

Copyright Undertaking

This thesis is protected by copyright, with all rights reserved.

By reading and using the thesis, the reader understands and agrees to the following terms:

1. The reader will abide by the rules and legal ordinances governing copyright regarding the use of the thesis.
2. The reader will use the thesis for the purpose of research or private study only and not for distribution or further reproduction or any other purpose.
3. The reader agrees to indemnify and hold the University harmless from and against any loss, damage, cost, liability or expenses arising from copyright infringement or unauthorized usage.

IMPORTANT

If you have reasons to believe that any materials in this thesis are deemed not suitable to be distributed in this form, or a copyright owner having difficulty with the material being included in our database, please contact lbsys@polyu.edu.hk providing details. The Library will look into your claim and consider taking remedial action upon receipt of the written requests.

RESEARCH ON AIRCRAFT TRAJECTORY
PREDICTION METHODS BASED ON DEEP
LEARNING

HAOYUAN ZHANG

PhD

The Hong Kong Polytechnic University

2026

The Hong Kong Polytechnic University
Department of Land Surveying and Geo-Informatics

Research on aircraft trajectory prediction methods based on deep learning

Haoyuan ZHANG

A thesis submitted in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

April 2025

CERTIFICATE OF ORIGINALITY

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgement has been made in the text.

Signature: _____

Name of student: Haoyuan ZHANG

ABSTRACT

Air transportation is essential for global social and economic development, yet a gap persists between air transport services and market needs. To address this, ICAO advocates trajectory-based operations (TBO), highlighting the importance of accurate four-dimensional (4D) trajectory prediction (TP) for improved airspace capacity, efficiency, and safety. Traditional physics-based TP methods face limitations due to complex parameter requirements and observability issues, while machine learning (ML) approaches, though promising, still lack a robust framework and sufficient accuracy for medium- and long-term predictions. Data imbalance, especially under severe weather conditions, further hinders TP performance. Overcoming these challenges is vital for reliable trajectory prediction in air traffic management (ATM).

This research focuses on three critical areas of aircraft TP: 1) developing a new TP technology framework based on deep learning, 2) improving medium- and long-term TP accuracy, and 3) enhancing TP ability under severe weather conditions. The main findings from this research are as follows:

In terms of overall technical framework, a typical TP system architecture is proposed, consisting of several information processing units such as data preprocessing, data updating, prediction output, TP models, and trajectory integration fusion. A basic evaluation index system is also introduced, comprising system functionality, TP accuracy, confidence in prediction results, response speed, stability, reliability, environmental adaptability, user experience, and system maintainability. Typical structural patterns of hybrid TP models are proposed, including the recurrent neural network (RNN) + convolutional neural network (CNN), Attention + RNN, generative adversarial network (GAN), and clustering algorithm-based models. Methods for determining and optimizing the parameters and hyperparameters of deep learning TP models are also outlined.

In this framework, a novel 4D TP method based on the conditional tabular generative adversarial network (CTGAN) model is proposed, which leverages automatic dependent surveillance-broadcast (ADS-B) data to achieve high

accuracy and stability in predicting multi-dimensional parameters under complex airspace conditions. The improved CTGAN model outperforms conditional generative adversarial networks (CGAN) in terms of accuracy, adaptability, and resilience, particularly in handling small sample data. It also surpasses long short-term memory (LSTM) in medium- to long-term TP accuracy over extended time spans. The model was trained and tested using historical 4D trajectory data from Hong Kong International Airport, and its performance was evaluated using metrics such as Jensen-Shannon (JS) divergence, maximum mean discrepancy (MMD), mean Euclidean distance (MED), and average running time. The results show a high degree of similarity between the predicted and real 4D trajectories, even with reduced input data. Comparative experiments demonstrate that the improved CTGAN achieves 10% faster computational speed than CGAN, with a 20% reduction in MMD and an 81.55% decrease in MED. Additionally, it significantly reduces prediction errors compared to LSTM, with greater improvements as the prediction time span increases. These findings validate the feasibility and potential of CTGAN for practical 4D TP applications.

In terms of improving medium- and long-term TP accuracy, based on the CTGAN architecture, the C-CTGAN model—combining K-medoids clustering with CTGAN—was developed. The model's accuracy was evaluated from the perspectives of data distribution and trajectory distance using MMD and MED metrics. Mean absolute error (MAE) was used to compare the C-CTGAN LSTM, CNN-LSTM, CNN-LSTM-Attention, CNN-BiLSTM, and the original CTGAN models. The proposed C-CTGAN model significantly outperformed LSTM-based models and the original CTGAN in predicting medium- and long-term trajectories. Clustering proved effective in enhancing TP precision. Experiments show the C-CTGAN model reduced MAE of core trajectory parameters (latitude, longitude, geometric altitude, and ground speed) by 69.89%, 15.00%, 74.07%, and 84.21%, respectively, compared to the CNN-LSTM model. When compared to the original CTGAN model, MAE reductions were 20.43% (latitude), 39.09% (longitude), 31.98% (geometric altitude), and 17.07% (ground speed).

Regarding TP under severe weather conditions, a solution to the data imbalance problem is proposed by combining a data enhancement network and a TP network. A hybrid model architecture of the CTGAN and TP network is designed, and the

CTGAN network is optimized. After using the optimized CTGAN to enhance minority-class samples, the TP network (LSTM is selected as a case) is applied for TP, which can significantly improve TP accuracy. Experiments showed that increasing the proportion of minority-class data from 4.7% to 13% reduced the average root mean square error (RMSE) by 20.46%. Compared with the classic data enhancement method, synthetic minority oversampling technique (SMOTE), the improved CTGAN is more suitable for aircraft TP. Research shows that the data enhancement method effectively solved the problem of data imbalance in TP and provides a new method for TP based on minority-class sample data.

LIST OF PUBLICATIONS

- [1] H. Zhang and Z. Liu, “Four-Dimensional Aircraft Trajectory Prediction Based on Generative Deep Learning,” *Journal of Aerospace Information Systems*, pp. 554-567, 2024.
- [2] H. Zhang and Z. Liu, “Four-Dimensional Aircraft Trajectory Prediction with a Generative Deep Learning and Clustering Approach,” *Journal of Aerospace Information Systems*, pp. 1-13, 2025.
- [3] H. Zhang and Z. Liu, “4D Aircraft Trajectory Prediction Considering the Severe Weather Effect.” *The Aeronautical Journal*, pp. 1-25, 2025.
- [4] H. Zhang and Z. Liu, “Medium- and long-term 4D Aircraft Trajectory Prediction Based on the Transfer Learning.” *Journal of Aerospace Information Systems* (**Finished**).
- [5] H. Zhang and Z. Liu, "Research on Air Traffic Situation near the Hong Kong Region Based on ADS-B Surveillance Data." the 2019 CPGPS Forum, Wuhan, China, 20-22 July 2019.

ACKNOWLEDGMENTS

It has been more than six years since I joined The Hong Kong Polytechnic University in September 2018, and this journey has become one of the most treasured memories of my life. However, these years were not without their challenges. I spent two years battling the aftereffects of Long COVID-19, which greatly impacted my well-being. During this difficult time, I was fortunate to receive unwavering support from my supervisor, parents, and friends, whose encouragement gave me the strength to persevere and continue my studies. I am truly fortunate to have met so many wonderful people along this extraordinary journey.

First and foremost, I wish to express my profound gratitude to my supervisor, Prof. Zhizhao Liu, for his steadfast academic and financial support throughout my studies. His invaluable guidance and dedication at every stage of this process have been pivotal to the completion of this thesis.

I am also thankful to my fellow labmates in the Micro-LARGE Group: Dr. Jia He, Dr. Shiwei Yu, Dr. Yangzhao Gong, Dr. Dabin Xue, Dr. Jiafei Xu, and Yusong Cheng, for the insightful discussions and collaboration we shared. Additionally, I extend my heartfelt thanks to my friends, including Dr. Mingchen Zhu and Rui Wang, for their joy, companionship, and support over the past six years.

Last but certainly not least, my deepest and most sincere gratitude goes to my mother and father for their endless love and unwavering support. Their belief in me has been a constant source of strength and motivation.

