircraft motions. A hierarchy of language generating engines is modeled and each engine can perform modification over a trajectory at different levels. Another recent work focuses on improving the accuracy of TP during landing by including voice communications into the Bayesian framework [50].

Issues with Deterministic TP

The major issue of the deterministic approach for TP is the lack of ability to predict the uncertainty of the trajectory. There has been stated that the deterministic techniques suffers from the degraded accuracy in several cases [45; 51; 46; 41]. Changing into probabilistic TP is a feasible solution to improve the robustness of the models. Furthermore, most of the trajectory prediction algorithms focuses on the short-term prediction, which cannot be used to predict the future trajectory in the long-term horizon.

2.2.2 Probabilistic Methods

The probabilistic TP research experienced a transition from conventional methods to statistical ML-based methods. The conventional methods include Sequential Monte Carlo (SMC) [52; 53], Hidden Markov Model (HMM) [54], and others [55; 56; 57]. ML-based methods are brought into the TP field due to the availability of surveillance data and the increased computing power. ML-based methods include the Gaussian Mixture Model (GMM), recurrent neural network (RNN), and several other statistical generalized regression models (GLMs).

Conventional TP

Probabilistic Monte Carlo (MC) methods and deterministic worst-case methods are proposed to perform model-based TP through a case study [52] based on current observations. The uncertainty of this method is assumed to come from the inexact knowledge of wind and aircraft mass, which represents the aleatoric uncertainty. Similarly, another research tries to identify the source of uncertainty that affecting a 3-DOF point mass model of aircraft motion [53]. These uncertainties are incorporated into their mathematical TP model. Other works also identify the different sources of uncertainties for their probabilistic TP models

[55; 56]. Another probabilistic model is proposed for predicting position in the near-term and mid-term future [57]. They derive an expression for the probability of conflict using the provided quantitative bounds. Based on the expression, the model is able to generate potential fields for probabilistic path planning of the aircraft. MC simulations is used to validate the purposed method. Their following work builds a probabilistic aircraft position forecasting model using the flight plan and current position [58]. Aircraft's intent is also an important factor in probabilistic TP models. Researchers have incorporated aircraft's intent into flight dynamics to predict future positions as a series of probability density functions [10].

Weather Impact on Conventional TP: Several existing studies focus on the impact of convective weather to the special circumstances of a TP task [59; 60; 61; 62]. Genetic algorithm (GA) is used to provide fuel and time-saving flight plans [59]. This method takes tailwind into account for the best route to meet specific requirements. Dynamic programming has been used to search for the best fuel and time-saving model, utilizing the convective weather avoidance model [60]. The development of dynamic weather reroutes (DWR) [63; 64; 65] is another approach to studying the weather impact on TP. It is proposed as a ground-based concept to automatically and efficiently propose short-time trajectories avoiding convective weather regions for in-flight aircraft in the en-route airspace [63]. Based on this concept, the system Terminal AutoResolver was deployed to find the auxiliary waypoint of the optimal path around the convective weather polygon [65]. A tactical rerouting model around convective weather was proposed in [61], and their later work found flight information further improves the prediction results [62].

Machine Learning Methods

The ML-based approach is also widely used for TP. Due to the nature of ML techniques, most of the work utilized a massive amount of real history recorded data for the model

training. A model using GLMs to perform TP is proposed in [66]. The model is trained with labeled inputs such as aircraft type, ground speed, altitude, surface wind, etc. The prediction is made through a step-wise regression. Recent work on dropout-based BDL is proposed for TP-based safety assessment in [67], which adopts dropout-based BDL to predict the trajectory deviations. Based on the predictions, this chapter proposes a method for aircraft safety assessment.

Weather Impact on Machine Learning Methods: Weather information is included and modeled as grids or cubes in several research papers [19; 68; 69; 70]. The airspace is considered as a 3D grid network and the weather information is stored in each grid in [19]. The raw trajectories are aligned with weather information and made these grid 4D joint cubes, which contains the latitude, longitude, altitude, and the weather condition (representing the time dimension). Then an HMM model is used to predict trajectories with weather uncertainties. Another study using a deep generative convolutional recurrent neural network approach for 4D trajectory prediction is the first paper using an encoder-decoder recurrent neural structure for TP [68]. The paper proposes an end-to-end convolutional recurrent neural network that consists of a long short-term memory (LSTM) encoder network and a mixture density LSTM decoder network. The model can predict the aircraft 4D trajectories using high-dimensional weather features and last filed flight plans. The prediction error metrics show that average absolute horizontal errors are around 50 nautical miles and 2800 feet for average vertical errors. Our previous work on TP follows this direction, in which a conditional generative adversarial network (CGAN) is proposed for TP within each TRACON [69] using the data from the Sherlock Data Warehouse (SDW). The convective weather data are stored in a high-dimensional tensor. The drawbacks of CGAN are training difficulties and the source of uncertainty comes from the sampled input. Another previous work also adopts the dropout-based BDL for probabilistic trajectory prediction and shows effectiveness [70]. However, the dropout ratio is defined manually during the train-

ing and the predicted TP uncertainties are strongly correlated with this model parameter. No theoretical method is available to determine the dropout ratio and the trial-and-error is performed. Thus, the uncertainty associated with this subjective ratio is inevitable. A more reliable and robust approximation for the BDL model is needed.

2.3 Bayesian Deep Learning

In this section, I will give a brief review of the concept and mechanism of NN in Section 2.3.1. Following this, I will describe the formulation of approximate variational inference (VI) to solve Bayesian neural network, which treats NN weights as probabilistic distributions in Section 2.3.2.

2.3.1 Deterministic TP: Deep Learning

The deep learning model is defined as the stacking of many NN layers as a deep model to perform supervised learning, semi-supervise learning, or unsupervised learning task. Different types of deep learning models may consist of different types of layers. Feedforward neural network consists of multiple fully-connected layers. Recurrent neural network (RNN) consists a stack of recurrent layers (long short-term memory (LSTM) [71], gated recurrent unit (GRU) [72]) and full-connected layers. Similarly, a convolutional neural network (CNN) is the combination of convolutional layers and fully-connected layers [73]. The recent work built a recurrent-convolutional neural network (RCNN) model by integrating deterministic CNN and RNN for spatial-temporal learning [74].

Fig. 2.1 shows a simple fully connected neural network (FCNN) architecture. The inputs are X_1 and X_2. The output is Y. The hidden units are H_1, H_2, H_3 and H_4. The arrows represent the parameters or weights of the model. Typical NN parameters include weights, and also bias terms. The fully connected layer is named as such because each output of the fully connected layer is a weighted sum of each input element. In Eq. 2.1,

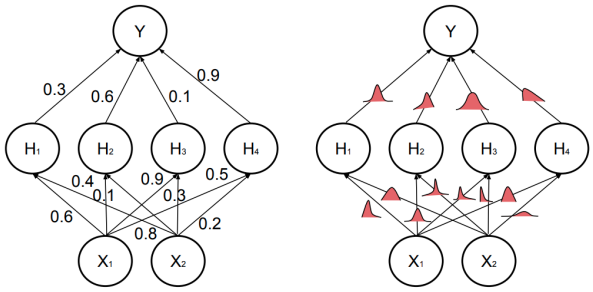


Figure 2.1: Left: Classical Neural Network, Each Weight has a Fixed Value. Right: Bayesian Neural Network, Each Weight is Represented by a Distribution.

the input is X and the parameters are the weight W and the bias b. f_{FC} stands for a fully-connect layer in NN.

$$f_{FC}(X) = X * W^T + b \tag{2.1}$$

The architecture shown in Fig. 2.1 has three layers. The input layer, the hidden layer, and the output layer. The hidden layer has two input nodes, the hidden layer has three nodes and the output layer has one node. The output of the neural network is passed through an activation function to cap the output into the desired range. The optimization objective is defined as the difference measure between the NN output and the true label data. I minimize the optimization objective to get the best parameters that can generate the closest output to the true labels. This process is also called *training*. The prediction output is made by passing the test data inputs into the NN with the best parameters.

The formulation of CNN follows a similar procedure as feedforward NN but with convolutional operations as shown in Eq. 2.2. The symbol $\otimes$ stands for convolution operations. In practice, learning the function with fully connected layers may be difficult because of the large number of parameters. For example, if our input is a $224 \times 224 \times 3$ image, and our fully connected layer has a modest output size of 500, then it will need $3 \times 224 \times 224 \times 500 \approx 75$ million weights for a single layer. This is not desirable as our dataset may has much fewer

samples than this, so the model will suffer from underfitting. One solution to this problem is to utilize weight sharing, of which convolutional neural networks are one popular type.

$$f_{CONV}(X) = X \otimes W^T + b \tag{2.2}$$

Convolution operations are useful to model the spatial data, when I expect our data to exhibit translation invariance in some dimensions, such as the image data. For example, if I am trying to detect a car, it would be useful to detect wheels. These wheels could be anywhere in the image, so I would like our wheel detector to be translationally invariant. As in fully connected networks, a stacked network of convolutional layers can learn complex functions of the input data.

The convolution operation uses a kernel, or filter, to compute outputs. In convolutional networks for image recognition, the filter size is usually small (3×3 or 5×5 for 2D convolution). The weights of the filters are learnable parameters. Additionally, the convolutional layer usually incorporates a learnable bias term, which is added to the filter output. As the filter slides across the input, outputs are generated for each input location. At the borders of the image, input data for the convolution is missing. This data can be replaced with zeros (zero paddings), or the convolution output at these locations can be ignored (valid paddings). RNN is relatively a complicated flow of computation. LSTM is a special form of RNN and achieves great success in the learning of sequence data. GRU is viewed as a simplified version of LSTM. All recurrent neural networks have a repeating module of the neural network called the recurrence. I list the calculation of one LSTM recurrence here.

$$f_t = \sigma(W_f \cdot h_x + b_f) \tag{2.3}$$

$$i_t = \sigma(W_i \cdot h_x + b_i) \tag{2.4}$$

$$\hat{c_t} = tanh(W_c \cdot h_x + b_c) \tag{2.5}$$

$$c_t = f_t \cdot c_t + i_t \cdot \hat{c_t} \tag{2.6}$$

$$o_t = \sigma(W_o \cdot h_x + b_o) \tag{2.7}$$

$$h_t = o_t \cdot tanh(c_t) \tag{2.8}$$

The LSTM cell consists of three cell gates, the forget gate in Eq. 2.3, the input gate in Eq. 2.4, and the output gate Eq. 2.7. These three gates are stringed together by the cell state tensor as Eq. 2.5 and Eq. 2.6. The forget gate is to decide whether the value needs to be passed through the cell state or not. The input gate is to control if the value needs to be stored in the cell state. The output of the model is decided by the calculation in Eq. 2.7 and Eq. 2.8. The readers should notice the usage of the activation functions σ and $tanh$ here. The σ and $tanh$ push the output into the range (0, 1) and (-1, 1), respectively. One of the key ideas behind LSTM is to connect the previous state to the current state, which is useful to model the complex sequential dependency. This is accomplished by the hidden tensor and the cell tensor through the calculation of three different gates.

To avoid overfitting, various stochastic regularization techniques (SRTs) are developed such as the use of early stop, data augmentation, parameter penalties, dropout, and Bayesian regularization. The network with Bayesian regularization is prone to resist overtraining and overfitting because the prior information provides a natural tendency to select simpler models. Furthermore, the Bayesian criterion provides a natural way of stopping training and tuning parameter selection [75].

2.3.2 Probabilistic TP: Bayesian Deep Learning

The formulation of Bayesian Deep Learning relies on Bayesian probabilistic modeling. The simple idea is to have stochastic NN parameters, as the right part in Fig. 2.1. For a regression task given a training input sequence $X = \{x_1, ..., x_n\}$ and their corresponding output sequence $Y = \{y_1, ..., y_n\}$, I try to find the parameters ω for the approximation function $y = f^{\omega}(x)$ that are *most likely* to generate the outputs. That is the inference of $p(\omega|X, Y)$.

Table 2.1: Notations in BDL

Notations	**Meaning**
X	The training inputs.
Y	The prediction outputs.
ω	The model parameters.
x^*	The new observed inputs.
y^*	The predictions corresponding to x^*.
q_θ	The approximated variational distribution

$$\overbrace{p(\omega|X,Y)}^{\text{Posterior}} = \frac{\overbrace{p(Y|X,\omega)}^{\text{Likelihood}}\overbrace{p(\omega)}^{\text{Prior}}}{\underbrace{p(Y|X)}_{\text{Evidence}}} \tag{2.9}$$

The Bayesian approach gives a space of parameters ω as a distribution $p(\omega)$ called the *prior*. The *prior* is defined based on prior knowledge and gives us an adversarial of what parameters are likely to generate our data before I have any data points. The Bayesian approach also defines a likelihood distribution $p(y|x, \omega)$, which is a probabilistic model of the model outputs given the data points and model parameters. The normaliser in Eq. 2.9, $p(Y|X)$, is the evidence [76]. The posterior $p(\omega|X, Y)$ is evaluated with Eq. 2.9.

For the prediction of the model, if I have a new observed data input sequence x^*, the predicted output sequence y^* is a distribution, marginalized over the posterior, as in Eq. 2.10. I can also view it as a weighted average of the model where the weights are determined by the posterior distribution of ω, mathematically, $\mathbb{E}_{p(\omega|X,Y)}[p(y^*|x^*,\omega)]$. This is also equivalent to using an ensemble of NNs for prediction. Unfortunately, this is intractable for any practical case [77].

$$p(y^*|x^*, X, Y) = \int p(y^*|x^*, \omega)p(\omega|X, Y)d\omega \tag{2.10}$$

The normal way to define the prior distributions is to place a standard Gaussian distribution over each of the NN weights W while the bias b is a deterministic value [78]. Due to the non-conjugacy and non-linearity of the complex structure of NNs, the closed-form derivation for Bayesian inference to posterior is intractable. On the other hand, the traditional sampling-based methods lead to surprising prohibitive computational complexity, and lack of scalability to large scale practical application problems.

Variational inference, as an approximated posterior inference algorithm, has been applied to the posterior inference of Bayesian neural networks. First, I define an approximated variational distribution $q_\theta(\omega)$ for the posterior $p(\omega|X, Y)$. Kullback-Leibler (KL) divergence [79] is adopted here to measure the difference between these two distributions. Thus minimizing the KL divergence w.r.t. θ leads to the best approximated variational distribution.

$$
\begin{aligned}
& KL(q_\theta(\omega)||p(\omega|X,Y)) \\
&= \int q_\theta(\omega) log \frac{q_\theta(\omega)}{p(\omega|X,Y)}\, d\omega \\
&= \int q_\theta(\omega) log \frac{q_\theta(\omega) p(X,Y)}{p(\omega,X,Y)}\, d\omega \\
&= \underbrace{\log p(X,Y)}_{\text{Constant}} - \underbrace{\int q_\theta(\omega) log \frac{p(\omega,X,Y)}{q_\theta(\omega)}\, d\omega}_{\text{Evidence Lower Bound (ELBO)}}
\end{aligned}
\tag{2.11}
$$

After rearranging Eq. 2.11, the minimization of KL divergence is equivalent to minimize the negative Evidence Lower BOund (ELBO).

$$
\begin{aligned}
-ELBO :&= KL(q_\theta(\omega)||p(\omega)) - \int q_\theta(\omega)\, log(p(Y|X,\omega))\, d\omega \\
&= \underbrace{KL(q_\theta(\omega)||p(\omega))}_{\text{Prior dependent}} - \underbrace{\mathbb{E}_{q_\theta(\omega)}[log(p(X,Y|\omega))]}_{\text{Data dependent}}
\end{aligned}
\tag{2.12}
$$

The negative ELBO is a sum of a prior dependent part and a data-dependent part (Eq. 2.12). The prior dependent part can be referred to as the complexity cost and the data-dependent part can be viewed as the likelihood cost. The negative ELBO embodies a trade-off between meeting the complexity of the dataset (X, Y), and satisfying the simplicity of prior [77]. Again, the minimizing of negative ELBO is intractable but luckily approximations and gradient-based methods can help us to find the optimum during inference.

MC integration is introduced to approximate the data-dependent part. The basic concept of MC integration is to substitute integrals with summations. The mean-field assumption of variational inference is used in MC integration, for the sake of simplicity. The mean-field variational family [80] assumes each of the latent variables is mutually independent and each is governed by a distinct factor. In Eq. 2.13, each of the latent variable ω_i is governed by its own variational factor, the density $q_{\theta_i}(\omega_i)$.

$$q_\theta(\omega) = \prod_{i=1}^{m} q_{\theta_i}(\omega_i) \tag{2.13}$$

Under this assumption, the data-dependent part can be evaluated in a relevant simple manner. Specifically, I sample $\hat{\omega}$ from the variational distribution $q_\theta(\omega)$ and optimize towards the objective function w.r.t. θ in Eq. 2.14 each time. By doing this recursively, I find the best $q_\theta(\omega)$ that approximated the true posterior. The final objective, or loss function for NN, is defined as the summation of KL divergence between the prior and the approximated variational distribution, and the negative log-likelihood (NLL). The method that samples the weight of a neural network stochastically at training time is referred to as the weight perturbation method [81].

$$-\widehat{ELBO} := \underbrace{KL(q_\theta(\omega)||p(\omega))}_{\text{KL divergence}} \underbrace{-log(p(X,Y|\omega))}_{\text{Negative Log-Likelihood (NLL)}} \tag{2.14}$$

Once the training process is finished, the uncertainty can be estimated through MC tests as in Eq. (3.13). K is the number of tests performed.

$$\begin{aligned} p(y^*|x^*, X, Y) &= \mathbb{E}_{p(\omega|X,Y)}[p(y^*|x^*, \omega)] \\ &\approx \frac{1}{K}\sum_{k=1}^{K} p(y^*|x^*, \hat{\omega_k}) \end{aligned} \tag{2.15}$$

Gal and Ghahramani showed that a neural network with dropout applied before every layer is mathematically equivalent to the Bernoulli approximate VI of NNs [30]. The following work of them shows similar behavior to convolutional layers [82], and recurrent layers with variational dropout applied [83]. The dropout approximation to Bayesian NNs is also known as the Monte Carlo Dropout (MCDropout) method. This work draws large attention because only minimal change is needed to switch the current deep learning models into Bayesian deep learning models. The uncertainty is estimated by simply applying dropout during the testing procedure. Consequently, abundant applications of dropout

approximation to a Bayesian posterior raised, as well as in the field of ATM [67; 70]. However, researchers have queried MCDropout [84; 85; 86; 87]. A few major concerns are,

- The improper prior in variational dropout leads to irremediably pathological behavior of the true posterior.
- The dropout ratio defined manually leads to arbitrarily poor decision making.
- The unsatisfactory on lack of theoretical grounding for dropout, without which the choice of dropout variant remains arbitrary.

A reliable uncertainty estimate is critical for enhancing the safety in aviation operations, such as uncertainty quantification of TP for early trajectory conflict awareness. It's better to constraint the KL divergence explicitly in the objective functions, as in Eq. 2.14. Weight perturbation methods such as the local reparameterization trick (LRT) [88] and Flipout [81] has enabled the possibility of optimizing towards Eq. 2.14 directly using sampling and gradient-based optimization during inference. The recent development of probabilistic programming languages build the pathway of fast and scalable application of neural network uncertainty estimations when dealing with real-world big data [89; 90; 91; 92].

2.4 Proposed Methodology for TP under Uncertainties

The problem definition and overview of the proposed method will be discussed in Section 2.4.1. Fig. 2.2 visualizes the innovative information fusion pipeline of the proposed problem-solving framework. The architecture can be divided into three modules, the data processing and filtering module, the feature extraction module, and the Sequence to Sequence (Seq2Seq) learning module. The data processing module is introduced in Section 2.4.2. Then the weather feature extraction module and the Seq2Seq learning module are described in Section 2.4.3. Finally, the implementation and model inference details are discussed in Section 2.4.4.

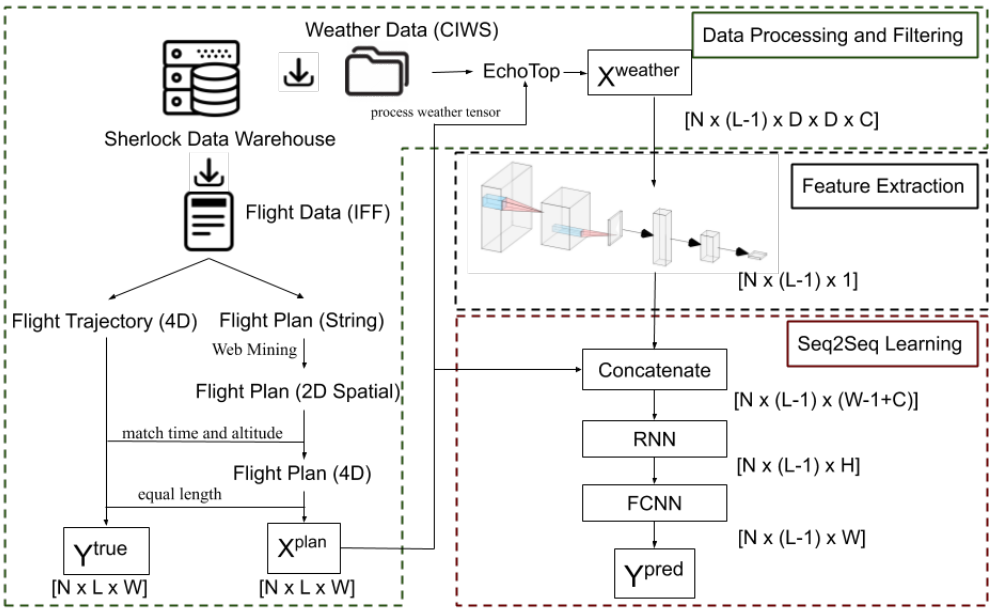


Figure 2.2: Overview of the Proposed Trajectory Prediction Framework. The ML Pipeline Consists of the Data Processing and Filtering Module, Feature Extraction Module, and Seq2seq Learning Module.

2.4.1 Overview

Problem Definition

First, I would like to introduce several abbreviations involved, as shown in Table. 2.2.

The following data is used in our problem:

- **Trajectory data** Y^{true}, which is the actual trajectory of the aircraft, with dimension $[N \times L \times W]$.
- **Flight plan data** X^{plan}, which refers to the processed last on-file flight plan of the aircraft prior to takeoff, with dimension $[N \times L \times W]$.
- **Weather Data** $X^{weather}$, which refers to the processed look ahead convective weather data around each position of the y^{plan}. The dimension of cube is $[N \times L \times D \times D \times C]$.

Table 2.2: Abbreviations

Notations	**Meaning**
N	Number of Samples/The number of flight trajectories in each sector. The total number of the data is separated for model training and testing. N is different for different sections.
L	Length of sequence. L represents the number of aircraft positions in one recorded trajectory, corresponding to one flight call sign. L is set to be 50 in our case, for the simplicity of the demonstration.
W	The dimension the prediction of the flight trajectories. W is 3 in our case, which represents the latitude, longitude, and altitude dimension.
D	Spatial resolution of the weather window. In our case, D is set to be 32, which implies that the weather window is 32×32. The selected window size is based on the approximated distance to cover the look-ahead region at the current coordinates.
C	Channel number of weather features. C is 1 in our experiments, which only includes the convective weather EchoTop values. Wind, pressure, humidity, etc, can be added to expand the channel number.
H	Hidden dimension of recurrent cell. The number of recurrence in the recurrent layer. A parameter to be determined during the parameter tuning process. 64 is determined in our case.

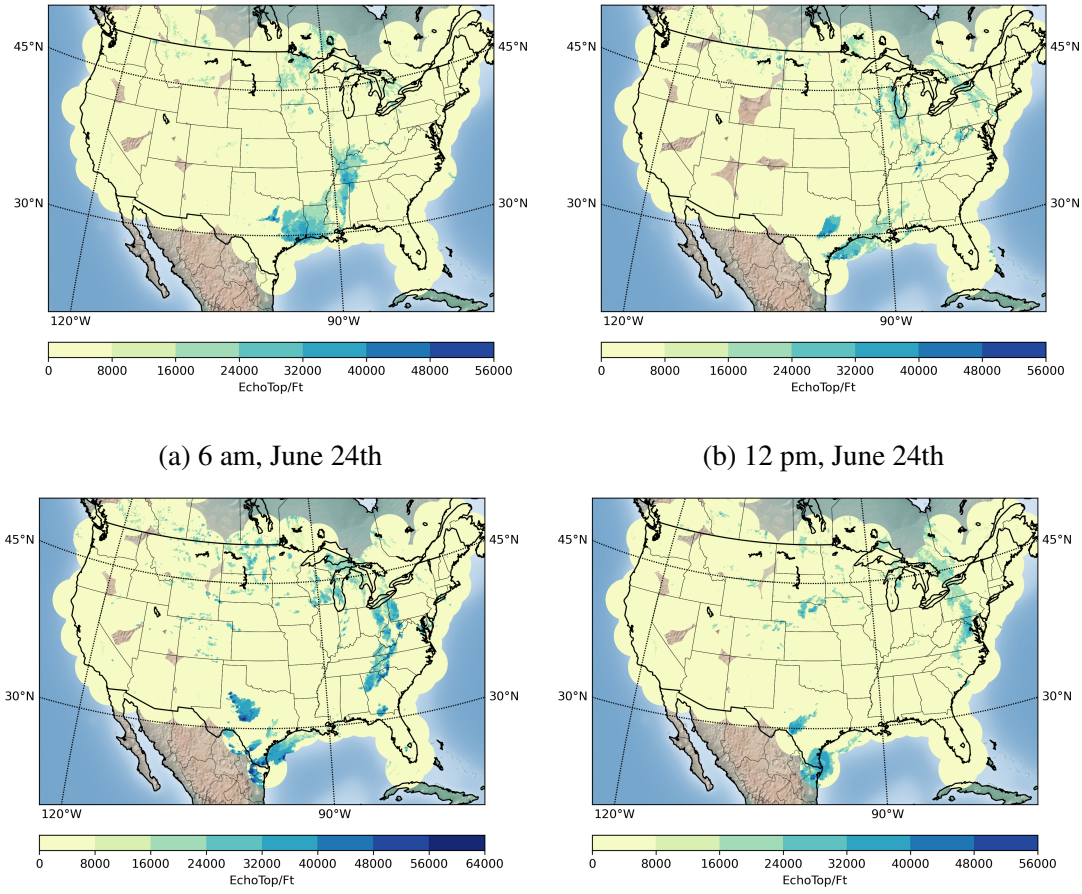
A more detailed description of the used data is given in Section 2.4.2. Finally, the following output sequence will be given from the proposed neural network model:

- **The predicted trajectory** Y^{pred}, with the dimension $[N \times (L-1) \times W]$.

The model aims to perform strategic TP based on the last on-file flight plan and weather forecast prior to takeoff. According to this, the TP task is defined as predicting Y^{pred} based on the last on-file flight plan X^{plan} and weather conditions $X^{weather}$ along the planned route X^{plan}. Thus, the model inputs are $X^{weather}$ and X^{plan}, the model output is Y^{pred}. The objective is to minimize the distance between Y^{pred} and Y^{true}.

Both Deterministic TP and probabilistic TP models are implemented and evaluated under the same problem definition and dataset. Probabilistic TP model includes both the MC-Dropout approximation of BDL and the VI based BDL. A comparison study is performed on various methods in the Sec. 2.5 and Sec. 5.6.

2.4.2 *Data Processing and Filtering Module*



(a) 6 am, June 24th

(b) 12 pm, June 24th

(c) 6 pm, June 24th

(d) 12 am, June 25th

Figure 2.3: EchoTop Convective Weather Visualization on June 24th, 2019 at (a) 6 am; (b) 12 pm; (c) 6 pm; (d) 12 am.

The data used in this research is obtained from the Sherlock Data Warehouse (SDW) [20]. It is a platform for reliable aviation data collection, archiving, processing, query, and delivery to support ATM research. Data of Sherlock comes primarily from the federal aviation administration (FAA) and the National Oceanic Atmospheric Administration (NOAA) [93]. Multiple sources of raw surveillance data are processed and stored in SDW. Here

this work only uses data from two sources. The Integrated Flight Format (IFF) flight data and EchoTop (ET) convective weather data from CIWS. To properly address the impact of weather on trajectory prediction, this work looks into the historical weather recordings for a specific day with convective weather presence. The NOAA's storm prediction center reported convective weather was formulated, with tornado and high winds in the north-east U.S. airspace on June 24th, 2019.

Flight Data Processing

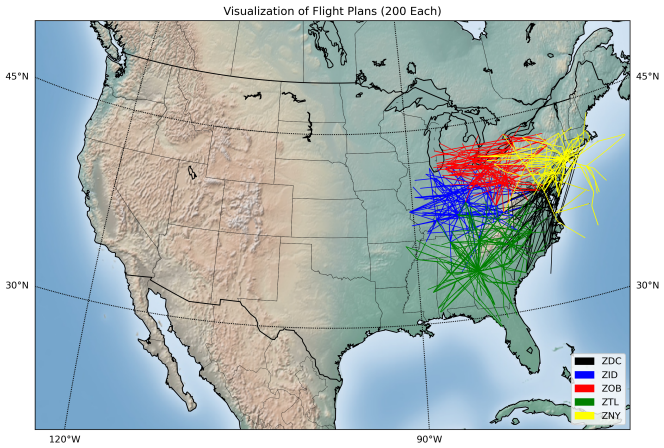


Figure 2.4: Visualization of Sampled Flight Plans in Different Sectors on 06/24/2019

The flight data is collected from different FAA facilities and stored in IFF format. It includes all raw source data plus the derived fields such as flight summary, track points, and flight plan. The flight summary is a general description of the flight, which contains flight time, flight call sign, aircraft type, origin, and destination information. The flight track points are the record of real flight operations. It includes the ground-measured aircraft position in both the spatial and temporal domains. The flight plan comes as a string of waypoints. I developed a web-based mining tool to translate it into WGS84 coordinates.

I choose five air traffic control centers for our demonstration since most of the tornadoes and high winds are reported around this area. Fig. 2.4 visualizes the flight tracks flying inside these control centers. These flight tracks with reference to the convective weather conditions are shown in Fig. 2.3.

Fig. 2.2 shows the procedure of processing sector flight data. The flight tracks are extracted from the sector IFF file and a linear interpolation is performed for each of the track points with a 1-second interval. The waypoint of each flight plan are removed if the waypoint is outside the investigated flight control centers in this study. The Euclidean distance is used to evaluate whether waypoints with respect to the current sector range. The same interpolation is performed for the flight plan to make it consistent with the track points. Finally, the sequence from the track points and flight plan interpolation are used in the proposed model.

Weather Data Processing

The weather data are obtained from Corridor Integrated Weather Systems (CIWS) in SDW [94]. CIWS is designed to improve convective weather decision support for congested en-route airspace. The two key features of CIWS, i.e., EchoTop (ET) and Vertically Integrated Liquid (VIL), come with the current and forecast dataset in the Sherlock database. EchoTop numbers are estimates in feet of the highest cloud tops associated with the radar echoes. The higher cloud tops usually indicate more intense precipitation. VIL is an estimate of the total mass of precipitation in the clouds. The measurement is obtained by observing the reflectivity of the air which is obtained with weather radar. Reflectivity represents the intensity of radar echoes returning from clouds. ET and VIL are updated and documented every 150 seconds, which results in 576 files daily.

In this research, I only use the current ET data for demonstration. At each given flight plan point in Y^{plan}, a weather cube ahead of the current location is selected from the origi-

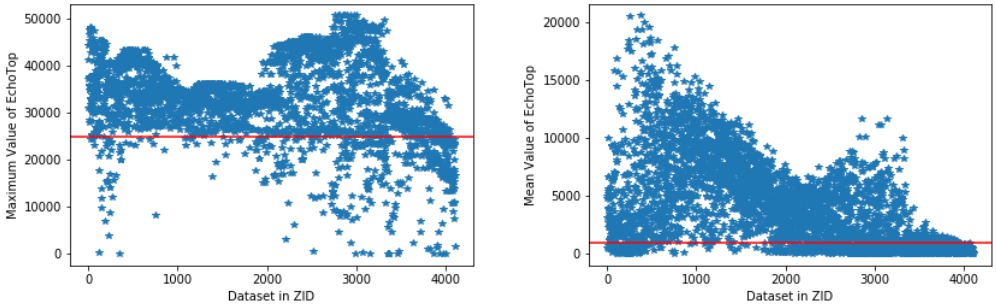
nal weather file. The cube direction is rotated to align with the heading angle of the aircraft. This is the same weather cube generator used in our previous work [95]. The cube size is selected to be 32×32. The reason for this selection is that the size will cover the look-ahead region for decision-making of possible weather avoidance. Thus the final dimension of the weather feature cubes is $N \times 49 \times 32 \times 32 \times 1$ and the dimension of the flight tracks and flight plans are $N \times 49 \times 3$, where N is the number of data processed for each sector. It should be noted that the weather data of the starting point in Y^{plan} is not processed, as it is the current location of the aircraft.

Data Filtering

Past study on the convective weather dataset indicates that only the flight tracks that have encountered convective weather ET of 25,000 feet or continued for at least 2 minutes are classified as weather encountered deviations [96]. In considering this, the flight track and convective weather dataset are filtered by the maximum and mean value of ET of each data pair, as shown in Table. 2.3. The threshold value for maximum value ET and mean value ET are 25,000 and 1,000, respectively. A visualization of the data screening process is shown in Fig. 2.5. The x-axis stands for the 4,103 pairs of data for sector ZID and the y-axis represents the maximum and mean values of each data pair. The red line visualizes the threshold values used in different metrics.

Table 2.3: Weather Encounters Classified for Different Sectors

	Total	**Weather encounters**	**Percentage**
ZID	4,103	2,817	68.7%
ZTL	6,255	3,642	58.2%
ZNY	6,135	3,757	61.2%
ZDC	5,420	2,907	53.6%
ZOB	4,771	2,333	48.9%



(a) Maximum EchoTop

(b) Mean EchoTop

Figure 2.5: Data Filtering with Threshold on EchoTop. Visualizing of ZID.

2.4.3 Feature Extraction Module and Seq2Seq Learning Module

The feature extraction module and Seq2Seq learning module can perform deterministic TP with classical deep learning layers as introduced in Section 2.3.1. They can also be easily changed into probabilistic TP when the Bayesian theory is integrated with Deep Learning as shown in Section 2.3.2. Feature extraction is performed to extract the indication of weather conditions at each point within the entire sequence. The extracted weather feature is combined with the flight plan as the input to the Seq2Seq learning module. The feature extraction module and Seq2Seq learning module form the complex spatial and tem-

poral structures of the weather information and flight trajectory. These two models can be made deterministic for the point estimate of flight trajectories or probabilistic to quantify the flight trajectory uncertainty.

The weather feature extraction module includes two three-dimensional convolutional layers and two fully-connect layers. The convolutional layers are used to extract useful features from $cube$ into a vector. The extracted weather vector will be concatenated with the flight plan tensor y^{plan} as the input to the Seq2Seq learning module. The Seq2Seq learning module consists of one recurrent layer and three fully-connect layers, which have been shown to have a supreme performance on sequence learning tasks in the literature [97]. The concatenation layer connects the feature extraction module and the Seq2Seq learning module. The prediction y^{pred} is the output from the last fully-connect layer. A detailed introduction of these two modules is also shown in Table. 2.4. In this table, N represents the batch size and empty cell means not applicable. The layer names that use bold font means need input.

The first input to the model is the weather cube tensor $cube$ with dimension $[N, 49, 32, 32, 1]$. The Conv3d_1 layer has a kernel size of $[1, 5, 5, 3]$ and a stride of $[1, 3, 3, 1]$ along the last four dimensions of the input tensor with no padding added for convolution operation. The output has a dimension of $[N, 49, 10, 10, 3]$. The second convolutional layer Conv3d_2 has a similar setup as the first one with a kernel size of $[1, 3, 3, 1]$, a stride of $[1, 3, 3, 1]$, and zero padding, respectively. The output has a size of $[N, 49, 16]$ after flattening along the last three dimensions of the tensor. The following two dense layers would compress the tensor to a dimension of $[N, 49, 1]$, which is the dimension of the weather feature tensor. I choose to concatenate it with the flight plan tensor p as the input to the RNN cell. The dimension of LSTM hidden state h_t and cell state c_t tensor is 64. Zero initialization is used in the LSTM cell. The output of the LSTM cell will be fed into three dense layers to calculate the output of the model. The training objective is defined based on Eq. (2.14)

which incorporates two parts: the KL divergence and the NLL.

Table 2.4: Architecture of Feature Extraction Module and Seq2Seq Learning Module with Processed Recorded Dataset

Layers	**Input Size**	**Output Size**	**Parameters**
Conv3d_1	[N, 49, 32, 32, 1]	[N, 49, 10, 10, 3]	Kernel: [1, 5, 5, 3], Stride: [1, 3, 3, 1], No Padding
Conv3d_2	[N, 49, 10, 10, 3]	[N, 49, 4, 4, 1]	Kernel: [1, 3, 3, 1], Stride: [1, 3, 3, 1], Zero Padding
Flatten	[N, 49, 4, 4, 1]	[N, 49, 16]	Keep first two dimension
Dense_1	[N, 49, 16]	[N, 49, 4]	4
Dense_2	[N, 49, 4]	[N, 49, 1]	1
Concat	[N, 49, 1]	[N, 49, 4]	Concatenate with flight plan p
LSTM	[N, 49, 4]	[N, 49, 64]	64
Dense_3	[N, 49, 64]	[N, 49, 32]	32
Dense_4	[N, 49, 32]	[N, 49, 16]	16
Dense_5	[N, 49, 16]	[N, 49, 3]	3

2.4.4 Integration and Inference

Metrics for Performance Evaluation

For the deterministic TP model, the training objective is defined as the minimization of the difference between the predicted sequence Y^{pred} and the ground truth sequence Y^{true}, as in Eq. (2.16). It is defined as the root mean squared error (RMSE) between the two sequences. K is the size of the training dataset.

$$RMSE = \sqrt{\sum_{k=1}^{K} \frac{(Y_k^{pred} - Y_k^{true})^2}{K}} \tag{2.16}$$

For the probabilistic TP model, the training objective is derived in Eq. (2.14). This can be reformulated to accommodate our proposed framework as in Eq. (2.17). The KL-divergence of the variational posterior $q_\theta(\omega)$ and prior $p(\omega)$ for each of the parameter ω

can be calculated analytically if a normal distribution is assumed for both of them. The NLL term is the RMSE loss defined above. Once the inferences of network parameters are finished, the mean predictions and corresponding uncertainties can be evaluated by MC tests, as in Eq. (3.13).

$$-ELBO = KL(q_\theta(\omega), p(\omega)) + RMSE \tag{2.17}$$

Implementation Details

The entire dataset is separated into the training set and the testing set with a ratio of 0.8 and 0.2. The training set is used in the model training process solely and the testing set is for model testing after the training procedure. Furthermore, the validation set is sampled from the training set to evaluate the model performance during the training process. Data normalization and denormalization are performed prior to and after the model training.

Adam optimizer [98] is used to minimize the objective until the negative ELBO converges. The initial learning rate is set to be $1e^{-3}$ by *best practice* during fine-tuning of the model parameters. Learning rate decay follows Eq. (2.18),

$$lr^{s+1} = \frac{lr^s}{(1 + r * s)^q} \tag{2.18}$$

where s is the global step during optimization, the decay rate r is $1e^{-4}$, and the power q equals to 0.75.

The training is performed with the deep probabilistic learning packages Edward [92; 90; 91] and Tensorflow-probability [89] based on Tensorflow 2.3.0 on a workstation with Intel Xeon E5-1620 v4 @3.50 GHz CPU and accelerated with a single NVIDIA GTX 1080 GPU.

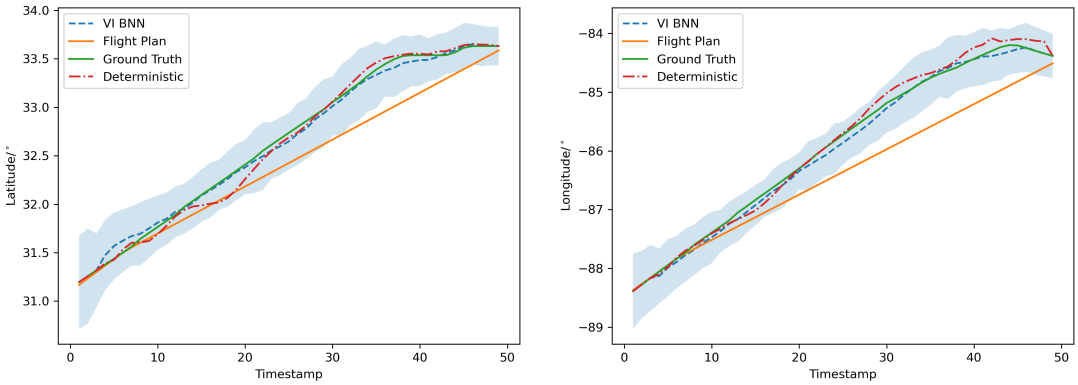
The comparison with the MCDropout method and deterministic model is conducted with the same model parameters and training setups. The use of dropout operation can

be summarized as, (a) applying regular dropout after each convolutional and dense layers; (b) performing variational dropout [83] to LSTM cell. It should be noted that there is no dropout operation after the last dense layer. The dropout ratio used in this chapter is set to 0.9.

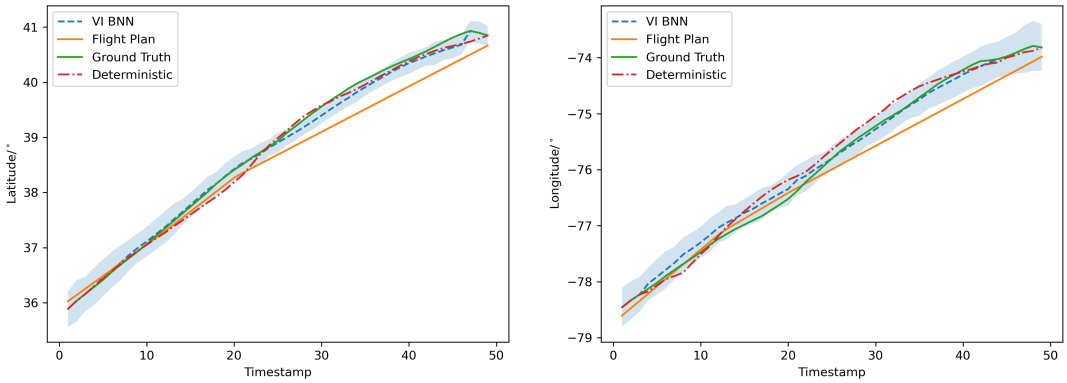
2.5 Demonstrations and Results

In this section, I first evaluate the prediction of the TP framework individually with the test data in Section 2.5.1. This includes discussions on several good prediction results and different reasons for failed cases. Next, a statistical study for the entire test dataset is performed to evaluate the overall TP uncertainty reduction performance in Section 2.5.2. The comparison of all methods mentioned in the above sections is also performed.

2.5.1 *Individual Evaluation*

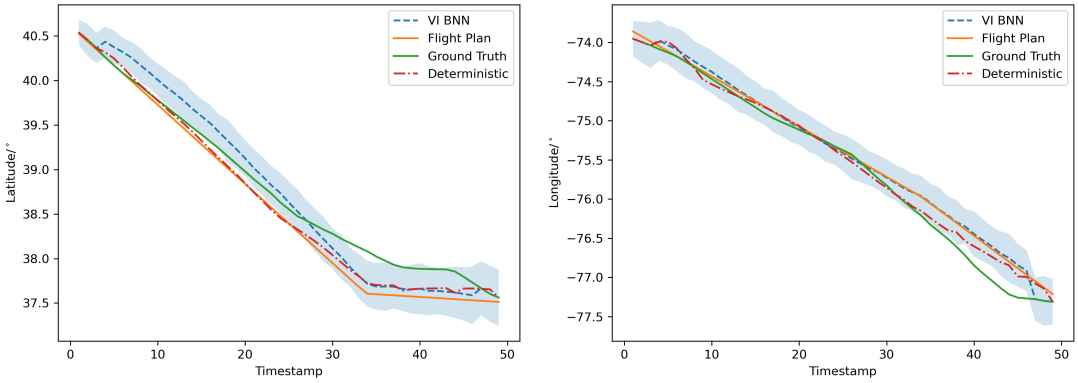


(a) Case 1: Both deterministic and probabilistic predictions show good prediction capabilities. The VI BNN prediction has smaller oscillations compared to deterministic prediction due to Bayesian regularization.

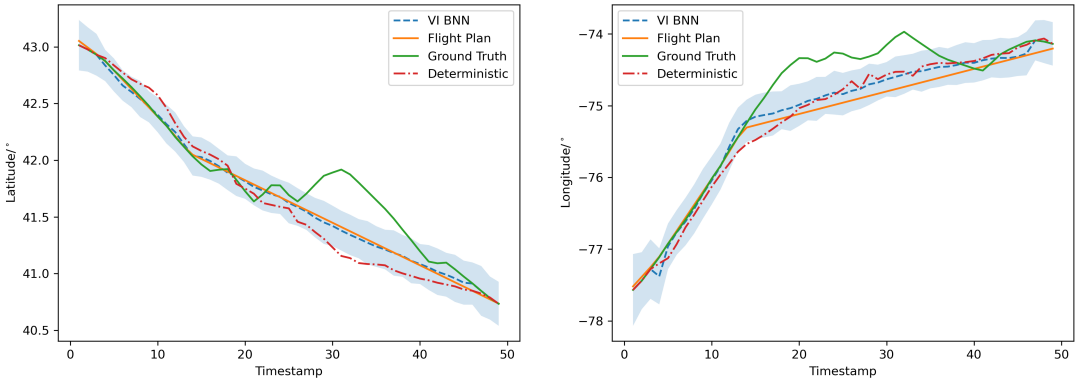


(b) Case 2: Both deterministic and probabilistic predictions show good prediction capabilities. The longitude shows a considerable accuracy compared to the latitude, potentially due to the dynamic formulation of the aircraft motion.

Figure 2.6: The Model Performs Good by Making Predictions Closer to The Ground Truth. The 95% Confidence Bound for VI BNN is Colored in Blue. The Start Point (Timestamp 0) is not Visualized.



(a) Case 3: The flight plan has a waypoint at timestamp 34 ignored by the ground truth. This waypoint has a large impact on model predictions. However, the 95% confidence interval of VI BNN covers most of the ground truth.



(b) Case 4: The ground truth has a large deviation due to severe weather conditions ($\approx 1^\circ$ latitude/longitude) with sharp turns starting from timestamp 14. The deterministic prediction fails at this case while the VI BNN still exhibits benefits from the confidence interval coverage.

Figure 2.7: The Model Fails at Making a Good Prediction Due to Various Reasons. The 95% Confidence Bound for VI BNN is Colored in Blue. The Start Point (Timestamp 0) is not Plotted.

Fig. 2.6 and Fig. 2.7 visualize model predictions from the test dataset. In order to better represent the 95% confidence interval on both latitude and longitude dimensions, I plot the latitude and longitude separately. The x-axis is the timestamp for the entire trajectory

sequence. As the inputs to the TP framework, it is shown that the actual flight trajectories Y^{true} (ground truth) deviate from the last on-file flight plans Y^{plan}. On the other hand, our model prediction Y^{pred} discovered the deviation from Y^{plan} and calibrated the outputs towards Y^{true}. The two cases in Fig. 2.6 reveal the effectiveness of the proposed ML pipeline. In case 1, both the deterministic prediction and the VI BNN prediction are closer to the ground truth compared with the flight plan. The 95% confidence interval acts as a reliable safety assurance on the predicted trajectories. The mean prediction of VI BNN shows smaller oscillations which comes from the intrinsic Bayesian regularization of NNs. Case 2 is another test example from the test dataset of ZDC. In this case, I noticed that the longitude shows a higher accuracy compared to the latitude, due to the dynamic formulation of the aircraft motion. This can be further investigated when the aircraft motion recording data (e.g. gyroscope measurements) is available.

On the other hand, Fig. 2.7 shows two typical failed cases of the proposed ML pipeline. In case 3, the aircraft ignored the assigned waypoint at timestamp 34. As the input of the ML pipeline, the flight plan has a large impact on the outputs. Although the overall trajectory deviation is reduced compared with the flight plan, the figures show unsatisfied predictions. In this case, the 95% confidence interval prediction based on VI BNN is able to cover the ground truth most of the time. Case 4 shows an extreme case where a large deviation and sharp turn in the ground truth is presented due to severe weather conditions. This is one of the drawbacks of data-driven TP models [99]. In this case, the deterministic method fails for trajectory deviation reduction. The VI BNN method exhibits benefits with the help of uncertainty quantification (UQ) capabilities. More accurate predictions for these cases will need other information, such as voice communication data between the pilot and controller, to fully explain the discrepancy.

2.5.2 Statistical Evaluation

In this section, I evaluate the effectiveness of the proposed model statistically on the entire test dataset. The key metric used is the percentage of the deviation variance reduction for the mean prediction. I compare the deterministic prediction, and the mean prediction of MCDropout and VI BNN. I also study the contribution of the feature extraction module toward deviation reduction in this section.

Metrics

The equations used for statistical evaluations are shown in Eq. (2.19) and Eq. (2.20). I denote the original deviations as l_{2k}^{ori}, defined as the L_2 distance of the true trajectory y_k^{true} and the flight plan x_k^{plan}. Similarly, the predicted deviation can be defined as the sum of the squared error of the true trajectory y_k^{true} and predicted trajectory y_k^{pred} as l_{2k}^{new}. The index k is the index of the test dataset. In Eq. (2.21), I denote ρ_k as the percentage of deviation variance reduction of the prediction l_{2k}^{new} compared to the original variation l_{2k}^{ori} in the kth test data. The subscripts k, n, and d represent the dimensions of the tensors corresponding to the size of the data, length of the sequence, and DOF for the prediction, respectively. In this evaluation, n is 49, and d is 2 for latitude and longitude dimensions. Eq. (2.21) is the equation I used to calculate the overall variance reduction among the data. The size of ρ is equal to the number of data points with deviation variance reduction.

$$l_{2_k}^{ori} = \sum_{i}^{n}\sum_{j}^{d}(y_{k,i,j}^{true} - y_{k,i,j}^{plan})^2 \tag{2.19}$$

$$l_{2_k}^{new} = \sum_{i}^{n}\sum_{j}^{d}(y_{k,i,j}^{true} - y_{k,i,j}^{pred})^2 \tag{2.20}$$

$$\rho_k = \frac{var(l_2^{ori})_k - var(l_2^{new})_k}{var(l_2^{ori})_k} \tag{2.21}$$

Deviation Reduction and Weather Effects

In Table. 2.5, I show the percentage of samples that can achieve deviation variation reduction (i.e., $\rho_k < 1$) based on the evaluation metrics described. I also examine the contributions of convective weather. This is achieved by omitting the weather feature extraction module in the ML pipeline. For example, using the deterministic model with a weather feature extraction module, 78.5% of the flight trajectory deviations are reduced. The total variance reduction is 22.4% for the test dataset of ZTL. While only 62.3% of the flight trajectory deviations are reduced by 14.5% without the weather data.

I discover that the proposed ML pipeline is effective and shows benefits in all cases, but with different performances for different experiment settings. I also notice that ZTL and ZID have a higher percentage of deviation reductions than the other sectors, especially weather-related reductions. One of the reasons is that they have a higher ratio of weather encounters in flight trajectories. This is verified by checking the movement of the meteorological weather contours in Fig. 2.3 and Table. 2.3. By feeding the weather features into the ML pipeline, the model achieves a higher percentage of flight deviation reduction and a better total variance reduction. Thus, adding the weather feature extraction module improves the model TP performance by explicitly considering weather uncertainties. On the other hand, the prediction without the weather data also shows a considerable deviation reduction. The reason is that our proposed ML pipeline implicitly learns the flight deviations caused by factors other than weather-related reasons from the historical record.

Comparing different methods, I realize that deterministic and VI BNN methods have better performance than MCDropout. There are multiple reasons behind this. It is worth pointing out that the dropout ratio used for MCDropout requires hyperparameter tuning. The training is unstable and hard to converge if I use a large dropout ratio. In contrast, the uncertainty will be underestimated if I choose a small dropout ratio. I perform a hyper-

Table 2.5: Deviation Reduction For Deterministic (NN) and Probabilistic Methods (BNN). The Probabilistic Methods include MCDropout and VI BNN. For Probabilistic Methods, the Results are Evaluated with the Mean Prediction. Note: In $A \mid B$, A is the Result with Weather Features, and B is the Result Without the Weather Features.

Model	**Flight Control Sector**	**Deviation Reduced/%**	**Variance Reduction/%**
Deterministic	ZTL	78.5% | 62.3%	22.4% | 14.5%
	ZID	84.6% | 46.6%	30.3% | 25.7%
	ZDC	54.4% | 60.8%	36.5% | 26.5%
	ZNY	59.6% | 67.1%	57.6% | 43.2%
	ZOB	66.7% | 44.1%	48.9% | 27.5%
MCDropout	ZTL	45.8% | 39.1%	8.7% | 7.4%
	ZID	36.1% | 11.5%	11.2% | 2.3%
	ZDC	25.4% | 23.7%	3.5% | 6.7%
	ZNY	28.7% | 29.7%	9.8% | 5.8%
	ZOB	14.6% | 17.6%	5.0% | 7.8%
VI BNN	ZTL	69.8% | 55.9%	24.1% | 16.4%
	ZID	79.0% | 22.5%	36.7% | 6.7%
	ZDC	60.7% | 52.5%	46.4% | 16.2%
	ZNY	40.6% | 39.0%	28.6% | 18.3%
	ZOB	25.0% | 11.6%	31.6% | 14.6%

parameter search over the dropout ratios, from 0.1 to 0.9 with an interval of 0.1 on a sector, and choose the best-performed dropout ratio (0.9) for all cases. Also, the Bernoulli assumption on NN parameters improves the difference between each draws during sampling thus introducing more fluctuations. In Table. 2.5, the deterministic approach performs slightly better than VI BNN in several flight control sectors. However, I only list results calculated using the mean prediction for probabilistic methods. The confidence interval of VI BNN significantly improves the confidence by inferring airspace coverage in a probabilistic sense. The oscillation behavior discussed in Sec. 2.5.1 of the deterministic method is also

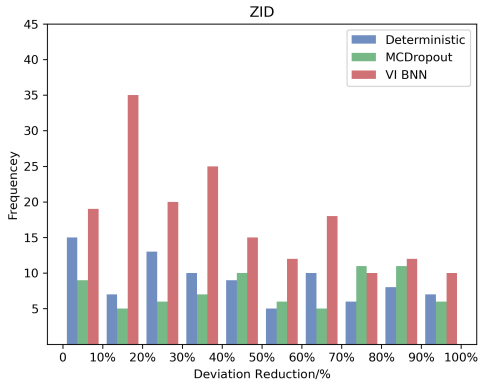
non-negligible.

In conclusion, our proposed ML pipeline shows considerable TP capabilities. The inclusion of convective weather products exhibits enhancements to the overall prediction power. The variational inference-based inference method for Bayesian neural network shows benefits from the aspect of training stability, model regularization, and uncertainty quantification. The deterministic setting also shows effectiveness but with conspicuous deficits.

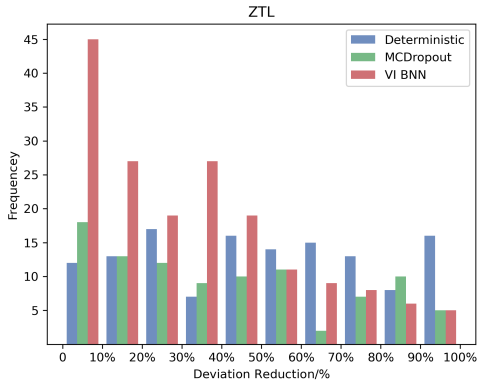
Furthermore, instead of showing only the deviation reductions based on the mean predictions, I compute the histogram of the variation reduction in ρ. To better understand the deviation reduction capability of the model, I visualize the results for all five flight control sectors in Fig. 2.8. For example, using VI BNN with the data of ZID, nearly 20 test samples achieve deviation reduction between 0 to 10%. It's obvious that VI BNN model performs better compared with MCDropout. The VI BNN method is good at generating predictions with lower percentages (e.g. less than 50%) of deviation reductions. And only a few samples have a deviation reduction larger than 50%. The deterministic method also performs well in most of the cases while I am only considering the mean predictions.

2.6 Conclusion

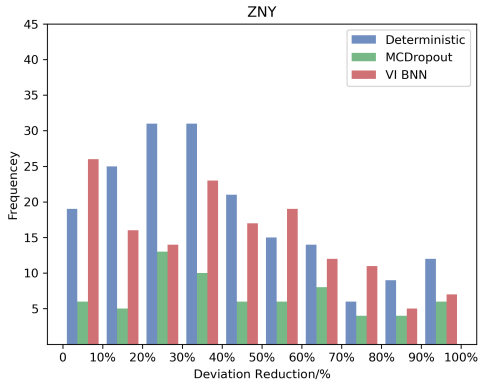
A major key requirement of NextGen pursuing accurate and reliable aircraft trajectory prediction. To accommodate this, I propose a ground-based strategic trajectory prediction Bayesian neural network with explicit consideration of convective weather products. After the model training, I individually and statistically justify the model performance. A comparison study is performed to evaluate the difference between the deterministic and probabilistic methods. Experiments demonstrate that the VI BNN method shows various benefits for our TP approach, with competitive deviation reduction and complexity. Here is a list of major contributions and conclusions on the proposed study,



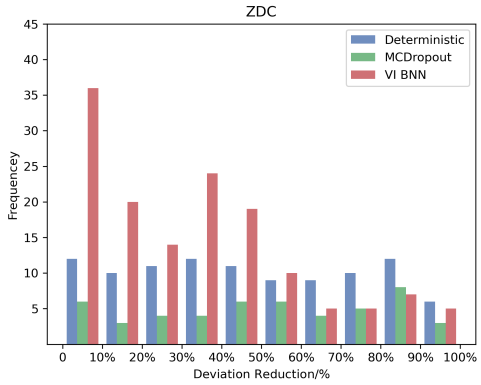
(a) ρ_{ZID}



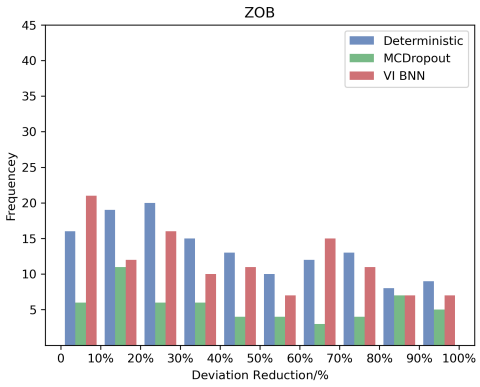
(b) ρ_{ZTL}



(c) ρ_{ZNY}



(d) ρ_{ZDC}



(e) ρ_{ZOB}

Figure 2.8: Histogram Showing Percentage of Overall Deviation Reduction of Deterministic, MCDropout, and VI BNN for Different Sections (a) ZID and (b) ZTL (c) ZNY (d) ZDC (e) ZOB

- The proposed nested CNN+RNN+FCNN structure is able to handle the complex spatial-temporal correlations of aircraft trajectory and weather conditions. It's scalable and flexible for information fusion and time-series modeling with various data sources and measurements.

- The data processing is critical for successful model learning and data discovery. This includes the proper conduct of data filtering and feature engineering based on domain knowledge expertise, either from the literature or human ATC experts.

- VI is shown to have robust performance compared to the MCDropout for BNN which depends on the subjective dropout ratio selection and requires hyper-parameter search. The deterministic method also shows benefits but lacks of uncertainty quantification power.

- Results show that variance reduction of the proposed method happens for both small and large flight deviation cases. The statistical study on deviation reduction indicates that the proposed method is more likely to reduce less than 50% of the deviations regardless of the choice of control sectors. I notice that the ML model implicitly considers the deviation caused by other factors such as onboard anomalies and the pilot's intent cached in the data. Although the prediction deviation can be reduced based purely on the last on-file flight plan, considering the weather explicitly into the ML pipeline can help further reduce the trajectory deviations.

For future work, I have the following suggestions. (a) It would be interesting to investigate the influence of adding other types of weather features, such as vertically integrated liquid (VIL). Also, it would be valuable to consider probabilistic weather forecasting in the model. All of these can be treated as additional dimensions of the weather feature cubes $X^{weather}$ in this proposed ML pipeline. (b) Other than the weather impact on TP,

another impact factor is the traffic management for neighboring aircraft due to potential conflict. This requires the data structure to change and search for the nearest neighbor in a dynamically changing spatial-temporal database. In addition, human factors are also critical uncertainty sources as some flights are operated by the visual approach of pilots. (c) The current application focus on the en-route flight and another interesting and important application will be for the near-terminal region, e.g., landing and taking-off phases. Many uncertain events happen near these phases due to increased density, vortex and turbulence effect, and altitude holding patterns. In addition, the standard terminal arrival procedures (STARs) and the standard instrument departure routes (SIDs) based procedures play an important role in near-terminal regions. The consideration of multi-aircraft awareness is critical and challenging for both data preparation and algorithm development. For example, the airport surface detection equipment, model X (ASDE-X) data, and the on-aboard Automatic Dependent Surveillance-Broadcast (ADS-B) data can be used for model validation of near-terminal TP with multi-aircrafts awareness. Significant additional research is required to consider the aforementioned factors.

Chapter 3

PROBABILISTIC MULTI-AIRCRAFT TRAJECTORY PREDICTION IN THE TERMINAL AIRSPACE

3.1 Introduction

The escalation of civil aviation operations leads to the concept of the next-generation air traffic management system (NextGen), which aims to efficiently and safely accommodate the growing air traffic flow within the United States airspace [8]. In 2020, FAA reported the statistical analysis results of U.S. air traffic operations before the COVID-19 pandemic [100]. It shows that the aviation operations handled by the core 30 airports increased by 1.8% annually. Moreover, the pilot certificates issued annually are also surging. Nonetheless, the total headcount of air traffic controllers is decreasing from 2018 to 2019. It has been shown that Air Traffic Controller (ATC) workload is one of the main limitations to the capacity of the current Air Traffic Management (ATM) system [101; 102]. This ATC workload increase urges the advancement of air traffic decision-support tools (DSTs) [9] of NextGen, which includes flight plan (FP) change, dynamic weather reroute (DWR), trajectory prediction (TP), and conflict detection and resolution (CDR). Furthermore, in NextGen, surveillance information sharing is greatly enhanced among the controllers and the pilots. In such a way, the aircraft itself can take over a portion of ATM tasks from ground air traffic controllers. And this leads to the prediction of multi-aircraft trajectories, which is beneficial in the relevantly congested, near-terminal air space.

In practice, a deterministic TP model is insufficient when dealing with increasingly congested airspace [19] and is not suitable for safety-related applications due to the inability to consider uncertainty. The uncertainty comes from a variety of sources. The environmental

factor is one of the major contributors to the TP uncertainty [16; 18; 103], which usually develops expeditiously and randomly. Human factors such as the pilot's intent or decision preference when dealing with an aviation event also contribute to TP uncertainty. Other factors such as aircraft performance and the pilot's physical condition can lead to an unreliable deterministic model prediction. This work focuses on modeling multi-aircraft social awareness happening in the real world. I demonstrate our framework using the terminal flight track data near one of the busiest airports (Class B Airspace) in the 30 major hubs.

The development of TP models is of fundamental importance to various advanced engineering application domains, e.g., autonomous systems and warning systems in the automotive and defense industry. Although most of the literature focuses on building TP models for a single agent, the multi-agent TP problem can be more challenging and practical in the real world. The difficulties come from a) the agent-to-agent interactions are hard to capture in a dynamic environment, e.g., two agents can have interactions in the current timestamps but no interactions in the future [104]; b) the action space for multiple agents are significantly larger than the single-agent case, e.g., at each step, the action of one agent has an impact on the actions to its neighboring agents; c) the temporal correlations are coupled with spatial interactions, e.g., the time-series motions of the current agent need to consider the motions of all the neighbors [105].

A few classical data-driven multi-agent TP models capture multi-agent interactions by energy functions, which requires significant feature engineering in a real case [106; 107]. The advancement of deep neural networks has demonstrated promising performance on a few pedestrian data benchmarks [37; 39; 105; 108]. The core idea behind these models is to capture the motion of each agent by the hidden space tensor and merge/share the hidden space across multiple agents. Social pooling is a widely used technique to equally combine the hidden space of each agent by the pooling mechanism. The attention mechanism weighs each agent using a learned score function, which treats each agent unequally. Transformer

adopts the efficient yet straightforward self-attention mechanism to improve the temporal modeling than the previous literature [109]. Recently, the Spatio-Temporal grAph tRansformer network (STAR) has been developed and shows state-of-the-art performance on the commonly used pedestrian dataset [99]. This work interleaves the spatial and temporal correlations by a separate spatial transformer and temporal transformer. The spatial transformer adopts transformer-based graph convolution (*TGConv*) to extract the spatial interactions. The temporal transformer is a classical Transformer with multi-head attention for each pedestrian. The experiment shows that the proposed framework achieves state-of-the-art performance on the ETH/UCY pedestrian benchmark dataset. I propose an uncertainty-aware multi-agent TP model with a multi-head temporal transformer and *TGConv*-based spatial transformer modules to capture the multi-aircraft interactions in the low-altitude near-terminal area. I combine Bayesian deep learning with STAR to achieve uncertainty quantification (UQ) in safety-critical air transportation systems. The Bayesian Spatio-Temporal grAph tRansformer network (B-STAR) for uncertainty-aware multi-aircraft TP in the near-terminal airspace. Like pedestrian TP, I model the near-terminal aircraft as a graph and leverage the spatiotemporal correlation by interleaving a graph-based spatial Transformer module and a temporal Transformer. While the aviation domain typically focuses on operation safety and regulations (e.g., separation assurance [110; 111]), I propose to address this domain-specific knowledge with uncertainty quantification techniques with Bayesian formulation and by encoding air traffic rules into the deep learning model. The uncertainty of our proposed B-STAR model directly comes from the data variations and is solved by the variational approximations with Gaussian priors over the network parameters. It should be noted that the simple dropout approximation to Bayesian deep learning [99] is not used in this study due to some known deficiencies related to the user-defined dropout ratio, the improper Bernoulli prior assumptions, and the lack of theoretical groundings [84; 85; 86; 87].

The test performance evaluation is done on the UCY/ETH benchmark pedestrian dataset to validate the prediction accuracy of the proposed B-STAR framework. Then, I propose a machine learning (ML) problem-solving pipeline to perform near-terminal multi-aircraft trajectory prediction using the data from Airport Surface Detection System — Model X (ASDE-X). Our proposed framework includes the efficient large-scale flight data querying module, the advanced graph transformer-based prediction module, and the web-based interactive visualization module. Graph-based spatiotemporal prediction models have been applied to engineering problems, such as power grid performance prediction [112] and traffic volume forecasting [113; 114]. However, to the best of our knowledge, the authors believe this is the first work utilizing the graph-based deep learning model for multi-aircraft interactive trajectory prediction.

One of our previous studies focused on the single aircraft trajectory prediction method [103], where a TP framework under convective weather conditions was proposed in the en-route phase. In the current work, the focus is on the impact of near-terminal multiple-aircraft interactions. Real-world flight recording data is used to demonstrate and validate the proposed methodology for multi-aircraft interactions in the near-terminal area. The proposed work refines the recent advancement in deep predictive modeling and adapts to the air transportation domain by encoding aviation regulations and physics knowledge into the deep learning model. Our contributions are listed below.

- This chapter proposes a multi-agent trajectory prediction model with uncertainty quantification capabilities called B-STAR. The uncertainties are inferred from the variations within the flight track data instead of pre-defined parameters or randomized inputs in other probabilistic frameworks.

- I encode the aviation regulations on the aeronautical separation into B-STAR. This is achieved by estimating the Haversine distance when selecting silent neighbors of

the current object to build the graph encoding. Additionally, I perform a sensitivity study on separation assurance distance to see the impact to the proposed framework.

- This chapter develops a module-based machine learning framework utilizing advanced computer software and hardware tools for data analysis. Open-source toolsets for data pre-processing and web-based interactive visualization are made available and integrated with the B-STAR framework.

The rest of the paper is organized as follows. First, Section 5.2 reviews the studies performed in the general field of trajectory prediction (TP), with a specific emphasis on air transportation. The development of various advanced data-driven TP models and necessary background knowledge are discussed in detail. Section 5.4 introduces the problem setup and the proposed module-based multi-agent trajectory prediction framework. In Section 5.5, the experimental results will be discussed in two parts. I first show that our B-STAR achieves state-of-the-art with comparison to various recent advancements in multi-agent TP. Then I show the testing results for the multi-aircraft trajectory prediction task with processed ASDE-X data. A discussion on limitations and future directions of the proposed work is described in Section 3.5. Section 5.6 discusses the limitations and potential future research directions.

3.2 Literature Review

In this section, I first review the research on TP across different domains, with an emphasis on air traffic trajectory prediction. Then, I introduce the necessary concept required in this study. This includes Transformer and Self-Attention, Graph Neural Networks, and Bayesian Deep Learning.

3.2.1 Related Works

Overview of Trajectory Prediction

Trajectory prediction is a critical functional component for the research focusing on different objects such as pedestrians, ground vehicles, aircraft, spacecraft, and rockets. Model-based methods are widely adopted in rigorously controlled, data-expensive environments (e.g., spacecraft and rockets), while data-driven models are more popular for less-controlled, data-intensive circumstances (e.g., ground vehicles, pedestrians, and civil aircraft). Model-based methods typically predict the control parameters in the differential kinematic equations, where the control parameters can be used to describe the motion state of the target [115]. Due to the limitation of available data, the data-driven approach in these areas adopts state-space estimations or intent inference [116; 117; 118] methods to compensate for the learning models. On the other hand, TP in data-intensive circumstances acquires data-driven methods or a combination of data-driven and model-based methods. Researchers have proposed numerous advanced data-driven models, such as classical deep neural networks and graph-based deep neural networks with attention. The ground vehicle TP mainly focuses on predicting the behaviors of vehicles, pedestrians, and environments based on the observations, where the decentralized communication between each agent is critically important [119]. Additionally, a hybrid approach combining data-driven methods with model-based methods is commonly adopted [120]. Other researchers propose to analyze the trajectory patterns (e.g., trajectory similarities) on the macro-scale for a high-level prediction of trajectories [121].

Trajectory Prediction in Air Transportation

Trajectory prediction for aircraft has been long regarded as a major topic in air transportation research. As a result, researchers have developed extensive TP frameworks but with

assumptions on different impact factors, in both deterministic and probabilistic senses. This includes TP with voice communication between the pilot and the tower [50], convective weather and other weather-related factors [59; 60; 61; 62; 103; 69], human factors such as pilot/aircraft intent [48; 49; 10], and aircraft conditions [52; 53].

The air transportation society has witnessed a significant increase in data-driven TP solutions in the last decade, associated with the advancement of machine learning techniques. The commonly used models are Kalman filtering [42], state-space model [41], simple neural network [43], Hidden Markov model [54], generalized linear regression [66], recurrent neural network (RNN) [68], and generative adversarial net [69]. Despite the aforementioned individual aircraft TP methods, there are very few works on multi-aircraft TP. A recent work using Social-Long Short-Term Memory (Social-LSTM) network was proposed for multi-aircraft trajectory prediction, where the social pooling layer learns interactions [122]. One limitation of this method is that it treats aircraft within certain airspace equally, which doesn't follow the aeronautical separation standards [123]. For example, in FAA Order 7110.65, different separation distances (from 3NM to 10NM) are allowed for different cases. In such cases, a dynamic determination of neighboring aircraft is preferred.

Multi-Agent Interactive Trajectory Prediction

The research on interactive trajectory prediction is primarily within the human pedestrian context. Existing methods can be divided into classical methods and deep learning-based methods.

Classical methods (e.g., Social Force models, Geometry-based methods) require handcrafted features to capture crowd behaviors. They are less data-intensive with increased interpretability. Social Force models (guided by virtual repulsive and attractive forces) are built upon the assumption that pedestrians are mission-driven for destination navigation and collision avoidance [106]. However, Social Force models perform poorly on TP tasks

[124]. Geometry-based models adopt optimization-based interactive TP with the geometry of each agent [125; 126]. Classical methods require extensive feature engineering and are hard to generalize in different scenes [105].

Deep learning-based methods learn crowd behaviors directly from the data, which achieves automatic feature engineering. Recurrent neural networks have been applied to TP and show satisfactory performance [37; 39; 104; 105; 108]. Behavior CNN captures crowd behaviors using CNN [127]. RNN-based approaches learn the pedestrian dynamical behavior with their latent state but generally perform poorly on complex temporal scenes [109]. Social-pooling layers merge the latent space between several nearby pedestrians, leading to a socially aware prediction. The attention mechanism [104; 105; 108] weighs each pedestrian with individual pedestrian importance through a learned function. In these works, the attention mechanisms are very simple with unsatisfied TP performance. More complex attention mechanisms, such as Transformer with self-attention, show effectiveness in modeling temporal dependencies [109]. Furthermore, the multi-head attention mechanism jointly learns multiple hypotheses from different positional embedding representations [109].

3.2.2 Preliminaries

Transformer and Self-Attention

The recurrent architecture of deep neural networks relies on the sequential encoding of inputs, which leads to computational inefficiency since the processing cannot be parallelized. To tackle this issue, researchers propose the *Transformer* architecture that replaces the RNN recurrence with a multi-head self-attention mechanism, which incorporates a richer context compared to RNN recurrence. The self-attention mechanism captures the mapping between input and output and demonstrates a shorter training time due to parallelization

and higher accuracy for the Machine Translation task [109]. *Transformer* has become the state-of-the-art framework for natural language processing (NLP). Variants of Transformers have shown success in a wide variety of problems, such as OpenAI's Generative Pre-trained Transformers (GPTs) [128; 129], and Bidirectional Encoder Representations from Transformer (BERT) for language representations [130].

In a typical temporal Transformer, the embedding h_t at t is pre-trained or is simply the output from the annotation functions. The self-attention of Transformers learns the query matrix $Q_t = f_Q(h_t)$, the key matrix $K_t = f_K(h_t)$, and the value matrix $V_t = f_V(h_t)$. The attention at time t computes at,

$$Attention(Q, K, V) = \frac{softmax(QK^T)}{\sqrt{d_k}}V \tag{3.1}$$

where $\frac{1}{\sqrt{d_k}}$ accounts for the numerical stability of attentions. In Equation (3.1), the compatibility function $f(Q, K) = QK^T$ defines how the keys and queries are matched together. The choice of compatibility functions defines the relationship between K and Q. The commonly used dot-product function, $f(Q, K) = QK^T$ is called machine-learned attention [131]. A similarity-based compatibility function replaces $f(Q, K) = QK^T$ by $f(Q, K) = sim(Q, K)$ called similarity attention [132], where the most relevant keys are the most similar to the query. $f(Q, K) = f(Q)$ is called location-based attention, where the relevance is solely a function of the key's location, independently of its content [131]. In Equation (3.1), $softmax$ is the distribution function. The choice of distribution function depends on what properties the model needs, for instance, probability scores or Boolean scores to enhance sparsity.

Decouple attention recurrence into multiple matrices allows self-attention to handle complex temporal dependencies. Multi-head attention is simply embedding the output from multiple self-attentions. With n heads, I have,

$$MultiAttention = f_h([Attention(Q, K, V)]_{i=1}^{n}) \tag{3.2}$$

where f_h is simply a fully connected layer merging the output from n heads. Additional position encoding is also required. Finally, the Transformer outputs the embeddings by a linear layer with two skip connections.

Graph Neural networks

Transformers are limited to non-structured data, e.g., linguistic languages. Graph neural networks (GNNs) are introduced to model complex social behaviors from the structured graph data with explicit message passing [133; 134]. I denote a graph with vertices and edges, represented as $G = (V, E)$, where the number of nodes $N_{nodes} = |V|$ and the number of edges $N_{edges} = |E|$. The adjacency matrix $A \in \mathbb{R}^{N_{nodes} \times N_{nodes}}$. The graph can be directed or undirected depending on whether the edges are directed from one node to another. The graph is considered a dynamic graph when the topology of the graph varies with time. Especially, the graphs in our work are undirected dynamical graphs. Graph convolutional networks (GCN) [135] convolve the input X using the derived compact form,

$$H = \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} XW \tag{3.3}$$

where $\tilde{A} = A + I_N$, $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$. $X \in \mathbb{R}^{N_{nodes} \times N_{input}}$ is the input matrix, $W \in \mathbb{R}^{N_{input} \times N_{output}}$ is the kernel parameters, and $H \in \mathbb{R}^{N_{nodes} \times N_{output}}$ is the updated graph embedding.

The attention mechanism has also been adopted on graphs where different neighbors are assigned different weights to alleviate noises and achieve better results [136; 137]. Graph Attention network (GAT) incorporates weighted message passing between nodes and multi-head attention to achieve state-of-the-art results on multiple domains [137]. The recently

introduced STAR model performs spatiotemporal TP with merely a self-attention mechanism and also demonstrates state-of-the-art performance on the UCY/ETH dataset [99]. They also introduce the Transformer-based graph convolution module (*TGConv*), which advances GAT with a self-attention Transformer. The spatial and temporal correlation is learned by simply interleaving the spatial Transformer and the temporal Transformer [138]. Furthermore, STAR also introduces a read-writable graph memory module to smooth the predictions and enforce temporal consistency continuously.

Bayesian Deep Learning

Bayesian Deep Learning (BDL) is a widely explored area of research on quantifying the uncertainty and improving the robustness of deep learning models. The idea behind BDL is to put distributions over neural network parameters. For a regression problem, given an input sequence $X = \{x_1, ..., x_n\}$ and the corresponding output sequence $Y = \{y_1, ..., y_n\}$, I try to estimate the parameters ω for the approximation function $y = f^{\omega}(x)$. That is, to inference $p(\omega|X, Y)$. During the test phase, if I have an observation data sequence x^*, the prediction sequence y^* is simply the weighted average of the model where the weights are determined by the posterior distribution of ω, mathematically, $\mathbb{E}_{p(\omega|X,Y)}[p(y^*|x^*, \omega)]$. Unfortunately, this is intractable in any practical case [77]. Consequently, Variational Inference (VI) is introduced to approximate the posterior of the Bayesian neural network. In VI, I aim to find the best variational distribution approximation $q_\theta(\omega)$ of $p(\omega|X, Y)$, where $q_\theta(\omega) \in Q_\Theta(\Omega)$, and $Q_\Theta(\Omega)$ is a family of i.i.d. Normal distributions,

$$Q_\Theta(\Omega) \sim \mathcal{N}(\mu, diag(\sigma_d^2)) \tag{3.4}$$

and with mean-field assumption on each component of Q_Θ,

$$Q_\Theta = \{q|q_\theta(\omega) = \prod_{i=1}^{d} q_{\theta i}(\omega_i)\} \tag{3.5}$$

Kullback–Leibler (KL) divergence is used to measure the discrepancy between $q_\theta(\omega)$ and $p(\omega|X,Y)$. That is, I want to solve the optimization,

$$min_{q_\theta \in Q_\Theta} \mathcal{KL}[q_\theta(\omega)||p(\omega|X,Y)] \tag{3.6}$$

Researchers showed that this objective is equal to minimizing the objective function in Equation (3.7), which is also known as *variational free energy*.

$$\begin{aligned} min_{q_\theta \in Q_\Theta} \quad & \mathcal{KL}(q_\theta(\omega)||p(\omega)) - \mathbb{E}_{q_\theta(\omega)}[log(p(Y|X,\omega))] \\ \approx & \frac{1}{N}\sum_{i=1}^{N}[log(q_\theta(\omega_i)) - log(p(\omega_i)) - log(p(Y|X,\omega_i))] \end{aligned} \tag{3.7}$$

From Equation (3.7), I show that the VI approximation to the Bayesian neural network can be inferred through the sampling of the three terms derived from the variational free energy. Various weight perturbation methods [88; 81] enable direct gradient-based optimization during VI. The recent advancement of probabilistic programming languages also established the pathway of building scalable real-world applications [89; 90; 91; 92].

3.3 Methodologies

In this section, I first define the setup for the problem I am trying to solve in Section 3.3.1. Then, I introduce the architecture of the proposed deep learning framework B-STAR in Section 3.3.2. B-STAR architecture decomposes spatiotemporal attention learning into *temporal modeling*, *spatial modeling*, and *uncertainty modeling*, which will also be discussed in detail.

3.3.1 Problem Setup

The task I am focusing on is the time-series forecasting of trajectories for multiple aircraft. I wish to predict future trajectories based on observed trajectories. The total times-

tamps of this sequence are size T for each individual aircraft. The first T_{obs} timestamps are observations, and the other $T - T_{obs}$ timestamps are the prediction horizon of our model. I set the total number of agents that show up in a scene as N. I use $p_t^i = (x_t^i, y_t^i)$ to denote the position of agents in a top-down view environment. An undirected edge for each aircraft pairs with a distance less than a threshold ζ. This leads to the dynamical undirected graph $G^t = (V^t, E^t)$ mentioned in previous sections. Similar setups have been used for the pedestrian TP case [99].

$$
\begin{aligned}
d_{i,j}^t &= 2 \times R \times \arcsin(\sqrt{\sin^2(\frac{y_t^i - y_t^j}{2}) + \cos(y_t^i)\cos(y_t^i)\sin^2(\frac{x_t^i - x_t^j}{2})}) \\
R &= 6371km
\end{aligned}
\tag{3.8}
$$

For the aircraft TP case, I use $p_t^i = (x_t^i, y_t^i)$ to denote the position of aircraft p^i's location shown in the radar measurement. The graph is built by calculating the distance between two aircraft pairs (p_t^i, p_t^j) at the same timestamp with Equation (3.8). It's also called Haversine distance [139] to calculate the great-circle distance between two WGS84 coordinates. If $d_{i,j}^t$ is greater than a neighboring threshold ζ, I establish a connection $E_{i,j}^t$ between aircraft i and j at timestamp t, and vice versa. This leads to the dynamical changing graph $G^t = (V^t, E^t)$ at each timestamp t, which includes all the aircraft pairs classified as neighbors.

3.3.2 Proposed Bayesian Spatio-Temporal Graph Transformer

B-STAR is a stacked encoder-decoder structure, where the encoder composes of several Transformer building blocks to leverage spatial and temporal modeling separately. In addition, the decoder accounts for uncertainty modeling and outputs predictions. I keep a deterministic encoder from a Bayesian decoder to reduce computational complexity while retaining the performance of the entire framework [140]. The encoder-decoder structure of B-STAR has been shown in Figure 3.1.

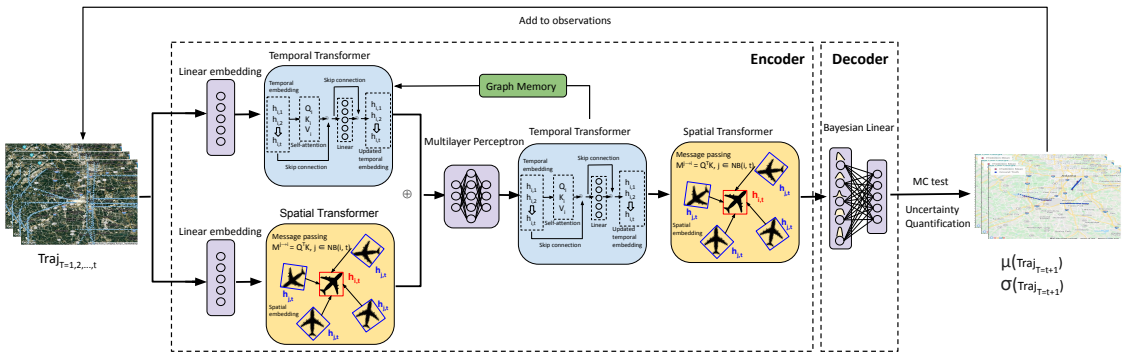


Figure 3.1: B-STAR: Multi-Aircraft Trajectory Prediction Network Architecture. In B-STAR, Trajectory Prediction is Achieved by Interleaving the Spatial Transformer and Temporal Transformer into an Encoder-Decoder Structure. The Inputs to Transformers are Embedded with Linear Layers and Concatenated to Feed into another Transformer Module. Here, I propose to Extend the Decoder by using a Bayesian Neural Network Approximated with Variational Inference Mentioned in Section 3.2.2. The Last Observation Timestamp is T_{obs}. The Prediction at T_{obs+1} is Added Back to the Observation Sequence to Predict the Aircraft Locations at Timestamp T_{obs+2}.

Transformer Encoder: The temporal Transformer block considers every single agent independently and learns each agent's dynamics from its own data. The temporal model is simply a standard temporal Transformer network, which has been shown to achieve better time-series learning performance compared to RNNs [99]. For spatial modeling, I adopt *TGConv* to perform Transformer-based graph convolution. The proposed architecture consists of two parallel-aligned Transformer blocks and two sequentially aligned Transformer blocks. I propose to use a multi-layer perception after the parallel-aligned Transformers, which gives a new spatiotemporal embedding of the data. The sequentially aligned module acts as the post-processing of the concatenated feature embeddings.

Bayesian Decoder: The decoder takes the updated embedding as inputs and outputs the predicted trajectory for each agent in the graph. Similarly, I propose a Bayesian version of

the decoder, which consists of a stochastic Bayesian fully-connected layer for predictions and uncertainty quantification. The parameters of Bayesian layers are initialized with standard normal distribution and sampled during inference of the posterior as Equation (3.7).

Specifically, the prediction of each timestamp is added back to the observation for future prediction. To improve the long-horizon temporal consistency of the predictions, an external graph memory is used to allow the temporal Transformer to condition the current temporal embeddings on the previous temporal embeddings. In such a way, I can get a consistent trajectory prediction to avoid unreasonable trajectory predictions. This graph memory module is read-writable.