TABLE OF CONTENTS

ABSTRACT	I
LIST OF PUBLICATIONS.....	IV
ACKNOWLEDGMENTS.....	V
TABLE OF CONTENTS	VI
LIST OF FIGURES	XI
LIST OF TABLES	XVI
LIST OF ABBREVIATIONS	XVIII
Chapter 1. Introduction.....	1
1.1 Research background and significance	1
1.2 Main research problems at present	5
1.2.1 TP technology framework based on deep learning	6
1.2.2 TP accuracy	7
1.2.3 TP based on minority-class sample data	7
1.3 Research content and technical approach of the thesis	8
1.3.1 Main research contents	8
1.3.2 Technical route	9
1.4 Organization of the thesis	10
Chapter 2. Fundamentals of aircraft trajectory prediction and deep learning	14
2.1 Fundamentals of aircraft TP	14
2.1.1 Relevant concepts in aircraft TP.....	14
2.1.2 TP based on trajectory-based operations	21
2.2 Current status of aircraft TP research	30
2.2.1 Dynamics methods	31

2.2.2 State estimation methods	34
2.2.3 Machine learning methods	37
2.2.4 Application of TP methods.....	51
2.2.5 Overview of research status.....	56
2.2.6 Comparative analysis of mainstream TP models	58
2.3 Typical deep learning network	60
2.3.1 Generative adversarial networks	60
2.3.2 Recurrent neural networks.....	68
2.3.3 Sequence-to-sequence generative model.....	77
2.3.4 Convolutional neural networks.....	81
2.4 Deep Learning TP support data	87
2.4.1 Automatic dependent surveillance–broadcast data	88
2.4.2 Meteorological data	93
2.5 Chapter summary.....	97
Chapter 3. Trajectory prediction technology framework based on deep learning	99
3.1 TP system composition.....	99
3.1.1 Basic system architecture	99
3.1.2 Trajectory calculation	102
3.2 Construction and application of TP system.....	104
3.2.1 Demand analysis.....	105
3.2.2 System design.....	105
3.2.3 Programming implementation	106
3.2.4 Model training and validation	106
3.2.5 System testing and evaluation	106
3.2.6 Deploy application.....	107
3.3 Evaluation of TP system.....	107

3.3.1 TP system evaluation framework	107
3.3.2 TP accuracy evaluation.....	113
3.4 Deep learning TP model.....	118
3.4.1 Typical structural model.....	118
3.4.2 Model parameter determination method	123
3.5 Chapter summary.....	126
Chapter 4. Aircraft trajectory prediction based on conditional tabular generative adversarial networks	129
4.1 Research background.....	129
4.2 Features analysis of CTGANs	131
4.3 TP method based on CTGANs	134
4.3.1 Improvements for CTGANs	137
4.3.2 Evaluation metrics design.....	141
4.3.3 Aircraft TP process	145
4.4 Experiment and analysis.....	149
4.4.1 Data preprocessing	149
4.4.2 Model hyperparameter confirmation and training.....	151
4.4.3 Improved CTGAN model testing and analysis	152
4.4.4 Comparison experiment and analysis with CGANs.....	160
4.4.5 Comparison experiment and analysis with Transformers	162
4.4.6 Comparison experiment and analysis with LSTMs.....	163
4.4.7 Experimental conclusion	166
4.5 Chapter summary.....	167
Chapter 5. Medium- and long-term aircraft trajectory prediction based on clustering and conditional tabular generative adversarial networks.....	169
5.1 Research background.....	169
5.2 Trajectory clustering algorithm	173

5.2.1 K-means.....	173
5.2.2 K-medoids	175
5.2.3 DBSCAN.....	176
5.3 Typical medium- and long-term aircraft TP methods	177
5.3.1 Flight plan interpolation	177
5.3.2 Stateful LSTM	178
5.3.3 Conv1D-GAN.....	178
5.3.4 Nominal route method.....	180
5.4 TP method based on C-CTGANs	181
5.4.1 Data preprocessing	182
5.4.2 Trajectory clustering.....	182
5.4.3 Model training	185
5.4.4 Trajectory prediction	189
5.5 Experiment and analysis.....	189
5.5.1 Data preparation	189
5.5.2 Test environment and model parameter settings	190
5.5.3 Model testing and analysis	192
5.5.4 Experimental conclusion	202
5.6 Chapter summary.....	204
Chapter 6. Aircraft trajectory prediction considering weather factors and minority- class sample types.....	206
6.1 Research background.....	206
6.2 Weather factor representation and data processing methods	209
6.2.1 Weather factor representation.....	209
6.2.2 Weather cube calculation	210
6.3 Minority-class sample data processing.....	211
6.3.1 Characteristics of minority-class sample types	211

6.3.2 Methods for processing minority-class samples.....	212
6.3.3 Synthetic minority over-sampling technique (SMOTE) data enhancement method	213
6.4 Aircraft TP methods under severe weather conditions.....	215
6.4.1 Overall framework.....	216
6.4.2 Data preprocessing	218
6.4.3 Data augmentation methods	219
6.4.4 Model training	223
6.4.5 Trajectory prediction	224
6.5 Experiment and analysis.....	224
6.5.1 Data preparation	224
6.5.2 Experimental environment and model hyperparameter settings	226
6.5.3 Model testing and analysis	227
6.5.4 Experimental conclusion	237
6.6 Chapter summary.....	239
Chapter 7. Summary and outlook.....	241
7.1 Main research work of this thesis	241
7.2 Main innovations of the thesis.....	242
7.3 Research directions and prospects.....	243
References	246

LIST OF FIGURES

<i>Figure 1.1 Evolution of the number of papers published each year in different directions of the ATM field.</i>	5
<i>Figure 1.2 Overall research plan of the thesis.</i>	10
<i>Figure 1.3 Thesis organization.</i>	11
<i>Figure 2.1 Stages of the entire aircraft trajectory.</i>	16
<i>Figure 2.2 ATM concept components.</i>	23
<i>Figure 2.3 TBO operation diagram [8].</i>	25
<i>Figure 2.4 Trajectory predictor process flow.</i>	26
<i>Figure 2.5 General TP data flow.</i>	27
<i>Figure 2.6 The model architecture of the original generative adversarial networks.</i>	63
<i>Figure 2.7 The transformation from GAN to CGAN.</i>	65
<i>Figure 2.8 (a) SRN recurrent unit structure, (b) SRN network structure expanded over time.</i>	70
<i>Figure 2.9 LSTM network unit structure.</i>	71
<i>Figure 2.10 Structure of gated recurrent unit.</i>	73
<i>Figure 2.11 Transformer architecture.</i>	78
<i>Figure 2.12 Basic structure of convolutional neural networks.</i>	82
<i>Figure 2.13 Convolutional layer operation process.</i>	83
<i>Figure 2.14 Schematic diagram of residual networks.</i>	84

<i>Figure 2.15 Composition of the automatic dependent surveillance–broadcast system.</i>	<i>88</i>
<i>Figure 3.1 Basic architecture of TP system.</i>	<i>100</i>
<i>Figure 3.2 Integrated application framework of TP models.</i>	<i>103</i>
<i>Figure 3.3 Construction and application process of TP system based on deep learning.....</i>	<i>105</i>
<i>Figure 3.4 Horizontal error of the aircraft.</i>	<i>114</i>
<i>Figure 3.5 The synchronous error measurement.</i>	<i>116</i>
<i>Figure 3.6 The common space error measurement method.</i>	<i>116</i>
<i>Figure 3.7 The closest point error measurement method.....</i>	<i>117</i>
<i>Figure 3.8 The framework mode of generative adversarial networks.</i>	<i>120</i>
<i>Figure 3.9 The framework mode combining convolutional neural networks and recurrent neural networks (using weather data and trajectory data).</i>	<i>121</i>
<i>Figure 3.10 The framework mode combining convolutional neural networks and recurrent neural networks (using only trajectory data).</i>	<i>121</i>
<i>Figure 3.11 The framework mode combining Attention mechanism and recurrent neural networks.</i>	<i>122</i>
<i>Figure 3.12 The framework mode combining clustering.</i>	<i>123</i>
<i>Figure 3.13 TP model parameter optimization process based on deep learning.</i>	<i>126</i>
<i>Figure 4.1 Simplified architecture for CTGANs.</i>	<i>134</i>
<i>Figure 4.2 The overall framework of the 4D aircraft TP approach based on an improved CTGAN architecture.....</i>	<i>137</i>
<i>Figure 4.3 Improved CTGAN architecture for ADS-B data.....</i>	<i>139</i>