Temporal Modeling

The input of the temporal Transformer block is the input embeddings from the raw input data, mathematically, $\{h_1^i, h_2^i, ..., h_t^i\}$, for agent i. Input embedding is commonly achieved using a simple linear layer. The output is the updated embedding sequence $\{\hat{h}_1^i, \hat{h}_2^i, ..., \hat{h}_t^i\}$. For agent i, the temporal Transformer first learns the query matrix Q^i, K^i, and V^i.

$$Q^i = f_Q(\{h_t^i\}_{t=1}^T), \quad K^i = f_K(\{h_t^i\}_{t=1}^T), \quad V^i = f_V(\{h_t^i\}_{t=1}^T) \tag{3.9}$$

where the functions f_Q, f_K, and f_V are shared across all agents. The computing of attentions for temporal Transformer follows the standard computation flow of the multi-head attention mechanism, as in Equation (3.1) and Equation (3.2).

Spatial Modeling

The spatial Transformer block learns the interaction between agents in a scene with *TGConv* proposed by [99]. As mentioned, *TGConv* is a Transformer-based graph convolution block that performs message passing between graph nodes. It is an attention-based graph convolution block, with a different spatial embedding h^i updating computation procedure.

In *TGConv*, the input is a spatial embedding set for multiple agents, mathematically, $\{h^1, h^2, ..., h^i\}$. The query vector for the agent i is $q^i = f_q(h^i)$, key vector is $k^i = f_k(h^i)$, and value vector is $v^i = f_v(h^i)$. The message passing from agent j to agent i is defined as,

$$m^{j \to i} = (q^i)^T k^j \tag{3.10}$$

The attention-based graph convolution in Equation (3.1) is reorganized into *TGConv* as,

$$Att^i(Q^i, K^i, V^i) = \frac{softmax([m^{j \to i}]_{j \in NB(i) \bigcup \{i\}})}{\sqrt{d_k}} [v_j]^T_{j \in NB(i) \bigcup \{i\}} + h^i \tag{3.11}$$

where $NB(i)$ is the neighbors set of agent i. The updated embedding $\hat{h^i}$ is calculated by Equation (3.12) with skip connections before the output function f_o.

$$\hat{h^i} = f_o(Att^i) + Att^i \tag{3.12}$$

The spatial Transformer outputs the updated spatial embedding $\{h^1_t, h^2_t, ..., h^i_t\}$, $t \in \{1, 2, ..., T_{obs+1}\}$ of the current agent i, which concatenates with the updated temporal embedding $\{\hat{h}^i_1, \hat{h}^i_2, ..., \hat{h}^i_t\}$, $t \in \{1, 2, ..., T_{obs+1}\}$ of agent i. The multi-layer perception takes the concatenation as the input and feeds the output into the second temporal Transformer. The output of the temporal Transformer finally feeds into the second spatial Transformer before getting into the Bayesian decoder. Also, the second temporal Transformer returns the output to the first temporal Transformer to account for a consistent recursive temporal prediction.

Uncertainty Modeling

The uncertainty comes from the Bayesian linear layers in the decoder. For example, it has been shown that a stochastic Bayesian classifier is sufficient for uncertainty quantification and improves ML systems' robustness [140]. Here, I believe that a Bayesian encoder is

sufficient instead of adopting Bayesian input embeddings or probabilistic attention in the decoder part. I leave probabilistic embeddings and probabilistic attention for future study. Finally, the output layer of the decoder remains deterministic to avoid training instability. In the test phase, the prediction uncertainty can be estimated through MC tests as in Equation (3.13). x^* and y^* are test data pairs, ω is the well-trained model parameters, and N is the number of tests performed.

$$\begin{aligned} p(y^*|x^*, X, Y) &= \mathbb{E}_{p(\omega|X,Y)}[p(y^*|x^*, \omega)] \\ &\approx \frac{1}{N}\sum_{n=1}^{K} p(y^*|x^*, \hat{\omega_k}) \end{aligned} \tag{3.13}$$

3.4 Experiments

In this section, I first discuss the evaluation metrics used in the experiments in Section 5.5.1. Then I report our results on the ETH (ETH and HOTEL) and UCY (ZARA1, ZARA2, and UNIV) pedestrian TP dataset in Section 3.4.2. ETH and UCY are human crowds datasets with a medium interaction density. They serve as the most used benchmark dataset for demonstrating state-of-the-art performances in multi-agent TP research works. I compare B-STAR to 8 state-of-the-art TP models developed on the pedestrian dataset with the same setup. I use the leave-one-out cross-validation strategy to get the test results. Lastly, I report the proposed machine learning problem-solving pipeline for near-terminal multi-aircraft TP, demonstrated with real-world radar recording data in Section 3.4.3. Additionally, a sensitivity study is performed on the aircraft case.

Briefly, I show that: a) The proposed uncertainty-aware B-STAR model can achieve state-of-the-art mean prediction performance on the ETH and UCY pedestrian dataset; b) B-STAR gives reliable uncertainty estimates without sacrificing the prediction power; c) The sensitivity study shows a larger prediction-observation ratio leads to a declined test performance.

3.4.1 Evaluation Metrics

The standard way to evaluate the model performance is to use the Average Displacement Error (ADE) and Final Displacement Error (FDE) as our minimization objectives. A lower ADE and FDE represent better model performance on the given dataset. I minimize the ADE and FDE on the training set and backpropagate them to update the parameters during training. After several epochs of training, I start to evaluate the ADE and FDE on the test set and save the best model.

- ADE: The averaged mean square error between the prediction and ground truth sequences.
- FDE: The L_2 distance between the predicted final position and the ground truth final position.

3.4.2 Case I: Pedestrian Crowd TP

This case study is conducted as a baseline study to validate our B-STAR performance, especially compared with the ML models proposed in the literature to understand the contributions of the proposed work. I use the same experiment set up on the 5 scenarios from ETH/UCY dataset for a fair comparison. To keep this case study simple, I record the results from the literature.

State-of-the-art Models

I compare our model with several state-of-the-art models as baselines. This includes,

- Social LSTM [37]: Each agent in the scenario is modeled with an LSTM, where the hidden states are shared between neighbors.

- State Refined LSTM (SR-LSTM) [104]: Similar to S-LSTM, each agent is modeled with an LSTM but with a state refinement module on the hidden state tensors. The refinement module includes a motion gate and pedestrian attention functions to extract useful features of each pedestrian.

- Social Attention [141]: The crowd of pedestrians is modeled with a spatiotemporal graph and adopts two LSTMs to handle the spatial and temporal dependencies.

- TrafficPredict [142]: TP motion prediction model using LSTMs. This method introduces a category layer to refine the predictions of the agents belonging to the same type, i.e., vehicles, and pedestrians.

- STAR [99]: The interleaved spatial and temporal Transformers learn spatiotemporal features from graph-structured data with *TGConv*.

- Social GAN [39]: The social Pooling Module (PM) is adopted into the general adversarial network (GAN). The generator module G learns from the data and outputs the predicted trajectory. G is an encoder-decoder framework where the hidden states of the encoder are linked with the hidden states of the decoder via PM. The discriminator D tries to classify the ground truth and the predicted trajectory from G. Similar to the classical GAN, the randomness comes from the random input to the Generator.

- Trajectron [143]: A variational encoder-decoder structure for multi-agent TP. The encoder encodes the history and future states of a given node and predicts the distribution of future trajectories using deep generative modeling. This chapter also adopts the β-weight ELBO [144]. The randomness comes from the random sampling of hidden variables.

- STAR-Dropout [99]: The STAR model with dropout applied on fully connected layers. The randomness comes from the pre-defined dropout ratio.

Comparisons

Table 3.1 shows the comparison of the proposed B-STAR with the literature on pedestrian datasets. I compare our method with 5 deterministic methods and 3 stochastic TP methods. I show that I achieve state-of-art test performance in these scenarios, except for ETH and HOTEL. Then, I apply the proposed ML framework to the air transportation problem.

Table 3.1: B-star Compares with State-of-the-art Multi-agent TP Models on the ETH/UCY Dataset. The Stochastic Methods Are Averaged over 20 Samples and Compared with the Mean Predictions. The Performance Reported is Based on the Evaluation Metrics ADE/FDE. B-star Achieves Comparable Performance to the Star with Dropout. However, Our Proposed B-star Doesn't Require a Pre-defined Dropout Ratio for Uncertainty Quantification.

Deterministic	ETH	HOTEL	ZARA1	ZARA2	UNIV
Social LSTM	0.77/1.60	0.38/0.80	0.51/1.19	0.39/0.89	0.58/1.28
SR-LSTM	0.63/1.25	0.37/0.74	0.41/0.90	0.32/0.70	0.51/1.10
Social Attention	1.39/2.39	2.51/2.91	1.25/2.54	1.01/2.17	0.88/1.75
TrafficPredict	5.46/9.73	2.55/3.57	4.32/8.00	3.76/7.20	3.31/6.37
STAR	0.56/1.11	0.26/0.50	0.41/0.90	0.31/0.71	0.52/1.15
Stochastic	ETH	HOTEL	ZARA1	ZARA2	UNIV
Social GAN	0.81/1.52	0.72/1.61	0.34/0.69	0.42/0.84	0.60/1.26
Trajectron	0.65/1.12	0.35/0.66	0.34/0.69	0.29/0.60	0.52/1.10
STAR-Dropout	**0.36/0.65**	**0.17/0.36**	0.26/0.55	**0.22**/0.46	**0.31/0.62**
B-STAR	0.44/0.72	0.24/0.43	**0.26/0.54**	0.25/**0.41**	0.36/0.68

3.4.3 Case II: Near-Terminal Multi-Aircraft TP

In this section, I report our implementation details for the near-terminal multi-aircraft TP as a module-based system. As mentioned in Section 5.1, I propose a module-based ML problem-solving pipeline with a refined state-of-the-art multi-agent prediction model. I discuss our framework from the data processing module, to the modeling training module, until the web-based interactive visualization module. The data used in this case is the ASDE-X near-terminal flight track surveillance recording of Hartsfield-Jackson Atlanta International Airport (KATL). I follow the classical leave-one-out cross-validation method during the training and evaluation process. Furthermore, I perform two sensitivity studies to fully understand the prediction capability of the proposed B-STAR architecture.

ASDE-X Near-Terminal Flight Data

I obtain the raw data streams from the Sherlock Data Warehouse (SDW) [20; 3]. SDW is a platform for reliable aviation data collection, archiving, processing, query, and delivery to support ATM research. The Sherlock data primarily comes from the FAA and the National Oceanic Atmospheric Administration (NOAA) [93]. In this work, I am focusing on the specific problem of near-terminal multi-agent TP. Thus, only the near-terminal surveillance recordings are required, which is the data from the Airport Surface Detection System — Model X (ASDE-X) system. ASDE-X is a surveillance system using radar, multilateration, and satellite technology that allows air traffic controllers to track the surface movement of aircraft and vehicles. It's reported that KATL has the highest average daily capacity and average hourly capacity among the core 30 airports of the United States [100]. Thus, I obtained one week's (Aug 1st, 2019 to Aug 7th, 2019) flight trajectory recordings from the ASDE-X of KATL, with the busiest operation period each day (2 pm to 10 pm), for the demonstration of our proposed method. The dataset has a time interval $\Delta t = 1s$ and

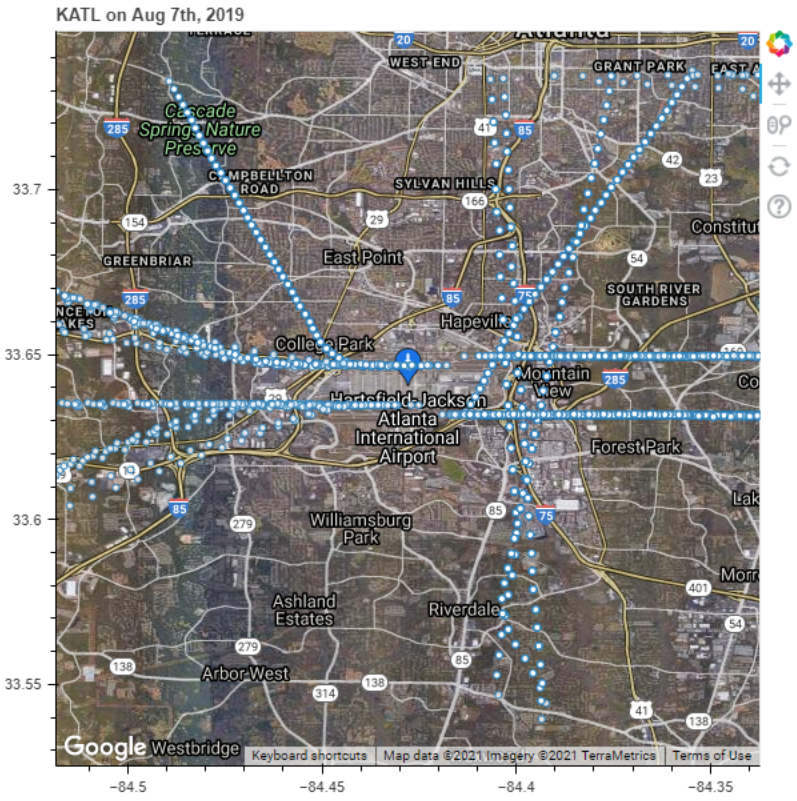


Figure 3.2: A Batch of the Filtered ASDE-X data on Aug 7th, 2019, Visualized with Google Cloud Map API, to Understand the Pattern of Aviation Operations Around the KATL. The Unix Timestamps and Flight IDs are Anonymized. The Figure Clearly Shows That There Are Four Runways on Duty at KATL. The Aircraft Departed to the West and Landed from the East. Several Flight Tracks are Crossing the Airspace of KATL.

is down-sampled into $\Delta t = 5s$ timestamp. I filter out the data from a rectangular area ($r = 0.2°$ latitude/longitude) around KATL, with the accelerated geospatial query tool in GeoSpark [145]. The first 6 days' data is used for training and validation, and the last day's data is kept for testing and performance visualization. I visualize a batch of the processed test dataset in Figure 3.2.

The processing of the raw ASDE-X data from Sherlock can be summarized as follows, a) perform a time window filter to find the flight tracks within the desired time range on

each day. In this work, the time range is from 2 pm to 10 pm locally; b) perform rectangular filter of the flight tracks within the area of interest, with the help of geometry large-scale data processing package. In this work, I define the range as 0.2° latitude by 0.2° longitude centered at the airport coordinates. Also, I filter along the altitude dimension to make sure the aircraft is above the ground level; c) anonymize the sensitive information in the ASDE-X dataset, e.g., the real flight callsign, flight ID, and Unix timestamps; d) downsample the time-series to a user-defined forecasting time interval. In our case, I have $\Delta t = 5\text{s}$. This sampled data sparsity will determine the capability of a well-trained model. I will discuss this in detail in Section 3.5.

Further pre-processing of the data follows the standard strategy as in [104; 99]. The origin of the input sequence shift to the last timestamp of observations. Also, the entire sequence has 20 timestamps for each aircraft. In this case, I have 12 timestamps as observations and 8 timestamps as predictions.

Module-Based Machine Learning Pipeline

I propose the machine learning problem-solving pipeline for the uncertainty-aware near-terminal multi-aircraft TP task. The schematic representations and technical stack are shown in Figure 3.3. The lower level of the proposed pipeline shows how is the raw radar recording data collected, processed, and stored in SDW, where data integration is adopted to handle data heterogeneity [146]. The user of SDW utilizes a user interface to acquire the data desired for the proposed research. I build our novel machine learning application, B-STAR, with an object-oriented programming language and large-scale data processing tools. With the help of multiple CPU & GPU cores, I perform parallelized training for rapid inference in our demonstration case. On the system output layer, I adopt web-based interactive visualization tools for the geometrical representation of our model predictions.

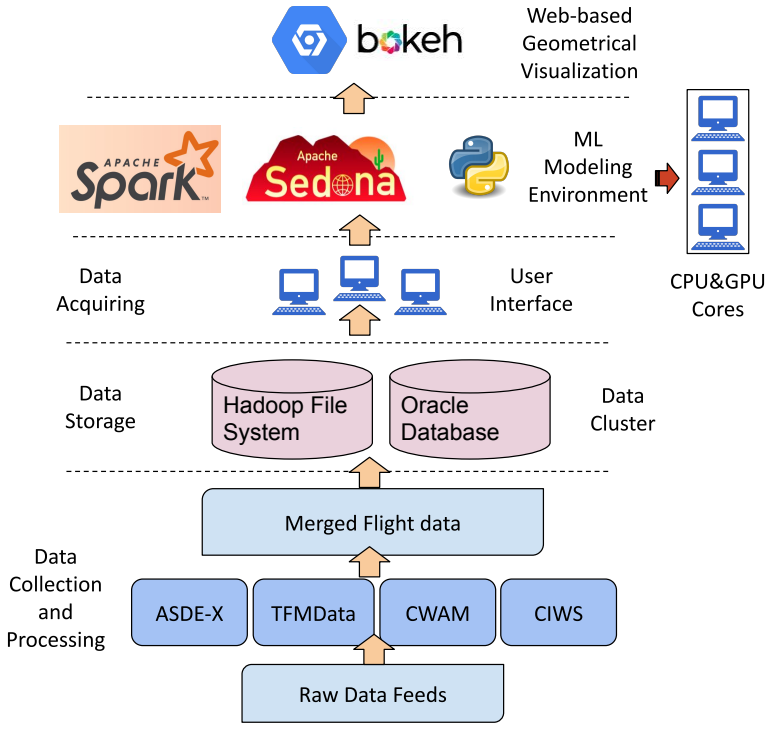


Figure 3.3: Schematic Representation of the Data Workflow. The Figure Shows the Flowchart of the Proposed Problem-solving Machine Learning Framework. Sherlock Data Warehouse Processes the Data from Multiple Surveillance Data Sources and Stores It in the Data Cluster. The User Who Has Access to Sherlock Can Acquire the Data from a Web-based User Interface. Then I Clean the Data into the Required Format Using Apache Spark, and Apache Sedona (Geospark) with Multi-cores. I Build the Training Environment Using Python with Cuda Acceleration. Finally, I Show the Forecasting of Aircraft Coordinates with the Web-based Geometrical Visualization Tool Bokeh and the Cloud-based Google Map API.

Visualization of Test Results

I present the prediction on the test dataset in Figure 3.4 and Figure 3.5. Also, I show the uncertainty values corresponding with both the observation timestamps and the prediction

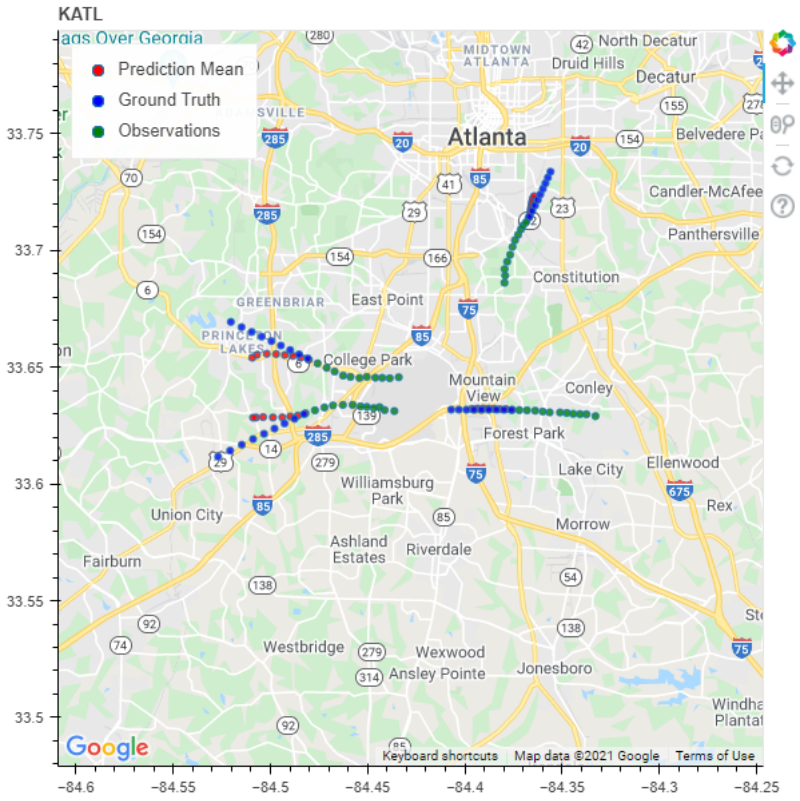


Figure 3.4: Visualization of a Sample Testing Case with Google Map API on Bokeh. Red: The Mean Predictions; Blue: The Ground Truth Track Points; Green: the Input Historical Track Points are Known as Observations.

timestamps in Figure 3.6. In Figure 3.4, I visualize one sample from the model output that belongs to the test dataset. To achieve this, I first determine the neighbor indices by plotting the adjacency matrix $\mathcal{A}$ of *TGConv*. Based on $\mathcal{A}$, I can find the correct neighbor indices and match the indices with the observations, predictions, and ground truth tracks. Additional coordinate shifting and rotating are performed.

I show the B-STAR model can predict temporal consistent trajectories in Figure 3.5a, where three aircraft are heading toward the airport in parallel. The uncertainty of these three aircraft in Figure 3.6a and Figure 3.6e is also small, given that they follow the same parallel pattern without possible intersection. In Figure 3.5b, the model predicts the aircraft

moving direction successfully but with slightly delayed locations. However, I find that the uncertainty value for the prediction also increases with a longer prediction horizon. In Figure 3.5c and Figure 3.5d, I present two cases where the moving direction is not correctly predicted. The typical case is departing aircraft fails at making a good prediction. It's a common pattern that the prediction uncertainty on the latitude dimension has a sudden drop in Figure 3.6. This is due to the FDE constraint on trajectory sequences. Also, the runway of KATL has an east-west layout, which has minimal changes in the latitude dimension. This explains no similar pattern in the longitude dimension.

Generally, B-STAR shows a significant predicting capability on the landing aircraft sequence but doesn't perform well on departing aircraft. As shown in Figure 3.2, the departing aircraft has multiple flight routes. On the contrary, the landing sequence is much simpler, while the aircraft are lined-up. I leave improving the prediction power for more complicated departing aircraft cases as a major future study, which will be discussed in section 3.5.

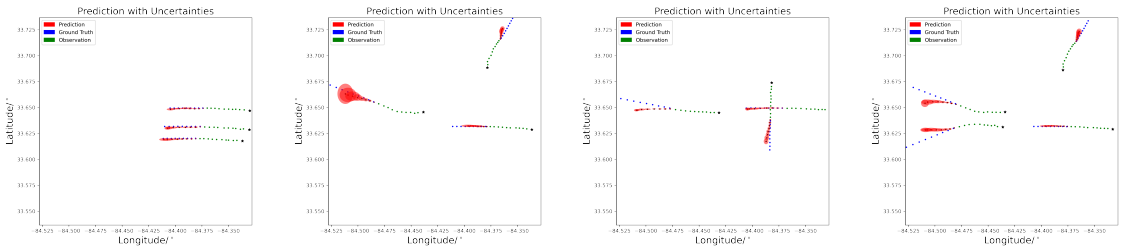
Sensitivity Study

To better understand the prediction capability and provide sufficient guidance for applying the machine learning model in the real case, I conduct the sensitivity study on the two parameters of B-STAR as follows:

- Prediction-Observation Ratio (PO-ratio) Θ: The ratio between the prediction horizon and the observation horizon. A larger PO-ratio stands for longer prediction power with less observed data.

- Graph Neighboring Threshold ζ: The separation distance threshold to determine if the connection between two aircraft was established.

The sensitivity study is performed by retraining the model, with different parameters in



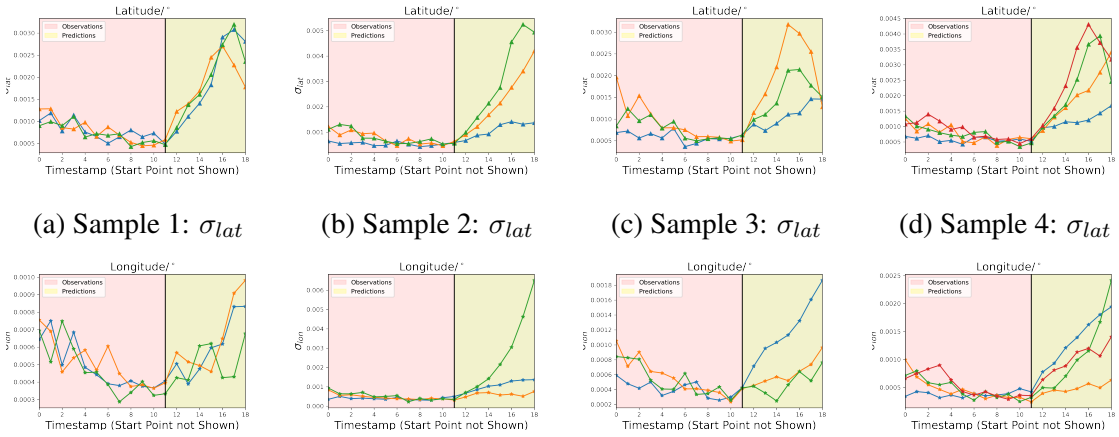
(a) Test Sample 1 (b) Test Sample 2 (c) Test Sample 3 (d) Test Sample 4

Figure 3.5: Visualization of B-STAR Trajectory Predictions. The Red Color Represents the Output from Our Model. Black Star Is the Departing Location for Each Aircraft. At Each Prediction Timestamp, I Visualize the 95% Confidence Bound along with the Mean Prediction, Where the Standard Deviations for Latitude and Longitude Are Measured with the Monte Carlo Test. It's Obvious That Our Well-trained B-STAR Makes Reasonable Trajectory Predictions, with the Contributions from the 95% Confidence Interval.

the preprocessing stage. It is obvious that in Figure 3.7a, the ADE and FDE increase with the enlargement of PO-Ratio. It's expected to have a larger error with a longer prediction horizon. I also notice that the ADE has a minor increase at $\Theta = 0.67, 1.0, 1.5$. Figure 3.7b shows the error when altering the graph neighboring threshold (10, 20, and 30 kilometers, corresponding to approximately 5, 10, and 15 nautical miles). I notice that there is not a significant change in the error metrics. The reason is ASDE-X data only covers a small range of flight tracks, $\zeta = 10km$ already covers all the aircraft in the airspace. In such a case, increasing ζ makes no difference to the neighboring graph.

3.5 Discussions

This work shows that the uncertainty-aware TP model can leverage the spatial and temporal coherence between multiple agents in the scenario. The model interactively forecasts the future location of multiple agents with a user-defined prediction time interval based on

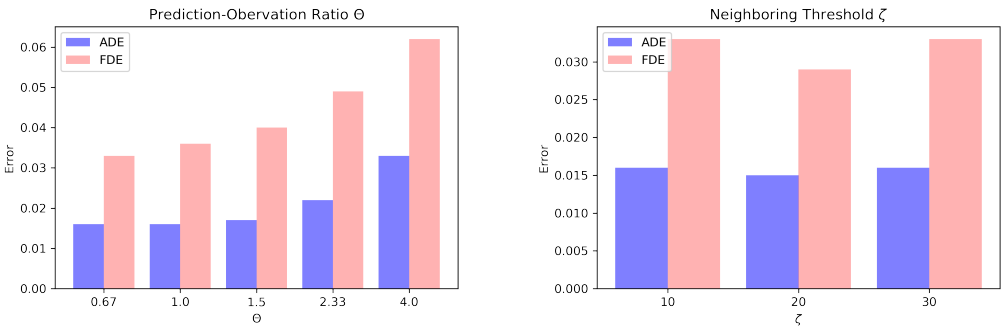


(a) Sample 1: σ_{lat} (b) Sample 2: σ_{lat} (c) Sample 3: σ_{lat} (d) Sample 4: σ_{lat}

(e) Sample 1: σ_{lon} (f) Sample 2: σ_{lon} (g) Sample 3: σ_{lon} (h) Sample 4: σ_{lon}

Figure 3.6: Uncertainty for Trajectory Prediction. The Standard Deviation for the Start Location Is Not Shown. The Left of the Vertical Line Shows the Standard Deviation of the Observation. During Inference, the Model Also Outputs Predictions for the Observation Steps. The Uncertainty for Observation Is Relevantly Stable and Minimal. The Right of the Vertical Line Demonstrates the Standard Deviation for the Prediction Timestamps on Both Latitude and Longitude Dimensions. Due to the Nature of the Time-series Forecasting Model, the Uncertainties Propagate Through Prediction Timestamps.

the current observations. The proposed model can handle an arbitrary number of agents shown in the current map. The uncertainty of the prediction increases with an elongated prediction horizon. I first perform a case study on the standard pedestrian dataset to show the effectiveness of B-STAR. Then I apply the proposed model to the safety-critical air transportation field. Furthermore, I propose an ML problem-solving pipeline to tackle the multi-aircraft TP problem. In this section, I will mainly discuss the limitations in Section 5.6.1 and insights in Section 5.6.2 from the air transportation aspects.



(a) Prediction-Observation Ratio Θ

(b) Graph Neighboring Threshold ζ

Figure 3.7: Sensitivity Study Results. The Error is Measured with ADE and FDE.

3.5.1 Limitations

There are several limitations of the existing methods. Firstly, the construction of the graph data requires a minimum distance threshold to fill the adjacency matrix. The threshold value is an assumption on the potential impact between nearby aircraft. Based on the aeronautical separation standards, I use $\zeta = 10$km as the threshold, where the tower believes two aircraft with a distance closer than $\zeta = 10$km will have interactions. Further validation on the best ζ is valuable. Also, this work doesn't consider the operation on the altitude dimension. This is based on the assumption that the near-terminal aircraft is generally at a low altitude where a vertical separation is not critical, as I am simulating the cases of surveillance radar. Lastly, the time interval of the down-sampled data is actually the prediction time interval. In real applications, to achieve real-time forecasting, the execution time of the model should be smaller than the prediction interval. I am using $\Delta t = 5$s at the moment. In the deployment phase, the model inference time for forecasting the next location of the current scenario is 2.74s. If using classmate enforcing [147], the prediction-response time for the controller/pilot will be 2.26s. However, in the current experiment setup, I am not feeding the ground truth into the model prediction in real-time, and the total

inference time for the entire 8 prediction timestamps is 21.66s. The corresponding response time for the pilot/controller will be 18.34s, which is acceptable in real-world practices.

3.5.2 Insights

This work is beneficial for the future development of safety-critical autonomous system applications, e.g., an accurate uncertainty-aware TP contributes to an early collision warning system. Probabilistic risk assessment is based on uncertainty quantification from the training data. Based on the limitations, I have the following initiatives,

- The demonstration of the proposed model can be improved in many aspects. I am expecting a further accuracy improvement if I include the orientations, the airspeed, and the altitude dimension in the prediction framework. In such a way, our approach is suitable for predicting trajectories in the enlarged airspace rather than the low-altitude, near-terminal region.

- The sensitive study on neighboring threshold can be further improved using the data from larger airspace. The current ASDE-X data has limited coverage around the airport control tower compared to the data from an air traffic control center (ARTCC).

- Future work combined with transfer learning and domain adaptation is desired to increase the generalizability of the proposed framework. For instance, I train our model with the flight data from one airport and manage to get an accurate prediction on the flight data from another airport using *domain adaption* methods, where different controllers' preferences existed. Also, I can try to generate the prediction power from commercial aircraft to other airborne agents, such as helicopters and drones, with the help of *domain generation* methods.

- Due to the characteristic of Bayesian deep learning, the execution time of the proposed model for online, multi-aircraft trajectory forecasting is impacted by the sam-

pling procedure during the test phase. From the algorithm-development aspect, the model compression technique can be beneficial for reducing computational cost, where *knowledge distillation* is a feasible direction to look into. From the engineering perspective, both software and hardware upgrades are available. TensorRT [148] with C++ will significantly reduce the inference time. Hardware such as NVIDIA Orin will boost performance by a wide margin.

3.6 Conclusions

In this chapter, a graph-structured Transformer-based deep neural network architecture is proposed for the uncertainty-aware multi-agent trajectory prediction task. It is shown that uncertainty-aware prediction can be achieved with a particular Bayesian formulation of the trajectory prediction deep learning network. The proposed model is demonstrated and validated using the commonly used standard pedestrian TP dataset (ETH and UCY) and a near-terminal aircraft TP dataset from Sherlock. I analyze the predictions individually and statistically, with a sensitivity study to further understand the prediction power. I visualize several cases on the geographical map around the ASDE-X data range of KATL. Following this, the sensitivity studies focused on the PO-ratio Θ and the neighboring threshold ζ. These studies show the optimal future prediction horizon w.r.t. the observation horizon. One unique focus of the proposed study is on the impact of different aviation regulation encodings on air traffic predictions. The current limitations are discussed and the potential research directions are suggested.

Chapter 4

AIRCRAFT LANDING SCHEDULING USING MACHINE LEARNING-ENHANCED OPTIMIZATION

4.1 Introduction

The civil aviation industry is losing air traffic control talents, while the need for maintaining daily operations keeps surging [100]. This situation leads to increased operational costs, higher safety concerns, the elevated workload for air traffic controllers, and frequent flight delays [149]. Flight delay is a major problem of interest faced by domain experts, which results in both economic and customer loyalty losses [150]. It's reported that 20% of the civil flights in the U.S. were delayed from 2010 to 2018, and the annual cost of delays before the pandemic is estimated to be $\$30$ billion [151]. The initial flight delays come from various resources (e.g., extreme weather conditions, carrier and controller issues) and can propagate through several hours [152; 153]. Moreover, the aviation industry is encountering a shortage of experienced operation talents after the COVID-19 pandemic due to various reasons (e.g., loss of operational and airline experience, staffing, and changing customer demand patterns). All of this urges the automation and digitization of the aviation industry in a regulated fashion, which heavily relies on innovative data-driven modeling techniques.

Automated computer-aided decision support tools (DSTs) are practical solutions to address safety and efficiency concerns (e.g., flight delays), with the help of modernized data monitoring and recording equipment. DSTs will help maximize the operational capacity of the terminal maneuvering area (TMA), where the optimization of departure/arrival operations in the TMA is a critical problem of air traffic control (ATC). In most cases,

the heuristic decision by the ATC will be suggested together with a graphic view of each corresponding location and speed of the aircraft near the TMA. This setup is efficient on normal operations but leads to flight delays and elevated controller workload during extreme scenarios. The unfolded diamond shape symbols in the graphic view may overlap and lead to significant delays during certain extreme cases [154]. DSTs are developed to alleviate flight delays and maximize operational capacity during certain cases and busy traffic. For instance, the measurement coverage of NextGen will be enlarged to hundreds of nautical miles (NM) due to the advanced surveillance radar for the Automatic Dependent Surveillance-Broadcast (ADS-B) system [155]. The enlarged surveillance measurement space enables the possibility of developing optimization-based DSTs, to be applied in the en-route phase. Lastly, DSTs assist controllers in suggesting reasonable resolutions by searching from historical data or learning from human preferences. Various government agencies proposed advanced DST system concepts. *Airport Collaborative Decision Making* (A-CDM) [156] concept and *Next Generation Air Transportation System* (NextGen) [8] was proposed by the European Organization for the Safety of Air Navigation (EUROCONTROL) and Federal Aviation Administration (FAA) to assist air traffic controllers in decision makings, with enhanced safety, efficiency, and capacity. Field demonstrations on either single-airport or multi-airport scenarios show great safety enhancements and efficiency improvements [157; 158].

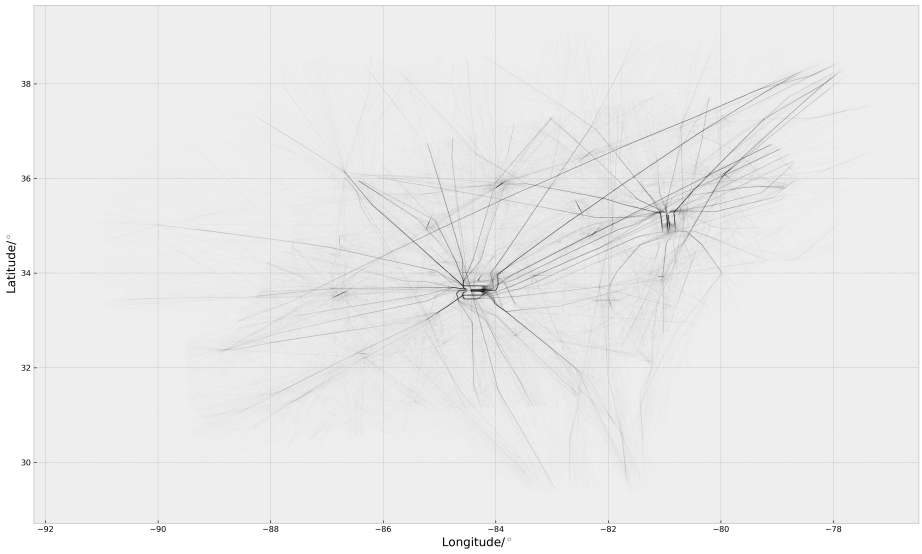


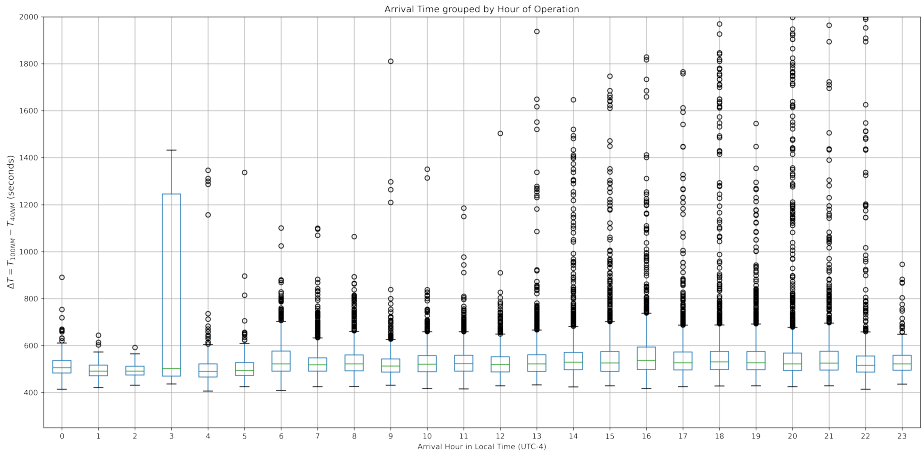
Figure 4.1: Flight Tracks on Aug 1st, 2019. Archived in Sherlock Data Warehouse [3] for ARTCC ZTL.

While the government-led efforts mostly focus on building the system workflow for on-board deployment, academic research focuses on algorithmic development and advanced data analytics to enable automated decision-making to support aviation digitization. The Aircraft Landing Scheduling (ALS) problem is vital to overcome flight delays and achieve efficient aviation operations in the TMA [159]. ALS studies the planning of the landing schedule for all the aircraft landed on the same runway in a short time period [160], where the runway capacity is pre-defined based on the existing infrastructure [161]. In aviation, the ALS problem is viewed as a critical element of the general planning system of aircraft around the TMA [159]. Researchers who are studying ALS focus on the following objectives,

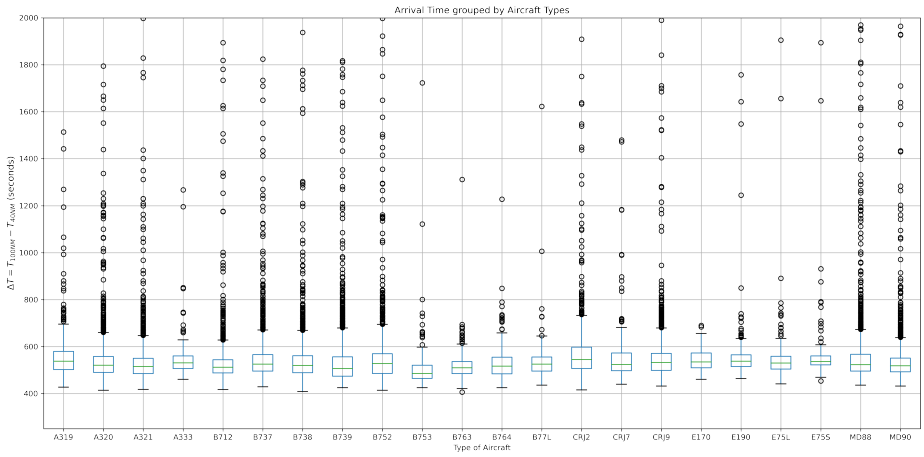
- Maximize the fuel efficiency by arranging the landing aircraft at the most economic landing times and speed profiles [162; 163; 164].

- Minimize the difference to the flight schedules [165; 166; 167].
- Maximize the runway throughput by minimizing the total landing time [168; 169; 170].

This chapter focuses on the last item, i.e., maximizing the runway throughput. During arrivals, the air traffic controllers (ATCs) give instructions to the pilots when the aircraft enters the range of the terminal surveillance radar. Thus, ATCs provide guidance for safe and effective landings. Landing safety is enforced by the minimum separation time (MST) between two landing aircraft. The MST is introduced to account for aerodynamic safety considerations [161]. For instance, when the leading aircraft is much heavier than the following aircraft, the leading aircraft's wake vortices will result in hazardous conditions for the following lighter aircraft within MST and poses immediate safety concerns. The ALS problem has been formulated into two sub-problems [160]. Firstly, the order of the aircraft entering the TMA is determined. Then, the exact scheduled landing time is determined based on the landing sequence and MST. These two steps can be collaboratively solved with proper optimization algorithms. The extension of the surveillance area enables the possibility of developing a novel landing scheduling scheme that can be performed in the en-route phase rather than only in the terminal area to prevent congestion and reduce congestion-related safety concerns. However, the current literature either focuses on formulating the optimization problem in both static [163] and dynamic [171; 161] scenarios with synthetic examples, or considering one of the related factors during sequencing to formulate the mathematical model (e.g., ground staff workload [172], airline preferences [173]). The above pure simulated demonstration limits the applicability and generalizability of the developed algorithms to be deployed in the real world.



(a) Boxplot Grouped by Daily Hours. It's Obvious That The Busy Hour Typically Starts From 13:00 to 22:00 Each Day.



(b) Boxplot Grouped by Aircraft Types. There Is Not a Clear Correlation Between Time Spent in TMA to Aircraft Types.

Figure 4.2: Time Spent from 100NM to 40NM for Landing Aircraft Entering the TMA of KATL During the Entire Month of August 2019.

The availability of a well-maintained aviation data warehouse [3; 21] has enabled the

possibility of learning and generating aviation operational decisions from realistic operational data. Machine Learning (ML) is an example of data analytics that draws interest from both academia and industry. In the ATM domain, the use of ML techniques also surges in recent years [122; 174; 95; 103; 175; 176], although multiple challenges (i.e., data privacy/collection/storage/integrity, system reliability, and scalability) still exist when deploying ML systems into real-world (MLOps) [177; 29]. Compared to the conventional methods, ML methods show the following benefits: a) A ML-based DST takes advantage of realistic historical data to simulate the human experience accumulation process, where the model can provide *experienced* guidance within the machine response time; b) ML methods are highly flexible to fuse structured or unstructured data from various sources for decision-making. Nonetheless, criticisms against ML methods also rise regarding model interpretability/explainability, prediction generalizability, and output trustworthiness. The authors believed that ML-based DSTs are beneficial for computer-assisted decision-making under the supervision of human controllers.

In view of the above discussion, there is a need and gap to develop data-driven aircraft landing scheduling algorithms from extended airspace to maximize runway throughput and reduce flight delays. In this chapter, I first investigate and identify several factors causing flight delays through data analysis. Then, I propose a data-enhanced optimization technique for ALS, where the statistics of minimum separation time (MST) is incorporated into the safety-critical constraints under the TSP formulation. The probabilistic MST is learned with a conditional tree-based ML method, namely a conditional gradient boosting machine (conditional GBM) with quantile distributions to retrieve the upper and lower bounds of MST. Following this, an optimal method using TSP formulation solved by MILP is proposed for sequencing to minimize total delay while taking into account the uncertainty of arrival time prediction.

The contributions of this work can be summarized as

- I investigate several arrival delay scenarios that occurred in the historical data and gain the following insights, a) the arrival time of aircraft has a highly multimodel distribution conditioned on the flight events; b) go around (looping) - the event happened in most arrival delay scenarios - occurs at approximately 100 nautical miles away from the terminal, where FCFS rule starts to take effect; c) the preference of landing scheduling made by human controllers may not be optimal (e.g., landing aircraft from west of terminal should yield to other directions on a west heading runway). This observation gives insights into identifying relevant impact factors in building ALS solutions.

- The statistics of MSTs are predicted with a tree-based probabilistic machine-learning algorithm from the historical flight recordings. The obtained probabilistic MSTs are incorporated as safety constraints to the time-constrained traveling salesman problem. To the author's best knowledge, this type of probabilistic scheduling setting is the first time.

- I propose to use a conditional ML predictor based on the event counts within a certain distance of the target aircraft to improve the prediction performance. Geographical location, speed profiles, flight event counts, and airspace complexity measures are integrated together for probabilistic prediction of arrival time, which has not been explored in the open literature.

- The proposed framework shows a reduction in total aircraft landing time compared to the FCFS rule, through case studies during busy operation hours at KATL. The proposed method takes effect from extended airspace (e.g., en-route phase flights 200NM away from the terminal), such that early adjustment of aircraft speed profiles can be issued to avoid holding patterns.

The rest of the paper is organized as follows. In Section 5.2, I review the related

literature on this topic first. The methodology proposed in this research is discussed in Section 5.4, where the optimization formulation and machine learning predictor are discussed. Investigations on flight delay scenarios and insights are discussed in Section 4.4. The optimization case studies and experimental results are shown in Section 5.5. Finally, conclusions and future insights are given based on the current investigations.

4.2 Literature Review

This section discusses the related literature to our proposed study on data-enhanced ALS. I first review the studies for the prediction of aircraft estimated arrival time (ETA) and MST in Section 4.2.1. Then, I review the research on aircraft landing scheduling problems in Section 4.2.2.

4.2.1 Estimated Arrival Time Prediction and Minimum Separation Time (MST)

Landing aircraft move along the predefined landing procedures with standard descending profiles when entering the TMA, with the help of necessary guidance from ATCs. The MST between two consecutive landing aircraft should be guaranteed in the approaching phase. The MST depends on the types and relative positions of two consecutive landing aircraft, which can be translated by considering the speed profiles [159]. Once a landing aircraft enters TMA, it should line up and proceed to the runway. However, delays happen on a daily basis and can propagate from ground to mid-air airplanes due to the sub-optimal scheduling of runway usage. In this case, ATC issues a *holding* order to the approaching aircraft and forces the aircraft to circle around and wait for the clearance to land. The conservative determination of the landing safety buffer will result in lower runway throughputs, with larger landing intervals between landing aircraft. In extreme cases (e.g., severe convective weather conditions), the delay might be very significant and prolonged due to high congestion and weather uncertainties. Real-time traffic management systems (e.g.,

Integrated Arrival Departure Surface Traffic Management by NASA) consider potential conflicts by constantly adjusting the group of aircraft within TMA in terms of re-routing, re-timing, and holding [159].

To properly include MST as the safety-critical *landing buffer time* with various operational uncertainties, I predict the ETA along with the corresponding ETA confidence interval. The prediction of ETA usually happens upon the aircraft entering the TMA, which is usually 40 minutes ahead of landing [178]. Early works to predict arrival time focus on using physics-based trajectory models, which are usually associated with the aircraft performance, flight plan, and the predicted atmospheric conditions provided by flight-desk systems [179]. In [180], a method is proposed to predict the arrival time in heavy weather conditions using the aircraft dynamics and weather avoidance algorithm. Estimated time of arrival time prediction is approached from a hybrid linear system in [181], then the chosen route probability is further incorporated for stochastic arrival time prediction [182]. A state-dependent hybrid estimation method is used for improved prediction accuracy in [183]. Many 4D trajectory prediction algorithms with various kinematic assumptions can also provide an estimated time of arrival [184; 185; 186].

Data-driven methods for arrival time prediction have increased rapidly in recent years, due to the rise of machine learning and well-maintained data storage facilities. Tree-based methods have been used to predict air traffic delays [187; 188; 189; 190], where the weather-related features are taken into account to enhance arrival time prediction capabilities. However, tree-based methods with quantile regression [191] have not been used for uncertainty quantification of arrival time predictions. Deep learning methods, such as recurrent neural networks (RNN), are also adopted for arrival time prediction under different circumstances [192; 193]. Moreover, the importance of feature selection in air traffic prediction is discussed in [149]. The experiments with Changi extended TMA conclude that when building machine learning models for air traffic prediction tasks, feature selec-

tion with the help of domain knowledge is critical. The model performance is less sensitive to the selection of machine learning algorithms itself. This also guides the discoveries on feature studies and case analysis in the later sections of this chapter.

4.2.2 Aircraft Landing Scheduling

The definition of the ALS problem is as follows. Assume that there are n aircraft lining up for landing on a single runway. The objective of the ALS problem is to find a schedule of the respective landing time $\{t_1, t_2, \ldots, t_i\}$ for each aircraft $\{1, 2, \ldots, i\}$. In ALS, there are two constraints to be satisfied: 1) the aircraft must land within a specific time period; 2) the minimum separation time between each pair of landing aircraft should be guaranteed [194]. The common practice for ALS used by ATC is following the First-Come-First-Served (FCFS) rule, where the scheduled landing sequence is consistent with the time for each aircraft entering the TMA. FCFS is convenient to maintain safe landing operations but can lead to severe delays during busy hours (e.g., Figure 4.4). The FCFS rule has several known drawbacks, a) the lower speed leading aircraft will impact the following aircraft, even if the following aircraft has higher ground speed; b) the FCFS rule can create unnecessary long separations for aircraft with different weight classes; c) when terminal area congestion presents, the aircraft reaching TMA will hold and poses significant delays to wait for the clearance of congested aircraft; d) the delay will easily propagate to several hours in extreme scenarios.

Thus, many researchers have proposed different approaches to optimizing the aircraft landing sequence within the scheduling range. The ALS problem can be formulated into mixed-integer program [163], where the relationship to machine scheduling problem has been exploited in the literature [195; 196]. Researchers propose a variety of algorithms to address the ALS problem: 1) the ALS problem can be classified into dynamic and static scheduling approaches, depending on whether the environment is dynamically changed or

not [178; 163; 171; 161]; 2) the scheduling algorithm itself considers various impact factors and objective functions, such as airlines' preferences [173], ground workload [172], and cellular automation [197]; 3) consider the ALS problem from limited airspace or extended airspace. A detailed review of the above-mentioned three major perspectives is given below.

Static Scheduling v.s. Dynamic Scheduling Static aircraft landing scheduling defines the ALS problem with a predetermined time window, such that the scheduling constraints are ensured. [163] proposes a mixed-integer zero-one formulation of ALS for both single and multiple runway scenarios, to consider commonly encountered issues in practice (e.g., restricting the number of total landings in a given period). The problem is further solved with linear programming-based tree search. [198] proposes a static optimization algorithm for aircraft landing in a single-runway, uncontrolled airport, with performance metrics such as total holding time and total landing time. Some other researchers view dynamic programming as a feasible approach to ALS. [199] adopts [163]'s formulation but with a novel dynamic constraints generation algorithm. The proposed algorithm approximates the MST into a rank two matrix, which leads to linear programming with relaxation. The dynamic ALS problem received less attention in the literature and is usually achieved with the same approach called rolling horizon [161]. Rolling horizon is as simple as rolling the time window of agents for optimization. Firstly, the aircraft inside TMA within the rolling horizon (typically several minutes) are optimized. Then, the landed aircraft are removed from the rolling horizon, and the new aircraft just entered the rolling horizon are added to the algorithm. [200] solves dynamic ALS with genetic algorithms using data from Sydney airport, and shows that the genetic algorithm can perform good results in real-time with a rolling horizon of 3 minutes.

Optimization Objectives & Related Factors Researchers working on the ALS problem consider various impact factors with different optimization objectives. In [163], the authors focus on reducing the deviation from the scheduled landing times. A linear programming-

based tree search method was proposed for landing scheduling, building upon the pioneering work of mixed-integer programming formulation for ALS [201]. Similarly, [165] extend the work to reduce deviations from scheduled landing times under time window constraints, but the MSTs are pre-defined for five different aircraft weight classes. Based on the tree search approach proposed in [163], [173] considers airline preferences into the optimization framework, in which the optimal landing sequences are given by tree search and MILP is used to determine the optimal landing time. Dynamic programming-based landing sequencing method is proposed in [202] to maximize the runway throughput. [202] achieves a highly satisfactory result, but the concern on computational complexity limits the real-world applicability. Studies on alleviating computational complexity are also conducted, such as the cellular automaton optimization method [160], ant colony optimization method [203], genetic algorithm [178; 204], and population heuristic algorithm [165; 164].

TMA Scheduling Range There have been several studies focusing on changing the range of the TMA for ALS. Some of the researchers propose to perform landing scheduling on the entire TMA, to consider the ALS problem from a systematic view. [205; 206] divide the ATC controls into routing decisions, scheduling decisions, and air segments and runways. Then, a job shop formulation is used to reduce the delay caused by conflicts in TMA. More recently, researches on arrival management suggest that performing aircraft sequencing in an extended area rather than in TMA is actually an effective solution. This concept allows ATCs to monitor and control traffic into a busy terminal area from the enroute phase, enabling aircraft to adjust their speed before their top of descent. Thus, time spent in mid-air holding in the TMA can be reduced. In [207], an algorithm is developed using the merging optimization method to simultaneously optimize trajectories, arrival sequence, and allocation of aircraft to parallel runways. A two-stage stochastic mixed-integer programming model is proposed in [208]. Another study [209] assessed the effect of flights departing on extended arrival management, in terms of the flight crew and air traffic control

task load, sequence stability, and delay. Two-stage stochastic programming is presented in [210] to address the arrival sequencing and scheduling problem under uncertainty.

Several existing gaps can be identified from the above review. For example, the existing landing scheduling methods assume the actual arrival times to deviate randomly from target times (calculated using the en-route speeds) to infer MST. Also, the scheduling algorithm assumes a pre-defined MST based on the aircraft weight classes. In practice, there is tremendous uncertainty associated with the arrival time prediction, which violates the assumptions of deterministic separation. Several factors, such as aircraft type, weather conditions, and airspace density information can be explicitly acquired in the aviation database and should be used to reduce the uncertainties of arrival time prediction. In addition, the assumption of static and fixed arrival time distributions is not valid and may cause ineffective landing scheduling and/or unsafe separation between aircraft (examples shown later using realistic data). The exact arrival time prediction with accurate uncertainty quantification for each landing aircraft should be determined, which further optimizes landing schedules for all of the landing aircraft with an ensured confidence level. Thus, the main focus of this chapter is to develop a real-world data-enhanced landing scheduling algorithm to achieve optimal landing scheduling with uncertainties.

4.3 Methodologies

This section demonstrates the methodologies for ML-enhanced ALS. I first illustrate the tree-based machine learning selected – Gradient Boosting Machine (GBM) with quantile regression in Section 4.3.1. Then, I provide the necessary background to the Traveling Salesman Problem (TSP) and introduce the formulation of time-constrained TSP formulation to solve the ALS problem in Section 4.3.2. Following this, I describe our proposed approach to integrating machine learning prediction of arrival time into time-constrained TSP formulation in Section 4.3.3.

4.3.1 Gradient Boosting Machine

Although the literature has concluded that selecting the correct feature set is more advantageous than pursuing the most advanced machine learning algorithms [149], I choose tree-based machine learning algorithms due to their proven outstanding performances on structured data. Furthermore, across various tree-based machine learning algorithms, I select boosting over simple trees or bagging. Gradient Boosting Machine (GBM) is a commonly used model [211; 212; 213]. Boosting methods add new base learners to the ensembles at each iteration, and each base learner has trained w.r.t. the residual from the current ensembles. GBM connects boosting and optimization [212; 213] to perform gradient descent on both the loss functions and the base learners.

Considering a supervised learning problem with structured data $\mathcal{D} = \{(x_i, y_i)|i = 0, ..., n\}$, where $x_i \in \mathbb{R}^M$ is also called the feature vector of the i-th sample with M different features, and y_i is the continuous response as the label of x_i in a regression problem. In GBM, I have a set of base learners $\mathcal{B} = \{b_{\gamma_m}(x) \in \mathbb{R}^M\}$ parameterized by γ_m. In GBM, the predictions are the linear combination $\mathsf{lin}\{B\}$ of the predictions from each of the base learner $b_{\gamma_m}(x) \in \mathcal{B}$, where each of the base learners is additively learned with pseudo-residuals (τ_m). The corresponding predicted label to feature vector x_i is $f(x_i) \in \mathsf{lin}\{B\}$, in the form of,

$$f(x_i) = \sum_{m=0}^{M} \alpha_m b_{\gamma_m}(x_i) \tag{4.1}$$

where α_m is the coefficient for each base learner $b_{\gamma_m}(x) \in \mathcal{B}$. Examples of base learners include linear models, support vector machines, classification, and regression trees [212; 214]. Additionally, for the most popular tree-based learners set, the GBM turns into Gradient Boosted Decision Trees (GBDTs). GBM aims to obtain the best function set $\hat{f}(x_i) \in \mathsf{lin}\{B\}$ to minimize the given data-fidelity evaluation function (e.g., least squared

residual loss for simple regressions).

$$\hat{f}(x_i) = \mathsf{argmin}_{f(x_i)\in \mathsf{lin}\{B\}} \sum_{i=0}^{n} \xi_i(y_i, f(x_i)) \tag{4.2}$$

where $\xi_i(y_i, f(x_i))$ is the data-fidelity evaluated at the i-th feature vector.

Using the defined notations above, the GBM minimizes the loss function by calculating the steepest descent to the objective function defined in Equation (4.2), where the steepest gradient is determined with line-search on the best base learner parameter set $\hat{\gamma}_m$.

The following research explores the possibility of improving the boosting method performance from many perspectives [213].

- Introduce a learning rate λ to the updating equation of $f(x)$: $f^{m+1}(x) = f^m(x) + \lambda\rho_m b_{\gamma_m}(x)$. Multiplying λ provides the damping of controlling the rate of descent on the error surface.

- Sampling without replacement from the dataset before the gradient calculation step gives stochasticity to GBM and greatly improved the performance of the algorithm.

- Using ANOVA decomposition can restrict the depth of the trees, which further controls the order of approximations of GBM: $f(x) = \sum_i f_i(x_i) + \sum_{ij} f_{ij}(x_i, x_j) + \sum_{ijk} f_{ijk}(x_i, x_j, x_k) + \cdots$

Specifically, GBMs can be turned into probabilistic predictors when applying quantile distributions to the response variable. The gradient calculation of τ_m changes to the quantile pseudo-redisual $\tau_m = \beta\xi(y_i \geq f(x_i)) - (1-\beta)\xi(y_i \leq f(x_i))$, where $\beta = \frac{\sum_{i=0}^{n} \omega_i \xi_i(y_i \leq q)}{\sum_{i=0}^{n} \omega_i}$ and q denotes the weighted quantile. With the GBM prediction label of the defined quantile, given a test sample $\hat{x_i}$, I can obtain the confidence interval $\sigma\hat{y_i}$ along with the prediction $\hat{y_i}$.

4.3.2 Traveling Salesman Problem with Time Windows (TSP-TW)

The ALS problem involves landing sequencing and landing scheduling, which is a discrete optimization problem in nature. Combinatorial optimization tackles discrete optimization problems from the intersection of combinatorics and theoretical computer science. Combinatorial optimization is widely utilized to solve tasks like resource allocation and scheduling for transportation and supply chains. TSP-TW is a classical combinatorial optimization problem [215]. The original definition of TSP-TW aims at finding the optimal tour that minimizes the length of the tour and visits each node once within the specified time window $[l_i, u_i]$, where l_i and u_i are the lower and upper bound for visiting time of node i. The bounded time windows set time constraints to the agent traveling within the node graph, and mark the significant difference to classical TSP problems. TSP-TW has been applied to bus scheduling and delivery systems [215]. In this work, I propose to use TSP-TW for ALS and incorporate the machine learning predicted aircraft ETAs into the constraints of TSP-TW.

Define an undirected graph $G = (V, A)$ with a finite set of nodes, $V = \{0, 1, \ldots, n\}$, and a finite set of edges, $A = \{(i, j)|i \neq j, i, j \in V\}$. TSP-TW determines the time t_i that the agent visits node $i \in \{0, 1, \ldots, n\}$. Meanwhile, an additional variable, t_{n+1}, is introduced to represent the completion time of the tour, as the agent has to return to node 0 at the end of the tour. A distance matrix, t_{ij}, records the shortest distance between each node pair, which can be further treated as the scalar transformation of time distance between node pairs [216]. Mathematically, the classical formulation of TSP-TW is shown in Equations (4.3) to (4.8).

$$\min \quad t_{n+1} \tag{4.3}$$

subject to

$$t_i - t_0 \geq t_{0i} \qquad i = 1, 2, \ldots, n \tag{4.4}$$

$$|t_i - t_j| \geq t_{ij} \qquad i = 2, 3, \ldots, n; 1 \leq j < i \tag{4.5}$$

$$t_{n+1} - t_i \geq t_{i0} \qquad i = 1, 2, \ldots, n \tag{4.6}$$

$$t_i \geq 0 \qquad i = 0, 1, \ldots, n+1 \tag{4.7}$$

$$l_i \leq t_i \leq u_i \qquad i = 1, 2, \ldots, n \tag{4.8}$$

To solve TSP-TW, there are several methods spanning from mathematical programming approaches to heuristic approaches. Mixed Integer Linear Programming (MILP) techniques are commonly used approaches to solve TSP-TW. Despite the difference between problem setups and applications, researchers propose various methods to solve TSP-TW with MILP for up to 200 clients [217; 216; 218]. Additionally, constraint programming methods are proposed to develop both exact [219] and heuristic [220] solvers for TSP-TW.

$$\min \quad t_{n+1} \tag{4.9}$$

subject to

$$t_i \geq t_{0i} \cdot y_{0i} \qquad i = 1, 2, \ldots, n \tag{4.10}$$

$$t_i - t_j + (u_i - l_j + t_{ij}) \cdot y_{ij} \leq u_i - l_j \qquad \forall i, j = 1, 2, \ldots, n : i \neq j \tag{4.11}$$

$$\sum_{i=0}^{n} y_{ij} = 1 \qquad j = 1, 2, \ldots, n \tag{4.12}$$

$$\sum_{j=0}^{n} y_{ij} = 1 \qquad i = 1, 2, \ldots, n \tag{4.13}$$

$$t_i + t_{i0} \leq t_{n+1} \qquad i = 1, 2, \ldots, n \tag{4.14}$$

$$l_i \leq t_i \leq u_i \qquad i = 1, 2, \ldots, n \tag{4.15}$$

$$y_{ij} \in \{0, 1\} \qquad \forall (i, j) \in \{(i, j) : i, j \in 0, 1, ..., n\} \tag{4.16}$$

$$t_i \geq 0 \qquad \forall i = 0, 1, ..., n + 1 \tag{4.17}$$

In this work, I introduce the minimum separation time (MST) into the constraints of the TSP-TW model (Equations (4.9) to (4.17).). The objective is to minimize the total landing time for all aircraft. u_i and l_i denote the earliest and latest time for aircraft i to land, respectively. $u_i - l_i$ indicates the maximum allowed flight time of aircraft i, which can reflect the aircraft conditions (e.g., fuel, pilot fatigue level, etc). y_{ij} is the adjacency matrix, defined as,

$$y_{ij} = \begin{cases} 1, & \text{if aircraft } j \text{ lands right after aircraft } i, \\ 0, & \text{otherwise.} \end{cases} \tag{4.18}$$

Equation (4.11) describes the constraint on the separation requirement between two consecutive intermediate aircraft. t_{ij} denotes the MST from aircraft i to aircraft j. Section 4.3.3 will discuss the method used to incorporate GBM predicted MST into t_{ij}. As MST depends on the wake turbulence generated by the leading aircraft, the formulation is an asymmetric TSP-TW problem, indicating $t_{ij} \neq t_{ji}$. The time window for the agent to

visit a node corresponds to the specified time range for the aircraft to start to land, considering the fuel consumption and aircraft dynamics. Equations (4.10) and (4.14) guarantees that the smallest and largest t_i values. Equations (4.12), (4.13) and (4.16) ensure that each aircraft will land exactly once. Equation (4.15) introduces the pre-determined time schedule of each aircraft. In practice, I solve the model in Equation (4.9) with GLPK solver [221] and Python Optimization Modeling Objects (Pyomo) package [222].

4.3.3 Incorporating Uncertainties of MST Constraints to TSP-TW

As I reviewed earlier, there are tremendous uncertainties associated with the estimated arrival time and minimum separation time (MST). Thus, the proposed study will include uncertainties in the landing scheduling problem to ensure confidence.

For each successive landing aircraft pair (i, j), GBM with weighted quantile gives the predicted landing time distributions for variable t_i and t_j from real-world data. It is assumed that arrival time for landing aircraft follows i.i.d. Gaussian distributions. The MST is defined as the difference between the two arrival time for the two aircraft (i, j). Thus, the MST can be expressed

$$t_{ij} \sim \mathcal{N}\left(\mathcal{T}_{ij}, \sigma_{ij}\right) \tag{4.19}$$

where $\mathcal{T}_{ij}$ is the referenced MST between aircraft i and j by the related authorities [1]. $\sigma_{ij} = \sqrt{\sigma_i^2 + \sigma_j^2}$ represents the uncertainty of MST from the quantified uncertainties (standard deviation) from the arrival time of the two aircraft (i, j). The major reference values of $\mathcal{T}_{ij}$ are listed in Table 4.1.

Table 4.1: $\mathcal{T}_{ij}$: Minimum Required Time-Space (s) Used by Arrival Manager [1].

		Trailing Aircraft			
		Heavy	B757	Large	Small
Leading Aircraft	Heavy	82	118	118	150
	B757	60	64	64	94
	Large	60	64	64	94
	Small	60	64	64	94

Given a fixed spacing conflict probability P_c, the MST between landing aircraft i and j, $\check{t}_{ij}$ can be calculated,

$$\check{t}_{ij} = \Phi^{-1}_{t_{ij}}(P_c) \tag{4.20}$$

where $\check{t}_{ij}$ forms the separation constraints in Equation (4.9). By Equation (4.20), the minimum allowable separation time between two successive landing aircraft pair (i, j) is obtained as $\check{t}_{ij}$. It is worth pointing out that $\check{t}_{ij}$ is different from $\check{t}_{ji}$, since the MSTs are significantly impacted by the leading aircraft. Additionally, the predicted mean values of estimated landing times μ_i and μ_j are included in the upper and lower bound (u_i and l_i) of Equation (4.11) and Equation (4.15).

In this work, I aim to predict the arrival time from 200 miles of the TMA, and the aircraft can adjust the speed in the en-route phase to reach the scheduled arrival time. The fuel consumption can be limited to a low level if the scheduled arrival time is constrained to a time window around the optimal speed. Thus, fuel consumption is also considered in the constraints. I incorporate the fuel consumption constraints into the calculation of upper bound u_i and lower bound l_i of the time window constraints, adjusted based on the distribution of t_{ij}.

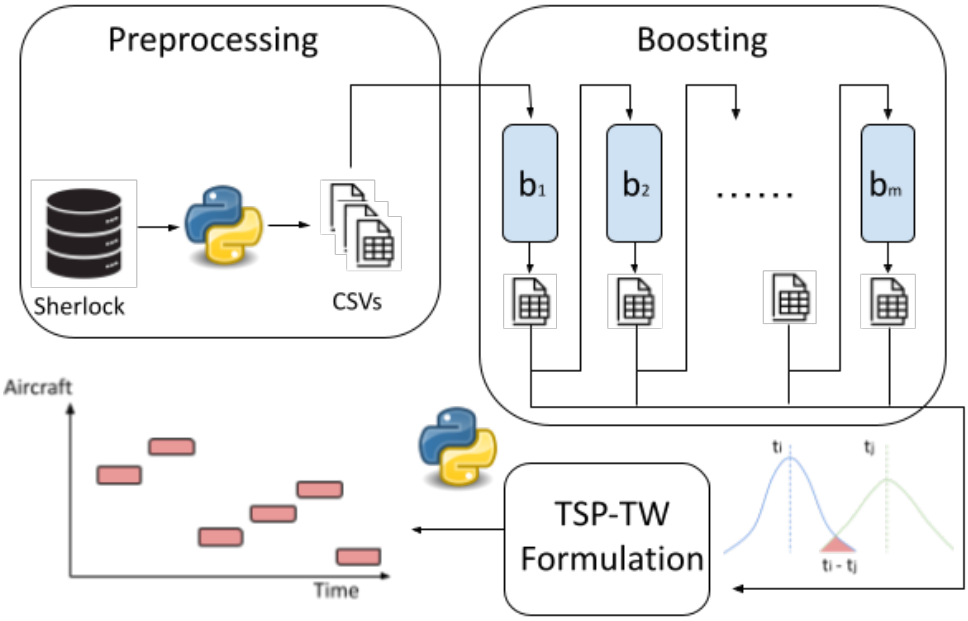


Figure 4.3: Proposed Machine Learning-Enhanced Optimization Model for Aircraft Landing Scheduling. The Aviation Source Data Are Obtained from the Sherlock Data Warehouse and Processed into the Well-organized Tabular Dataset. The Boosting Model Takes the Tabular Dataset and Fits into Base Learners Sequentially, Where the Residuals Are Concatenated for the Best Set of Base Learners. The Boosting Model Predicts the Distribution of the Landing Time Difference Between Two Successive Flights and Formulates the TSP-TW Constraints for ALS.

The complete ALS procedure is shown in Section 4.3.3. GBM is trained with real-world air traffic data, to predict the estimated aircraft landing time with associated uncertainty intervals. First, a look-ahead horizon is defined to determine the rolling time window for scheduling. For instance, the TMA ranges from 100 nautical miles to 200 nautical miles from the destination airport. The detected number of aircraft in this area is n. The maximum number of aircraft to be scheduled is constrained to n_{max} to limit the computation complexity for real-time implementation. If $n \geq n_{max}$, the first n_{max} aircraft is selected

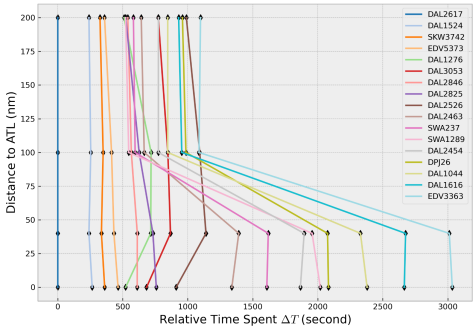
and performs the landing scheduling, otherwise select all the aircraft in the current horizon. The trained model is used to predict the arrival time of all the affected aircraft. Then the scheduled arrival time is calculated using the algorithm described above.

4.4 Empirical Data Analysis

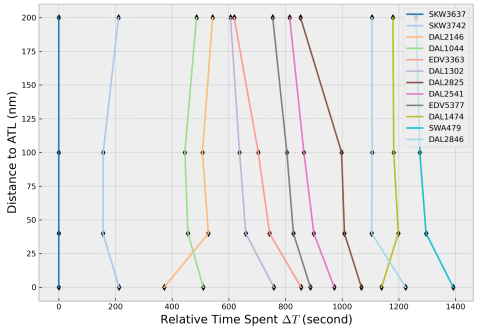
In this section, I first investigate several flight delay scenarios via real-world aviation flight recordings. Through the investigations, I discover that the holding pattern is one of the major impact factors leading to flight delays. I propose to explicitly include safety-related flight event counts as the features for GBM to demonstrate effectiveness. A short description of the flight track and flight event data is given.

4.4.1 Investigation on Flight Delays

Figure 4.4 shows the relevant time spent from three distance intervals ($[200, 100], [100, 40], [40, 0]$ nautical miles) away from the terminal. I obtain and visualize the flight track data from Sherlock Data Warehouse (SDW) [3]. SDW is a distributed big data platform for data visualization to support air traffic management research. Sherlock includes a database, a web-based user interface, a few data visualization tools, and other services. The flight data is stored in the Integrated Flight Format (IFF). It includes all raw data plus the derived fields such as flight summary, track points, and flight plan. The flight summary is a general description of the flight which contains flight time, flight call sign, aircraft type, origin, and destination information. The flight track points are the record of real flight operations. It includes the ground-measured aircraft position in both the spatial and temporal aspects. The format of flight track data in SDW and conversion between WGS84 coordinates to absolute distance has been discussed in previous work [175].



(a) Aug 1st, 2019@13:20 - 13:45



(b) Aug 8th, 2019@13:20 - 13:45

Figure 4.4: Flight Landing Progress Recording During Busy Hours. A Comparison Between Two Consecutive Mondays. Both Days Present Normal Weather Conditions.

In Figure 4.4(a), I visualize the landing aircraft between the time window $13:20$ and $13:45$ on Monday, Aug 1st, 2019, during which severe landing flight delays happened. For reference, I also visualize the normal flight landing process of the next consecutive Monday in Figure 4.4(b). During the given time window, 17 flights entering the 100 nautical mile range from the terminal are captured by the surveillance radar. Figure 4.4(a) shows flight delay starts at the 5-th flight.

I draw the complete tracks for 3 landing flights (DAL1276, DAL3053, DAL2526) coming from the northeast towards an east landing, and 7 landing flights coming from the northwest towards an east landing. In Figure 4.6 and Figure 4.7, the diamond marks represent the location of reaching $200, 100, 40$ NM from the airport center. I gain valuable insights from Figure 4.6 and Figure 4.7, a) holding command is the common practice during ALS, and the flight receives a holding command loop around in the airspace; b) holding pattern contributes to the long arrival time within the TMA, and can last for various period; c) holding patter usually happens when the aircraft is in TMA range $[100, 40]$NM, where FCFS rule takes effect for air traffic control.

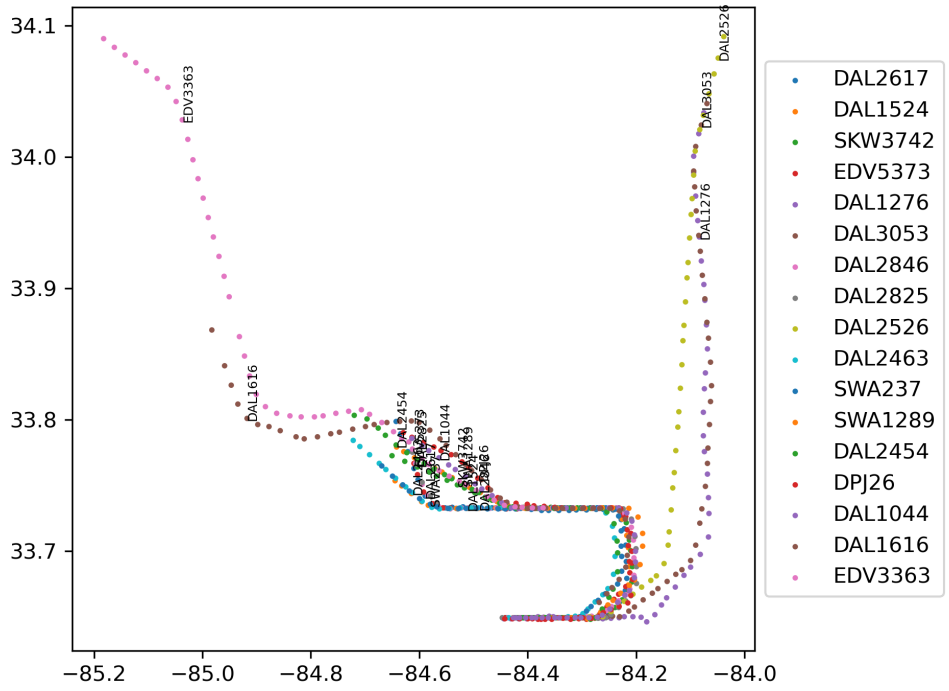
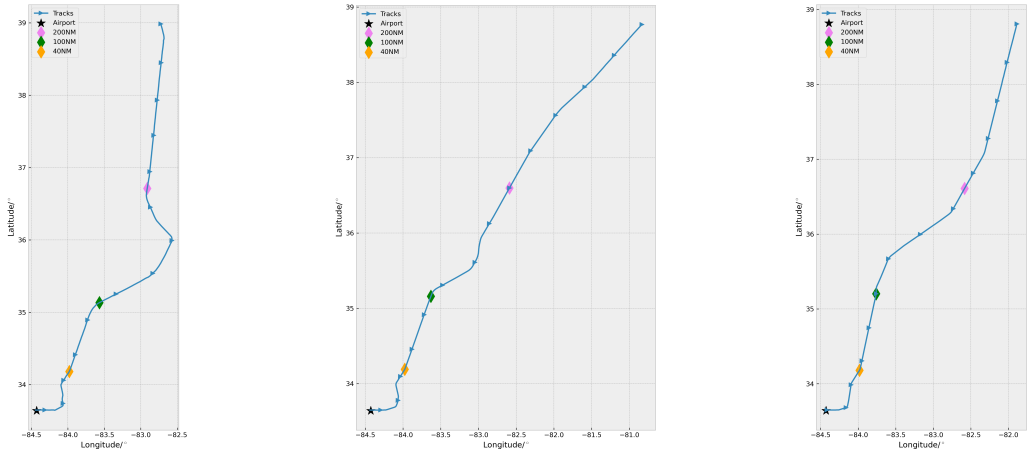


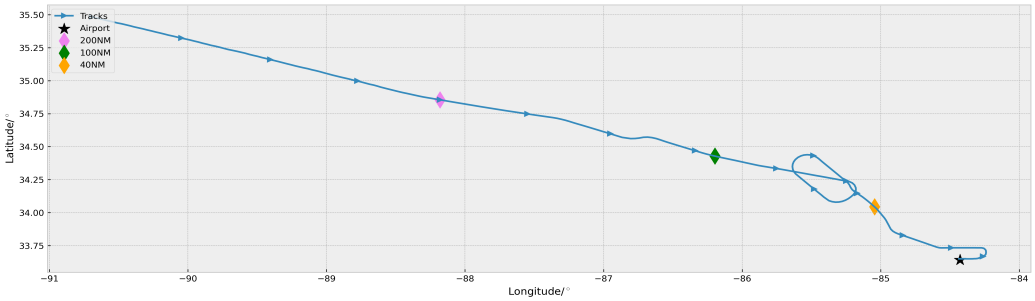
Figure 4.5: Flight Tracks for Landing Aircraft in Figure 4.4(a).

It's obvious that holding in the congested near terminal airspace poses a safety concern to air traffic operations, which motivates our proposed method to perform from an extended TMA. In this work, I propose to do ALS from an extended TMA, such that an early landing scheduling can be issued. By doing this, I can alleviate the near terminal airspace complexity, and landing aircraft can adjust the speed profile to account for the issued arrival time, over 100NM away from the terminal.

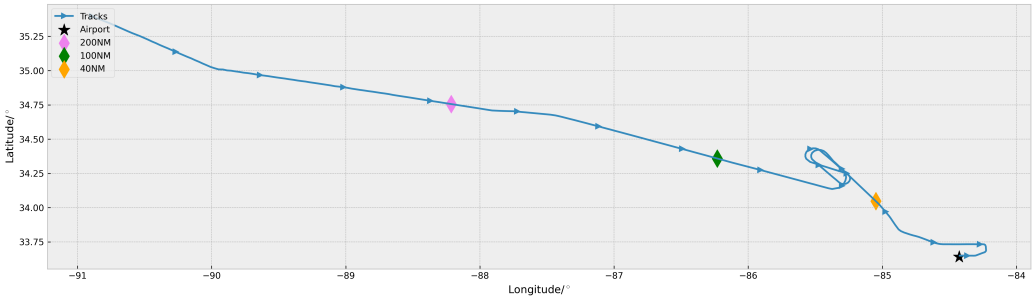


(a) DAL1276 (b) DAL3053 (c) DAL2526

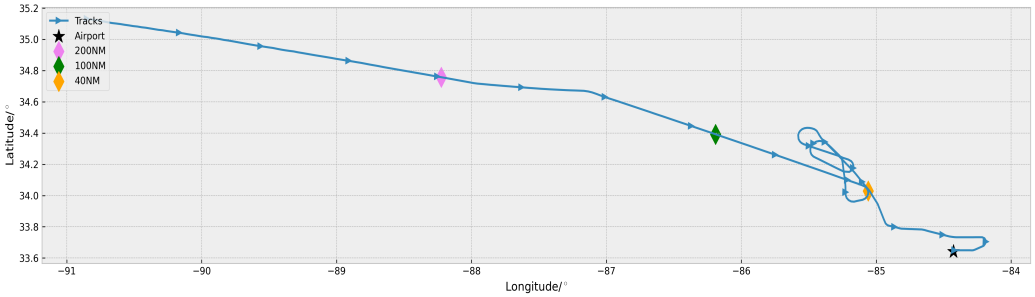
Figure 4.6: Landing Aircraft Coming From the Northeast for West Landing.



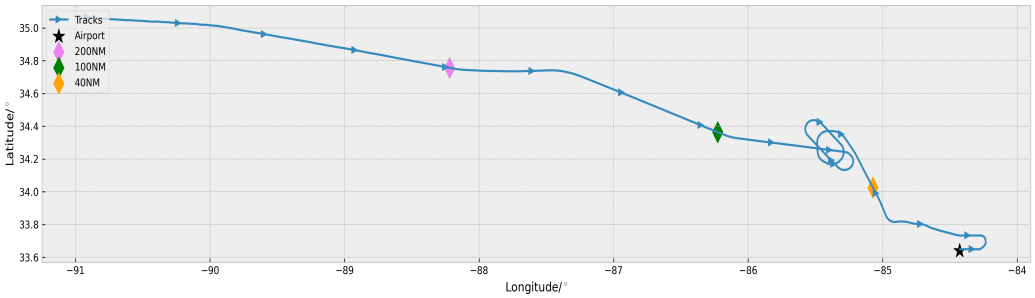
(a) DAL2463



(b) SWA237



(c) SWA1289



(d) DAL2454

Figure 4.7: Landing Aircraft Coming From the Northwest for West Landing.

4.4.2 Aviation Data Mining

The aviation data used in this work are obtained from the SDW, where the flight tracks and flight event recordings are of interest.

Flight Track Recordings

The flight track data takes the standard Integrated File Format (IFF) for aviation standards. The IFF flight track data contains the processed raw flight data collected from FAA facilities across the United States territories, as well as some derived features such as flight summary. I use IFF flight track data from the FAA Atlanta Air Route Traffic Control Center (ARTCC ZTL). ARTCC ZTL covers airspace across Alabama, Georgia, South Carolina, Tennessee, and North Carolina. For a better illustration of ARTCC ATL, Figure 4.1 shows the flight tracks recorded in ARTCC ZTL.

IFF flight track data contains the flight operational features (e.g., flight plans, flight callsign), positional features (e.g., coordinates, speed, course), and flight identifiers/codes (e.g., Beacon code, operations type). As discussed in previous sections, I are interested in the features that can represent the status of the target aircraft, as well as the nearby airspace complexity. I select and construct a proper set of features for the prediction of landing aircraft arrival times, as shown in Table 4.2. These features show an impact on the prediction performances. The aircraft type is obtained from the Sherlock data. I use latitude, longitude, and altitude as the spatial features, each coordinate is associated with a timestamp. I also round the timestamp to full hours with the assumption that the hours of operation will impact the aircraft's landing time. Additionally, I count the number of aircraft ahead or behind the target aircraft as the indicators for airspace complexity measure. The airspace complexity largely impacts the workload of the controller, which further leads to potential flight delays due to ATC.

Table 4.2: IFF Flight Track data: Processed Features for GBM

Feature Group	Feature Name	Descriptions
Flight	acType	Type of the aircraft
Spatial	Latitude	Latitude
	Longitude	Longitude
	Altitude	Altitude
	Distance	Distance to the destination
Temporal	Time	Timestamp
	Hour	Round to the nearest full hour
Speed	GroundSpeed	Aircraft ground speed
Complexity	AC_10mins_ahead	Aircraft count 10 mins ahead
	AC_10mins_behind	Aircraft count 10 mins behind
	AC_30mins_ahead	Aircraft count 30 mins ahead
	AC_30mins_behind	Aircraft count 30 mins behind
	AC_60mins_ahead	Aircraft count 60 mins ahead
	AC_60mins_behind	Aircraft count 60 mins behind

Flight Event Recordings

Flight event recordings are also processed and archived in SDW. IFF flight event data are well-organized tabular format data, instead of time-series coordinates combined with tabular information in flight track data. The timestamp, location, and flight callsign associated with the flight event are stored. Table 4.3 shows the detailed descriptions of fifteen different flight event types recorded in the data. I identify three safety-related flight events. Similarly, I process the feature based on the timestamp that the target landing aircraft reaches the defined TMA, which has the same levels (e.g., 10min, 30min, 60min). I count the num-

ber of flight events that happened ahead or behind the target aircraft for each level. In such a way, I obtain the flight event recordings as the predicted safety indicators for the target aircraft.

Table 4.3: List of Flight Event Types in IFF Flight Event Data. The Three Safety-Related Flight Events are EV_RRT, EV_LOOP, and EV_GOA.