<i>Figure 4.4 The ADS-B data preprocessing flow chart.</i>	<i>146</i>
<i>Figure 4.5 The diagram of alignment of arrival flight time parameters.</i>	<i>147</i>
<i>Figure 4.6 Schematic diagram of data alignment between the test dataset and generated data.</i>	<i>148</i>
<i>Figure 4.7 Flight trajectory from KADLO to VHHH on December 23, 2016. .</i>	<i>150</i>
<i>Figure 4.8 Distribution comparison of generated and real data for 80 flights from way KADLO to VHHH during December 23-27, 2016, in the Hong Kong region.</i>	<i>154</i>
<i>Figure 4.9 Loss function value of the generator and discriminator for training.</i>	<i>154</i>
<i>Figure 4.10 Distribution comparison of real and generated data for different numbers of input flights. (a) includes a single flight, (b) includes 5 flights, (c) includes 10 flights, and (d) includes 80 flights.</i>	<i>156</i>
<i>Figure 4.11 Distribution comparison of real and generated data for different levels of thinning from 80 flights. (a) 20% data remaining, (b) 40% data remaining, (c) 60% data remaining, and (d) 80% data remaining.</i>	<i>158</i>
<i>Figure 4.12 Distribution comparison of real and generated data based on CGAN and improved CTGAN models from 30 flights. (a) CGAN, (b) CTGAN.</i>	<i>161</i>
<i>Figure 4.13 Comparison of distribution of real data and generated data from 30 flights based on Transformer and improved CTGAN models. (a) Transformer, (b) CTGAN.</i>	<i>163</i>
<i>Figure 4.14 Comparison among the prediction results of improved CTGAN and LSTM models for flight AAR723. (a) latitude, (b) longitude, (c) geometric height, (d) ground speed, and (e) track angle.</i>	<i>166</i>
<i>Figure 5.1 Conv1D-GAN structure [21].</i>	<i>179</i>
<i>Figure 5.2 Basic process of TP using the nominal route method.</i>	<i>180</i>

<i>Figure 5.3 The general framework of the 4D aircraft TP method based on clustering and an improved CTGAN architecture.....</i>	<i>182</i>
<i>Figure 5.4 The SSE curve for determining the appropriate K value via the elbow method.</i>	<i>184</i>
<i>Figure 5.5 Improved CTGAN architecture based on the K-medoids clustering method.</i>	<i>186</i>
<i>Figure 5.6 The selected flight trajectories from waypoints KADLO to VHHH: (a) without holding, (b) with holding.</i>	<i>190</i>
<i>Figure 5.7 The clustered flight trajectories from waypoints KADLO to VHHH: (a) without holding, (b) with holding.</i>	<i>192</i>
<i>Figure 5.8 Distribution comparison between the generated and real data: (a) Flight trajectories without holding, (b) flight trajectories with holding.</i>	<i>194</i>
<i>Figure 5.9 The relationship between the prediction error of the LSTM model and the prediction time span after z-score standardization. The ordinate represents the MAEs of different parameters.</i>	<i>195</i>
<i>Figure 5.10 Comparison of the prediction results of the CNN-LSTM, CTGAN, and C-CTGAN models for flight CPA405 (trajectory without holding).....</i>	<i>199</i>
<i>Figure 5.11 Comparison of the prediction results of the CNN-LSTM, CTGAN, and C-CTGAN models for flight ANA859 (trajectory with holding).....</i>	<i>200</i>
<i>Figure 6.1 Weather cube concept diagram.</i>	<i>210</i>
<i>Figure 6.2 The general framework of the 4D aircraft TP method based on the improved CTGAN and LSTM-based architecture.</i>	<i>218</i>
<i>Figure 6.3 Visualization of the data filtering process using thresholds on ET data. (a) represents maximum ET value. (b) represents mean ET value.....</i>	<i>219</i>
<i>Figure 6.4 The CTGAN architecture considering severe weather conditions. .</i>	<i>220</i>
<i>Figure 6.5 Inbound flight trajectories from KADLO to VHHH in the Hong Kong</i>	

<i>region.....</i>	<i>225</i>
<i>Figure 6.6 Distribution comparison between the generated and real data.</i>	<i>228</i>
<i>Figure 6.7 Loss function value of the generator and discriminator for training.</i>	<i>228</i>
<i>Figure 6.8 Comparison of the prediction RMSE of the LSTM, CNN-LSTM, CNN- LSTM-Attention, and CNN-BiLSTM models for different data percentages.....</i>	<i>232</i>
<i>Figure 6.9 Comparison of the prediction results of the LSTM model for flight CPA475.....</i>	<i>233</i>
<i>Figure 6.10 RMSE values of different parameters for various data percentages using improved CTGAN and SMOTE data augmentation techniques.</i>	<i>236</i>

LIST OF TABLES

<i>Table 1.1 Categorization of aircraft trajectory prediction applications.....</i>	<i>3</i>
<i>Table 2.1 Application of flight phase navigation.</i>	<i>20</i>
<i>Table 2.2 Comparative analysis of aircraft trajectory prediction methods.</i>	<i>58</i>
<i>Table 2.3 ASTERIX category 021 message fields.</i>	<i>89</i>
<i>Table 4.1 Correlation between CTGAN advantages and aircraft TP characteristics.</i>	<i>135</i>
<i>Table 4.2 MED calculation under different hyperparameters of the CTGAN model.</i>	<i>151</i>
<i>Table 4.3 Training parameters for the improved CTGAN.</i>	<i>151</i>
<i>Table 4.4 Evaluation results of different parameters for 80 flights from KADLO to VHHH from December 23, 2016, to December 27, 2016, in the Hong Kong region.....</i>	<i>153</i>
<i>Table 4.5 Multiflight model evaluation results from KADLO to VHHH during December 23-27, 2016, in the Hong Kong region.</i>	<i>156</i>
<i>Table 4.6 Evaluation results of models with different levels of thinning from KADLO to VHHH during December 23-27, 2016 in the Hong Kong region. ..</i>	<i>157</i>
<i>Table 4.7 Comparison of CTGAN model evaluation metrics before and after improvement.</i>	<i>159</i>
<i>Table 4.8 Comparison of CGAN and improved CTGAN models.</i>	<i>161</i>
<i>Table 4.9 Comparison of Transformer and improved CTGAN models.....</i>	<i>162</i>
<i>Table 4.10 Comparison of MAE between LSTM and improved CTGAN Models.</i>	<i>164</i>

<i>Table 5.1 Training parameters for the C-CTGAN.</i>	<i>190</i>
<i>Table 5.2 MAE comparison among the LSTM, CNN-LSTM, CNN-LSTM-Attention, CNN-BiLSTM, original CTGAN, and C-CTGAN models based on flight trajectories without holding.</i>	<i>196</i>
<i>Table 5.3 MAE comparison among the LSTM, CNN-LSTM, CNN-LSTM-Attention, CNN-BiLSTM, original CTGAN, and C-CTGAN models based on the flight trajectories with holding.....</i>	<i>196</i>
<i>Table 6.1 Training parameters for the improved CTGAN.</i>	<i>221</i>
<i>Table 6.2 The parameter setting for different LSTM-based models.....</i>	<i>226</i>
<i>Table 6.3 Evaluation results of the improved CTGAN model under different encoding methods.</i>	<i>227</i>
<i>Table 6.4 RMSE of LSTM model with different data percentages of severe weather conditions.</i>	<i>229</i>
<i>Table 6.5 RMSE of CNN-LSTM model with different data percentages of severe weather conditions.....</i>	<i>229</i>
<i>Table 6.6 RMSE of CNN-LSTM-Attention model with different data percentages of severe weather conditions.</i>	<i>230</i>
<i>Table 6.7 RMSE of CNN-BiLSTM model with different data percentages of severe weather conditions.....</i>	<i>230</i>
<i>Table 6.8 Comparison of mean horizontal and vertical errors of the LSTM model for flight CPA475. The prediction time span is 5 minutes.</i>	<i>234</i>
<i>Table 6.9 RMSE of LSTM model with different data percentages using SMOTE data augmentation.</i>	<i>234</i>