Event Type	Descriptions
EV_TOF	Take off event; defined when the aircraft crosses the departure runway threshold/aircraft wheels come off the pavement.
EV_USER	User event; a flexible event defined by users preference (e.g., how many times an airplane goes into a center, different volume definitions).
EV_MOF	Mode of flight; a measure of aircraft trajectory in the vertical domain (e.g., descending level, climbing level).
EV_INIT	Indicator of flight track beginning recorded by the facility surveillance system.
EV_TRNS	Transition from above/below altitude. Not often used.
EV_XING	Crossing event; defined when aircraft is crossing between different spaces.
EV_RRT	**Reroute event; defined when there is a new flight plan issued. Issuing reroutes significantly increases the workload of the ATC, which poses safety concerns.**
EV_TOC	Top of climb; defined when the aircraft reaches the cruise altitude.
EV_TOD	Top of descend; defined when the aircraft starts to descend from the cruise altitude.
EV_PXCP	Unknown event type.
EV_LND	Landing event; defined when the arrival aircraft passes the arrival runway threshold.
EV_LOOP	**Holding event; defined when the aircraft is circling around in the trajectory. Looping in the TMA increases airspace complexity and leads to safety concerns.**
EV_STOL	Holding end; defined when the LOOP event ended.
EV_STOP	Indicator of stopping flight track recording by the surveillance systems.
EV_GOA	**Go around event; defined when the aircraft aborted landing in final approach or after touchdown. Go around is a major safety concern.**

4.5 Case Study on Scheduling

In this section, I introduce machine learning prediction and flight delay optimization case studies. First, the performance evaluation metrics are briefly discussed. Then, I explain the condition-based machine learning predictor for improved performance, where the processed features are classified based on the number of looping event counts. Last, I show that the proposed machine learning-enhanced TSP-TW solution can achieve a shorter total landing time compared to FCFS, for all of the landing aircraft within a time window.

4.5.1 Performance Evaluation Metrics

Proper performance evaluation metrics are required to select the best parameter setup for the machine learning model. Considering a supervised regression problem, I propose three cost functions to address various statistical behaviors. Define a predicted label y_i and the ground truth $\hat{y}_i$, I have,

Mean Absolute Error (MAE): MAE is the arithmetic average of the absolute errors between predicted labels and ground truth labels. MAE is a commonly used metric in forecasting and prediction objectives. MAE weighs each sample at the same scale.

$$\mathsf{MAE} = \frac{1}{n}\sum_{i=0}^{n}|y_i - \hat{y}_i| \tag{4.21}$$

Root Mean Squared Error (RMSE): RMSE is an alternative to MAE, which share the same drawbacks. RMSE is sensitive to outliers, where a significantly bad prediction aggravates the overall performance measure. This skews the evaluation results towards overestimating the models' badness.

$$\mathsf{RMSE} = \sqrt{\frac{1}{n}\sum_{i=0}^{n}(y_i - \hat{y}_i)^2} \tag{4.22}$$

Root Mean Squared Log Error (RMSLE): In ALS predictions, severe delays can hap-

pen due to various reasons, which are treated as outliers in data-driven prediction. These outliers are unlikely to be captured by predictors and result in overestimating of model's badness. This can be misleading. I propose to use RMSLE, as shown in Equation (4.23). RMSLE is viewed as the RMSE of log-transformed prediction and log-transformed ground truth. RMSLE is preferred as I need to avoid over-penalizing severe delay scenarios, which helps select the best model parameters.

$$\mathsf{RMSLE} = \sqrt{\frac{1}{n}\sum_{i=0}^{n}[\log(y_i+1) - \log(\hat{y}_i+1)]^2} \tag{4.23}$$

4.5.2 Prediction Analysis

The model development/training phase has two objectives, predicting the ETA distributions with GBM and formulating the predicted values into optimization for the demonstration of case studies. The first part requires model-tuning efforts. I tackle this from three aspects,

Table 4.4: GBM Parameters and Search Spaces for Grid Search

Parameters	Definition	Search Space
learn_rate	The learning rate for the optimizer.	[0.05, 0.1]
max_depth	The maximum tree depth.	[7, 8, 9, 10, 11, 12]
sample_rate	The sample without replacement rate along the feature dimension.	[0.8, 1.0]
col_sample_rate	The sample without replacement rate along the sample dimension.	[0.5, 0.6, 0.7, 0.8]

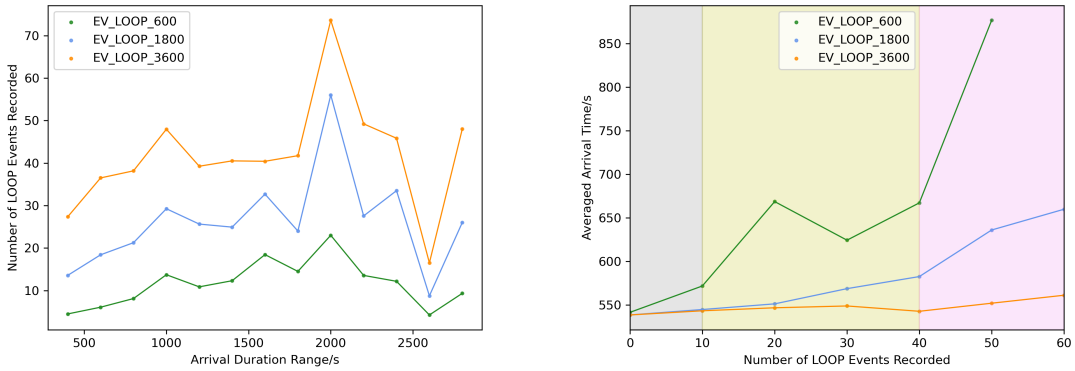
Grid Search Grid search is common practice to fine-tune parameterized machine learning predictors, to find the best combination of modifiable parameters. For the GBM used in

this work, I am especially interested in the following parameters, a) the learning rate controlling the step size of optimization (efficiency); b) the maximum tree depth to control the order of approximations (accuracy); c) the data sampling rate along the feature and sample dimension (stochasticity). More discussion of b) and c) is in Section 4.3.1. I list the search space of this study in Table 4.4. Refinement of search spaces can certainly improve the predictor performance and is beyond the scope of this study.

Domain Knowledge Domain expertise and human intelligence can benefit data-driven models. In Section 4.4.1, I have discovered the impact of holding patterns on flight delays. The holding pattern is recorded as a looping event in the IFF flight event recordings. Airspace complexity is represented by the number of nearby aircraft of the target landing aircraft. As discussed in Section 4.4.2 and Section 4.4.2, I implicitly include the airspace complexity measurements and flight event number that happened within the certain time range of the target landing aircraft.

Divide-and-Conquer The renowned strategy *Divide-and-Conquer*, or *Divide-and-Rule* stands for gaining and maintaining supremacy divisively. I propose conditional GBM, which pre-filters the data samples based on the number of looping event counts, to gain exceptional prediction capability. Figure 4.8 shows the statistical analysis between the number of flight events and the arrival aircraft landing time within the $[100, 40]$NM range. Figure 4.8(a) shows a non-monotonic trend, where an obvious drop of looping events is presented when the arrival time is greater than $2,500$ seconds. However, from Figure 4.8(b), I notice there are approximately three stages along with the increase of looping event counts. I separate the entire dataset into three parts and trained with separate GBM models. I discuss each of the separations here,

- Stage I (EV_LOOP ≤ 10): In this stage, minimum flight event conditions exist, where



(a) Looping Event Counts v.s. Time Spent from 100NM to 40NM

(b) Averaged Time Spent from 100NM to 40NM v.s. Looping Event Counts

Figure 4.8: Data Analysis to Identify the Reasonable Conditions for Conditioned Prediction. The Legend Stands For The Number Of Looping Events $600/1, 800/3, 600$ Seconds Ahead/Behind The Current Timestamp.

the arrival time duration is near optimal. At this stage, the arrival time increase is not significant.

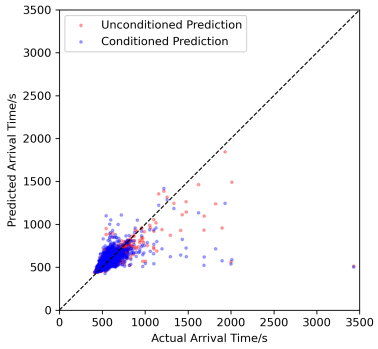
- Stage II ($10 < \text{EV\_LOOP} \leq 40$): The steady growth stage. At this stage, the arrival time duration is steadily growing with the increase of looping event counts.
- Stage III ($\text{EV\_LOOP} > 40$): The rapidly increasing stage. The arrival time sharply increases when the number of looping events increases.

I collect and process the flight track and flight event data for August 2019 at ARTCC ZTL. A total of $28,181$ well-structured arrival flight information are obtained, with the feature set described in previous sections. Then, the data is filtered for three stages based on the EV_LOOP_600 feature. For each stage, I further separate the data into training, validations, and testing set. The training and validation sets are used for GBM training and fine-tuning, and the testing set is used for prediction case studies.

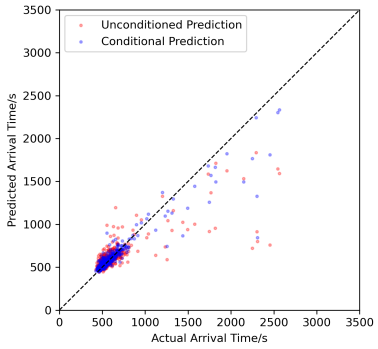
In Table 4.5, I evaluate the performance of the three-stage model using the testing dataset and compare it with the unconditional method without considering different growth behaviors. Including the flight event-related features boost the model performance by a large margin, while the conditioned predictor further refines the results. Figure 4.9 shows the evaluation results in comparison for unconditioned predictions and conditioned predictions. Figure 4.10 shows the corresponding variable (feature) importance for conditioned predictors. For stage I, the conditioned prediction doesn't significantly improve the prediction accuracy. At this stage, the number of looping events is not a critical variable for the GBM. The ground speed is prominently dominant in the stage I predictions. However, for stage II and stage III, the conditioned predictor greatly enhanced the performance. At stage II, the type of aircraft shows nearly the same variable importance as the ground speed. Notably, at stage III, the ground speed turns into a less critical factor contributing to the overall model performance. The airspace complexity factors, geological information, and flight safety-related events are essential.

Table 4.5: GBM Evaluation Results on the Testing Dataset

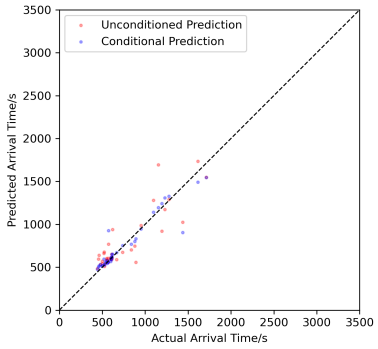
Model	RMSE	MAE	RMSLE
Predictor w/o Flight Events	118.147	42.139	1.136
Predictor w/ Flight Events	93.784	38.085	1.011
Conditioned Predictor w/ Flight Events	90.944	36.259	0.098



(a) EV_LOOP $\leq$ 10

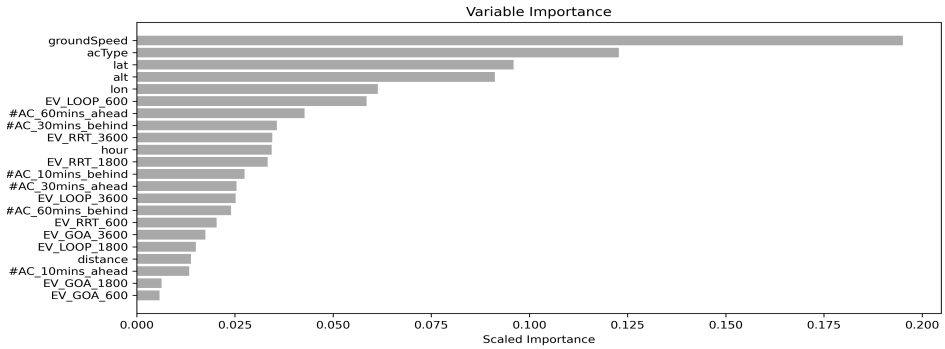


(b) 10 $<$ EV_LOOP $\leq$ 40

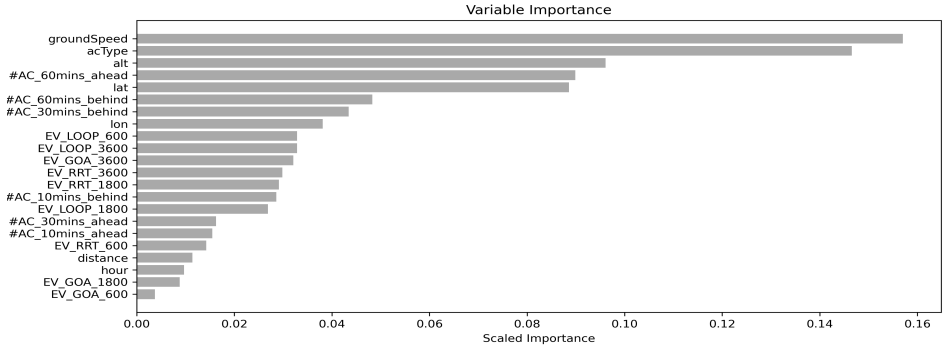


(c) 40 $<$ EV_LOOP

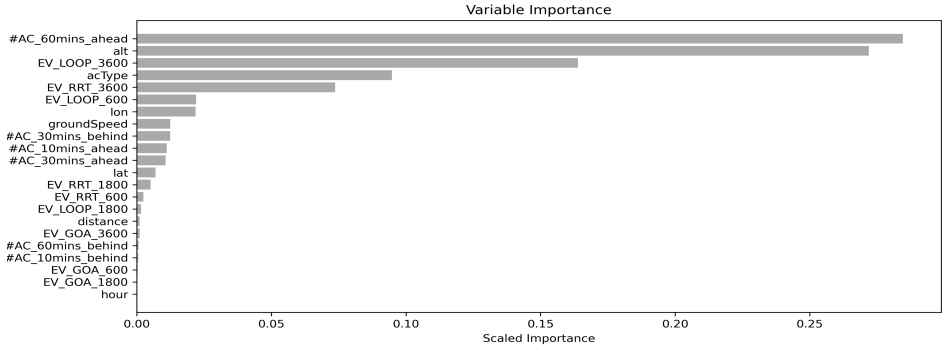
Figure 4.9: Splitting the Testing Set: Unconditioned Prediction v.s. Conditioned Prediction



(a) Variable Importance: EV_LOOP ≤ 10



(b) Variable Importance: $10 <$ EV_LOOP ≤ 40



(c) Variable Importance: $40 <$ EV_LOOP

Figure 4.10: Variable Importance for Three Divisive Predictors. The Importance of the Speed Profile Keeps Decreasing with the Increase in the Number of Looping Events. In (C), the Dominant Variables Are Airspace Complexity, and Safety-related Event Counts.

4.5.3 ALS Case Studies

In this section, I discuss real-world demonstration case studies. At first, I define the problem horizon by visualizing and analyzing the real-world data. Figure 4.11 shows the scatter plot of time spent for approaching flights around KATL TML on Aug 1st, 2019. The increased scatter density indicates the business of aviation operations at a certain timestamp. The Y-axis denotes the flight time from 100NM to 40NM away from the terminal TMA. It's clear that the normal time spent should be lower than $1,000$s, which are also cross-referenced by Figure 4.2(a) and Figure 4.4(b). Based on these, I conclude that there are three severely delayed time ranges, starting from 13:00, 16:00, and 20:00, approximately. In this demonstration, I present two case studies. In the first scenario, I take 9 successive landing flights from 13:20 to 13:45 and show the trajectories in Figure 4.12(a). These landing flights mostly follow two groups of landing procedures. The first group of 6 flights came from the west side of KATL for a west landing on KATL runway 26R, while another group of landing aircraft came from the northeast for the same west landing at 26R. Similarly, in the second case study, I set the time duration starting from 16:00 to 16:30. Case II has three groups of landing procedures, where flight EDV3441 and DAL2794 maneuvered for west landing ahead of time. The selection of time windows is based on the availability of aircraft landing on the same runway, and the scalability of the TSP-TW solver.

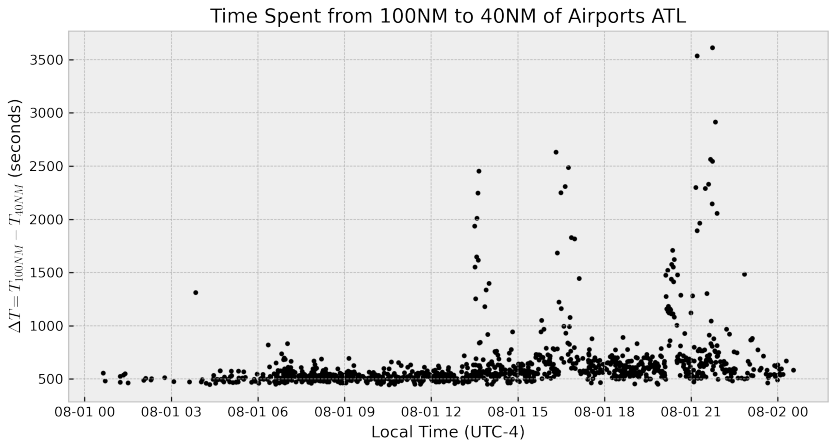
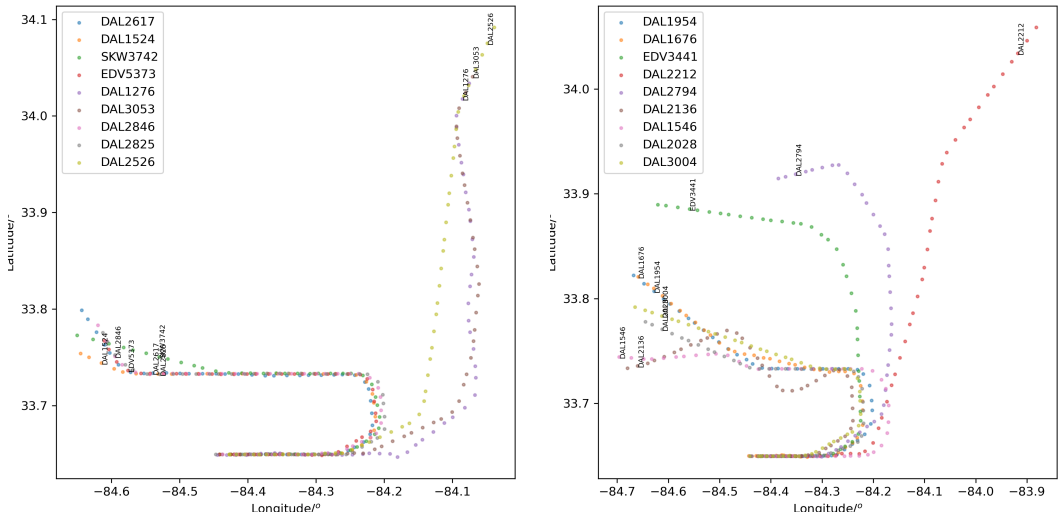


Figure 4.11: Scatter Plot of Flight Times Around KATL TMA on Aug 1st, 2019. This Figure Shows That There Are Three Severe Delay Periods, Starting from Around 13:20, 16:00, 20:00, Respectively.



(a) Case Study I: Landing Trajectories on KATL Runway 26R

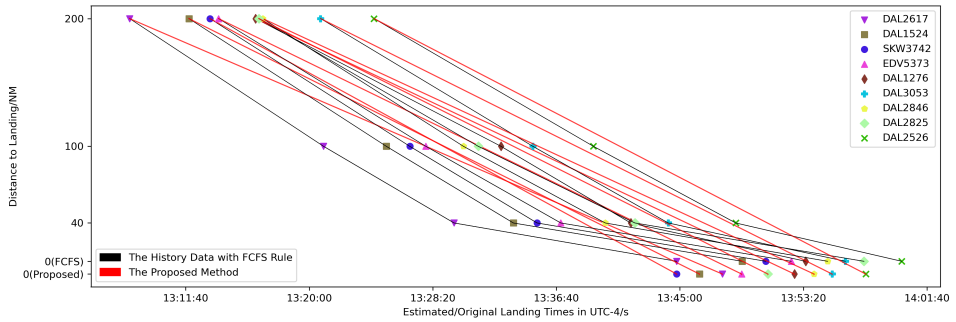
(b) Case Study II: Landing Trajectories on KATL Runway 26R

Figure 4.12: Visualization of Landing Trajectories for Two Case Studies on KATL Runway 26R: a) 13:20 to 13:45, Aug 1st, 2019; b) 16:00 - 16:30, Aug 1st, 2019

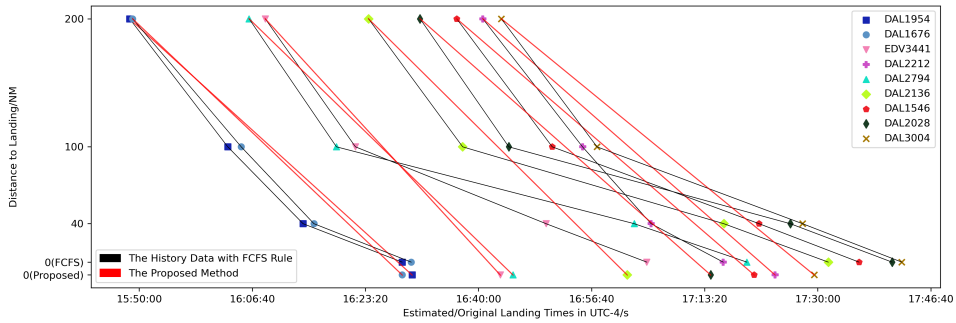
From GBM, I obtain the distributions for successive landing aircraft pair $t_i \sim \mathcal{N}(\mu_i, \sigma_i)$, $t_j \sim \mathcal{N}(\mu_j, \sigma_j)$. To determine the $\mathcal{T}_{ij}$ for various aircraft types, I refer to FAA Order 7360.1 [2]. Following Equation (4.19), I obtain the distribution of t_{ij}. Given the separation violation probability, I can estimate the probability intervals with numerical tools. In this way, I obtain $\check{t_{ij}}$, u_i, and l_i from the learning of historical data. Then, I incorporate the learned parameters into the formulation of TSP-TW and solve with Python optimization solvers [222]. It's worth pointing out that, the aircraft involved in our case studies are classified as medium size aircraft. Based on Table 4.1, I choose the *Large-Large* minimum separation threshold, 64 seconds, for our case studies. I set the number of landing aircraft to 9 for both cases due to the computational complexity, which corresponds to at least $\sim$ 10min optimization horizon in extreme scenarios. The exploration of other efficient solvers for MILP is beyond the focus of this study.

Table 4.6: The Detailed ALS Results for Two Case Studies. The Determination of the Case Study Windows Has Been Discussed. Depending on FAA-Order 7360.1 [2], All Of The Aircraft Involved In These Case Studies Belong To The Medium Weight Class. Referring to Table 4.1, I Use The Large-Large $\mathcal{T}_{ij}$ to Be Fixed at 64. All Of the Unix Timestamps Contained In The Data Have Been Transformed To The Local Time Zone At ARTCC ZTL (UTC-4).

Case Study I: 13:20-13:45							Case Study II: 16:00 - 16:30					
Entering Sequence	Callsign	AcType	$\mathcal{T}_{ij}$/s	Landing Sequence	Landing Time/s	σ_{ij}/s	Callsign	AcType	$\mathcal{T}_{ij}$/s	Landing Sequence	Landing Time/s	σ_{ij}/s
1	DAL2617	B712	64	3	13:47:52	4.45	DAL1954	A321	64	2	16:30:12	18.11
2	DAL1524	B752	64	6	13:52:44	21.91	DAL1676	MD88	64	1	16:28:45	1.13
3	SKW3742	CRJ2	64	1	13:44:47	2.95	EDV3441	CRJ9	64	3	16:43:12	16.11
4	EDV5373	CRJ7	64	2	13:46:19	10.09	DAL2212	B712	64	8	17:23:42	26.33
5	DAL1276	MD88	64	8	13:55:16	11.63	DAL2794	B739	64	4	16:45:04	34.25
6	DAL3053	MD88	64	7	13:54:03	6.44	DAL2136	MD90	64	5	17:01:55	3.16
7	DAL2846	MD88	64	4	13:49:10	3.25	DAL1546	A321	64	7	17:20:37	90.57
8	DAL2825	B738	64	5	13:50:57	31.54	DAL2028	A321	64	6	17:14:13	143.43
9	DAL2526	MD88	64	9	13:57:32	27.21	DAL3004	B739	64	9	17:29:28	146.37



(a) Case Study I: Proposed Method v.s. FCFS



(b) Case Study II: Proposed Method v.s. FCFS

Figure 4.13: Experiment Results for Two Case Studies with Landing Flight: a) 13:20 to 13:45, Aug 1st, 2019; b) 16:00 - 16:30, Aug 1st, 2019. Looking At the Landing Time of The Last Aircraft In This Time Window, It's Obvious The Proposed Method Takes a Shorter Time Than The Actual Landing Time Recorded in the Data, At Which ATC Applies The FCFS Rules.

I list the detailed landing scheduling results in Table 4.6 and Figure 4.13. Table 4.6 gives the original sequence of landing aircraft entering the 100 NM TMA, and their predicted corresponding landing sequence, and estimated landing time from the proposed method. The MSTs between two successive landing aircraft comply with FAA regulations while incorporating additional uncertainties learned from aviation history recordings. Figure 4.13 compares the total landing time between the FCFS rule (history recordings) and the pro-

posed methods. As mentioned in Section 5.4, the straightforward objective is to achieve a shorter total landing time for all the landing aircraft heading the same runway. In Figure 4.13, I color the optimized landing time in red, and the total landing time for all the aircraft is the landing time difference between the first landed aircraft and the last landed aircraft (i.e., the time duration between 13:44:47 for SKW3742 and 13:57:32 for DAL2526 is 12 minutes and 45 seconds).

4.6 Conclusions

In this work, I propose a novel machine learning-enhanced aircraft landing scheduling algorithm, which provides a new conceptual design to avoid significant delays with safety constraints. First, the aircraft landing scheduling algorithm is formulated into a time-constrained traveling salesman problem. Being machine learning-enhanced, I incorporate machine learning-predicted results into several safety-related constraints of the time-constrained traveling salesman problem formulation. Regarding the machine learning prediction algorithm, I propose explicitly introducing nearby flight event situations and airspace complexity measurements into the conditional data-driven learner, which greatly enhances the prediction accuracy. The variable importance analysis suggests that aircraft type, ground speed, distance to destination, and airspace density are key factors affecting arrival time prediction accuracy. Finally, I evaluate and compare the performance of the proposed method through real-world case studies during peak hours at ARTCC ZTL. Various uncertainties from aircraft, speed, and airspace density are included. The key concept is to optimize the scheduling using enhanced operational predictability combining advanced instruments (e.g., ADS-B) and data analysis (e.g., arrival time prediction model in this study).

Insights A few insights are discussed based on the current investigation, and a few potential research directions are suggested.

- The scalability of this work can be improved. In extreme cases, the current optimization horizon corresponds to a planning horizon of $\sim$ 10mins. Dynamic scheduling with rolling window horizons can be integrated with the current method to extend the proposed method for a longer planning horizon. The performance of this extension can also be evaluated.

- The proposed study focuses on the methodology demonstration and only uses limited data at one center. A significant amount of data collection and validation at multiple airports is required.

- Weather is an important factor affecting arrival time prediction. The proposed model did not include the weather feature explicitly. The complexity measure indirectly reflects some weather conditions. Additional theoretical work and validation are required for explicit weather feature inclusion.

- Another important research direction is multi-runway aircraft landing scheduling, which can be especially important for ATC of major international airports. The multi-runway scheduling problem is more challenging, and significant further study is needed. Both hierarchical and concurrent optimization can be used based on our beliefs. Performance evaluation and scalability need to be balanced for decision support.

Chapter 5

AIR TRAFFIC CONTROLLER WORKLOAD PREDICTION USING CONFORMALIZED DYNAMICAL GRAPH LEARNING

5.1 Introduction

The rapid advancement of intelligent vehicles substantially reduces the operational effort from the individual user level but escalates the system-level complexity of real-time decision-making and corporate planning due to the dynamically changing environments, time restrictions, and tactical constraints [223; 224; 225; 154]. The exponential growth of computation power enables swift individual decision-making and can lead to serious task overhead on the system level, especially during certain unexpected events (i.e., traffic flow restrictions, communication errors, and severe weather). In complex semi-automated systems (i.e., vehicle piloting [226; 227; 228; 229; 230; 231]). Prediction and assessment of the cognitive workload of operating such complex systems have long been regarded as critical research objects [232; 233; 234; 235; 236]. The cognitive workload is defined as the mental resources required to accomplish task demands placed on the human operator. Workload overhead can occur when the demands exceed the human operator's capacity and can lead to efficiency drop and operational safety concerns. Various factors such as level of automation, transparency of automation, and task complexity have been identified as factors that can influence cognitive workload [237; 238; 239].

Air traffic control (ATC) is a crucial part of aviation safety, ensuring that aircraft are safely guided through the airspace and landed or taken off from airports. Air traffic controllers (ATCos) are responsible for managing the flow of aircraft, communicating with pilots, and making critical decisions in real-time to ensure the safety of all involved. As

air traffic continues to increase [100], it puts more pressure on ATCos, who already have highly demanding and stressful daily routines [240; 149; 241]. Quantifying the effort made to meet these task requirements lead to the concept of cognitive workload as an air traffic controller, which denotes the subjective qualitative measure of perception demand placed by the current air traffic situation [242; 154; 234]. Moreover, proper workload management and scheduling are vital to ensure ATCos can perform their duties effectively and faultlessly without being overwhelmed. Human performance is a crucial factor in ensuring the safe operations of the National Airspace System (NAS). In the past, human operators have been identified as significant contributors to accidents involving air carrier operations governed by the 14 Code of Federal Regulations (CFR) Part 121, which covers commercial airliners frequently used by the public [243]. For instance, around 80% of the 446 air carrier accidents that occurred between 1997 and 2006 were attributed to personnel-related factors, while environmental factors were cited in approximately 40% of the accidents and aircraft-related factors in 20% [244]. Workload prediction can help in several ways, such as ensuring that enough ATCos are available to manage the traffic, preventing fatigue and burnout, optimizing shift schedules, and improving overall efficiency. By accurately predicting the workload, air traffic organizations can ensure that they have the necessary resources and personnel to maintain a safe and efficient air traffic system [245; 246; 247]. All of these objectives are based on reliable ATCo cognitive workload level modeling and predictions. Moreover, artificial intelligence (AI)-enabled aviation human factor studies have been identified as one of the core elements of AI taxonomy by related authorities [5; 6], where ATCo workload management is a key dedicated objective.

A tremendous amount of research has been done to understand the impact factors and demand patterns that drive the workload of a controller, such that a better workload prediction performance can be discovered. Two types of factors are studied extensively in the literature, (a) subjective features including ATCo mental stress, fatigue level, communica-

tion difficulties, and situation awareness [248; 249; 250; 251]; (b) objective factors such as traffic and airspace complexity measures (i.e., operational errors (OE)), abnormal events, level of automation, and weather situations [13; 252; 253; 254; 255; 256; 257]. In order to collect objective features for workload prediction, researchers have proposed to collect human-subject data (i.e., eye movement, communications, heartbeat rates, and Electroencephalography (EEG) signals) [258; 259; 260], in an intrusive and non-intrusive sense. On the other hand, objective features can be directly obtained from computer flight recordings and operational recordings. However, some specific objective features need *post-hoc* processing (i.e., loss of separations (LoS), OEs). Specifically, in this work, I am interested in objective features since it is very unlikely to collect real-time biological features (i.e., EEG/ECG signals or heartbeat rates) in the near future due to privacy concerns and regulatory requirements. Moreover, I discover that the existing model on workload prediction is mostly using handcrafted features, even if a graph data structure and simple neural networks (i.e., minimum spanning trees) have been proposed [252]. To the best of the authors' knowledge, there is no investigation on utilizing advanced data-driven learning techniques (i.e., graph neural networks (GNNs)) for workload prediction possibilities that directly leverage the spatiotemporal relationships contained in the traffic data and airspace layout.

In this work, I investigate the possibility of using the graph neural network to predict ATCo cognitive workload levels, with an additional post-processing technique, namely conformal prediction, to boost the accuracy with a set of prediction labels. The data is collected by conducting experiments with retired ATC participants who have experience at FAA Radar Approach Control (TRACON) facilities, under three different scenarios, (a) baseline conditions; (b) high workload nominal conditions; (c) high workload off-nominal conditions. The major difference scenarios are the peak traffic densities and the presence of off-nominal events (i.e., runway switch, communication errors, etc). The simulation sce-

nario is limited to a few Phoenix approach procedures for a duration of 25 minutes for each scenario. A detailed description of the experimental setup is in Section 5.3. Specifically, the ATCo workload I investigated is the executive (R-side) controllers' workload [245]. Predicting controller workload levels can be viewed as a pattern recognition problem [253] and thus is suitable for data-driven learning algorithms. In this work, the problem of predicting workload based on the spatiotemporal layout of airspace is viewed as a time-series dynamically evolving graph classification task. Being time-series classification, I propose to input multiple historical timestamp graphs into the model for the prediction of workload level at the next timestamp. Also, the spatiotemporal layout of the graph structure varies at each timestamp (i.e., number of nodes, graph edge connections), resulting in a dynamical graph classification problem.

Our contributions are summarized as,

- This chapter investigates the possibility of predicting executive controller workload during approach scenarios directly from the recorded air traffic data with graph neural networks and discovers that traffic conflict is a nontrivial contributor to improving workload prediction capabilities.

- I propose to formulate the ATCo workload prediction task into a dynamical time-series graph classification problem and show that the Evolving Graph Convolutional Network (EvolveGCN) can achieve a higher prediction accuracy than both statistical (i.e., regression, handcrafted features) and classical learning methods (i.e., MLP, GCN). I show that graph neural networks have great potential for predicting controller workload with varying spatiotemporal airspace layouts.

- A moving window approach is proposed to build the correct input-output matching from the collected sparse workload data. The moving window size represents the temporal length of the historical information used in workload prediction. The se-

lection of parameters can be alternated to fit into the operational need. The data structure formulation transfer complex structured traffic features into a lucid format for research and development purposes.

- To further improve the classification accuracy of the experimental data. I explore conformal prediction to expand the prediction as set predictions. I show that conformal prediction has better ground truth label coverage by giving multiple possible predictions as indicators of model uncertainties. I suggest that conformal prediction is a valuable machine learning *post-hoc* processing tool to boost performance further as well as indicate prediction uncertainties.

The rest of the paper is organized as follows. First, Section 5.2 reviews related studies on air traffic controller workload prediction. I first introduce the impact factors of ATCo workload in Section 5.2.1, then list the current practices in predicting workload Section 5.2.2. In Section 5.3, I introduce the detailed workflow of human-in-the-loop simulations to collect the traffic data and ground truth ATC workload labels, along with data analysis of the collected data. Section 5.4 describes the flowchart of the proposed machine learning framework, from experiment data handling to innovative modeling. The prediction performance and evaluation of the conformal prediction set are discussed in Section 5.5. Section 5.6 concludes this chapter by giving limitations of this study and provides future insights.

5.2 Related Works

Due to the surging number of daily aviation operations, the aviation industry is in urgent need of advanced decision support tools that can accommodate the rapid annual air traffic growth. Numerous studies have investigated into ATCo workload. It's an aviation researchers' consensus that understanding the impact factors that drive mental workload

can help improve airspace capacity, thus reducing aviation safety concerns [261; 154; 262]. With meaningful impact factors collected or modeled, predictive modeling is critical to building an accurate workload prediction algorithm. In this section, I discuss the related works from two aspects, (1) understand the impact factors that drive the mental workload in Section 5.2.1; (2) discuss the current practice in predicting ATCo workload from open literature with a focus on predicting workload from traffic factors Section 5.2.2.

5.2.1 Task Demands and Impact Factors to ATCo Workload

In air traffic control, task demand refers to the level of mental and physical effort required for ATCos to complete their duties effectively. These tasks are classified as either primary tasks, which are the core duties required to ensure the safe and efficient flow of air traffic, or secondary tasks, which are supportive or administrative tasks that support the primary duties. Primary tasks for ATCos include monitoring the position and movement of aircraft, communicating with pilots, issuing clearances and instructions, and managing potential conflicts between aircraft. Secondary tasks include administrative duties such as filling out paperwork, conducting briefings, and updating flight progress information. The level of task demand for air traffic controllers varies depending on several factors, including *air traffic factors, airspace complexity measurements, operational constraints, and ATCo cognitive states during working hours*. High task demand leads to increased workload, which negatively impacts aviation safety. Therefore, understanding the level of task demand and appropriately managing workload is essential for ensuring that ATCos can perform their duties effectively and safely. Correctly modeling task demand is viewed as the prerequisite for workload prediction for a long history. It's noteworthy to mention that ATCos workload is not a simple function of task demands; the ATC strategy the controller adapted to meet the increased task demands also provided a feedback loop to ATCo workload [154]. I discuss each of these aforementioned grouped impact factors separately.

Air traffic factors refers to both the aircraft count under the ATCo control and their spatiotemporal relationships. The number of aircraft under control is viewed as the most important factor that drives ATCo's mental workload [263; 264]. A high aircraft count leads to higher communication frequency and a higher possibility of safety events, resulting in a higher mental and physical workload. Traffic density is typically measured in aircraft per unit of airspace, such as aircraft in unit time and unit airspace sector area. Measurements of traffic density have been developed based on the averaged vertical/horizontal separation distances [253], as they directly infer loss of separations. Another research investigates the necessity to consider flight interactions and flight characteristics, which includes the changes/variability in heading, speed, or altitude, the pattern of how air traffic flows merges and separates into a set of air traffic complexity metrics [265; 253; 264; 266; 234]. Their regression analysis shows subjective workload depends on both aircraft count and other air traffic complexity measures. Additionally, some other studies also suggest that a lower aircraft count also can lead to task overload if these aircraft are interacting in a complex fashion [267; 268].

Airspace complexity factors include the number of routes, altitudes, and restrictions, which can also impact the workload of ATCos, as they need to monitor and manage multiple variables simultaneously. Airspace-related factors are another key contributor to ATCo mental workload [269]. Larger airspace size indicates a higher aircraft count and higher metrics on traffic complexity, while small airspace size reduces conflict resolution options and higher traffic evolving rates. It is noteworthy to mention another work considering both the traffic factors and airspace structure complexity and proposes Structural Complexity Metric (SCM), which incorporates a measure of the organization, hierarchy, and interdependence into the complexity calculation [270]. Furthermore, this chapter suggests using well-defined ingress and egress points in the airspace to distinguish normal and abnormal flights based on real-time monitoring.

Operational Constraints are another major contributor that drives the ATCo workload. Operational constraints refer to the temporal variability within the operational conditions of the airspace, as well as the conditions of related technology and equipment. Several factors are viewed as operational constraints; (1) pilot-controller communications are critical for maintaining safe aviation operations. Malfunctions of communication devices can disrupt air traffic control operations. This is known as loss of ratio communication (NORDO) (2) convective weather conditions, such as thunderstorms or heavy fog. These types of objective factors can affect air traffic control operations by reducing visibility and creating unsafe flying conditions. (3) subjective airspace restriction is another type of operation constraint. The restrictions come from multiple sources, i.e., aircraft holding, no-fly zones, or special-use airspace [154]. In addition, certain other off-nominal events are considered operational constraints, i.e, runway switch, and minimum fuel reported [271; 272].

Cognitive states directly contribute to the cognitive task demands of ATCo. To measure cognitive states, the researchers propose to measure the psychological states of the air traffic controller, including brain activities, eye movements, and heartbeat rates. These states can be quantitatively measured by sensors signals such as electrocardiography (ECG) signals, electroencephalography (EEG) signals, galvanic skin response (GSR), blood pressure (BP), and certain biochemical analysis [273; 274; 275]. However, using intrusive psychological state measurements is disruptive to controllers' normal working conditions, as it creates additional mental stress and discomfort in maintaining ATC operations. Alternatively, computer vision (CV) based non-intrusively psychological state measurements are proposed to collect distractions, drowsiness, head poses, eye movements, and fatigue levels [276; 277; 247]. However, these types of measurements can lead to information security and privacy concerns.

5.2.2 *Workload Prediction Algorithms*

The dynamic density model builds a regression model to find the linear relationships between traffic complexity factors and ATCo workload. The Dynamic Density metric uses a combination of traffic density and complexity measures to estimate workload in real-time, with the goal of providing a more accurate and responsive measure of controller workload [278]. However, dynamic density metrics fail to consider the human cognitive workload, which is the primary source of real ATCo cognitive workload. In [264], the traffic complexity, as well as the airspace complexity of different sections, are considered. The results show that the airspace factor can actually contribute to workload prediction in a multi-sector study. Similarly, in [249], the authors find that the ATCo workload is proportional to the number of aircraft controlled by the enroute sector. They conduct HITL experiments and found a linear relationship between aircraft count and workload ratings.

However, it's still difficult to identify the most contributing impact factors to workload prediction from regression analysis. The first reason is the multi-collinearity within these factors. For instance, the number of conflicts depends on the speed, altitude, and heading variabilities. The complexity of traffic situations, such as traffic density and potential conflicts, also mediate the causal connection between traffic count and workload. The inter-relationship of these factors makes it challenging to determine the relative importance of each predictor in a regression equation. The second query is the debate on the linear relationship between these factors and workload ratings. For instance, the ATCo can alternate control strategies during a certain period to meet the increased task demands. Additionally, the online processing or post-processing of these factors only considers the current situations, and there is no inference on ATCo's intent and air traffic intent information. Trajectory prediction from flight plans helps with estimating the workload of ATCos, where prediction and reduction of trajectory uncertainties can help alleviate ATCo work-

load [15; 256].

Machine learning algorithms have been adopted for workload prediction. In [279], tree-based models, and support vector machines are included for workload measurements. The authors consider both traffic complexity and operational constraints as features and show a high F1 score of over 0.9. However, these types of works are not dynamically considering the *traffic pattern* of the airspace, but still formulate the data as a tabular format, which fails to address the aforementioned concerns [154]. On the other hand, [236] proposes to use a 3-layer simple neural network to forecast ATCo workload. However, the ATCo workload in this work is assessed from voice communication data, which fails to model the task demand factors mentioned earlier. Impressively, direct prediction from the spatiotemporal air traffic layout is actually proposed decades ago. In [13; 253], the authors propose to model air traffic at each timestamp into graph-structured data and calculate the second-order statistics of time-series of graphs as extracted air traffic complexity measurements. Then a simple neural network is adopted to do classification from these features.

In summary, understanding the factors that drive mental workload has been a challenging yet unresolved open question for decades. Instead of investigating the linear relationship between impact factors and workload, one should look into the dynamic properties of these factors and workload. This leads to our study on workload prediction – I model the spatiotemporal airspace layout into time-series graph structures and propose to use a time-series learning algorithm to predict workload levels, in consideration of historical dynamic variabilities contained within the air traffic data.

5.3 Human-In-The-Loop (HITL) Simulations

In this section, I provide an overview and the detailed simulation setup of the Human-In-The-Loop (HITL) experiment as in Figure 5.1. The scope and objectives will be discussed. Then, data analysis on the collected data is presented in Section 5.3.3.

5.3.1 Simulation Overview

The HITL simulation is the first human factor study of our aviation big data project, which aims at addressing the safety needs and technology solutions for future NAS [280]. The backbone of the project lies in information fusion and uncertainty management for real-time system-wide safety assurance, where human factors like cognitive workload play a key role. As mentioned, accurately predicting the ATCos workload can improve operational efficiency and reduce safety concerns such that aviation authorities can ensure a reasonable resource allocation and workload management.

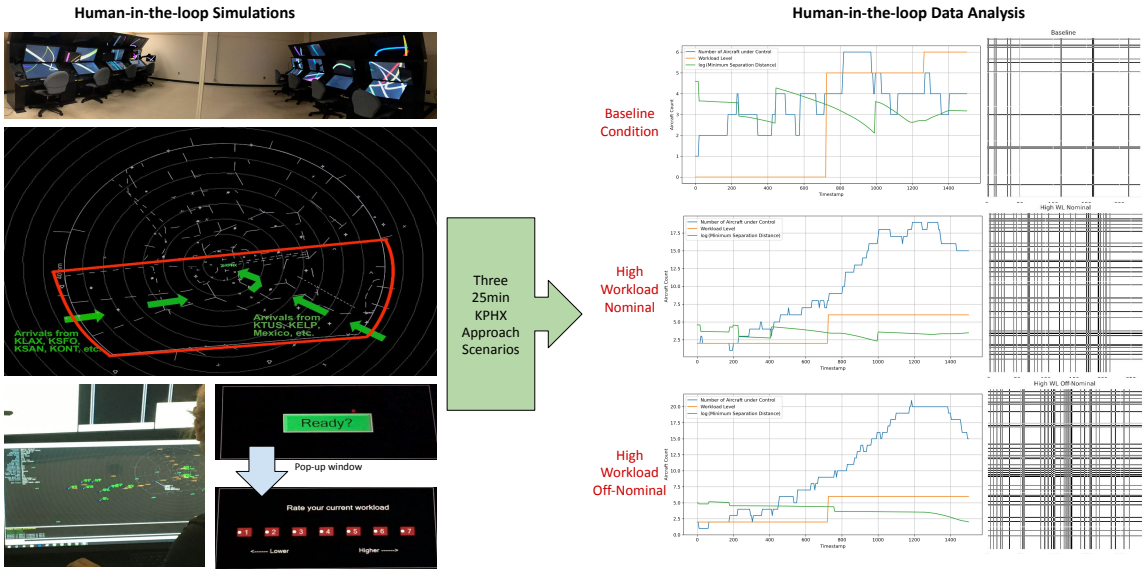


Figure 5.1: Overview of the Human-In-The-Loop (HITL) Experiments. The Left Part Shows the Atc Simulation Platforms Equipped with a Simple Demonstration of the Graphical User Interface. The Primary Focus of the Simulation Is on the Kphx Arrivals from Two Directions of Three Pseudo Pilots, with Flight Procedures Including 1 RNAV (HYDRR1) and 3 STARs (ARLIN4, BLYTHE5, SUBSS8). THe Pop-up Window Shows Either a Workload Question or a Situational Question Every 3 Minutes. The Experiment Has Three Working Scenarios, with Time Duration of 25 Minutes Each. After Label Interpolation, the Collected Data Will Show the Workload Ratings and Corresponding Aircraft Densities at Each Timestamp. I Also Obtain the Recurrence Plot (Rp) from Communication Transcripts as Indicators of System Tendency In Figure 5.2

Table 5.1: A Short View of the Communication Transcripts Post-processed from Radio Recordings in HITL Simulation. Three Cut-off Sections Are Listed Here, Which Correspond to Off-nominal Events, (1) Turbulence Reported; (2) No Radio Communications; (3) Landing Runway Switch. Indicator of Communication Deviations Is Also Shown in the Right-most Column.

Speaker	Start Time	Transcriptions	End Time	Deviation Indicators
... ...				
Speed bird 281	05:26.9	apporach speed bird two eighty-one I am experiencing motor turbulence at one three thousand	05:30.5	1
PHX approach (Speed bird 281)	05:34.9	speed bird two eighty-one heavy, roger	05:37.7	1
... ...				
PHX approach (Cumpacity 250)	11:32.8	cumpacity two fifty descending maintain four thousand one hundred	11:35.4	0
PHX approach (Cumpacity 250)	11:38.8	cumpacity two fifty four thousand one hundred	11:40.5	1
Cumpacity 250	11:41.8	cumpacity two fifty dropping down to four one hundred	11:44.6	1
... ...				
Local south	14:29.1	court this is local south I am switching to runway seven left and right effect immediately	14:35.9	1
PHX approach (Local south)	14:38.8	Ok. . . move to runway seven	14:40.7	1
PHX approach (Ascer 4527)	14:49.8	shuttle forty-five twenty-seven expect dail is runway seven left turn left in two seven zero maintain five thousand	14:55.1	1
... ...				

The primary objective of the HITL experiment is to investigate the correlation between communication patterns (such as content, volume, and flow patterns) and both controller workload and human performance. Figure 5.1 gives the overview of the HITL simulation

process. The simulation contains two arrival flows including four Phoenix in-bound Procedures. The first arrival flow represents flights from the west coast (KLAX, KSFO, KSAN, KONT, etc), with procedures HYDRR1, ARLIN4, and BLYTHE5. The second flow stands for arrival flights from the southeast, including KTUS, KELP, and Mexico. SUNSS8 arrival procedure is used here. The left panel of Figure 5.1 displays the layout of the simulation environment. Each retired ATCo participant study employs a study within-subjects design and measures performance across three simulation trials. Each trial spans 25 minutes and varies in workload level by manipulating two variables, namely traffic density, and occurrence of off-nominal events.

Baseline: Baseline trials contain up to 6 aircraft in the airspace at a given timestamp. There are no off-nominal events in baseline trials. Typically, a moderate workload is expected.

High Workload Nominal: High workload nominal trials can have up to 20 aircraft showing up in the current simulation environment. Again, there are no off-nominal events.

High Workload Off-Nominal: In addition to the experimental setup in high workload nominal trials, high workload off-nominal trials incorporate four off-nominal events during the 25 min duration. I list the name of these off-nominal events here,

- Turbulence: Moderate turbulence is simulated in several arrival flows. In Table 5.1, speed bird 281 reported experiencing turbulence at a certain altitude, starting from 05:26.9 of simulation time.

- No radio (NORDO): The pilot has no radio communication with the approach ATCo, which can happen during a radio failure. In Table 5.1, the KPHX approach controller repeated the order when the first order at 11:32.8 was not confirmed.

- Runway switch: In this simulation, the landing runway switch from KPHX 25L to 07R. The order is given by the local tower, as in Table 5.1 14:29.1.

- Minimum fuel: At the end of each simulation trial, the aircraft encountered fuel issues.

These corresponding timestamps t_1, t_2, t_3, and t_4 to apply these off-nominal events is indicated in Figure 5.2, respectively.

5.3.2 Simulation Setup

HITL is conducted with eight air traffic management system Metacraft facilities located at the Arizona State University TRACON Simulation Lab, which can be operated as either ATC terminal radar positions or pseudo-pilot stations. The human controllers are retired ATCos who have experience with civilian TRACON facilities within the past 15 years but do not persist possess experience with Phoenix TRACON [240]. Six retired ATCos are involved in this study. There are also three researchers who act as pseudo-pilots to fly along the assigned arrival routes during each simulation scenario. Metacraft is the name of the TRACON radar simulation computer cluster system. It provides ATC functions to maneuver the aircraft in a simulation environment, including altitude, speed, and heading. Metacraft collects and maintains data logs such as spatiotemporal tracks, LoS events, and distance measures.

During each 25 minutes experiment trial, the pop-up window shows a questionnaire probe every 3 minutes, asking either a question on workload rating or situational awareness questions. Specifically, the workload rating questions are showing three times at exact 3 min, 12 min, and 21 min timestamps. Figure 5.1 right visualize the collected data for one participant. Features include minimum separations (traffic conflict), number of aircraft (traffic density), and workload ratings are reported. A recurrence plot indicating communication tendency is also provided, visually representing the communication efforts between the tower controller and the pseudo-pilots.

The workload rating probe is designed based on the modified subjective workload as-

sessment method (SPAM) [281]. The SPAM method is a subjective workload assessment technique that is commonly used in human factors research. It involves participants rating their perceived workload using various rating scales, such as the NASA task load index (TLX) scale [282; 283]. The SPAM method also includes measures of mental effort and task difficulty to provide a more comprehensive assessment of workload. The modified SPAM is adopted following the related literature [284; 285; 286]. The workload probe employs a two-step process for administering questions related to situation awareness or workload. Participants first press a ready button, followed by selecting a response. The timing of both actions is recorded, following the methods used in the aforementioned studies.

The controller workload is self-evaluated by the workload question pop-up window, and the human performance is indicated by the count of separation violations. To facilitate this investigation, I have gathered preliminary data on three types of metrics: 1) aeronautical separation violations, which are viewed as traffic conflicts existed in the airspace and an indicator of ATCo performances; 2) real-time workload ratings, taken at three different points in each 25-minute scenario; and 3) audio recordings of controller-pilot transmissions during the workload ratings. These initial settings will serve as a foundation for further analyses using additional measures, such as facial recognition, heart rate variability, situation awareness probes, and operational efficiency. Ultimately, this research will guide the development of real-time models for predicting human performance.

Table 5.2: Description of Selected Features Recorded in the HITL Simulations. Traffic Density Is Directly Obtained from the Metacraft. The Latency Variables Are Defined and Collected Following Modified SPAM. Workload Ratings Are Collected from the Question Probe. Additionally, There Will Be an Evaluation of ATCo Conditions Based on the Correctness of the Answers to Situation Awareness Questions.

Feature Names	**Feature Descriptions**	**Feature Values**
traffic_density	Total number of aircraft under ATC participant's control	Integer: 0-23
ready_latency	Time spent from screen appearing "Ready?" to participant pressing "Ready?" on the pop-up questionnaire.	Decimal: 0.01-60.00 sec
query_latency	Time spent from pressing "Ready?" to selecting answers on the pop-up questionnaire	Decimal: 0.01-60.00sec
wl_rating	Workload rating select from the pop-up window	Integer: 1-7
sa_correct	Evaluation of selected responses on situation awareness questions from the pop-up window	0: no resp; 1: correct; 2: incorrect

In this chapter, I obtain the flight traffic recordings and the real-time workload ratings from real-world human-in-the-loop simulations. The subjective workload rating is collected from the question pop-up windows showing at 3 min, 12 min, and 21 min for each trial. The originally collected data and adjusted workload rating score has been discussed in the literature [287]. By doing this, I obtain realistic human workload levels, or the ground truth, from participants' honest ratings of their mental status. This is the most reasonable data source for obtaining features and labels for building real-world machine-learning pipelines.

5.3.3 Empirical Data Analysis

Communication transcription analysis is performed based on post-processed radio recordings. There are three major components in this part, (a) use a speech recognition tool or

manual transcription tool to translate voice to text; (b) identify the named entity of each communication transcript (i.e., controllers or pilots); (c) perform either statistical analysis or keyword extraction [288; 289; 290]. As mentioned, the deviation indicator represents the deviation in communications. A mark of "0" if the ATCo communication followed a pilot's communication, and vice versa. On the other hand, "1" will be noted if the communications are not effective and are immediately followed between the ATCo and the pilot.

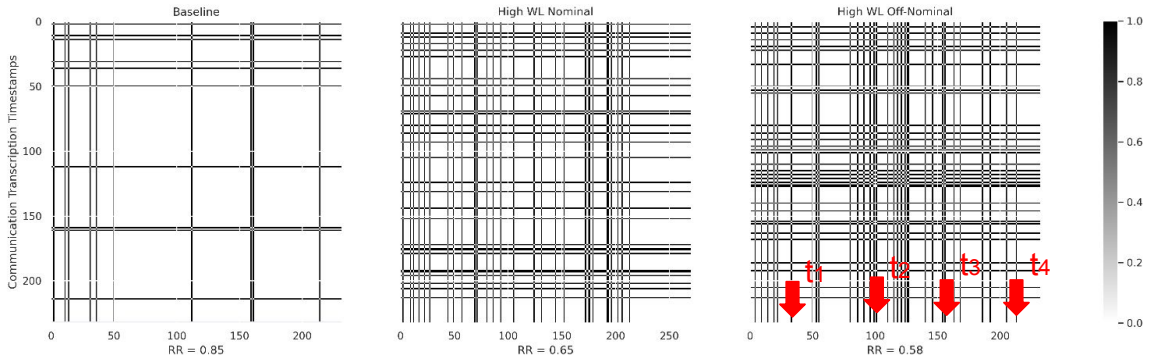
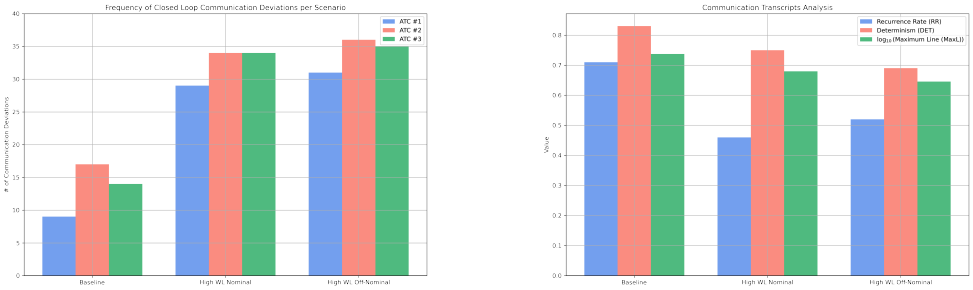


Figure 5.2: Communication Transcription Visual Analysis: Recurrence Plot (RP). RP Is Used to Quantify the Overall Tendency of Recurrence in the System. Vertical/Horizontal Lines Indicate the Laminar States Don't Change or Change Slowly over Time [4].

Figure 5.2 shows the recurrence plot (RP) of the communication deviation analysis. RP was originally proposed to visualize the complexity of dynamical systems [291; 292], where a detailed mathematical formulation can be found. In this work, I define the phase vector as the communication deviations and build the recurrence matrix $\mathbf{R}$ as,

$$\mathbf{R}_{i,j} = \begin{cases} 1, & \text{for} \quad \vec{x_i} \approx \vec{x_j} \\ 0, & \text{for} \quad \vec{x_i} \not\approx \vec{x_j} \end{cases} \quad i,j = 1,...,N \tag{5.1}$$

where N indicates the number of current states. $\vec{x_i} \approx \vec{x_j}$ means that they are approximately equal up to an error round defined as ϱ. In general, the recurrent matrix $\mathbf{R}$ compares the state and indicates the state similarity across the entire series [292]. The selection of



(a) Number of Deviations that Happened in Each Scenario. Showing First Three ATCs.

(b) Recurrence Quantification Analysis (RQA).

Figure 5.3: Communication Data Analysis on Different Scenarios. In (a), I Show the Histogram of the Frequency of the Communication Deviations for Each Scenario (up to 3 ATCos). As Discussed, the Number of Communication Deviations Indicates Communication Difficulties. In (B), I Show the QRA under Three Different Scenarios, Showing the Scenario Complexities Obtained from Communications.

similarity threshold ϱ is critical. Researchers have investigated the selection criterion in the literature based on the system states [293; 294; 295]. In Figure 5.2, I choose $\varrho = 0.1$ for the visualization. Moreover, Figure 5.2 suggest a typical vertical and horizontal lines pattern, which is suggested to be a laminar state or state idle case, while the sparse region indicates lower system complexity and vice versa [292].

Beyond the visual approach, recurrence quantification analysis (RQA) is also widely used to measure system-level complexity [294; 296; 297]. Typically, RQA is based on the diagonal and vertical patterns of the RP. I use three types of complexity measures here.

Recurrence Rate (RR) is the simplest measure of RP, which is the averaged density of recurrence points in RP. RR represents the likelihood of a state returning to its ϱ-neighborhood in the phase space [298]. It's the measure of correlations between the ATCo and pilot communications.

Determinism (DET) refers to the degree of predictability or orderliness in a system's dynamics over time. A deterministic system is one in which future states can be precisely predicted from knowledge of the present state and the system's dynamics. In the context of recurrence plots, a high degree of DET is indicated by the presence of diagonal lines in the plot, which represent points in the system's trajectory that are close to each other in phase space and recur with a high degree of regularity. Conversely, a low degree of DET is indicated by a more random or chaotic pattern in the recurrence plot, with fewer or no diagonal lines. It indicates the predictability of ATCo and pilot interaction.

Maximum Line (MaxL) is a diagonal line that represents the longest connected sequence of recurrent points in the plot. It is the diagonal line that has the most points along it, and it indicates the most persistent pattern of recurrence in the system's dynamics. The length and frequency of maximum lines can provide insights into the regularity and predictability of the system's behavior over time. MaxL quantifies the stability of ATCo and pilot interaction.

Figure 5.3 shows the histograms of communication deviations in (a), and the calculated measures of complexity in (b). As shown in Figure 5.3(a), there is a notable rise in communication challenges from the baseline scenarios to the high workload scenarios. The prolonged occurrence of off-nominal events could contribute to a slightly heightened level of complexity. Additionally, the deviations in communication patterns can differ across Air Traffic Control Officers (ATCos), possibly due to variations in individual experience and seniority. Therefore, it can be concluded that both airspace density and off-nominal events contribute to an increase in communication complexity. As depicted in Figure 5.3(b), it is evident that the correlation, predictability, and stability of the system all exhibit a decrease from the baseline scenarios to the high workload scenarios. These findings provide valuable insights into understanding the behavior of different scenarios and guide our further studies on workload prediction.

5.4 Proposed ATC Workload Prediction Framework

In this section, I describe the proposed workload machine learning prediction framework in Figure 5.4. I first demonstrate the formulation and basic concepts of the graph learning problems in Section 5.4.1. The dynamical graph convolution learning algorithm and conformal prediction setup are explained in Section 5.4.2 and Section 5.4.3, respectively.

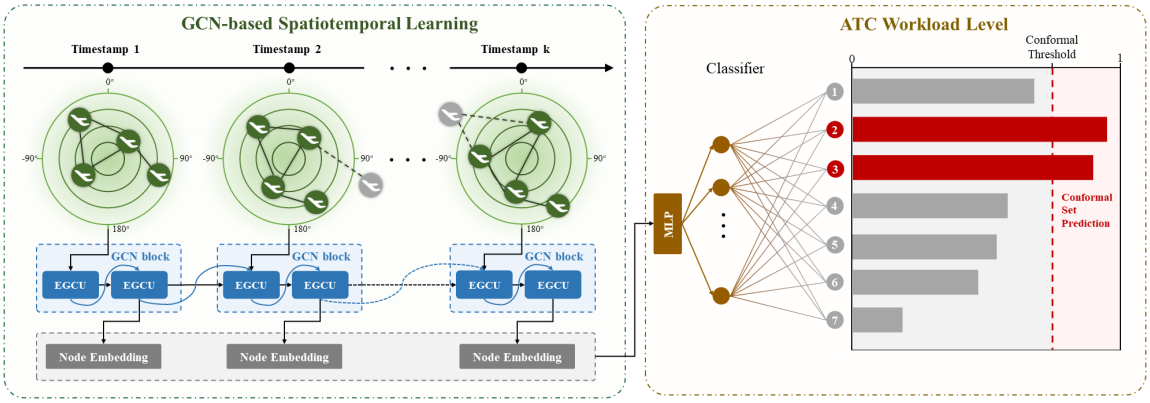


Figure 5.4: Schematic Illustration of Conformalized EvolveGCN Set Prediction Framework. I Formulate the ATC Workload Level Prediction as a Time-series Graph Classification Task, Where Each Graph Node Represents *Each Aircraft under the ATC's Control*. The Number of Nodes and Weight (Distance) of Each Edge Can Change Across Different Timestamps. On the Classifier Side, I Propose the Conformal Prediction Set for Improved Ground Truth Coverage. Conformalization Acts as a *Post-hoc* Procedure to Post-process the Prediction Labels, Where the Softmax Probability Threshold Is Inferred on the Calibration Set.

5.4.1 Problem Formulation

As mentioned, the ATCo workload prediction task is viewed as a time-series dynamical graph classification task. In this chapter, I use subscript $t \in \{1, ..., T\}$ to demonstrate the

timestamp and superscript $l \in \{1, ..., L\}$ to denote the layer index. Graph neural networks (GNNs) are introduced to model the spatiotemporal layout of the airspace from the structured graph data with explicit message passing [133; 134]. I denote a graph at timestamp t with vertices and edges, represented as $G_t = (V_t, E_t)$, where the number of nodes at timestamp t is $N_t^{nodes} = |V_t|$ and the number of edges at timestamp t is $N_t^{edges} = |E_t|$. The adjacency matrix $A_t \in \mathbb{R}^{N_t^{nodes} \times N_t^{nodes}}$. The constructed graph structure can be either directed or undirected depending on whether the edges are directed from one node to another. The dynamic graph is mentioned when the graph topology varies with time. Especially, the graphs in our work are undirected dynamical graphs. The constructed graph inputs are $A_t \in \mathbb{R}^{N_t^{nodes} \times N_t^{nodes}}$ and $X_t \in \mathbb{R}^{N_t^{nodes} \times N_t^{features}}$, where A_t is the adjacency matrix at each timestamp t and X_t is the node feature matrix.

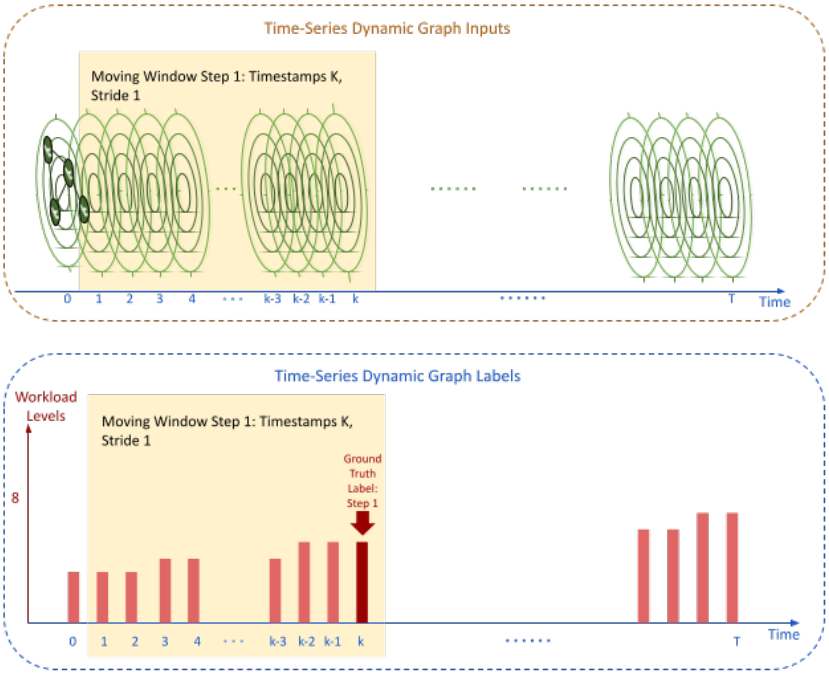


Figure 5.5: Schematic Illustration of the Moving Window Approach. At Each Step, the Moving Window Moves 1 Timestamp (5s) along the Temporal Dimension (Stride 1). Each Series of Graphs Contains a Graph of κ Timestamps. The Workload Ground Truth Label for the Graph Series Input Is the Workload Level (1-7) Reported at the Last Timestamp, Collected from the Human-in-the-loop Experiment. This Setup Allows the Model to Capture Long-term Spatial Relationships and Result in a Prediction at Every Timestamp since κ. For Abbreviation, the Graph Input Is Represented by a Radar Plot.

Graph structure represents the spatial layout of the airspace. However, the dynamical graph constructed A_t is based on the geo-distance between two aircraft pairs, resulting in geospatial graphs on a two-dimensional space. This type of graph construction is widely adopted with acceptable complexity. Consequently, in this work, I adopt a scaling formula described in minimum-spanning-tree-based workload prediction task [13] to scale the horizontal and vertical distance between two aircraft pairs into one distance metric. Specifically, the graph is built by calculating the distance $\tilde{d_{ij}}$ between two aircraft pairs (i, j) at

the same timestamp t. I use $d_{t,ij}$ to represent the horizontal distance and $h_{t,ij}$ to present the vertical separation distance between aircraft pairs (i, j). The scaling function is described in Equation (5.2).

$$\tilde{d_{t,ij}} = \sqrt{d_{t,ij}^2 + s^2 h_{t,ij}^2} \tag{5.2}$$

where s is the spatial scaling factor which equalizes the separation on the horizontal and vertical dimension, as in Equation (5.3).

$$s = \begin{cases} 0.005, & \text{for} \quad \mathsf{alt}_i \leq 29,000 \text{ and } \mathsf{alt}_j \leq 29,000 \\ 0.0025, & \text{for} \quad \mathsf{alt}_i > 29,000 \text{ or } \mathsf{alt}_j > 29,000 \end{cases} \quad i, j = 1, ..., N \tag{5.3}$$

In this workload prediction task, I first obtain the constructed input X_t and A_t at each timestamp t. In such a way, I obtain the series of graphs $G_t, t \in \{1, ..., T\}$. Then, I fill the workload ratings into another time series based on the self-evaluated workload rating during the HITL simulations, representing the prediction labels. Last, I use a moving window approach to build the correct input-output matching for supervised machine learning. The schematic illustration of the moving window process is shown in Figure 5.5. I define a window of size κ and move the window along the time axis of the series of inputs, with a stride of 1. The time-series graph of size κ is denoted as $\{G_t\}_\kappa$. Similarly, I move the window function along the prediction labels and obtain the workload ratings by claiming the last reported workload value within the current window. Then, if Y_t denotes the ground truth workload label at timestamp t, I mathematically formulate the problem into,

$$Y_t = \mathsf{EvolveGCN}(\{G_t\}_\kappa) \tag{5.4}$$

5.4.2 Evolving Graph Convolution Network

Spatial graph convolutional networks (GCN) [135] convolve the input A_t and X_t using the derived compact form,

$$H_t^{l+1} = \sigma(\hat{A}_t H_t^l W_t^l), \quad with \quad H_t^0 = X_t \tag{5.5}$$

where σ is the activation function (i.e., ReLU). $\hat{A}_t$ is a normalized version of A_t, to account of numerical instability. Specially, $\hat{A}_t$ is defined as, $\hat{A}_t = \tilde{D}_t^{-\frac{1}{2}} \tilde{A}_t \tilde{D}_t^{-\frac{1}{2}}$, $\tilde{A}_t = A_t + I_t$, $\tilde{D}_{t,ii} = \sum_j \tilde{A}_{t,ij}$. It's clear that H_t has the same dimension as X_t as $H_t \in \mathbb{R}^{N_t^{nodes} \times N_t^{features}}$. $W_t \in \mathbb{R}^{N_{features} \times N_{featrues}}$ is the kernel parameters. For multiple graph convolutional layer setup, H_t^{l+1} stands for the updated graph embedding of convolutional layer $l+1$ at timestamp t. Specifically, in classification problems, the activation function σ at the output layer L is the softmax function.

Evolving Graph Convolution Network (EvolveGCN) improves GCN by introducing recurrence layers to capture the dynamism underlying a time-series graph. Two types of EvolveGCN are presented [299], depending on the recurrent updating architecture.

The first variant treats the GCN kernel parameter W_t^l as the hidden state of recurrent learning function and updates W_t^l with a gated recurrent unit (GRU), while the node embeddings of node features are still contained within the GCN hidden state tensor H_t^l. EvolveGCN-H is used to denote this variant Equation (5.6). This requires a special design of GRU computation flows as described in [299].

$$W_t^l = GRU(H_t^l, W_{t-1}^l) \tag{5.6}$$

$$H_t^{l+1} = \sigma(\hat{A}_t H_t^l W_t^l) \tag{5.7}$$

Another variant of EvolveGCN is the -O version Equation (5.8), where the kernel parameter W_t^l is treated as input of recurrent learning without considering the temporal cor-

relations between node embeddings. The implementation of EvolveGCN-O is straightforward by extending dimensions.

$$W_t^l = LSTM(W_{t-1}^l) \tag{5.8}$$

$$H_t^{l+1} = \sigma(\hat{A}_t H_t^l W_t^l) \tag{5.9}$$

Either an EvolveGCN-H or EvolveGCN-O is denoted as an Evolving Graph Convolution Unit (EGCU), as shown in Figure 5.4. In both ways, the EGCU first updates the GCN weights and then propagates the hidden states through the layers. Several layers of EGCU form a GCN block in Figure 5.4. For a graph learning problem with large feature space, EvolveGCN-H is more effective since the feature embedding recurrence is also considered. Otherwise, EvolveGCN-O is more focused on learning the graph topology structure changes.

5.4.3 Conformal Prediction

In this section, I provide a brief overview of conformal prediction (CP). For the classification task mentioned above, I have EvolveGCN acted as the classifier $\mathcal{C}$, which outputs an estimated probability for each class, i.e., $p \in [0,1)^{\varkappa}$ for $\varkappa$ classes. I reserve a small amount of data called *calibration set* to calculate the probability score threshold $\hat{q}$ such that the following condition holds on the test set,

$$1 - \alpha \leq \mathbb{P}(Y_{test} \in \mathcal{C}(X_{test})) \leq 1 - \alpha + \frac{1}{n+1} \tag{5.10}$$

where the test dataset is the unused test set to evaluate model performance. $\alpha \in (0,1]$ is the pre-defined tolerated error rate. This is to guarantee that the model is $1-\alpha$ confident that the model prediction set contains the correct ground truth label. This equation is also known as the marginal conformal coverage guarantee, which has been proved in the literature

[300; 301]. Notably, the *calibration* is the key step to find $\hat{q}$. Suppose I define the concept of the conformal score by one minus the softmax probability of the true class, $\hat{q}$ is defined to be the $\frac{\lceil (n+1)(1-\alpha) \rceil}{n}$ quantile of the conformal scores. $\lceil \rceil$ is the ceiling function to correct the quantile. Then, the prediction of a new test sample will be all classes with a softmax score higher than $\hat{q}$. The prediction set will be larger if the model is uncertain about the prediction labels or if the input is out-of-distribution. Intrinsically, the size of the prediction set is the indicator of model uncertainty.

Conformal prediction (CP) has been studied from various angles by researchers. However, the classical CP method is susceptible to coverage issues due to its tendency to produce the smallest average size of prediction sets [302]. Specifically, CP tends to overcover hard data samples while undercover simple ones. To address this issue, researchers have proposed an approach called adaptive conformal prediction [303; 304]. The underlying principle of this method is to compute the conformal threshold $\hat{q}$ based on the cumulative softmax score across $\varkappa$ classes.

It's also noteworthy to discuss conformal evaluation methods, which are adopted in evaluating our model. To determine the model's performance, a straightforward method is to examine the histogram of prediction set sizes visually. Essentially, a larger size of the prediction set implies that the model is facing certain data quality problems, while the variation in the set size can provide insights into the model's ability to differentiate between easy and difficult input samples.

$$\mathbb{P}[Y_{test} \in \mathcal{C}(X_{test}) | X_{test}] \geq 1 - \alpha \tag{5.11}$$

Conditional coverage is a feasible approach to evaluate the adaptivity of conformal prediction. For instance, in a classification setting, I seek to find the prediction sets with exactly $1 - \alpha$ coverage for any input data sample, as in Equation (5.11). The conditional coverage concept is a stronger metric than the marginal coverage mentioned above. Some

literature mentioned that conditional coverage is impossible to achieve in most general cases [305]. **Size-stratified coverage (SSC) metric** is a general metric to evaluate how close the model is able to achieve Equation (5.11). SSC metric is a way to evaluate the performance of conformal prediction models. It is based on the idea that prediction sets of different sizes may have different properties and should be evaluated separately. The key is to group test samples into different size strata based on the size of their prediction sets and compute the averaged empirical coverage on each size strata. SSC metric can be useful to diagnose specific issues, such as overcoverage or undercoverage, that may be related to the size of the prediction sets. In addition, another conformal prediction evaluation method has been proposed recently [306], where a calibration plot of the prediction error versus the specified significance level (α) is used. Remarkably, In Section 5.5, I evaluate our model with these metrics.

CP provides a rigorous way to measure the uncertainty associated with the predictions made by a machine learning model and to express this uncertainty in the form of prediction intervals or regions that can be used to guide decision-making. This can be particularly useful in critical engineering applications where accurate prediction intervals or regions are essential [307]. CP has been widely adopted in drug discovery [308], medical diagnosis [309], and robotics [310]. CP is a unified *post-hoc* softmax score calibration process to generate prediction sets for any classification model [311; 312; 313]. In this work, I propose to use CP for aviation decision support. Specifically, the prediction set comes from CP gives uncertain prediction label suggestions, i.e., workload rating of 3, 5, 7. While the isolated classification label in workload prediction is not reasonable, I propose to fill the intermediate workload ratings based on the minimum predicted rating and the maximum predicted ratings.

5.5 Experiments

In previous sections, I have introduced the HITL data collection process, the problem definition, the machine learning model system design, and conformal prediction for better ground truth label coverage Section 5.4. In this section, I present a comprehensive of experiments to test and evaluate the proposed model. I first discuss several evaluation metrics used for this classification task. Then I report the classification accuracy from the machine learning model with a few implementation details. Lastly, conformal prediction set results are reported with several conformal coverage evaluation methods mentioned in Section 5.4.3. The validation set is used to tune hyper-parameters, and the testing dataset results are evaluated based on the best validation epoch.

5.5.1 Evaluation Metrics

The F-score, or F-measure, is a binary classification metric used in statistical analysis to assess the accuracy of test samples. It is computed using both precision and recall, where precision measures the number of correctly identified true positive results relative to all positive results (including incorrect ones), and recall measures the number of correctly identified true positive results relative to all samples that should have been identified as positive. Precision is also referred to as positive predictive value, while recall is also called sensitivity in binary diagnostic classification.

Specifically, the F1-score is defined as the symmetrical harmonic mean of precision and recall [314]. F1-score can also be used for multi-class classification by taking either the micro-averaging (MicroF1) or the macro-averaging (MacroF1).

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{5.12}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{5.13}$$

$$\text{MicroF1} = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \tag{5.14}$$

$$\tag{5.15}$$

where TP, FP, and FN stands for true positive, false positive, and false negative, respectively. TP means the model predicted correctly for the positive class, FP means the model predicted incorrectly for the positive class, and FN means the model predicted incorrectly for the negative class. Equation (5.12) gives the mathematical formulation of MicroF1, where each sample is considered independently without considering which class this sample belongs to. MicroF1 treats each data input equally but is biased on class frequency. MicroF1 is useful when the classification task is unbalanced, meaning that some classes have many more instances than others. In this case, the MicroF1 gives equal weight to each instance, regardless of its class.

$$\text{MacroF1} = \frac{\text{F1}_1 + \text{F1}_2 \cdots, + \text{F1}_\text{n}}{\text{n}} \tag{5.16}$$

On the contrary, MacroF1 average the F1 score across all classes as in Section 5.5.1, where n is the number of classes, and $\text{F1}_1 + \text{F1}_2 \cdots, + \text{F1}_\text{n}$ are the F1 scores for each class. MacroF1 treats each class equally regardless of the size of samples within each class thus, it's biased on the number of samples. MacroF1 is useful when the classification task is balanced, meaning that each class has approximately the same number of instances. In this case, MacroF1 gives equal weight to each class, regardless of its frequency.

In this work, I am encountering a highly-imbalanced classification problem Figure 5.1. Thus, MicroF1 is a better indicator than MacroF1. It is noteworthy to mention that n in

should be adjusted to exclude the class labels that are not presented in either ground truth or predictions. I report both metrics in our studies.

5.5.2 Implementation Details

Although the commonly adopted node-level or link-level classification objective is prevalent, the proposed workload prediction framework is instead a graph-level classification task [299]. Thus, on the output layer, I take the aggregated class probability score across each node to get a unified score of the entire graph. I adopted the grid-search strategy to search for key parameters and fine-tuned the neural network model with the data collected from three different scenarios. The key parameters used are the number of EvolveGCN layers (EGCU layers), and the dimensions of these layers. Dropout is used in the classifier to address overfitting. Table 5.3 lists the fine-tuned key model parameters under three simulation scenarios. Other parameters, such as the dimension of classifiers, are kept the same as the original implementation [299]. Parameter tuning on the classifiers might be useful, but it's beyond the scope of this study.

Table 5.3: List of Fine-tuned Model Parameters Used in EvolveGCN Training under Three Different Simulation Scenarios.

	Baseline	High Workload Nominal	High Workload Off-Nominal
Number of EGCU Layers	2	2	4
EGCU Layer Dimensions	64	128	64
Dropout Ratio	0.25	0.5	0.25
Learning Rate	0.001	0.0015	0.0005

Moreover, as discussed in Section 5.3, the first reported workload rating starts at 3 minutes of the 25 minutes duration. This corresponds to the $36th$ timestamp with $5s$ interval in the collected flight traffic data. Consequently, the moving window size κ in our experiment is 36, with a stride of 1. I separate the data into train, validation, and test sets with a ratio of $[0.4, 0.3, 0.3]$. The validation set is used for deep learning model hyper-parameter tuning. Moreover, the validation set is also used as the calibration set to find the CP threshold $\hat{q}$.

5.5.3 Experiment Results

Table 5.4: Workload Level Prediction: Comparison between Different Workload Prediction Methods.

ATC Workload	**Baseline**		**High Workload Nominal**		**High Workload Off-Nominal**	
Level Prediction	MicroF1	MacroF1	MicroF1	MacroF1	MicroF1	MacroF1
Simple LR w/ Density	0.306	0.218	0.307	0.198	0.323	0.195
Simple LR w/ Graph Feature	0.331	0.236	0.383	0.263	0.350	0.255
2-layer MLP	0.364	**0.283**	0.455	0.386	0.459	0.367
GCN	0.404	0.218	0.580	0.401	0.526	0.352
EvolveGCN-O	**0.545**	0.277	**0.740**	**0.632**	**0.695**	**0.472**
EvolveGCN-H	0.413	0.221	0.593	0.474	0.581	0.414

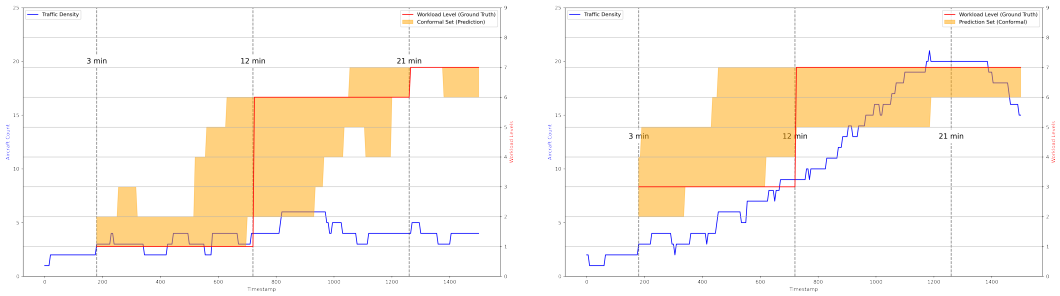
In this work, I compare our model with both classical methods (i.e., linear regressions (LRs)) and simple data-driven learning methods (i.e., fully-connected neural networks/multilayer perceptions (MLPs)). In [249], the authors also conducted high-fidelity human-in-the-loop simulations to study the impact of traffic density features on controller workload. They found that the workload rating of the enroute center controller is propor-

tional to the number of aircraft with a slope of 0.306 and bias of -3.373. They also identified the primary sources of workload for controllers, including airspace and traffic management, communication, and coordination tasks with workload management suggestions. In [13; 253], the authors create graph-structured airspace data structure – minimum-spanning trees but propose several handcraft features based on the histogram of node features. Then a two-layer fully-connected neural network is used for prediction based on handcrafted features and shows remarkable performance. However, the workload ratings are directly generated from the traffic density, where thresholds of 7 and 17 separate workload ratings into low, medium, and high scenarios. Likewise, inspired by this work, our proposed method adopts a graph structure to represent the spatiotemporal layout. I utilize the recent advancement in graph learning and learn from the graph structure without handcrafted features.

In Table 5.4, comparing the first two rows, I first show that including additional graph node features can achieve higher prediction accuracy, even for simple LRs. Despite the traffic density features, additional graph node features are traffic conflict features (i.e., horizontal/vertical minimum separation to nearby aircraft). For the MLP with handcraft features, I generate second-order statistics of the sum and difference histograms introduced in [13]. As a reference to EvolveGCN, I also conduct an experiment on vanilla GCN. This can be easily achieved by removing the LSTM layer in Equation (5.8). I show that EvolveGCN can achieve significantly higher MicroF1 and MacroF1 than LP, MLP, and GCN. Moreover, the -O variant EvolveGCN outperforms the -H variant. One of the major reasons is the selection of top K indices reduces the hidden dimension of EGCU, due to the low node feature dimensions and a small number of nodes in graphs (i.e., only one aircraft showing up at the first timestamp) at certain timestamps.

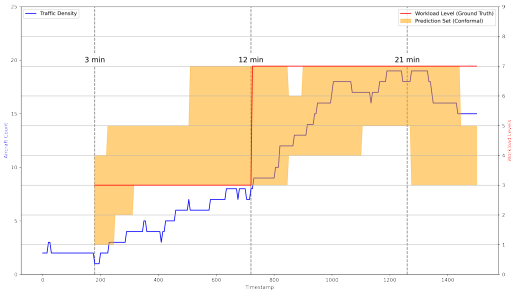
5.5.4 *Conformal Prediction Results*

As mentioned, I use conformal prediction to improve prediction accuracy further. In Figure 5.6, the prediction on one test participant is shown. The conformal prediction set coverage is the shaded region. The conformal prediction is generated with a tolerated error rate of $\alpha = 5\%$. The blue solid lines show the real-time aircraft density (left axis) in the simulation, and the red lines are the interpolated workload ratings. At 3 min, 12 min, and 21 min, the participants are required to submit their workload rating to the computer. The ground truth workload ratings are colored in red (right axis). The conformal prediction set covers most of the ground truth but is undercover at several spots. As discussed in Section 5.4.3, I further evaluate the conformal prediction coverage.



(a) Baseline Condition

(b) High Workload Nominal Condition

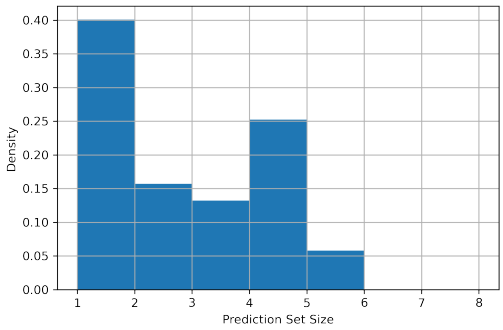


(c) High Workload Off-Nominal Condition

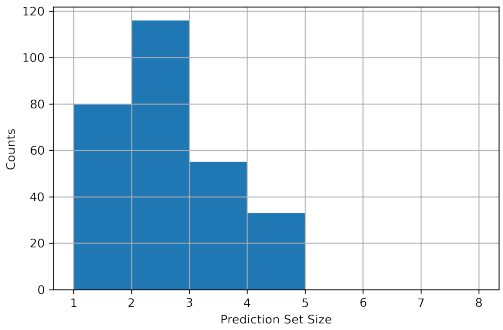
Figure 5.6: Visualization of Conformal Predictions on the Test Sample. For the Workload Level Prediction Task, I Set Our Prediction as the Range Between the Lowest Predicted Workload Level and the Highest Predicted Workload Level.