LIST OF ABBREVIATIONS

ADS-B	Automatic Dependent Surveillance–Broadcast
AI	Artificial Intelligence
AMDAR	Aircraft Meteorological Data Relay
AOM	Airspace Organization and Management
APM	Aviation Propulsion Model
ARP	Airport Reference Point
ASBU	Aviation System Block Upgrade
ATC	Air Traffic Control
ATM	Air Traffic Management
ATFM	Air Traffic Flow Management
ATS	Air Traffic Service
BADA	Base of Aircraft Data
BPTT	Backpropagation Through Time
BNNs	Bayesian Neural Networks
C-CTGAN	Clustering and Conditional Tabular Generative Adversarial Network
CA	Constant Acceleration
CAT	Clear Air Turbulence
CD&R	Conflict Detection and Resolution
CMA	China Meteorological Administration
CMDC	China Meteorological Data Service Center
CNNs	Convolutional Neural Networks
CT	Constant Turn
CTGANs	Conditional Tabular Generative Adversarial Networks
CV	Constant Velocity
CVAEs	Conditional Variational Autoencoders
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
DCB	Demand-capacity Balancing
DCGAN	Deep Convolutional Generative Adversarial Network

DenseNet	Densely Connected Convolutional Network
DSTs	Decision Support Tools
DSW	Dynamic Space Warp
DTW	Dynamic Time Warping
ECMWF	European Centre for Medium-Range Weather Forecasts
EE	Euclidean Error
ET	Echo Top
FAA	Federal Aviation Administration
FID	Frechet Inception Distance
FMS	Flight Management System
GANs	Generative Adversarial Networks
GANP	Global Air Navigation Plan
GAT	Graph Attention Network
GCRNs	Graph Convolutional Recurrent Networks
GD	Gradient Descent
GFS	Global Forecast System
GMMs	Gaussian Mixture Models
GRIB	Grid Binary
GRU	Gated Recurrent Unit
HKO	Hong Kong Observatory
HMM	Hidden Markov Model
HKIA	Hong Kong International Airport
HPL	Horizontal Protection Level
ICAO	International Civil Aviation Organization
IMM	Interacting Multiple Models
IRU	Inertial Reference Unit
IS	Inception Score
JS	Jensen-Shannon
KF	Kalman Filter
KL	Kullback-Leibler
LSTM	Long Short-Term Memory
MED	Mean Euclidean Distance

MDE	Mean Deviation Error
ML	Machine Learning
MLP	Multilayer Perceptron
MMD	Maximum Mean Discrepancy
MSE	Mean Squared Error
NACp	Navigation Accuracy Category for the Position
NAM	North American Mesoscale Forecast System
NCEI	National Centers for Environmental Information
NIC	Navigation Integrity Category
NOAA	National Oceanic and Atmospheric Administration
NSGA-II	Non-Dominant Sorting Genetic Algorithm II
NUCp	Navigation Uncertainty Category for the Position
PBN	Performance-Based Navigation
PDFs	Probability Density Functions
PMM	Point-Mass Model
RMS	Root Mean Square
RMSprop	Root Mean Square Propagation
RNAV	Integrates Regional Navigation
RoPE	Rotary Positional Embeddings
RNNs	Recurrent Neural Networks
RTA	Required Time of Arrival
SA	Self-Attention
SDM	Service Delivery Management
SGD	Stochastic Gradient Descent
SLHS	Stochastic Linear Hybrid System
SMOTE	Synthetic Minority Over-Sampling Technique
SS-d	Sequence-to-Sequence Deep
SSE	Sum of Squared Errors
SWIM	System-Wide Information Management
TCNs	Temporal Convolutional Networks
TBO	Trajectory-Based Operation
TMP	Atmospheric Temperature
TOE	Time of Overfly Errors

TMA	Terminal Maneuvering Area
VI	Variational Inference
VGM	Variational Gaussian Mixture
VIL	Vertical Integrated Liquid
WGAN-GP	Wasserstein GAN-Gradient Penalty
WMO	World Meteorological Organization

Chapter 1.

Introduction

1.1 Research background and significance

Air transport is vital in promoting global social development and sustainable economic growth. According to statistics, the aviation industry directly and indirectly supports 58.1 million jobs yearly and contributes more than \$2.4 trillion to global GDP. The industry carries over 3.3 billion passengers and \$6.4 trillion of cargo each year [1]. Global air transport volume is projected to grow at an average annual rate of 4.4% through 2037, while China's air traffic is expected to expand by 250% during the same period [2].

The global air travel demand maintains a consistent growth momentum. On the one hand, it has promoted the prosperity and development of air transportation, and on the other hand, it has brought tremendous pressure and challenges to air traffic management (ATM). The sustained expansion of air travel demand is projected to elevate air traffic density significantly. This growth trajectory may precipitate route congestion, exacerbate flight delays, and ultimately degrade air traffic control (ATC) system performance, potentially elevating midair conflict risks and compromising aviation safety margins. The growing imbalance between escalating air traffic volumes and constrained airspace capacity is reaching critical levels. The rising demand for air travel has necessitated greater demands on ATM systems and service quality standards. As artificial intelligence (AI) technology advances at an accelerated pace, new types of transportation, such as unmanned aircraft systems, have begun to be applied in some fields. It can be foreseen that soon, uncrewed aircraft will become an essential member of air transportation. Adding new types of transportation will further increase the complexity of ATM and intensify the contradiction between aviation demand and service capabilities.

In 1998, the International Civil Aviation Organization (ICAO) introduced the global ATM operational framework in response to existing ATC challenges and anticipated future demands. Through system reconstruction, a new generation of

ATM system was built to fundamentally solve the problems faced by the ATC system [3]. Trajectory-based operation (TBO) is a vital element of the new generation of ATM systems. Real-time exchange and continuous updating of flight trajectories among aviation stakeholders form the foundation for coordinated operational planning and management [4]. The trajectory in the TBO environment refers to the set of four-dimensional (4D) coordinates of the trajectory points the aircraft passes through from the boarding gate to the arrival gate, including but not limited to three-dimensional geographic spatial position information and one-dimensional time information. That is, the 4D trajectory synthesizes flight path coordinates through geospatial positioning, temporal parameters, and kinematic attributes (including velocity and acceleration) [5].

4D aircraft trajectory is the information used to describe the motion status of an aircraft. Through the distribution, sharing, and collaborative management of 4D trajectory information, real-time or near-real-time situational awareness of the entire aircraft operation process can be achieved. On this basis, trajectory information-based collaborative decision-making, thereby improving the flexibility of flight plan adjustments, the utilization of airspace and airport resources, and the air transportation system safety. Enhanced operational efficacy becomes particularly pronounced in congested airspace corridors and busy airport zones requiring intensive flight coordination [6]. Enhanced trajectory prediction (TP) precision constitutes the essential foundation for optimizing aircraft separation minima, increasing airspace capacity, improving transportation efficiency, and ensuring air transportation safety. In the TBO environment, measures to enhance the capabilities of the ATM system, such as traffic synchronization, orderly organization of traffic flow, flexible airspace capacity management, dynamic adjustment of flight trajectories, and entrusting aircraft separation decisions to crew members when conflicts may occur, all place higher demands on the accuracy of aircraft trajectories (especially current and future positions). Since TP is an indispensable and essential function of most (if not all) ATM decision support tools (DSTs) [4], TP is not only an important support for the new generation of ATM systems but also proving instrumental in advancing the current ATM infrastructure. To provide a clearer picture of its practical significance, the applications of aircraft TP are categorized in Table 1.1, followed

by illustrative case studies.

Table 1.1 Categorization of aircraft trajectory prediction applications.