5.5.5 *Conformal Coverage Evaluation*

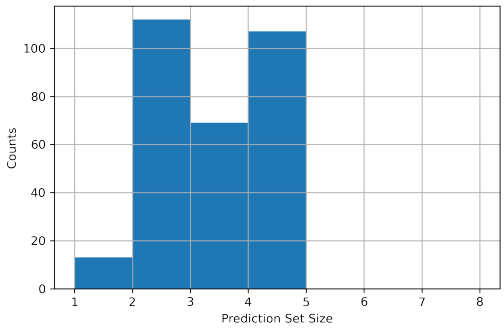
I adopt different conformal coverage evaluation metrics to examine the performance of our conformal set. Firstly, I plot the histogram of set sizes. A high average set size suggests that the conformal prediction procedure is imprecise, which could indicate issues with the score or underlying model. Secondly, the range of set sizes indicates whether the prediction sets adapt properly to the complexity of examples. A wider range is typically preferred because it implies that the procedure accurately differentiates between simple and challenging inputs. I show the histograms in Figure 5.7. The size of conformal prediction is typically around 5. The spread for baseline and high workload nominal conditions looks reasonable. The model is able to distinguish hard and easy samples. However, the spread for high workload off-nominal conditions indicates potential scoring issues or simply difficult data [301].



(a) Baseline Condition



(b) High Workload Nominal Condition



(c) High Workload Off-Nominal Condition

Figure 5.7: Histogram of Set Sizes on Test Set Predictions. The Spread of the Histogram Shows the Difficulty of Making a Correct Prediction.

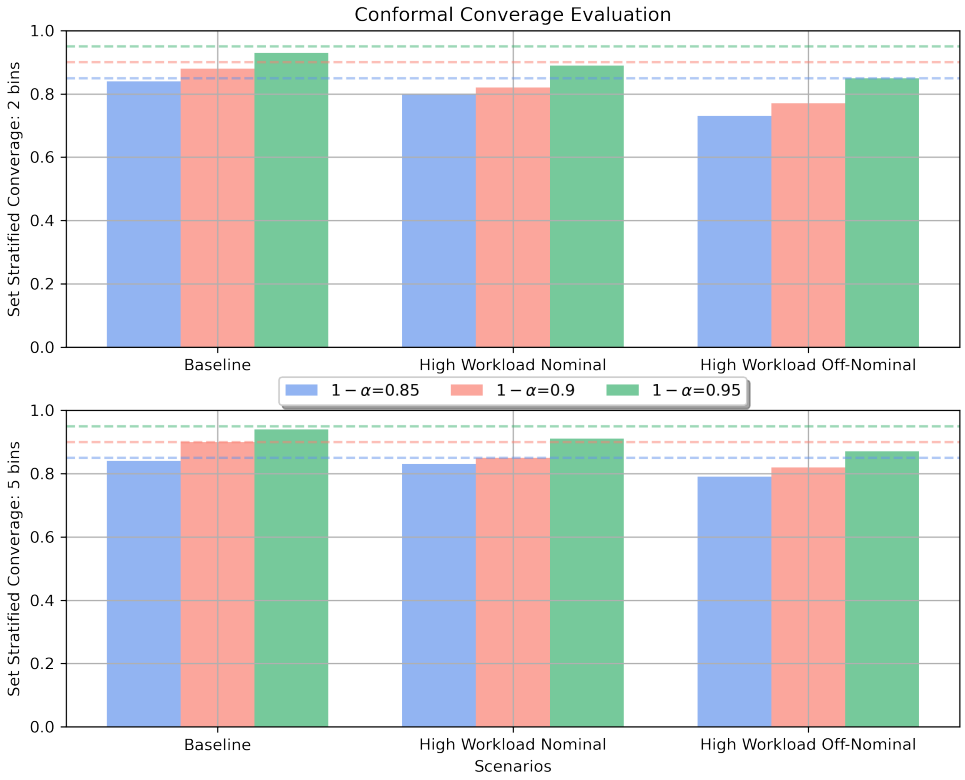
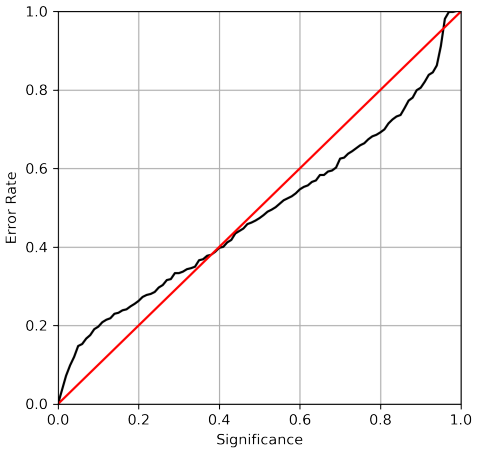


Figure 5.8: Conformal Coverage Evaluation with Various Desired α Values Under Three Workload Simulation Conditions. I Use the Size-stratified Coverage (SSC) Metric Better to Represent the Adaptive Coverage of the Conformal Set Coverage.

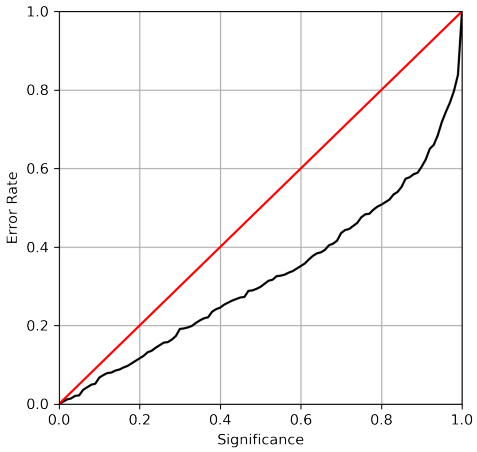
Following the discussion in Section 5.4.3, I investigate the size-stratified coverage (SSC) to evaluate the condition coverage and plot the figures in Figure 5.8. The dash lines are desired coverage values. In this figure, I consider three desired error rates, $\alpha = 0.15, 0.1, 0.05$, for three cognitive workload scenarios. I use two possible $\mathcal{C}(x)$ cardinalities of two bins and five bins. In other words, I divide the predicted sets into different size categories (e.g., sets of size 2, sets of size 5.) and calculate the percentage of times that the true value falls within each category. I discover that the prediction coverage of baseline

conditions shows a good sign, but the two high workload scenarios tend to under-coverage. Again, the reasons still come from the unsatisfactory of collect data, which leads to lower reported F1 values.

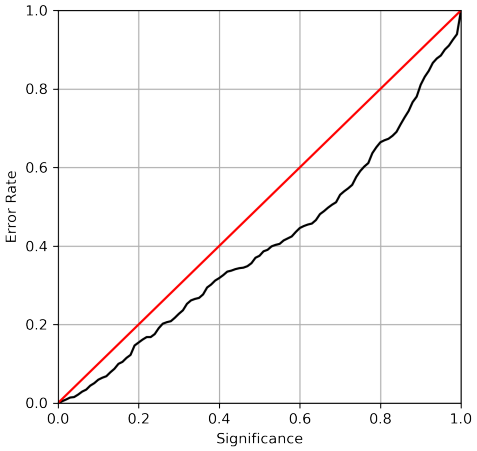
To further look at the coverages, I adopt another recently proposed figure to better show the prediction coverage violations [306]. In Figure 5.9, I show three figures for three simulation scenarios. The x-axis shows the tolerated error rate (the specified significant level), and the y-axis shows the fractions of failed prediction sets (the number of prediction samples where the ground truth label is not in the conformal prediction set). This property holds true in high workload scenarios but not in the baseline scenarios when the significance level is lower than 0.4 or higher than 0.95. One of the key advantages of conformal predictors is their ability to offer valid predictions even when new examples are independently and identically distributed with the training examples. Additionally, in Figure 5.9, I use Kolmogorov-Smirnov (K-S) test to test the distributions of predictions in the calibration set and test data under three conditions. In the K-S test, the null hypothesis is that the calibration and test samples are drawn from the same data distribution. The corresponding p-values obtained for three conditions are, (a) $1.98e-12$; (b) $3.68e-48$; (c) $2.18e-55$. All three values below 5%, are considered two-sided statistically significant. Thus, the hypothesis holds true.



(a) Baseline Condition



(b) High Workload Nominal Condition



(c) High Workload Off-Nominal Condition

Figure 5.9: The Calibration Plot Illustrates the Observed Prediction Error, Which Is the Proportion of True Labels That Are Not Included in the Prediction Set, Plotted Against the Pre-specified Significance Level ε, or the Tolerated Error Rate. The Conformal Predictor Is Deemed Valid Only When the Observed Error Rate Is Within the Limit of ε, I.E., The Observed Error Rate Should Align Closely with the Diagonal Line Representing the Tolerated Error Rate for All Significance Levels.

5.6 Conclusions

In this chapter, I investigate the workload prediction problem. I formulate the problem into a time-series dynamic graph classification task with changing graph topologies. I demonstrate the effectiveness of this proposed method from real-world human-in-the-loop air traffic control simulations, where participants are retired air traffic controllers. I show that traffic density features and traffic conflict features have a positive influence on workload predictions. Algorithm-wise, the graph-structured data-driven learning model outperforms the existing practices in workload prediction research literature (i.e., simple regressions, simple neural networks with handcrafted features).

5.6.1 Limitations

There are several limitations. Firstly, I only have limited resources to conduct the HITL experiments in a simulation environment. Real-world scenarios can be immensely different from simulation scenarios. Secondly, data quality is critical for developing a successful machine-learning algorithm. In this work, I have to use the corrected workload rating data due to the poor quality of the originally collected workload ratings. Only 6 out of 10 ATCos HITL simulation data are considered valid for use in this work, and I also have to use the manually adjusted workload ratings [287]. Another critical part is modeling the different ATC strategies adopted by different controllers, which can result in a multi-modal machine learning setup. The benefit of including ATCo strategies has also been discussed in the literature [154]. Lastly, better algorithm development can help with improved workload prediction performance. The single prediction label made by either EGCU-O or EGCU-H can be improved, despite the data quality issue.

5.6.2 Insights

This work is beneficial for the non-intrusive, uninterrupted executive controller workload prediction, and it is purely based on the flight traffic data. First, I show that both traffic density and traffic conflict features contribute to higher prediction accuracy. Then, I show that model of the spatiotemporal airspace layout as a dynamic time-series graph learning problem has great potential for ATC workload level predictions. Additionally, I explore the possibility of further accuracy improvement by introducing a *post-hoc* classification score processing process, namely conformal prediction, which can be used to generate multiple classification labels adaptively.

Based on these insights, I propose several research directions that might be interesting to researchers,

- I am expecting a significant performance improvement by conducting more HITL simulations or real-world ATC experiments, collecting additional high-quality data, and data-driven model refinement. Spatiotemporal graph learning is a popular theoretical research direction, and better graph learning model architecture is expected, which results in better workload prediction performances.

- The window function setup for input-output data matching is flexible for any practical requirement in the real world. The length of the window determines the history length to be considered, while the stride size defines the prediction horizon.

- The workload prediction problem formulation can be alternated from workload rating classification to workload rating regression task. This setup is algorithm-wise more reasonable for uncertainty calibration with conformal prediction but requires significant experiment setup change. For instance, the current rating-based question prob in modified SPAM and NASA TLX will be modified to continuous variables. A

considerable modification of the HITL simulations is desired.

- There are still several important questions that remain unanswered, such as how to incorporate real-time data and feedback into the prediction model and how to adapt the model to different types of air traffic control systems. Future research in this area could also explore the impact of other factors, such as weather conditions and aircraft type, on controller workload and safety.

- Several works of literature quires the validity of only using traffic-related factors to predict mental workload, where a significant part of pilot-controller interactions and feedback are missing [154]. In further studies, I propose to build a predictive model that can consider reciprocal feedback interactions (i.e., the communication deviations [234] in Section 5.3) from a learning perspective.

- Further studies can also combine trajectory prediction models such that the task demands can be predicted first and then perform workload forecast in real-time. In such a way, either deterministic or probabilistic trajectory prediction models can act as moderators of workload models [256; 95; 103; 175].

I believe the above discussions have practical implications for aviation authorities, airlines, and air traffic management providers. Specifically, our workload prediction model could be used to inform scheduling and staffing decisions, optimize resource allocation, and support proactive safety management. However, it is important to note that successfully implementing such interventions will require collaboration and communication across stakeholders.

Chapter 6

CONCLUSION

Throughout its history, aviation has been a pioneer in innovation, driving significant advancements in passenger comfort, efficiency, and safety. The introduction of jet engines in the 1950s and fly-by-wire in the 1980s were two notable milestones that revolutionized air transportation. Today, as we enter the 2020s, we stand on the cusp of another unprecedented evolution in the industry. With the exponential growth in computer processing power and emerging data storage facilities, the possibilities of AI in aviation are endless, with the potential to enable multiple sectors of aviation, including autonomous flights, predictive maintenance, air traffic management, etc. AI is a fairly old concept that stands for any technology that is trying to *emulate* human behaviors and performances, meaning a mature AI model has cognitive inference capabilities to generate existing knowledge to a new phase. This stands in contrast to statistical models, which are merely numerical fittings based on historical data. Lifelong learning remains an essential principle, although adapting to new technologies can be challenging and push individuals out of their comfort zones. The future will be shaped by open-minded outstanding multidisciplinary researchers, through the integration of AI technologies, paving the way for a safer, more efficient, and sustainable aviation industry.

6.1 Contributions

This thesis has demonstrated the significant potential for AI to improve various sectors of aviation, to address the increasing challenges faced by the ATFM. In this study, I investigate trajectory prediction capabilities for TBO, environmental impacts, flight operations, and aviation human factors. A detailed literature review is conducted along with each of

these topics mentioned, and presented in the chapter. Also, the itemized contributions of each work have been discussed at the beginning or end of each chapter. Here, I want to highlight the most important ones.

- Chapter 2 investigates the convective weather impacts on trajectory prediction models. I propose a modular-designed probabilistic deep neural network structure, which is a scalable and flexible architecture for handling complex spatial-temporal correlations between aircraft trajectory and weather conditions. Variational Inference-based Bayesian Deep Neural Network is shown to be a robust probabilistic method compared to MCDropout for BNN, and the proposed method shows variance reduction for both small and large flight deviation cases. This work reveals the relationships between convective weather, flight plans, and flight trajectories. The performance is validated with convective weather forecasts and flight recording data from the Sherlock data warehouse. Effective data processing is essential for successful machine learning problem-solving. This includes proper data filtering and feature engineering, which should be informed by domain knowledge and expertise in order to extract relevant and meaningful features from the raw data.

- Chapter 3 aims to further reduce trajectory prediction deviations in the near-terminal airspace, where weather-related is no longer a dominating factor. I propose a novel multi-agent trajectory prediction model called B-STAR that can quantify uncertainties in flight track data without relying on pre-defined parameters or randomized inputs. The model incorporates aviation regulations on aeronautical separation by estimating the Haversine distance to select silent neighbors and building a graph encoding. A sensitivity study is also conducted on separation assurance distance to evaluate the impact on the proposed framework. The chapter also presents a module-based machine learning framework that utilizes advanced computer software and

hardware tools for data analysis. Open-source toolsets for data pre-processing and web-based interactive visualization are integrated with the B-STAR framework to enable efficient and effective data analysis.

- Chapter 4 examines several arrival delay scenarios in historical flight data and gains insights into the multimodal distribution of aircraft arrival times conditioned on flight events, the occurrence of go-around events at approximately 100 nautical miles from the terminal, and suboptimal landing scheduling preferences made by human controllers. The statistics of minimum safe times are predicted using a tree-based probabilistic machine learning algorithm and incorporated as safety constraints to the time-constrained traveling salesman problem. A conditional machine learning predictor based on event counts within a certain distance of the target aircraft is proposed to improve prediction performance, and multiple factors including geographical location, speed profiles, flight event counts, and airspace complexity measures are integrated for probabilistic prediction of arrival time. The proposed framework shows a reduction in total aircraft landing time compared to the FCFS rule, even in extended airspace, where early adjustment of aircraft speed profiles can be issued to avoid holding patterns.

- Chapter 5 proposes a novel approach for predicting executive controller workload during approach scenarios using graph neural networks (GNNs). The task is formulated as a dynamical time-series graph classification problem, and the Evolving Graph Convolutional Network (EvolveGCN) is shown to achieve higher prediction accuracy compared to statistical and classical learning methods. The use of a moving window approach to build the correct input-output matching from the collected sparse workload data is proposed, with the window size representing the temporal length of the historical information used in workload prediction. The data struc-

ture formulation transfers complex structured traffic features into a lucid format for research and development purposes. The chapter also explores conformal prediction to expand predictions as set predictions, which has better ground truth label coverage and can indicate prediction uncertainties. The results suggest that GNNs have great potential for predicting controller workload with varying spatiotemporal airspace layouts and that conformal prediction is a valuable machine learning *post-hoc* processing tool to boost performance and indicate prediction uncertainties.

6.2 Future Works

At the conclusion of each chapter, I identified the limitations of my work and presented several potential areas for future research on each individual task. In light of the broader scope of this thesis on AI-enabled aviation studies, I now wish to discuss some high-level aviation research topics that merit further investigation to advance the digitalization of aviation. To this end, the future plans consist of two main objectives: (i) continue exploring AI applications to enhance aviation safety and efficiency, and (ii) investigate research directions to improve the robustness and trustworthiness of aviation systems-level assurance.

6.2.1 AI Application Directions

Air Traffic Management Automated decision-making assistance tools have served as essential features of air traffic management systems for several decades. The degree of automation directly influences the mental workload of air traffic controllers. Additional automated decision support systems can assist pilots and controllers in prioritizing safety-critical tasks while also facilitating the seamless exchange of information and promoting collaboration among all stakeholders, including those involved in airborne operations. Notably, there are two distinct definitions of automation that are recognized by domain experts. Some focus on developing *pattern matching* or *service discovery* algorithms to identify the

best operational procedures for specific scenarios from an in-house operation database, typically using semantic web services. Such systems provide ATCos of any seniority level with a wealth of real-world operational experience and ensure regulatory compliance. The second type of automation leverages the generalization capabilities of AI models, which are mostly recognized by the general AI community. The decision recommendations generated by such systems do not guarantee regulatory compliance and have been criticized for their trustworthiness. However, as mentioned, AI models are designed to emulate human intelligence, and the decision recommendations should not be limited by human performance. The author believes this track will eventually overcome manpower and thus has greater potential. Future research directions include enhancing strategic planning capabilities for safety and efficiency, improving trajectory prediction accuracy by proposing more powerful models, enabling ATC operation matching capabilities, increasing operational efficiency with data-enhanced optimization models, as well as many other research directions.

Autonomy Cockpit automation and autonomous flights are considered as types of autonomous operations. Cockpit automation refers to the use of automated systems and technology in the cockpit to assist with various tasks and processes, including navigation, communication, flight control, and monitoring. This technology aims to replace or at least human co-pilots with computer-dominant decision-making processes, which is why it is considered autonomous. Autonomous flight, on the other hand, is one of the most prominent applications of machine learning, with the drone industry leading the way. The increasing demand for urban air mobility (UAM) has increased interest in developing air taxi systems. However, autonomous vehicles will require highly complex systems to make decisions, such as ensuring safe flight and landing, as well as managing the separation between air vehicles, which may involve reduced distances compared to current air traffic management (ATM) practices. To achieve full autonomy, advanced algorithms will be necessary to

perform information fusion and agent-to-agent interactions. Research in this area includes corporate multi-agent planning and task completion, UAM path planning, and human-AI teaming, among others. Despite the potential of autonomous flight, the author notes that achieving full autonomy still has a long way to go considering the difficulties of earning the public trust of commercial air transport systems and meeting the aviation standards, and will require a collaborative effort from multiple stakeholders, including manufacturers, operators, and regulatory agencies.

6.2.2 AI Model Trustworthiness

Safety and Risk Management Safety and risk investigations are critical for ensuring the safe and reliable operation of autonomous systems that use AI models. These investigations aim to evaluate, detect, and monitor the risks associated with AI models in real-time to minimize potential harm. To mitigate these risks, extensive testing is necessary to ensure that the AI models are at least as safe as traditional airborne systems that use existing safety assessment tools. Regulatory agencies are currently working on developing standards for testing and validating AI-based air traffic management (ATM) systems. Research into real-time risk assessment and detection, risk control techniques such as event classification, information leaking risks, AI model performance metrics, and off-nominal event risk mitigation are essential areas to investigate. Additionally, the uncertainties involved in the risk assessment step cannot be ignored. Probabilistic machine learning can be highly useful in these situations, leading to research topics such as identifying sources of uncertainty, reducing variance, and estimating failure probabilities.

Verification and Validation Verification and validation (V&V) are crucial processes to guarantee that any system satisfies regulatory standards and design requirements. In the field of critical engineering systems, such as flight control systems, formal methods play a vital role in ensuring safety, reliability, and performance. Three significant branches of for-

mal methods, namely model checking (exhaustive testing), static analysis (error analysis), and theorem proving (logical inference), require exploration to enhance the verification and validation of these systems. Mathematical developments, such as advanced sampling algorithms and temporal logic, are promising directions.

Aviation Cybersecurity Aviation cybersecurity involves securing both ground-based and airborne systems, which include communication networks, navigation systems, air traffic control, flight control systems, and avionics. Integrated space-air-ground ATM systems rely on communication, which is prone to malicious activities. Information security assurance has been a research area for decades, and game theory provides insights into effective security measures by modeling adversarial interactions. Threats such as ADS-B false data injection, data tempering, location information spoofing, and DDoS attacks can affect both AI model training and inference processes, necessitating extensive data transmission. Protection techniques on data access such as firewalls and Intrusion Detection and Prevention Systems (IDPS) can enhance data integrity. Federated learning is a promising technique for collaboratively training AI models, allowing each party to train the model on their local server, reducing data transmission needs, and associated cybersecurity concerns. On the model inference side, adversarial robustness is critical to defend against unforeseen information security threats. Bayesian neural networks have shown excellent robustness against attacks and have the potential to defend against adversarial attacks. Possible research directions include data access protection, federated learning, adversarial robustness, and Bayesian neural networks.

6.3 Closure

AI is a fast-evolving technique, that demands constant awareness and in-depth thinking from highly multidisciplinary researchers. Aviation research is a typical system engineering branch, requiring solid knowledge basis on aerospace, AI, meteorology, remote sensing,

ergonomics, dynamics, and flight operations build upon math and computer engineering skills. While adapting to new technologies can be challenging and push individuals out of their comfort zones, lifelong learning is an essential principle that every researcher should embrace. As the aviation industry continues to evolve and become more complex, it will require a group of open-minded, outstanding, and multidisciplinary researchers to integrate AI technologies and pave the way for safer, more efficient, and sustainable practices. With continued investment in research and development along the track, we can unlock the full potential of AI for aviation and create a brighter future for the aviation industry and human beings. I firmly believe that this thesis has provided valuable insights that can inform and guide future research in this area. I hope that my work can serve as an initiative for further exploration and contribute to the technological advancement in the evolution of aviation digitalization.

REFERENCES

[1] H. Erzberger, T. Nikoleris, R. A. Paielli, Y.-C. Chu, Algorithms for control of arrival and departure traffic in terminal airspace, Proceedings of the Institution of Mechanical Engineers, Part G: Journal of aerospace engineering 230 (9) (2016) 1762–1779.

[2] FAA, FAA ORDER JO 7360.1, https://www.faa.gov/documentlibrary/media/order/order_7360.1.pdf (2019).

[3] H. M. Arneson, P. Hegde, M. E. La Scola, A. D. Evans, R. M. Keller, J. E. Schade, Sherlock data warehouse (2019).

[4] N. Marwan, M. Carmen Romano, M. Thiel, J. Kurths, Recurrence plots for the analysis of complex systems, Physics Reports 438 (5) (2007) 237–329. doi:https://doi.org/10.1016/j.physrep.2006.11.001.
URL https://www.sciencedirect.com/science/article/pii/S0370157306004066

[5] EASA, Easa concept paper: Artificial intelligence roadmap: A human-centric approach to ai in aviation, https://www.easa.europa.eu/en/downloads/109668/en Accessed: 3-20-2023 (2021).

[6] EASA, Easa concept paper: First usable guidance for level 1&2 machine learning applications: A deliverable of the easa ai roadmap, https://www.easa.europa.eu/en/downloads/137631/en Accessed: 3-20-2023 (2023).

[7] S. Mondoloni, N. Rozen, Aircraft trajectory prediction and synchronization for air traffic management applications, Progress in aerospace sciences 119 (2020) 100640.

[8] M. FAA, Nextgen implementation plan, Washington, DC, Available: http://www.faa.gov/nextgen/media/ng2011implementation plan.pdf (accessed March 18, 2011) (2013).

[9] W. Schuster, W. Ochieng, Performance requirements of future trajectory prediction and conflict detection and resolution tools within sesar and nextgen: Framework for the derivation and discussion, Journal of Air Transport Management 35 (2014) 92–101.

[10] W. Liu, I. Hwang, Probabilistic trajectory prediction and conflict detection for air traffic control, Journal of Guidance, Control, and Dynamics 34 (6) (2011) 1779–1789.

[11] E. H. Phillips, Free flight poses multiple challenges, Aviation Week & Space Technology (1996).

[12] R. A. Paielli, H. Erzberger, Conflict probability estimation for free flight, Journal of Guidance, Control, and Dynamics 20 (3) (1997) 588–596.

[13] B. Sridhar, K. S. Sheth, S. Grabbe, Airspace complexity and its application in air traffic management, in: 2nd USA/Europe Air Traffic Management R&D Seminar, 1998, pp. 1–6.

[14] D. Thipphavong, Analysis of climb trajectory modeling for separation assurance automation, in: AIAA Guidance, Navigation and Control Conference and Exhibit, 2008, p. 7407.

[15] D. Knorr, L. Walter, Trajectory uncertainty and the impact on sector complexity and workload, SESAR Innovation Days 29 (2011).

[16] A. Nilim, L. El Ghaoui, V. Duong, M. Hansen, Trajectory-based air traffic management (tb-atm) under weather uncertainty, in: Proc. of the Fourth International Air Traffic Management R&D Seminar ATM, 2001.

[17] A. Lee, S. Weygandt, B. Schwartz, J. Murphy, Performance of trajectory models with wind uncertainty, in: AIAA modeling and simulation technologies conference, 2009, p. 5834.

[18] S. Mondoloni, A multiple-scale model of wind-prediction uncertainty and application to trajectory prediction, in: 6th AIAA Aviation Technology, Integration and Operations Conference (ATIO), 2006, p. 7807.

[19] S. Ayhan, H. Samet, Aircraft trajectory prediction made easy with predictive analytics, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 21–30.

[20] H. M. Arneson, Sherlock data warehouse (2018).

[21] D. G. Jones, M. R. Endsley, Sources of situation awareness errors in aviation., Aviation, space, and environmental medicine (1996).

[22] J. Zhang, F.-Y. Wang, K. Wang, W.-H. Lin, X. Xu, C. Chen, Data-driven intelligent transportation systems: A survey, IEEE Transactions on Intelligent Transportation Systems 12 (4) (2011) 1624–1639.

[23] K. He, G. Gkioxari, P. Dollár, R. Girshick, Mask r-cnn, in: Proceedings of the IEEE international conference on computer vision, 2017, pp. 2961–2969.

[24] C. D. Manning, C. D. Manning, H. Schütze, Foundations of statistical natural language processing, MIT press, 1999.

[25] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, P. Kuksa, Natural language processing (almost) from scratch, Journal of machine learning research 12 (Aug) (2011) 2493–2537.

[26] C. Hong, J. Yu, J. Wan, D. Tao, M. Wang, Multimodal deep autoencoder for human pose recovery, IEEE Transactions on Image Processing 24 (12) (2015) 5659–5670.

[27] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, in: Advances in neural information processing systems, 2014, pp. 2672–2680.

[28] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al., Human-level control through deep reinforcement learning, Nature 518 (7540) (2015) 529.

[29] L. Zhu, F. R. Yu, Y. Wang, B. Ning, T. Tang, Big data analytics in intelligent transportation systems: A survey, IEEE Transactions on Intelligent Transportation Systems 20 (1) (2018) 383–398.

[30] Y. Gal, Z. Ghahramani, Dropout as a bayesian approximation: Representing model uncertainty in deep learning, in: international conference on machine learning, 2016, pp. 1050–1059.

[31] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, The Journal of Machine Learning Research 15 (1) (2014) 1929–1958.

[32] P. Lauret, E. Fock, R. N. Randrianarivony, J.-F. Manicom-Ramsamy, Bayesian neural network approach to short time load forecasting, Energy conversion and management 49 (5) (2008) 1156–1166.

[33] J. Wiest, M. Höffken, U. Kreßel, K. Dietmayer, Probabilistic trajectory prediction with gaussian mixture models, in: 2012 IEEE Intelligent Vehicles Symposium, IEEE, 2012, pp. 141–146.

[34] A. Houenou, P. Bonnifait, V. Cherfaoui, W. Yao, Vehicle trajectory prediction based on motion model and maneuver recognition, in: 2013 IEEE/RSJ international conference on intelligent robots and systems, IEEE, 2013, pp. 4363–4369.

[35] S. Ammoun, F. Nashashibi, Real time trajectory prediction for collision risk estimation between vehicles, in: 2009 IEEE 5th International Conference on Intelligent Computer Communication and Processing, IEEE, 2009, pp. 417–422.

[36] B. Kim, C. M. Kang, J. Kim, S. H. Lee, C. C. Chung, J. W. Choi, Probabilistic vehicle trajectory prediction over occupancy grid map via recurrent neural network, in: 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC), IEEE, 2017, pp. 399–404.

[37] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, S. Savarese, Social lstm: Human trajectory prediction in crowded spaces, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 961–971.

[38] H. Xue, D. Q. Huynh, M. Reynolds, Ss-lstm: A hierarchical lstm model for pedestrian trajectory prediction, in: 2018 IEEE Winter Conference on Applications of Computer Vision (WACV), IEEE, 2018, pp. 1186–1194.

[39] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, A. Alahi, Social gan: Socially acceptable trajectories with generative adversarial networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 2255–2264.

[40] B. D. Ziebart, N. Ratliff, G. Gallagher, C. Mertz, K. Peterson, J. A. Bagnell, M. Hebert, A. K. Dey, S. Srinivasa, Planning-based prediction for pedestrians, in: 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2009, pp. 3931–3936.

[41] H. Ryan, M. Paglione, State vector based near term trajectory prediction, in: AIAA Guidance, Navigation and Control Conference and Exhibit, 2008, p. 7144.

[42] I. Lymperopoulos, J. Lygeros, Sequential monte carlo methods for multi-aircraft trajectory prediction in air traffic management, International Journal of Adaptive Control and Signal Processing 24 (10) (2010) 830–849.

[43] Y. Le Fablec, J.-M. Alliot, Using neural networks to predict aircraft trajectories., in: IC-AI, 1999, pp. 524–529.

[44] Y. Yu, H. Yao, Y. Liu, Aircraft dynamics simulation using a novel physics-based learning method, Aerospace Science and Technology 87 (2019) 254–264.

[45] J. V. Benavides, J. Kaneshige, S. Sharma, R. Panda, M. Steglinski, Implementation of a trajectory prediction function for trajectory based operations, in: AIAA Atmospheric Flight Mechanics Conference, 2014, p. 2198.

[46] G. Avanzini, Frenet-based algorithm for trajectory prediction, Journal of guidance, control, and dynamics 27 (1) (2004) 127–135.

[47] I. Lymperopoulos, J. Lygeros, A. Lecchini, Model based aircraft trajectory prediction during takeoff, in: AIAA Guidance, Navigation, and Control Conference and Exhibit, 2006, p. 6098.

[48] J. L. Yepes, I. Hwang, M. Rotea, New algorithms for aircraft intent inference and trajectory prediction, Journal of guidance, control, and dynamics 30 (2) (2007) 370–382.

[49] J. A. Besada, G. Frontera, J. Crespo, E. Casado, J. Lopez-Leones, Automated aircraft trajectory prediction based on formal intent-related language processing, IEEE Transactions on Intelligent Transportation Systems 14 (3) (2013) 1067–1082.

[50] Y. Wang, Y. Pang, S. Gorceski, P. Kostiuk, M. T. Mohen, P. K. Menon, Y. Liu, A voice communication-augmented simulation framework for aircraft trajectory simulation, IEEE Transactions on Intelligent Transportation Systems (2021).

[51] J. Bronsvoort, G. McDonald, J. Lopez-Leones, H. Visser, Improved trajectory prediction for air traffic management by simulation of guidance logic and inferred aircraft intent using existing data-link technology, in: AIAA Guidance, Navigation, and Control Conference, 2012, p. 4928.

[52] E. Crisostomi, A. Lecchini-Visintini, J. Maciejowski, Combining monte carlo and worst-case methods for trajectory prediction in air traffic control: a case study, Journal of Guidance, Control and Dynamics, submitted (2008).

[53] E. Casado, C. Goodchild, M. Vilaplana, Identification and initial characterization of sources of uncertainty affecting the performance of future trajectory management automation systems, in: Proceedings of the 2nd International Conference on Application and Theory of Automation in Command and Control Systems, 2012, pp. 170–175.

[54] S. Qiao, D. Shen, X. Wang, N. Han, W. Zhu, A self-adaptive parameter selection trajectory prediction approach via hidden markov models, IEEE Transactions on Intelligent Transportation Systems 16 (1) (2014) 284–296.

[55] S. Swierstra, S. Green, Common trajectory prediction capability for decision support tools, in: 5th USA/Eurocontrol ATM R&D Seminar, 2003, pp. 23–27.

[56] T. Mueller, J. Sorensen, G. Couluris, Strategic aircraft trajectory prediction uncertainty and statistical sector traffic load modeling, in: AIAA Guidance, Navigation, and Control Conference and Exhibit, 2002, p. 4765.

[57] M. Prandini, J. Hu, J. Lygeros, S. Sastry, A probabilistic approach to aircraft conflict detection, IEEE Transactions on intelligent transportation systems 1 (4) (2000) 199–220.

[58] J. Lygeros, M. Prandini, Aircraft and weather models for probabilistic collision avoidance in air traffic control, in: Proceedings of the 41st IEEE Conference on Decision and Control, 2002., Vol. 3, IEEE, 2002, pp. 2427–2432.

[59] S. Mondoloni, A genetic algorithm for determining optimal flight trajectories, in: Guidance, Navigation, and Control Conference and Exhibit, 1998, p. 4476.

[60] H. K. Ng, S. Grabbe, A. Mukherjee, Design and evaluation of a dynamic programming flight routing algorithm using the convective weather avoidance model, in: AIAA guidance, navigation, and control conference, 2009, p. 5862.

[61] T. Stewart, L. Askey, M. Hokit, A concept for tactical reroute generation, evaluation and coordination, in: 12th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference and 14th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, 2012, p. 5586.

[62] T. Stewart, J. DeArmon, D. Chaloux, Using flight information to improve weather avoidance predictions, in: 2013 Aviation Technology, Integration, and Operations Conference, 2013, p. 4377.

[63] D. McNally, K. Sheth, C. Gong, J. Love, C. H. Lee, S. Sahlman, J. Cheng, Dynamic weather routes: a weather avoidance system for near-term trajectory-based operations, in: 28th International Congress of the Aeronautical Sciences, 2012, pp. 23–28.

[64] H. Erzberger, T. A. Lauderdale, Y.-C. Chu, Automated conflict resolution, arrival management and weather avoidance for atm (2010).

[65] H. Erzberger, T. Lauderdale, Y. Chu, Automated conflict resolution, arrival management, and weather avoidance for air traffic management, Proceedings of the Institution of Mechanical Engineers, Part G: Journal of aerospace engineering 226 (8) (2012) 930–949.

[66] A. De Leege, M. van Paassen, M. Mulder, A machine learning approach to trajectory prediction, in: AIAA Guidance, Navigation, and Control (GNC) Conference, 2013, p. 4782.

[67] X. Zhang, S. Mahadevan, Bayesian neural networks for flight trajectory prediction and safety assessment, Decision Support Systems (2020) 113246.

[68] Y. Liu, M. Hansen, Predicting aircraft trajectories: a deep generative convolutional recurrent neural networks approach, arXiv preprint arXiv:1812.11670 (2018).

[69] Y. Pang, Y. Liu, Conditional generative adversarial networks (cgan) for aircraft trajectory prediction considering weather effects, in: AIAA Scitech 2020 Forum, 2020, p. 1853.

[70] Y. Pang, Y. Liu, Probabilistic aircraft trajectory prediction considering weather uncertainties using dropout as bayesian approximate variational inference, in: AIAA Scitech 2020 Forum, 2020, p. 1413.

[71] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural computation 9 (8) (1997) 1735–1780.

[72] J. Chung, C. Gulcehre, K. Cho, Y. Bengio, Empirical evaluation of gated recurrent neural networks on sequence modeling, arXiv preprint arXiv:1412.3555 (2014).

[73] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, in: Advances in neural information processing systems, 2012, pp. 1097–1105.

[74] S. Xingjian, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, W.-c. Woo, Convolutional lstm network: A machine learning approach for precipitation nowcasting, in: Advances in neural information processing systems, 2015, pp. 802–810.

[75] F. Burden, D. Winkler, Bayesian regularization of neural networks, in: Artificial neural networks, Springer, 2008, pp. 23–42.

[76] Y. Gal, Uncertainty in deep learning, Ph.D. thesis, PhD thesis, University of Cambridge (2016).

[77] C. Blundell, J. Cornebise, K. Kavukcuoglu, D. Wierstra, Weight uncertainty in neural networks, arXiv preprint arXiv:1505.05424 (2015).

[78] R. M. Neal, Bayesian learning for neural networks, Vol. 118, Springer Science & Business Media, 2012.

[79] S. Kullback, R. A. Leibler, On information and sufficiency, The annals of mathematical statistics 22 (1) (1951) 79–86.

[80] D. M. Blei, A. Kucukelbir, J. D. McAuliffe, Variational inference: A review for statisticians, Journal of the American statistical Association 112 (518) (2017) 859–877.

[81] Y. Wen, P. Vicol, J. Ba, D. Tran, R. Grosse, Flipout: Efficient pseudo-independent weight perturbations on mini-batches, arXiv preprint arXiv:1803.04386 (2018).

[82] Y. Gal, Z. Ghahramani, Bayesian convolutional neural networks with bernoulli approximate variational inference, arXiv preprint arXiv:1506.02158 (2015).

[83] Y. Gal, Z. Ghahramani, A theoretically grounded application of dropout in recurrent neural networks, in: Advances in neural information processing systems, 2016, pp. 1019–1027.

[84] T. Pearce, M. Zaki, A. Brintrup, N. Anastassacos, A. Neely, Uncertainty in neural networks: Bayesian ensembling, stat 1050 (2018) 12.

[85] J. Hron, A. G. d. G. Matthews, Z. Ghahramani, Variational bayesian dropout: pitfalls and fixes, arXiv preprint arXiv:1807.01969 (2018).

[86] I. Osband, J. Aslanides, A. Cassirer, Randomized prior functions for deep reinforcement learning, in: Advances in Neural Information Processing Systems, 2018, pp. 8617–8629.

[87] G. Melis, C. Blundell, T. Kočiskỳ, K. M. Hermann, C. Dyer, P. Blunsom, Pushing the bounds of dropout, arXiv preprint arXiv:1805.09208 (2018).

[88] D. P. Kingma, T. Salimans, M. Welling, Variational dropout and the local reparameterization trick, in: Advances in neural information processing systems, 2015, pp. 2575–2583.

[89] J. V. Dillon, I. Langmore, D. Tran, E. Brevdo, S. Vasudevan, D. Moore, B. Patton, A. Alemi, M. Hoffman, R. A. Saurous, Tensorflow distributions, arXiv preprint arXiv:1711.10604 (2017).

[90] D. Tran, A. Kucukelbir, A. B. Dieng, M. Rudolph, D. Liang, D. M. Blei, Edward: A library for probabilistic modeling, inference, and criticism, arXiv preprint arXiv:1610.09787 (2016).

[91] D. Tran, M. W. Hoffman, D. Moore, C. Suter, S. Vasudevan, A. Radul, Simple, distributed, and accelerated probabilistic programming, in: Advances in Neural Information Processing Systems, 2018, pp. 7598–7609.

[92] D. Tran, M. Dusenberry, M. van der Wilk, D. Hafner, Bayesian layers: A module for neural network uncertainty, in: Advances in Neural Information Processing Systems, 2019, pp. 14633–14645.

[93] M. M. Eshow, M. Lui, S. Ranjan, Architecture and capabilities of a data warehouse for atm research, in: 2014 IEEE/AIAA 33rd Digital Avionics Systems Conference (DASC), IEEE, 2014, pp. 1E3–1.

[94] D. Klingle-Wilson, J. Evans, Description of the corridor integrated weather system (ciws) weather products, Project Report ATC-317, MIT Lincoln Laboratory, Lexington, MA (2005).

[95] Y. Pang, H. Yao, J. Hu, Y. Liu, A recurrent neural network approach for aircraft trajectory prediction with weather features from sherlock, in: AIAA Aviation 2019 Forum, 2019, p. 3413.

[96] R. DeLaura, M. Robinson, M. Pawlak, J. Evans, Modeling convective weather avoidance in enroute airspace, in: 13th Conference on Aviation, Range, and Aerospace Meteorology, AMS, New Orleans, LA, Citeseer, 2008.

[97] I. Sutskever, O. Vinyals, Q. V. Le, Sequence to sequence learning with neural networks, in: Advances in neural information processing systems, 2014, pp. 3104–3112.

[98] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980 (2014).

[99] C. Yu, X. Ma, J. Ren, H. Zhao, S. Yi, Spatio-temporal graph transformer networks for pedestrian trajectory prediction, in: European Conference on Computer Vision, Springer, 2020, pp. 507–523.

[100] FAA, Air traffic by the numbers, `https://www.faa.gov/air_traffic/by_the_numbers/media/Air_Traffic_by_the_Numbers_2020.pdf` Accessed: 1-19-2021 (2020).

[101] A. Majumdar, J. Polak, Estimating capacity of europe's airspace using a simulation model of air traffic controller workload, Transportation Research Record 1744 (1) (2001) 30–43.

[102] G. Mercado-Velasco, M. Mulder, M. Van Paassen, Analysis of air traffic controller workload reduction based on the solution space for the merging task, in: AIAA Guidance, Navigation, and Control Conference, 2010, p. 7541.

[103] Y. Pang, X. Zhao, H. Yan, Y. Liu, Data-driven trajectory prediction with weather uncertainties: A bayesian deep learning approach, Transportation Research Part C: Emerging Technologies 130 (2021) 103326.