Application domain	Specific application/function	Application timing/phase	Key TP requirement
Air traffic control (ATC)	Conflict detection and resolution	Tactical/real-time (0-20 min)	High accuracy, low latency
	Arrival/departure management	Tactical to pre-tactical (20 min - 2 hours)	High accuracy in terminal area
	Traffic flow management	Strategic to pre-tactical (2 - 8+ hours)	Capacity & demand prediction
Flight deck/cockpit	Airborne conflict avoidance	Tactical/real-time (0-5 min)	Extreme reliability, integrity
	Station keeping (interval management)	Tactical/real-time	High precision & update rate
Airline operations	Fleet management & scheduling	Strategic (days before flight)	Aggregate fleet trajectories
	Flight (re-)planning & rerouting	Pre-tactical to tactical	Fuel & time optimization
	Estimated time of arrival (ETA)	Entire flight (en-route to arrival)	Accuracy at meter fixes

1). Case study 1: Metroplex arrival management

In congested terminal areas like the New York or Shanghai Metroplex, multiple airports operate in close proximity. Arrival streams from different directions must be merged efficiently onto final approach paths. Here, highly accurate short-term TP (e.g., 5-10 minute look-ahead) is indispensable for tools like Time-Based Metering (TBM). The controller uses TP to calculate precise arrival times at the runway threshold and issues speed adjustments to aircraft to maintain optimal

separation and sequence, thereby maximizing runway throughput while minimizing vectoring and holding delays. Any significant error in the TP can lead to missed spacing, requiring last-minute manual interventions that reduce capacity and increase controller workload.

2). Case study 2: In-flight trajectory negotiation (4D trajectory based operations - TBO)

The core of TBO, as mentioned in the introduction, is the negotiation and adherence to a precise 4D trajectory (latitude, longitude, altitude, time). Consider a scenario where an aircraft needs to change its flight path mid-flight to avoid weather or save fuel. The airline's operations center would generate a new, optimized 4D trajectory and submit it as a formal amendment request to the ground-based ATM system. The system must then evaluate this proposal against all other active and planned trajectories. This conflict detection process relies entirely on medium-term TP (e.g., 20-60 minutes) for all affected aircraft. The accuracy and, crucially, the uncertainty quantification of these predictions determine whether the new trajectory can be safely approved. Poor TP could either lead to rejecting efficient trajectories (reducing flexibility) or approving potentially conflicting ones (compromising safety).

TP is, therefore, a hot topic in current research on ATMs. Unless otherwise specified, the term TP refers specifically to aircraft TP in this study.

Augustin Degas et al. [7] collected articles in ATM published in English from 2010 to the end of December 2021 from relevant top-level conferences and journals. The analysis results are shown in Figure 1.1. Compared with optimization, analysis, and simulation, the number of articles on prediction is the largest, and the growth trend is the most significant. Although the prediction direction is not limited to TP, such as traffic flow or runway occupancy time prediction, it is closely related to TP. In particular, TP is redefined as the prediction of the entire door-to-door trajectory, and many related predictions can be included in the scope of TP. The analysis results show that TP is a hot research issue in ATMs. In summary, it is necessary to research aircraft TP methods.

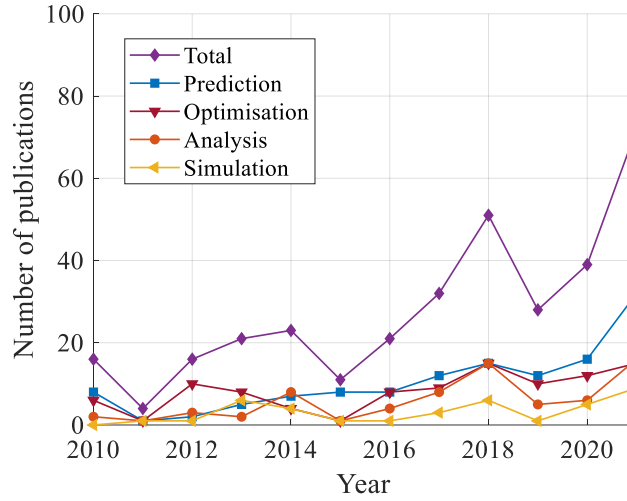


Figure 1.1 Evolution of the number of papers published each year in different directions of the ATM field.

This study examines deep learning and its application characteristics in aircraft TP. From the perspective of business applications for TP methods, it investigates the technical features of deep learning-based TP systems. It proposes a technical framework that includes system composition, construction and application, an evaluation index system, and typical structural patterns of hybrid TP models. Given the notable advantages of the recent research achievement of conditional tabular generative adversarial networks (CTGANs) in handling non-Gaussian distributions and imbalanced tabular data, which align with the characteristics of aircraft trajectory data, an exploratory study on CTGAN-based TP was conducted. This involved assessing CTGAN's application potential in TP, investigating its application methods for medium- to long-term predictions, and exploring its capability to enhance minority-class sample data, effectively improving prediction accuracy under adverse weather conditions. The studies based on CTGAN demonstrate its potential and typical application methods in TP. The proposed technical framework for deep learning TP serves as a reference for applying deep learning methods in this field, ultimately promoting the development of TP technologies and their applications.

1.2 Main research problems at present

While deep learning has demonstrated promising results in aircraft TP, its

practical implementation remains largely experimental and requires further validation. A relatively complete theoretical and technical system has not yet been established, which cannot effectively guide research practice, promote innovation and development, and promote the substantial improvement and practical application of TP capabilities. The existing TP accuracy, mainly the medium- and long-term TP accuracy, is challenging in meeting the requirements of TP for the rapidly growing demand for air transportation. An effective TP method has not yet been formed for problems such as sample data imbalance in severe weather. These are technical problems that urgently need to be solved and are also the main content of this article. Of course, for an industry like ATM that takes safety as its primary principle, there is almost zero tolerance for errors, and the requirements for mistakes are rigorous. Like emerging technologies, deploying deep learning for TP at scale necessitates resolving critical technical barriers in robustness, interpretability, and operational integration.

1.2.1 TP technology framework based on deep learning

TP is the core technology of modern ATC automation systems. It is the essential function of ATM DSTs.

TP serves as a foundational component in contemporary ATC automation frameworks and a critical operational capability within ATM decision-support ecosystems. Such as demand and capacity balance of air traffic flow management (ATFM) automation system, traffic synchronization of the arrival management system, conflict management (CM) of conflict detection system (conflict detection-CM), decision unit of flight planning system (flight planning-AUO), and decision unit of flight management system (flight management-AUO). Different application scenarios usually have different requirements for the operating environment, usage, function, and performance of TP tools. Although the prediction models, implementation technologies, and product forms of different TP tools may differ, the critical factors involved in building a TP system, the principles for handling problems, the primary data flow, and other logical methods are consistent. Research on basic technical architecture can provide methodological and technical guidance and references for designing and implementing different TP tools. At present, ICAO has proposed a basic composition framework and data process for TP consisting of four modules:

preparation, calculation, update, and output. Some authors have also proposed an evaluation framework and indicator system for aircraft TP models and a model based on machine learning (ML). However, the current framework remains fragmented in implementation and lacks systematic alignment with deep learning's architectural cohesion and domain-specific learning paradigms. Deep learning is a new technology with good application prospects and is attracting much attention in this field. Advancing foundational theoretical studies to establish a deep learning -driven TP framework will critically enhance deep learning's practical applicability in this domain and facilitate collaborative innovation through research sharing.

1.2.2 TP accuracy

The sustained expansion of global air traffic demands enhanced optimization of aviation resource utilization, particularly airspace efficiency, while requiring aircraft TP systems to achieve superior reliability and precision. Based on higher TP accuracy, the safe separation distance of aircraft can be reduced, the airspace capacity can be effectively increased, and the refined management level of ATM can be improved, providing critical technical support for significantly improving air transportation capacity. Due to the complexity and uncertainty of the environment, the current TP is difficult, especially in the terminal area, and current prediction systems struggle to achieve the precision demanded by the exponential growth in air traffic operations. The method based on physical mechanisms can obtain higher prediction accuracy in short-term prediction. Due to the uncertainty of various environmental factors, whether a method based on a physical mechanism or a method based on ML, the prediction accuracy usually drops sharply as the prediction time span increases. Improving the accuracy of TP has always been the unrelenting goal of the aircraft TP field, significantly improving the accuracy of medium- and long-term TP, which is a significant problem currently faced.

1.2.3 TP based on minority-class sample data

Severe environments such as electromagnetic solid storms and severe convective weather significantly impact air traffic. They are environmental factors that must be focused on in TP. Since flight data under severe weather conditions are relatively scarce, the proportion of historical trajectory data under normal weather

conditions is tiny, which belongs to minority-class sample data characterized by data imbalance in the sample data set. Existing TP models often perform poorly in learning the features of minority-class sample data in imbalanced data. This is because general data-driven methods usually treat minority-class sample data as outlier data, or it is difficult to capture the characteristics of minority-class sample data due to insufficient sample numbers, resulting in decreased prediction accuracy. There is no research report on solving the data imbalance problem faced by TP under severe weather conditions. Conducting TP based on minority-class sample data and solving the problem of TP under severe weather conditions will help improve the scenario adaptability of TP, make up for the shortcomings of TP, and enhance the TP model system.

1.3 Research content and technical approach of the thesis

1.3.1 Main research contents

(1) TP technology framework based on deep learning

This contemporary AI methodology, which has demonstrated itself as a scalable and effective big data solution across diverse domains, is increasingly being adopted for operational deployment in industry applications. The application of deep learning in aircraft TP is still in the exploratory research stage. Research on the application characteristics, application methods, and system characteristics of deep learning-based aircraft TP methods will help promote the field development. The study on the deep learning TP technology framework aims to summarize relevant knowledge and technologies to form systematic expertise and provide a reference for developing and applying TP technology based on deep learning. The research content includes system composition, system construction and application, system evaluation index system, TP hybrid model structure, and integrated application of the TP model.

(2) Medium- and long-term TP methods

The low accuracy of medium- and long-term TP has always plagued the industry. Methods based on physical mechanisms cannot accurately capture the uncertainty of aircraft maneuvers in the long term, which will cause significant errors. The RNN-based method commonly used in TP is limited by chaos phenomena, and

prediction reliability degrades exponentially with extended temporal horizons. Given that generative adversarial networks (GAN) supports unsupervised training, its generator implicitly learns the distribution of real data and can be regarded as an implicit density estimation method. The accuracy of its generated data is not independent of the prediction time span, and it has the primary conditions for application in medium- and long-term TP. The main research content includes exploring the application method of GAN in aircraft TP, designing a GAN-based aircraft TP model based on the characteristics and requirements of medium- and long-term predictions, and experimentally verifying its ability in medium- and long-term aircraft TP.

(3) TP method under severe weather conditions

Weather is an essential concern for flight, and severe weather significantly impacts ATM. Compared with average weather, flight data under severe weather conditions is relatively scarce and belongs to minority-class sample data, which causes data imbalance in sample data. The current TP method does not take special processing measures for minority-class sample data, making it challenging to implement TP effectively. To address this issue, we carried out research work on minority-class sample data enhancement, including minority-class sample data enhancement methods, the impact of different minority-class sample data ratios on prediction, the impact of various data enhancement methods on prediction, and the design and experiment of TP methods under severe weather conditions.

1.3.2 Technical route

The overall research plan of this thesis is illustrated in Figure 1.2. The fundamental idea of the research is as follows: first, based on the characteristics of deep learning TP, the study investigates the fundamental principles and methods for building applications, forming a technical framework (Chapter 3) to guide subsequent research. Building on this, the research addresses medium- to long-term TP by exploring the technical pathways for applying new AI research outcomes, discovering that CTGAN's advantages align well with several key characteristics of aircraft TP, indicating strong application potential (Chapter 4). To further leverage CTGAN's potential, a hybrid model combining clustering algorithms with improved CTGAN is constructed to improve medium- to long-

term TP accuracy (Chapter 5). In response to the challenges of predicting trajectories under adverse weather conditions, the study fully utilizes the improved CTGAN's strengths in data generation, exploring data augmentation methods to tackle the technical challenge of sample data imbalance in aircraft TP.

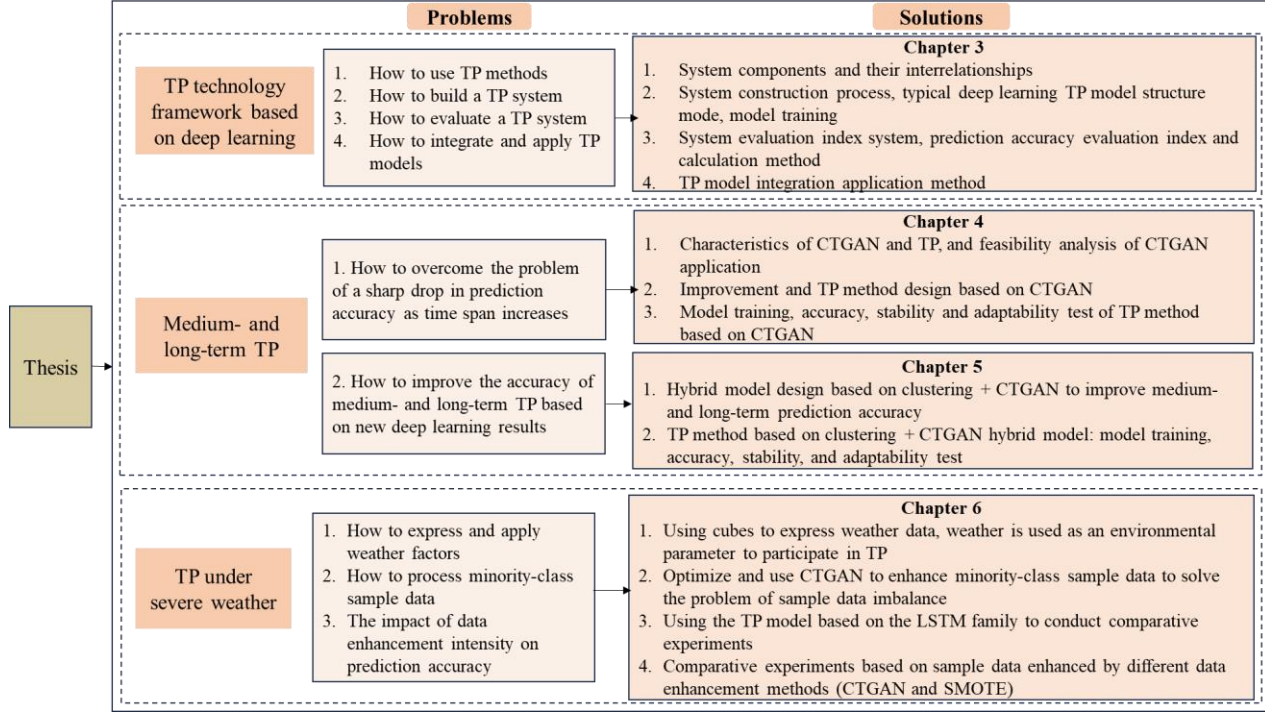


Figure 1.2 Overall research plan of the thesis.

1.4 Organization of the thesis

This thesis consists of 7 chapters, organized as shown in Figure 1.3.

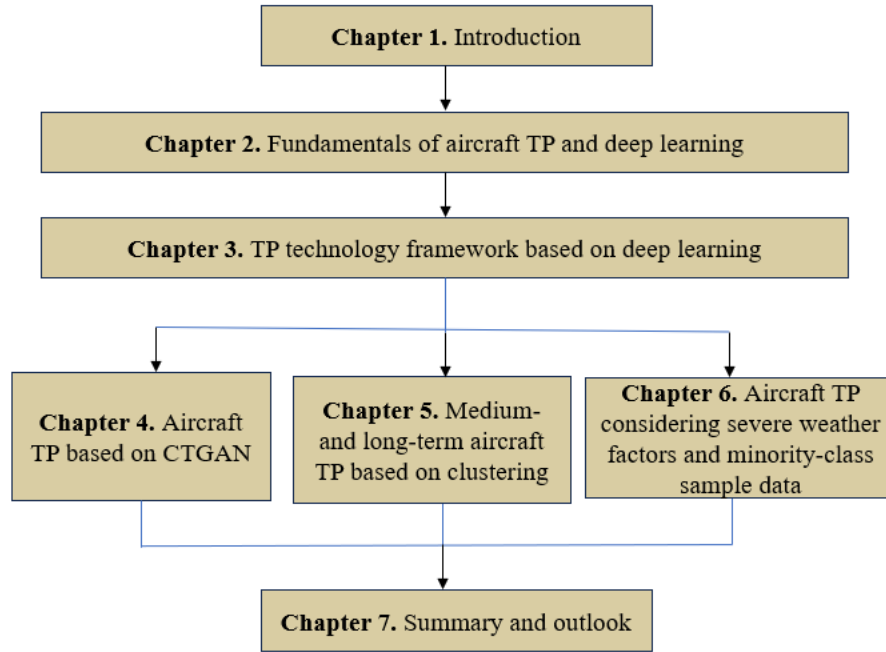


Figure 1.3 Thesis organization.