[104] P. Zhang, W. Ouyang, P. Zhang, J. Xue, N. Zheng, Sr-lstm: State refinement for lstm towards pedestrian trajectory prediction, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 12085–12094.

[105] Y. Huang, H. Bi, Z. Li, T. Mao, Z. Wang, Stgat: Modeling spatial-temporal interactions for human trajectory prediction, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 6272–6281.

[106] D. Helbing, P. Molnar, Social force model for pedestrian dynamics, Physical review E 51 (5) (1995) 4282.

[107] D. Helbing, L. Buzna, A. Johansson, T. Werner, Self-organized pedestrian crowd dynamics: Experiments, simulations, and design solutions, Transportation science 39 (1) (2005) 1–24.

[108] B. Ivanovic, M. Pavone, The trajectron: Probabilistic multi-agent trajectory modeling with dynamic spatiotemporal graphs, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 2375–2384.

[109] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, in: Advances in neural information processing systems, 2017, pp. 5998–6008.

[110] M. Brittain, P. Wei, Scalable autonomous separation assurance with heterogeneous multi-agent reinforcement learning, IEEE Transactions on Automation Science and Engineering (2022).

[111] J. Hu, X. Yang, W. Wang, P. Wei, L. Ying, Y. Liu, Obstacle avoidance for uas in continuous action space using deep reinforcement learning, arXiv preprint arXiv:2111.07037 (2021).

[112] A. M. Karimi, Y. Wu, M. Koyuturk, R. H. French, Spatiotemporal graph neural network for performance prediction of photovoltaic power systems, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 35, 2021, pp. 15323–15330.

[113] L. Zhao, Y. Song, C. Zhang, Y. Liu, P. Wang, T. Lin, M. Deng, H. Li, T-gcn: A temporal graph convolutional network for traffic prediction, IEEE Transactions on Intelligent Transportation Systems 21 (9) (2019) 3848–3858.

[114] Z. Cui, K. Henrickson, R. Ke, Y. Wang, Traffic graph convolutional recurrent neural network: A deep learning framework for network-scale traffic learning and forecasting, IEEE Transactions on Intelligent Transportation Systems 21 (11) (2019) 4883–4894.

[115] L. Fan, X. Jiajun, L. Xuhui, B. Hongkui, T. Xiansi, Hypersonic vehicle trajectory prediction algorithm based on hough transform, Chinese Journal of Electronics 30 (5) (2021) 918–930.

[116] W. Harlin, D. A. Cicci, Ballistic missile trajectory prediction using a state transition matrix, Applied mathematics and computation 188 (2) (2007) 1832–1847.

[117] Q. Wang, Z. Zhang, Z. Wang, Y. Wang, W. Zhou, The trajectory prediction of spacecraft by grey method, Measurement Science and Technology 27 (8) (2016) 085011.

[118] Y. Hu, C. Gao, J. Li, W. Jing, Z. Li, Novel trajectory prediction algorithms for hypersonic gliding vehicles based on maneuver mode on-line identification and intent inference, Measurement Science and Technology 32 (11) (2021) 115012.

[119] Z. Zhao, H. Fang, Z. Jin, Q. Qiu, Gisnet: Graph-based information sharing network for vehicle trajectory prediction, in: 2020 International Joint Conference on Neural Networks (IJCNN), IEEE, 2020, pp. 1–7.

[120] T. Zhao, G. Chen, X. Wang, E. Yong, W. Qian, Aerodynamic modeling using an end-to-end learning attitude dynamics network for flight control, Acta Mechanica Sinica (2021) 1.

[121] E. Cesario, C. Comito, D. Talia, An approach for the discovery and validation of urban mobility patterns, Pervasive and Mobile Computing 42 (2017) 77–92.

[122] Z. Xu, W. Zeng, X. Chu, P. Cao, Multi-aircraft trajectory collaborative prediction based on social long short-term memory network, Aerospace 8 (4) (2021) 115.

[123] FAA, Faa order jo 7110.65, `https://www.faa.gov/documentLibrary/media/Order/7110.65Y.pdf` Accessed: 6-4-2022 (2019).

[124] M. Kuderer, H. Kretzschmar, C. Sprunk, W. Burgard, Feature-based prediction of trajectories for socially compliant navigation., in: Robotics: science and systems, 2012.

[125] Y. Luo, P. Cai, A. Bera, D. Hsu, W. S. Lee, D. Manocha, Porca: Modeling and planning for autonomous driving among many pedestrians, IEEE Robotics and Automation Letters 3 (4) (2018) 3418–3425.

[126] J. Van Den Berg, S. J. Guy, M. Lin, D. Manocha, Reciprocal n-body collision avoidance, in: Robotics research, Springer, 2011, pp. 3–19.

[127] S. Yi, H. Li, X. Wang, Pedestrian behavior understanding and prediction with deep neural networks, in: European Conference on Computer Vision, Springer, 2016, pp. 263–279.

[128] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, et al., Language models are unsupervised multitask learners, OpenAI blog 1 (8) (2019) 9.

[129] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al., Language models are few-shot learners, arXiv preprint arXiv:2005.14165 (2020).

[130] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, arXiv preprint arXiv:1810.04805 (2018).

[131] M.-T. Luong, H. Pham, C. D. Manning, Effective approaches to attention-based neural machine translation, arXiv preprint arXiv:1508.04025 (2015).

[132] A. Graves, G. Wayne, I. Danihelka, Neural turing machines, arXiv preprint arXiv:1410.5401 (2014).

[133] X. Li, X. Ying, M. C. Chuah, Grip: Graph-based interaction-aware trajectory prediction, in: 2019 IEEE Intelligent Transportation Systems Conference (ITSC), IEEE, 2019, pp. 3960–3966.

[134] A. Mohamed, K. Qian, M. Elhoseiny, C. Claudel, Social-stgcnn: A social spatio-temporal graph convolutional neural network for human trajectory prediction, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 14424–14432.

[135] T. N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, arXiv preprint arXiv:1609.02907 (2016).

[136] J. Zhang, X. Shi, J. Xie, H. Ma, I. King, D.-Y. Yeung, Gaan: Gated attention networks for learning on large and spatiotemporal graphs, arXiv preprint arXiv:1803.07294 (2018).

[137] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio, Graph attention networks, arXiv preprint arXiv:1710.10903 (2017).

[138] B. Lim, S. Ö. Arık, N. Loeff, T. Pfister, Temporal fusion transformers for interpretable multi-horizon time series forecasting, International Journal of Forecasting (2021).

[139] C. C. Robusto, The cosine-haversine formula, The American Mathematical Monthly 64 (1) (1957) 38–40.

[140] Y. Pang, S. Cheng, J. Hu, Y. Liu, Evaluating the robustness of bayesian neural networks against different types of attacks, arXiv preprint arXiv:2106.09223 (2021).

[141] A. Vemula, K. Muelling, J. Oh, Social attention: Modeling attention in human crowds, in: 2018 IEEE international Conference on Robotics and Automation (ICRA), IEEE, 2018, pp. 4601–4607.

[142] Y. Ma, X. Zhu, S. Zhang, R. Yang, W. Wang, D. Manocha, Trafficpredict: Trajectory prediction for heterogeneous traffic-agents, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 33, 2019, pp. 6120–6127.

[143] B. Ivanovic, M. Pavone, The trajectron: Probabilistic multi-agent trajectory modeling with dynamic spatiotemporal graphs, in: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), 2019.

[144] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, A. Lerchner, beta-vae: Learning basic visual concepts with a constrained variational framework (2016).

[145] J. Yu, J. Wu, M. Sarwat, Geospark: A cluster computing framework for processing large-scale spatial data, in: Proceedings of the 23rd SIGSPATIAL international conference on advances in geographic information systems, 2015, pp. 1–4.

[146] C. Comito, D. Talia, Gdis: A service-based architecture for data integration on grids, in: OTM Confederated International Conferences" On the Move to Meaningful Internet Systems", Springer, 2004, pp. 88–98.

[147] C. Tang, R. R. Salakhutdinov, Multiple futures prediction, Advances in Neural Information Processing Systems 32 (2019).

[148] H. Vanholder, Efficient inference with tensorrt, in: GPU Technology Conference, Vol. 1, 2016, p. 2.

[149] I. Dhief, Z. Wang, M. Liang, S. Alam, M. Schultz, D. Delahaye, Predicting aircraft landing time in extended-tma using machine learning methods, in: ICRAT 2020, 9th International Conference for Research in Air Transportation, 2020.

[150] I. Vlachos, Z. Lin, Drivers of airline loyalty: Evidence from the business travelers in china, Transportation Research Part E: Logistics and Transportation Review 71 (2014) 1–17.

[151] FAA-APO-100, Cost of delay estimates, https://www.faa.gov/data_research/aviation_data_statistics/media/cost_delay_estimates.pdf Accesed: 05-15-2020 (2019).

[152] M. Jetzki, The propagation of air transport delays in europe, Master's thesis, RWTH Aachen University, Airport and Air Transportation Research (2009).

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

[154] S. Loft, P. Sanderson, A. Neal, M. Mooij, Modeling and predicting mental workload in en route air traffic control: Critical review and broader implications, Human factors 49 (3) (2007) 376–399.

[155] M. Micallef, K. Chircop, D. Zammit-Mangion, A. Sammut, Revised approach procedures to support optimal descents into malta international airport, CEAS Aeronautical Journal 5 (4) 461–475. doi:10.1007/s13272-014-0119-y.

[156] T. Bolić, P. Ravenhill, Sesar: The past, present, and future of european air traffic management research, Engineering 7 (4) (2021) 448–451.

[157] D. Huet, D. Booth, S. Pickup, A-cdm impact assessment-final report, EUROCONTROL, March (2016).

[158] A. Ging, S. Engelland, A. Capps, M. Eshow, Y. Jung, S. Sharma, E. Talebi, M. Downs, C. Freedman, T. Ngo, et al., Airspace technology demonstration 2 (atd-2) technology description document (tdd), Tech. rep. (2018).

[159] M. Sama, A. D'Ariano, P. D'Ariano, D. Pacciarelli, Optimal aircraft scheduling and routing at a terminal control area during disturbances, Transportation Research Part C: Emerging Technologies 47 (2014) 61–85.

[160] S.-P. Yu, X.-B. Cao, J. Zhang, A real-time schedule method for aircraft landing scheduling problem based on cellular automation, Applied Soft Computing 11 (4) (2011) 3485–3493.

[161] J. A. Bennell, M. Mesgarpour, C. N. Potts, Dynamic scheduling of aircraft landings, European Journal of Operational Research 258 (1) (2017) 315–327.

[162] A. T. Ernst, M. Krishnamoorthy, R. H. Storer, Heuristic and exact algorithms for scheduling aircraft landings, Networks: An International Journal 34 (3) (1999) 229–241.

[163] J. E. Beasley, M. Krishnamoorthy, Y. M. Sharaiha, D. Abramson, Scheduling aircraft landings—the static case, Transportation science 34 (2) (2000) 180–197.

[164] H. Pinol, J. E. Beasley, Scatter search and bionomic algorithms for the aircraft landing problem, European Journal of Operational Research 171 (2) (2006) 439–462.

[165] J. E. Beasley, J. Sonander, P. Havelock, Scheduling aircraft landings at london heathrow using a population heuristic, Journal of the operational Research Society 52 (5) (2001) 483–493.

[166] L. Bianco, P. Dell'Olmo, S. Giordani, Scheduling models for air traffic control in terminal areas, Journal of Scheduling 9 (3) (2006) 223–253.

[167] N. Zufferey, M.-S. Vié, R. Leus, Local search for aircraft-landing planning, in: Optimization in Artificial Intelligence and Data Sciences, Springer, 2022, pp. 163–171.

[168] L. Bianco, P. Dell'Olmo, S. Giordani, Minimizing total completion time subject to release dates and sequence-dependentprocessing times, Annals of Operations Research 86 (1999) 393–415.

[169] J. A. Atkin, E. K. Burke, J. S. Greenwood, D. Reeson, Hybrid metaheuristics to aid runway scheduling at london heathrow airport, Transportation Science 41 (1) (2007) 90–106.

[170] A. Wided, B. Fatimaa, Effective simulation-based optimization algorithm for the aircraft runway scheduling problem (2022).

[171] J. E. Beasley, M. Krishnamoorthy, Y. M. Sharaiha, D. Abramson, Displacement problem and dynamically scheduling aircraft landings, Journal of the operational research society 55 (1) (2004) 54–64.

[172] N. Boysen, M. Fliedner, Scheduling aircraft landings to balance workload of ground staff, Computers & Industrial Engineering 60 (2) (2011) 206–217.

[173] M. J. Soomer, G. J. Franx, Scheduling aircraft landings using airlines' preferences, European Journal of Operational Research 190 (1) (2008) 277–291.

[174] W. Zeng, X. Chu, Z. Xu, Y. Liu, Z. Quan, Aircraft 4d trajectory prediction in civil aviation: A review, Aerospace 9 (2) (2022) 91.

[175] Y. Pang, X. Zhao, J. Hu, H. Yan, Y. Liu, Bayesian spatio-temporal graph transformer network (b-star) for multi-aircraft trajectory prediction, Knowledge-Based Systems (2022) 108998.

[176] P. N. Tran, H. Q. Nguyen, D.-T. Pham, S. Alam, Aircraft trajectory prediction with enriched intent using encoder-decoder architecture, IEEE Access 10 (2022) 17881–17896.

[177] D. Sculley, G. Holt, D. Golovin, E. Davydov, T. Phillips, D. Ebner, V. Chaudhary, M. Young, J.-F. Crespo, D. Dennison, Hidden technical debt in machine learning systems, Advances in neural information processing systems 28 (2015).

[178] V. Ciesielski, P. Scerri, Real time genetic scheduling of aircraft landing times, in: 1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No. 98TH8360), IEEE, 1998, pp. 360–364.

[179] X. Bai, L. A. Weitz, S. Priess, Evaluating the impact of estimated time of arrival accuracy on interval management performance, in: AIAA Guidance, Navigation, and Control Conference. arXiv:https://arc.aiaa.org/doi/pdf/10.2514/6.2016-1852, doi:10.2514/6.2016-1852.
URL https://arc.aiaa.org/doi/abs/10.2514/6.2016-1852

[180] J. Krozel, C. Lee, J. Mitchell, Estimating time of arrival in heavy weather conditions, in: Guidance, Navigation, and Control Conference and Exhibit, 1999, p. 4232.

[181] K. Roy, B. Levy, C. Tomlin, Target tracking and estimated time of arrival (eta) prediction for arrival aircraft, in: AIAA Guidance, Navigation, and Control Conference and Exhibit, 2006, p. 6324. doi:10.2514/6.2006-6324.
URL https://arc.aiaa.org/doi/abs/10.2514/6.2006-6324

[182] H. Huang, K. Roy, C. Tomlin, Probabilistic estimation of state-dependent hybrid mode transitions for aircraft arrival time prediction, in: AIAA Guidance, Navigation and Control Conference and Exhibit, 2007, p. 6695. doi:10.2514/6.2007-6695.
URL https://arc.aiaa.org/doi/abs/10.2514/6.2007-6695

[183] J. Wei, J. Lee, I. Hwang, Estimated time of arrival prediction based on state-dependent transition hybrid estimation algorithm, in: AIAA Guidance, Navigation, and Control Conference, 2015, pp. 1081–1094. arXiv:https://arc.aiaa.org/doi/pdf/10.2514/6.2007-6695, doi:10.2514/6.2007-6695.
URL https://arc.aiaa.org/doi/abs/10.2514/6.2007-6695

[184] M. Porretta, M.-D. Dupuy, W. Schuster, A. Majumdar, W. Ochieng, Performance evaluation of a novel 4d trajectory prediction model for civil aircraft, The Journal of Navigation 61 (3) (2008) 393. doi:10.1017/S0373463308004761.

[185] L. Xi, Z. Jun, Z. Yanbo, L. Wei, Simulation study of algorithms for aircraft trajectory prediction based on ads-b technology, in: 2008 Asia Simulation Conference-7th International Conference on System Simulation and Scientific Computing, IEEE, 2008, pp. 322–327. doi:10.1109/ASC-ICSC.2008.4675378.

[186] J. V. Benavides, J. Kaneshige, S. Sharma, R. Panda, M. Steglinski, Implementation of a trajectory prediction function for trajectory based operations, in: AIAA Atmospheric Flight Mechanics Conference, 2014, p. 2198. arXiv:https://arc.aiaa.org/doi/pdf/10.2514/6.2014-2198, doi:10.2514/6.2014-2198.
URL https://arc.aiaa.org/doi/abs/10.2514/6.2014-2198

[187] J. J. Rebollo, H. Balakrishnan, A network-based model for predicting air traffic delays, in: 5th International Conference on Research in Air Transportation, 2012, pp. 22–25.

[188] F. Liu, J. Sun, M. Liu, J. Yang, G. Gui, Generalized flight delay prediction method using gradient boosting decision tree, in: 2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring), IEEE, 2020, pp. 1–5.

[189] S. Manna, S. Biswas, R. Kundu, S. Rakshit, P. Gupta, S. Barman, A statistical approach to predict flight delay using gradient boosted decision tree, in: 2017 International conference on computational intelligence in data science (ICCIDS), IEEE, 2017, pp. 1–5.

[190] N. Chakrabarty, T. Kundu, S. Dandapat, A. Sarkar, D. K. Kole, Flight arrival delay prediction using gradient boosting classifier, in: Emerging technologies in data mining and information security, Springer, 2019, pp. 651–659.

[191] Y. Glina, R. Jordan, M. Ishutkina, A tree-based ensemble method for the prediction and uncertainty quantification of aircraft landing times, in: American Meteorological Society–10th Conference on Aritificial Intelligence Applications to Environmental Science, New Orleans, LA, 2012.

[192] S. Ayhan, P. Costas, H. Samet, Predicting estimated time of arrival for commercial flights, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2018, pp. 33–42. doi:https://doi.org/10.1145/3219819.3219874.

[193] Z. Wang, M. Liang, D. Delahaye, Automated data-driven prediction on aircraft estimated time of arrival, Journal of Air Transport Management 88 (2020) 101840. doi:https://doi.org/10.1016/j.jairtraman.2020.101840.
URL http://www.sciencedirect.com/science/article/pii/S0969699719304429

[194] S. Yua, X. Cao, J. Zhang, A realytime schedule method for aircraft landing scheduling problem based on cellular automaton, Applied Soft Computing 11 (2011) 3485–3493.

[195] A. R. Brentnall, Aircraft arrival management, Ph.D. thesis, University of Southampton (2006).

[196] K. Artiouchine, P. Baptiste, C. Dürr, Runway sequencing with holding patterns, European Journal of Operational Research 189 (3) (2008) 1254–1266.

[197] S. Yu, X. Cao, M. Hu, W. Du, J. Zhang, A real-time schedule method for aircraft landing scheduling problem based on cellular automaton, in: Proceedings of the first ACM/SIGEVO Summit on Genetic and Evolutionary Computation, 2009, pp. 717–724.

[198] Y. Ding, J. Valasek, Aircraft landing scheduling optimization for single runway noncontrolled airports: Static case, Journal of guidance, control, and dynamics 30 (1) (2007) 252–255.

[199] A. Faye, Solving the aircraft landing problem with time discretization approach, European Journal of Operational Research 242 (3) (2015) 1028–1038.

[200] V. Ciesielski, P. Scerri, An anytime algorithm for scheduling of aircraft landing times using genetic algorithms, Australian Journal of Intelligent Information Processing Systems 4 (3/4) (1997) 206–213.

[201] J. Abela, D. Abramson, M. Krishnamoorthy, A. De Silva, G. Mills, Computing optimal schedules for landing aircraft, in: proceedings of the 12th national conference of the Australian Society for Operations Research, Adelaide, Citeseer, 1993, pp. 71–90.

[202] H. Balakrishnan, B. Chandran, Scheduling aircraft landings under constrained position shifting, in: AIAA guidance, navigation, and control conference and exhibit, 2006, p. 6320. doi:https://doi.org/10.2514/6.2006-6320.

[203] I. Farah, A. Kansou, A. Yassine, T. Galinho, Ant colony optimization for aircraft landings, in: 2011 4th international conference on logistics, IEEE, 2011, pp. 235–240.

[204] X.-B. Hu, W.-H. Chen, Genetic algorithm based on receding horizon control for arrival sequencing and scheduling, Engineering applications of artificial intelligence 18 (5) (2005) 633–642.

[205] A. D'Ariano, M. Pistelli, D. Pacciarelli, Aircraft retiming and rerouting in vicinity of airports, IET Intelligent Transport Systems 6 (4) (2012) 433–443.

[206] A. D'Ariano, D. Pacciarelli, M. Pistelli, M. Pranzo, Real-time scheduling of aircraft arrivals and departures in a terminal maneuvering area, Networks 65 (3) (2015) 212–227.

[207] D. Toratani, N. K. Wickramasinghe, E. Itoh, Study on the arrival manager maximizing the benefit of four-dimensional trajectory based operations, in: 31st Congress of the International Council of the Aeronautical Sciences (ICAS), Belo Horizonte, Brazil, 2018.

[208] A. Khassiba, F. Bastin, S. Cafieri, B. Gendron, M. Mongeau, Two-stage stochastic mixed-integer programming with chance constraints for extended aircraft arrival management, Transportation Science 54 (4) (2020) 897–919.

[209] A. Vanwelsenaere, J. Ellerbroek, J. Hoekstra, E. Westerveld, Effect of popup flights on the extended arrival manager, Journal of Air Transportation 26 (2) (2018) 60–69.

[210] A. Khassiba, F. Bastin, B. Gendron, S. Cafieri, M. Mongeau, Extended aircraft arrival management under uncertainty: A computational study, Journal of Air Transportation 27 (3) (2019) 131–143.

[211] J. Friedman, T. Hastie, R. Tibshirani, Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors), The annals of statistics 28 (2) (2000) 337–407.

[212] J. H. Friedman, Greedy function approximation: a gradient boosting machine, Annals of statistics (2001) 1189–1232.

[213] J. H. Friedman, Stochastic gradient boosting, Computational statistics & data analysis 38 (4) (2002) 367–378.

[214] H. Lu, S. P. Karimireddy, N. Ponomareva, V. Mirrokni, Accelerating gradient boosting machines, in: International Conference on Artificial Intelligence and Statistics, PMLR, 2020, pp. 516–526.

[215] M. G. Alharbi, A. Stohy, M. Elhenawy, M. Masoud, H. A. El-Wahed Khalifa, Solving traveling salesman problem with time windows using hybrid pointer networks with time features, Sustainability 13 (22) (2021) 12906.

[216] E. K. Baker, An exact algorithm for the time-constrained traveling salesman problem, Operations Research 31 (5) (1983) 938–945.

[217] N. Christofides, A. Mingozzi, P. Toth, State-space relaxation procedures for the computation of bounds to routing problems, Networks 11 (2) (1981) 145–164.

[218] Y. Dumas, J. Desrosiers, E. Gelinas, M. M. Solomon, An optimal algorithm for the traveling salesman problem with time windows, Operations research 43 (2) (1995) 367–371.

[219] G. Pesant, M. Gendreau, J.-Y. Potvin, J.-M. Rousseau, An exact constraint logic programming algorithm for the traveling salesman problem with time windows, Transportation Science 32 (1) (1998) 12–29.

[220] G. Pesant, M. Gendreau, J.-Y. Potvin, J.-M. Rousseau, On the flexibility of constraint programming models: From single to multiple time windows for the traveling salesman problem, European Journal of Operational Research 117 (2) (1999) 253–263.

[221] A. Makhorin, Glpk linear programming kit: Implementation of the revised simplex method, Glpk documentation, Moscow Aviation Institute, Moscow, Russia (2001).

[222] W. E. Hart, C. D. Laird, J.-P. Watson, D. L. Woodruff, G. A. Hackebeil, B. L. Nicholson, J. D. Siirola, Pyomo-optimization modeling in python, Vol. 67, Springer, 2017.

[223] P. A. Hancock, N. Meshkati, Human mental workload, North-Holland Amsterdam, 1988.

[224] T. B. Sheridan, T. B. Sheridan, K. Maschinenbauingenieur, T. B. Sheridan, T. B. Sheridan, Humans and automation: System design and research issues, Vol. 280, Human Factors and Ergonomics Society Santa Monica, CA, 2002.

[225] F. Nachreiner, P. Nickel, I. Meyer, Human factors in process control systems: The design of human–machine interfaces, Safety Science 44 (1) (2006) 5–26.

[226] E. J. Sirevaag, A. F. Kramer, C. D. W. M. REISWEBER, D. L. STRAYER, J. F. GRENELL, Assessment of pilot performance and mental workload in rotary wing aircraft, Ergonomics 36 (9) (1993) 1121–1140.

[227] G. F. Wilson, An analysis of mental workload in pilots during flight using multiple psychophysiological measures, The International Journal of Aviation Psychology 12 (1) (2002) 3–18.

[228] S. R. Dixon, C. D. Wickens, Control of multiple-uavs: A workload analysis, Tech. rep., ILLINOIS UNIV AT URBANA-CHAMPAIGN SAVOY AVIATION HUMAN FACTORS DIVISION (2003).

[229] S. P. Schipani, An evaluation of operator workload, during partially-autonomous vehicle operations, Tech. rep., Army Research Lab Aberdeen Proving Ground MD (2003).

[230] Z. Zhang, R. Tian, V. G. Duffy, L. Li, The comfort of the soft-safety driver alerts: Measurements and evaluation, International Journal of Human–Computer Interaction (2022) 1–11.

[231] K. van de Merwe, S. Mallam, S. Nazir, Agent transparency, situation awareness, mental workload, and operator performance: A systematic literature review, Human Factors (2022) 00187208221077804.

[232] D. Gopher, E. Donchin, Workload: An examination of the concept. (1986).

[233] D. Gianazza, Forecasting workload and airspace configuration with neural networks and tree search methods, Artificial intelligence 174 (7-8) (2010) 530–549.

[234] J. Djokic, B. Lorenz, H. Fricke, Air traffic control complexity as workload driver, Transportation research part C: emerging technologies 18 (6) (2010) 930–936.

[235] G. Tobaruela, W. Schuster, A. Majumdar, W. Y. Ochieng, L. Martinez, P. Hendrickx, A method to estimate air traffic controller mental workload based on traffic clearances, Journal of Air Transport Management 39 (2014) 59–71.

[236] H. Wang, D. Gong, R. Wen, Air traffic controllers workload forecasting method based on neural network, in: The 27th Chinese control and decision conference (2015 CCDC), IEEE, 2015, pp. 2460–2463.

[237] J. D. Lee, K. A. See, Trust in automation: Designing for appropriate reliance, Human factors 46 (1) (2004) 50–80.

[238] J. Y. Chen, S. G. Lakhmani, K. Stowers, A. R. Selkowitz, J. L. Wright, M. Barnes, Situation awareness-based agent transparency and human-autonomy teaming effectiveness, Theoretical issues in ergonomics science 19 (3) (2018) 259–282.

[239] P. Schmidt, F. Biessmann, T. Teubner, Transparency and trust in artificial intelligence systems, Journal of Decision Systems 29 (4) (2020) 260–278.

[240] S. V. Ligda, M. L. Seeds, M. J. Harris, C. S. Lieber, M. Demir, N. Cooke, Monitoring human performance in real-time for nas safety prognostics, in: AIAA Aviation 2019 Forum, 2019, p. 3411.

[241] C. S. Lieber, M. Demir, N. Cooke, S. Ligda, Deviations in closed loop communications between air traffic controllers and pilots as a predictor of loss of separation, in: AIAA AVIATION 2021 FORUM, 2021, p. 2320.

[242] B. Hilburn, Cognitive complexity in air traffic control: A literature review, EEC note 4 (04) (2004) 1–80.

[243] F. A. Regulations, Title 14-aeronautics and space, US government Printing Office, Washington, USA (2017).

[244] N. T. S. Board., Annual review of aircraft accident data, us air carrier operations: Calendar year 2001 (rep. no. arc-06–01) (2001).

[245] D.-T. Pham, S. Alam, V. Duong, An air traffic controller action extraction-prediction model using machine learning approach, Complexity 2020 (2020) 1–19.

[246] Y. Heng, M. Wu, X. Wen, et al., Identifying key risk factors in air traffic controller workload by seir model, Mathematical Problems in Engineering 2022 (2022).

[247] R. Xiong, Y. Wang, P. Tang, N. J. Cooke, S. V. Ligda, C. S. Lieber, Y. Liu, Predicting separation errors of air traffic controllers through integrated sequence analysis of multimodal behaviour indicators, Advanced Engineering Informatics 55 (2023) 101894.

[248] C. A. Manning, S. H. Mills, C. M. Fox, E. M. Pfleiderer, H. J. Mogilka, Using air traffic control taskload measures and communication events to predict subjective workload, Tech. rep., FEDERAL AVIATION ADMINISTRATION OKLAHOMA CITY OK CIVIL AEROMEDICAL INST (2002).

[249] S. Hah, B. Willems, R. Phillips, The effect of air traffic increase on controller workload, in: Proceedings of the Human Factors and Ergonomics Society Annual Meeting, Vol. 50, SAGE Publications Sage CA: Los Angeles, CA, 2006, pp. 50–54.

[250] J. Crutchfield, C. Rosenberg, Predicting subjective workload ratings: A comparison and synthesis of operational and theoretical models, Tech. rep., FEDERAL AVIATION ADMINISTRATION OKLAHOMA CITY OK CIVIL AEROMEDICAL INST (2007).

[251] T. Edwards, S. Sharples, J. R. Wilson, B. Kirwan, Factor interaction influences on human performance in air traffic control: The need for a multifactorial model, Work 41 (Supplement 1) (2012) 159–166.

[252] G. B. Chatterji, B. Sridhar, Neural network based air traffic controller workload prediction, in: Proceedings of the 1999 American Control Conference (Cat. No. 99CH36251), Vol. 4, IEEE, 1999, pp. 2620–2624.

[253] G. Chatterji, B. Sridhar, Measures for air traffic controller workload prediction, in: 1st AIAA, aircraft, technology Integration, and operations Forum, 2001, p. 5242.

[254] A. Majumdar, W. Y. Ochieng, Factors affecting air traffic controller workload: Multivariate analysis based on simulation modeling of controller workload, Transportation Research Record 1788 (1) (2002) 58–69.

[255] T. Edwards, L. Martin, N. Bienert, J. Mercer, The relationship between workload and performance in air traffic control: exploring the influence of levels of automation and variation in task demand, in: Human Mental Workload: Models and Applications: First International Symposium, H-WORKLOAD 2017, Dublin, Ireland, June 28-30, 2017, Revised Selected Papers 1, Springer, 2017, pp. 120–139.

[256] S. C. Corver, D. Unger, G. Grote, Predicting air traffic controller workload: trajectory uncertainty as the moderator of the indirect effect of traffic density on controller workload through traffic conflict, Human factors 58 (4) (2016) 560–573.

[257] K. Sharma, H. Iyer, R. Pant, Cognitive ability criterion for expertise in air traffic control task, in: AIAA SCITECH 2022 Forum, 2022, p. 2449.

[258] L. L. Di Stasi, M. Marchitto, A. Antolí, T. Baccino, J. J. Cañas, Approximation of on-line mental workload index in atc simulated multitasks, Journal of Air Transport Management 16 (6) (2010) 330–333.

[259] H. A. Abbass, J. Tang, R. Amin, M. Ellejmi, S. Kirby, Augmented cognition using real-time eeg-based adaptive strategies for air traffic control, in: Proceedings of the human factors and ergonomics society annual meeting, Vol. 58, SAGE Publications Sage CA: Los Angeles, CA, 2014, pp. 230–234.

[260] P. Aricò, G. Borghini, G. Di Flumeri, A. Colosimo, S. Bonelli, A. Golfetti, S. Pozzi, J.-P. Imbert, G. Granger, R. Benhacene, et al., Adaptive automation triggered by eeg-based mental workload index: a passive brain-computer interface application in realistic air traffic control environment, Frontiers in human neuroscience 10 (2016) 539.

[261] B. Hilburn, G. Flynn, Toward a non-linear approach to modeling air traffic complexity, in: 2nd Human Performance Situation Awareness and Automation Conference, 2004.

[262] F. T. Durso, A. L. Alexander, Managing workload, performance, and situation awareness in aviation systems, in: Human factors in aviation, Elsevier, 2010, pp. 217–247.

[263] R. M. Rose, C. D. Jenkins, M. W. Hurst, et al., Air traffic controller health change study: a prospective investigation of physical, psychological and work-related changes., Tech. rep., Civil Aerospace Medical Institute (1978).

[264] P. Kopardekar, S. Magyarits, Measurement and prediction of dynamic density, in: Proceedings of the 5th usa/europe air traffic management r & d seminar, Vol. 139, 2003.

[265] D. Delahaye, S. Puechmorel, Air traffic complexity: Towards an intrinsic metric, in: Proceeding of the 3rd USA/Europe Air Traffic Management R and D Seminar, 2000.

[266] D. Gianazza, K. Guittet, Selection and evaluation of air traffic complexity metrics, in: 2006 ieee/aiaa 25TH Digital Avionics Systems Conference, IEEE, 2006, pp. 1–12.

[267] R. H. Mogford, J. Guttman, S. Morrow, P. Kopardekar, The complexity construct in air traffic control: A review and synthesis of the literature. (1995).

[268] K. Kallus, D. Van Damme, A. Dittman, Integrated job and task analysis of air traffic controllers: Phase 2, Task analysis of en-route controllers (European Air Traffic Management Programme Rep. No. HUM. ET1. ST01. 1000-REP-04). EUROCONTROL, Brussels, Belgium (1999).

[269] B. Kirwan, R. Scaife, R. Kennedy, Investigating complexity factors in uk air traffic management, Human Factors and Aerospace Safety 1 (2) (2001).

[270] J. M. Histon, R. J. Hansman, G. Aigoin, D. Delahaye, S. Puechmorel, Introducing structural considerations into complexity metrics, Air Traffic Control Quarterly 10 (2) (2002) 115–130.

[271] C. D. Wickens, B. L. Hooey, B. F. Gore, A. Sebok, C. S. Koenicke, Identifying black swans in nextgen: Predicting human performance in off-nominal conditions, Human Factors 51 (5) (2009) 638–651.

[272] G. C. Fraccone, V. Volovoi, A. E. Colón, M. Blake, Novel air traffic procedures: investigation of off-nominal scenarios and potential hazards, Journal of Aircraft 48 (1) (2011) 127–140.

[273] J. H. Crump, Review of stress in air traffic control: Its measurement and effects., Aviation, Space, and Environmental Medicine (1979).

[274] J. Vogt, T. Hagemann, M. Kastner, The impact of workload on heart rate and blood pressure in en-route and tower air traffic control, Journal of psychophysiology 20 (4) (2006) 297–314.

[275] F. Trapsilawati, M. K. Herliansyah, A. S. A. N. S. Nugraheni, M. P. Fatikasari, G. Tissamodie, Eeg-based analysis of air traffic conflict: Investigating controllers' situation awareness, stress level and brain activity during conflict resolution, The Journal of Navigation 73 (3) (2020) 678–696.

[276] F. Li, C.-H. Lee, C.-H. Chen, L. P. Khoo, Hybrid data-driven vigilance model in traffic control center using eye-tracking data and context data, Advanced Engineering Informatics 42 (2019) 100940.

[277] H. J. Wee, S. W. Lye, J.-P. Pinheiro, An integrated highly synchronous, high resolution, real time eye tracking system for dynamic flight movement, Advanced Engineering Informatics 41 (2019) 100919.

[278] A. J. Masalonis, M. B. Callaham, C. R. Wanke, Dynamic density and complexity metrics for realtime traffic flow management, in: Proceedings of the 5th USA/Europe Air Traffic Management R & D Seminar, Budapest, Hungary, 2003, p. 139.

[279] D. Gianazza, Learning air traffic controller workload from past sector operations, in: ATM Seminar, 12th USA/Europe Air Traffic Management R&D Seminar, 2017.

[280] Y. Liu, K. Goebel, Information fusion for national airspace system prognostics, in: PHM Society Conference, Vol. 10, 2018.

[281] G. B. Reid, T. E. Nygren, The subjective workload assessment technique: A scaling procedure for measuring mental workload, in: Advances in psychology, Vol. 52, Elsevier, 1988, pp. 185–218.

[282] S. G. Hart, L. E. Staveland, Development of nasa-tlx (task load index): Results of empirical and theoretical research, in: Advances in psychology, Vol. 52, Elsevier, 1988, pp. 139–183.

[283] S. G. Hart, Nasa-task load index (nasa-tlx); 20 years later, in: Proceedings of the human factors and ergonomics society annual meeting, Vol. 50, Sage publications Sage CA: Los Angeles, CA, 2006, pp. 904–908.

[284] F. T. Durso, A. R. Dattel, The complexity of workload assessment: An overview of methods, in: Linking Expertise and Naturalistic Decision Making, Lawrence Erlbaum Associates, 2004, pp. 60–74.

[285] S. V. Ligda, M. A. Neerincx, J. Lindenberg, Measuring operator workload in a multitask environment, Interacting with Computers 22 (6) (2010) 532–541.

[286] K.-P. L. Vu, K. Procci, M. Mouloua, Methods for assessing operator workload: A critical review and synthesis, Journal of Cognitive Engineering and Decision Making 6 (3) (2012) 211–232.

[287] C. Lieber, Communications between air traffic controllers and pilots during simulated arrivals: Relation of closed loop communication deviations to loss of separation, Ph.D. thesis, Arizona State University (2020).

[288] J. C. Gorman, P. W. Foltz, P. A. Kiekel, M. J. Martin, N. J. Cooke, Evaluation of latent semantic analysis-based measures of team communications content, in: Proceedings of the Human Factors and Ergonomics Society annual meeting, Vol. 47, Sage Publications Sage CA: Los Angeles, CA, 2003, pp. 424–428.

[289] E. E. Salas, S. M. Fiore, Team cognition: Understanding the factors that drive process and performance., American Psychological Association, 2004.

[290] N. J. Cooke, J. C. Gorman, P. A. Kiekel, Communication as team-level cognitive processing, in: Macrocognition in teams, CRC Press, 2017, pp. 51–64.

[291] J.-P. Eckmann, S. O. Kamphorst, D. Ruelle, et al., Recurrence plots of dynamical systems, World Scientific Series on Nonlinear Science Series A 16 (1995) 441–446.

[292] N. Marwan, M. C. Romano, M. Thiel, J. Kurths, Recurrence plots for the analysis of complex systems, Physics reports 438 (5-6) (2007) 237–329.

[293] M. Koebbe, G. Mayer-Kress, Use of recurrence plots in the analysis of time-series data, in: SANTA FE INSTITUTE STUDIES IN THE SCIENCES OF COMPLEXITY-PROCEEDINGS VOLUME-, Vol. 12, Citeseer, 1992, pp. 361–361.

[294] J. P. Zbilut, C. L. Webber Jr, Embeddings and delays as derived from quantification of recurrence plots, Physics letters A 171 (3-4) (1992) 199–203.

[295] G. M. Mindlin, R. Gilmore, Topological analysis and synthesis of chaotic time series, Physica D: Nonlinear Phenomena 58 (1-4) (1992) 229–242.

[296] C. L. Webber Jr, J. P. Zbilut, Dynamical assessment of physiological systems and states using recurrence plot strategies, Journal of applied physiology 76 (2) (1994) 965–973.

[297] N. Marwan, N. Wessel, U. Meyerfeldt, A. Schirdewan, J. Kurths, Recurrence-plot-based measures of complexity and their application to heart-rate-variability data, Physical review E 66 (2) (2002) 026702.

[298] H. Kantz, Quantifying the closeness of fractal measures, Physical Review E 49 (6) (1994) 5091.

[299] A. Pareja, G. Domeniconi, J. Chen, T. Ma, T. Suzumura, H. Kanezashi, T. Kaler, T. Schardl, C. Leiserson, Evolvegcn: Evolving graph convolutional networks for dynamic graphs, in: Proceedings of the AAAI conference on artificial intelligence, Vol. 34, 2020, pp. 5363–5370.

[300] V. Vovk, A. Gammerman, C. Saunders, Machine-learning applications of algorithmic randomness (1999).

[301] A. N. Angelopoulos, S. Bates, A gentle introduction to conformal prediction and distribution-free uncertainty quantification, arXiv preprint arXiv:2107.07511 (2021).

[302] M. Sadinle, J. Lei, L. Wasserman, Least ambiguous set-valued classifiers with bounded error levels, Journal of the American Statistical Association 114 (525) (2019) 223–234.

[303] A. Angelopoulos, S. Bates, J. Malik, M. I. Jordan, Uncertainty sets for image classifiers using conformal prediction, arXiv preprint arXiv:2009.14193 (2020).

[304] Y. Romano, M. Sesia, E. Candes, Classification with valid and adaptive coverage, Advances in Neural Information Processing Systems 33 (2020) 3581–3591.

[305] A. N. Angelopoulos, S. Bates, A. Fisch, L. Lei, T. Schuster, Conformal risk control, arXiv preprint arXiv:2208.02814 (2022).

[306] H. Olsson, K. Kartasalo, N. Mulliqi, M. Capuccini, P. Ruusuvuori, H. Samaratunga, B. Delahunt, C. Lindskog, E. A. Janssen, A. Blilie, et al., Estimating diagnostic uncertainty in artificial intelligence assisted pathology using conformal prediction, Nature communications 13 (1) (2022) 7761.

[307] V. Balasubramanian, S.-S. Ho, V. Vovk, Conformal prediction for reliable machine learning: theory, adaptations and applications, Newnes, 2014.

[308] J. Alvarsson, S. A. McShane, U. Norinder, O. Spjuth, Predicting with confidence: using conformal prediction in drug discovery, Journal of Pharmaceutical Sciences 110 (1) (2021) 42–49.

[309] C. Lu, A. Lemay, K. Chang, K. Höbel, J. Kalpathy-Cramer, Fair conformal predictors for applications in medical imaging, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 36, 2022, pp. 12008–12016.

[310] R. Luo, S. Zhao, J. Kuck, B. Ivanovic, S. Savarese, E. Schmerling, M. Pavone, Sample-efficient safety assurances using conformal prediction, in: Algorithmic Foundations of Robotics XV: Proceedings of the Fifteenth Workshop on the Algorithmic Foundations of Robotics, Springer, 2022, pp. 149–169.

[311] H. Papadopoulos, K. Proedrou, V. Vovk, A. Gammerman, Inductive confidence machines for regression, in: Machine Learning: ECML 2002: 13th European Conference on Machine Learning Helsinki, Finland, August 19–23, 2002 Proceedings 13, Springer, 2002, pp. 345–356.

[312] V. Vovk, A. Gammerman, G. Shafer, Algorithmic learning in a random world, Vol. 29, Springer, 2005.

[313] J. Lei, L. Wasserman, Distribution-free prediction bands for non-parametric regression, Journal of the Royal Statistical Society: Series B: Statistical Methodology (2014) 71–96.

[314] A. A. Taha, A. Hanbury, Metrics for evaluating 3d medical image segmentation: analysis, selection, and tool, BMC medical imaging 15 (1) (2015) 1–28.