Chapter 1: Introduction

This chapter examines the operational requirements for aircraft TP, clarify the research significance. It further identifies systemic limitations in current TP paradigms and outlines this work's methodological innovations and integrative technical strategies to address these challenges.

Chapter 2: Fundamentals of aircraft TP and deep learning

This chapter introduces the basic knowledge of aircraft TP and summarizes the concepts, frameworks, and requirements related to TBO. It also presents the current status of aircraft TP research, typical TP models based on physical mechanisms, standard deep learning networks, and supporting data for deep learning TP.

Chapter 3: TP technology framework based on deep learning

This chapter approaches the TP system from a business operations perspective, proposing the fundamental components of the system, along with the construction and application processes and main tasks involved. A comprehensive evaluation index system is presented based on research into the TP system's evaluation framework. Additionally, it summarizes typical structural patterns and parameter

optimization methods for hybrid TP models based on deep learning and principles and techniques for their integrated application, forming a technology system for aircraft TP based on deep learning.

Chapter 4: Aircraft TP based on CTGAN

This chapter analyzes typical deep learning models' main characteristics and application directions, such as long short-term memory (LSTM), GAN, and Transformer in TP. It provides a detailed introduction to the characteristics of CTGAN, designs a TP method based on the improved CTGAN, conducts comparative experiments on TP, and presents a thorough comparative analysis of the experimental results.

Chapter 5: Medium- and long-term aircraft TP based on clustering

This chapter focuses on improving the accuracy of medium- and long-term TP by designing a hybrid model that combines a clustering algorithm with the improved CTGAN network. It conducts comparative experiments against models such as LSTM, convolutional neural network (CNN)-LSTM, CNN-LSTM-Attention, CNN-BiLSTM, and the original CTGAN, providing a detailed comparative analysis of the experimental results.

Chapter 6: Aircraft TP considering severe weather factors and minority-class sample data

This chapter addresses the significant impact of severe weather on aircraft TP and the challenge of limited sample data. It emphasizes research on methods and techniques for generating minority-class sample data under severe weather conditions. Initially, it uses an improved CTGAN to enhance minority-class sample data and conducts TP comparative experiments using standard models like LSTM, CNN-LSTM, CNN-LSTM-Attention, and CNN-BiLSTM. Additionally, it employs other methods to enhance minority-class sample data, conducting TP comparative experiments based on data enhanced by different methods and presenting a detailed comparative analysis of the results.

Chapter 7: Conclusion and outlook

This chapter synthesizes the core contributions of this research, delineates key

methodological advancements in deep learning-driven TP, and proposes targeted directions for subsequent investigation to address unresolved technical and operational gaps.

Chapter 2.

Fundamentals of aircraft trajectory prediction and deep learning

This chapter serves as the theoretical and technical foundation of the entire study. We can further explore deep learning-based TP methods by clarifying the new requirements for TP in a TBO environment, typical TP models, available deep learning models, and data resources. This chapter mainly investigates and summarizes these foundational issues.

2.1 Fundamentals of aircraft TP

Aircraft TP generates a 4D trajectory for an aircraft from gate to gate based on input conditions. To address the increasing demand for air transport capacity, the ICAO has promoted the construction of a system according to the global ATM concept. TBO is the cornerstone of this concept. In a TBO environment, the importance of TP is further emphasized, and it imposes higher demands on prediction accuracy and reliability.

2.1.1 Relevant concepts in aircraft TP

An aircraft trajectory represents the aircraft's motion in both airborne and ground operations, defined by parameters such as position, time, and derived kinematic attributes like velocity and acceleration. It integrates multidimensional data, including 3D spatial coordinates (latitude, longitude, altitude), temporal data (time), and calculated or inferred dynamic variables. This primarily involves continuously recording or modeling the aircraft's path during all phases of flight and ground maneuvers, ensuring comprehensive operational tracking and analysis [8]. A complete aircraft trajectory describes the motion state of the aircraft from gate to gate. Mathematically, a trajectory can be represented by a series of time-ordered trajectory state vectors or the geometric shape of the flight path. Each state vector encapsulates the attributes of a trajectory point, including basic information

across four dimensions: longitude, latitude, altitude, and time, typically supplemented by additional attributes like heading, speed, and acceleration. This comprehensive representation is commonly referred to as a 4D trajectory.

According to the ICAO, aircraft TP calculates and forecasts an aircraft's future position, heading, and other states. TP depends on integrating real-time aircraft status and planned flight intent, combined with predictive models for performance characteristics, meteorological factors, and airspace configuration data to estimate future flight paths [9]. TP is essential for ATM systems, enabling critical functions like conflict detection and resolution (CD&R), traffic flow optimization, and navigational guidance. TP models are at the core of TP and research, significantly determining the effectiveness of trajectory forecasting. These models' successful operation and performance depend on other factors, such as providing necessary input parameters from various components. To address this, the Federal Aviation Administration (FAA)/Eurocontrol Action Plan 16 (AP16) introduces a standardized TP framework structured into four interconnected modules: preparation, computation, updating, and output. These modules establish clear relationships and procedural workflows to ensure systematic trajectory prediction [9].

Research on aircraft TP methods typically encompasses two main aspects: the design and application of TP models. For deep learning-based aircraft TP methods, this includes model design (structuring the neural network and its interrelations), model training (determining model parameters), and model application (executing TPs). The deep learning-based aircraft TP model establishes a neural network structure and parameters suited for specific application scenarios, manifested as a runnable software module known as the trajectory computation engine.

2.1.1.1 Stages of Aircraft trajectory

An aircraft trajectory is a collection of spatial position coordinates and corresponding times experienced at all waypoints throughout the flight process. This can be defined as the aggregation of flight paths spanning from the departure gate at the origin airport to the arrival gate at the destination airport. Specifically, this includes seven stages: ground taxiing, takeoff and departure, climb, cruise, descent, approach, landing, and taxiing to the gate [10], as illustrated in Figure 2.1.

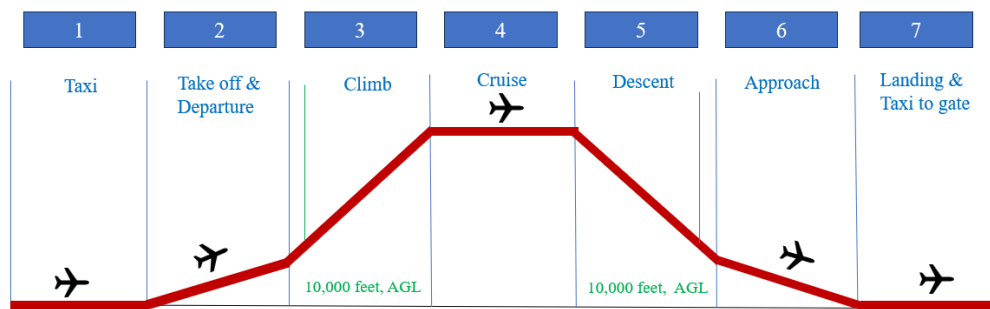


Figure 2.1 Stages of the entire aircraft trajectory.

Ground taxiing phase: This phase begins when the aircraft moves from parking and continues until takeoff. It includes pushback, taxiing, and proceeding to the takeoff point.

Takeoff and departure phase: The aircraft takes off from the ground. It involves operations such as the takeoff roll, acceleration, and becoming airborne.

Climb phase: This phase occurs after takeoff and continues until the aircraft reaches cruising altitude. The aircraft ascends vertically to its cruising height during this stage while gradually accelerating.

Cruise phase: In this phase, the aircraft flies horizontally at cruising altitude. Following a predetermined flight path, the aircraft maintains a constant altitude and speed.

Descent phase: This stage involves the aircraft descending from cruising altitude to approaching altitude near the airport. The aircraft gradually reduces speed and descends vertically to landing altitude.

Approach phase: In this phase, the aircraft enters the airspace near the airport and prepares for the final approach and landing. The aircraft adjusts its heading and altitude, entering a predefined path known as the "glideslope" to ensure a smooth landing. The approach phase is divided into three sequential sub-phases: initial, intermediate, and final.

Landing and taxiing to the Gate Phase: This phase encompasses the period from when the aircraft touches down on the runway until it taxis to the designated gate or parking position. During this stage, the aircraft lands on the runway, decelerates

to a safe speed, and then taxis along the airport taxiways to the gate or parking position.

Given the different environments faced during various flight phases, the influence of environmental factors on TPs varies. Therefore, constructing different TP models tailored to each phase is often necessary to meet specific prediction requirements. Each phase must adhere to relevant operational regulations, procedures, and air traffic controller instructions.

Factors such as airspace capacity, weather conditions, route congestion, aircraft performance, airline operational procedures, and pilot preferences play significant roles during the cruise phase. Other phases are also influenced by the airport's geographical environment, airport facilities and capacity (e.g., runway numbers), and airport operational and management capabilities. The impact of these factors on different phases can vary considerably, making it challenging to predict an entire trajectory with a single model. To reduce complexity, the ATM research community combines several closely related phases into four main research stages: airport, approach, enroute, and regional [11]. Trajectory modeling and prediction are typically conducted separately for each stage. The airport and approach stages correspond to the TMA, guided by airport tower ATC services. The airport stage encompasses taxiing, takeoff and departure, and climb phases, while the approach stage includes descent, approach, and taxiing to the gate. The terminal airspace is complex, characterized by high flight flow density and frequent changes in aircraft attitude, presenting challenges for high-precision and reliable TPs, which are a focus for decision support tool development [12]. The enroute stage corresponds to the cruise phase and is typically modeled and predicted as a standalone stage. The regional stage pertains to various segments of ATC zones along flight routes, primarily used for monitoring and managing traffic within these airspace sectors.

2.1.1.2 Classification of aircraft TP

Aircraft TP can be classified by time scale into short-term and medium-to long-term predictions [13]. Short-term TP covers intervals of minutes or less, focusing on immediate conflict risk detection and resolution. It relies less on long-term intent or weather data but assumes fixed maneuvers (e.g., constant turn rates), with accuracy improving as intervals shorten. Medium- to long-term TP spans 10+

minutes, requiring comprehensive data on flight intent, environmental factors, aircraft performance, and navigation. However, uncertainties in these inputs reduce accuracy over extended timeframes. While short-term TP enables real-time safety measures, medium- to long-term TP supports strategic planning for flight routing, fuel estimation, and airspace management.

TP is categorized as deterministic or probabilistic based on result formats [14]. Deterministic approaches, including nominal and worst-case methods, directly output predicted trajectories. Nominal methods cannot fully account for uncertainties in aircraft behavior, leading to reduced accuracy over extended prediction periods. Worst-case methods conservatively assume the aircraft may execute any maneuver within a predefined set, generating predictions under worst-case scenarios. In contrast, probabilistic methods model trajectory uncertainties using probability density functions (PDFs), offering enhanced accuracy for long-term predictions compared to deterministic techniques.

TP is categorized by timing into strategic/pre-tactical and tactical predictions [11]. Strategic predictions span from one year to one week before the operational day, while pre-tactical predictions cover the final week prior. Both focus on pre-takeoff trajectories, incorporating flight plans, weather trends, traffic flow, aircraft performance, and historical patterns to assess fuel consumption and airspace flow. Tactical predictions, executed post-takeoff, integrate dynamic data such as real-time flight status, route-specific weather forecasts, and traffic density to support operational decisions and conflict warnings.

TP is classified into ground-based and aircraft-based systems depending on deployment location. The ground-based TP employed by ATC operates from ground infrastructure and utilizes surveillance radar or ADS-B data to estimate aircraft status and environmental conditions as model inputs. Integrated into FMS, aircraft-based TP relies on onboard sensors to acquire real-time parameters. Data from aircraft systems, such as flight status and weather updates, often offers greater timeliness and accuracy compared to ground sources. Recent studies focus on harmonizing outputs from both approaches to enhance consistency [15].

TP is categorized into physics-based and deep learning-based approaches based on underlying technologies. Physics-based TP utilizes principles of kinematics

and dynamics, employing classical methods such as state estimation and dynamic modeling. These methods often require extensive input parameters and state data, some of which are challenging to acquire or estimate accurately due to their dynamic nature. Simplifying assumptions are frequently applied to reduce complexity, but uncertainties in input data can compromise prediction reliability. Deep learning-based TP, in contrast, bypasses the need for prior knowledge or assumptions about aircraft parameters by autonomously extracting trajectory features and identifying hidden patterns from historical data. This data-driven approach has emerged as a novel research focus, offering advantages over traditional physics-based methods [16].

TP is classified into two types based on model/algorithm integration: standalone model-based and hybrid model-based approaches. Basic model/algorithm-based prediction: This method relies on a single model or algorithm for TP, such as those based on GANs, RNNs, CNNs, or sequence-to-sequence generative networks (Transformers). Hybrid model-based prediction: This approach combines two or more models/algorithms to achieve TP, such as methods that integrate RNN and CNN or Attention mechanisms with RNN.

2.1.1.3 TP accuracy requirements

TP is widely applied in various ATM operations, including flight planning, anomaly monitoring, traffic management, arrival management, conflict detection, and aircraft operations. Different application scenarios impose varying accuracy requirements on TPs. Aircraft navigation systems are essential for determining an aircraft's position in the air and play a critical role in ensuring air traffic safety and orderly operation. While navigation accuracy requirements are not explicitly designed for TP, they serve as a valuable reference for trajectory accuracy standards [17]. When TP accuracy aligns with navigation accuracy, the results can be directly used for navigation. Therefore, the accuracy required for navigation can be seen as a fundamental target for TP, particularly in short-term predictions for conflict warning and resolution.

The ICAO introduced performance-based navigation (PBN), an operational framework that combines area navigation (RNAV) and required navigation performance (RNP) procedures and standards. To facilitate this concept

implementation in actual operations, ICAO published the "PBN manual" in 2006, with ongoing updates. The fourth edition, published in 2013, outlines the accuracy requirements for TPs under various navigation standards [18], as shown in Table 2.1.

Table 2.1 Application of flight phase navigation.

Navigation Specifications	Flight phase							
	Oceanic/remote land routes	Overland air routes	Descent	Approach				Departure
				Initial	Intermediate	Final	Go-around	
RNAV 10	10							
RNAV 5		5	5					
RNAV 2		2	2					2
RNAV 1		1	1	1	1		1	1
RNP 4	4							
RNP 2	2	2						
RNP 1			1	1	1		1	1
A-RNP	2	2 or 1	1	1	1	0.3	1	1
RNP APCH				1	1	0.3	1	
RNP AR APCHH				1-0.1	1-0.1	0.3-0.1	1-0.1	
RNP 0.3			0.3	0.3	0.3		0.3	0.3

The table values represent the lateral navigation accuracy requirements that must be met over 95% of flight time, measured in nautical miles (n mile). RNAV and RNP are both area navigation specifications, with RNAV not requiring onboard monitoring and alerting capabilities, while RNP does. The specifics of each standard are as follows:

- 1). RNAV 1: A navigation standard applied to routes requiring area navigation capabilities under air traffic services (ATS), also supporting instrument approach procedures beyond the final approach fix.
- 2). RNAV 2: Similar in scope and requirements to RNAV1 but allows for longer inertial reference unit (IRU) cruise times, more commonly applied in en route phases.

- 3). RNAV 5: Uses DME/DME, DEM/DEM/IRU, or GNSS for navigation and can be applied for standard instrument approaches starting from 30 nautical miles from the airport reference point (ARP) and above the minimum sector altitude.
- 4). RNAV 10: Utilizes GNSS as the navigation source, primarily for routes over oceans and remote areas.
- 5). RNP 1: Limited to standard instrument approaches, standard instrument departures, and initial climb segments following the missed approach.
- 6). RNP 2: Primarily used in en route phases, especially over oceans/remote areas and land.
- 7). RNP 4: Focused on en route phases, particularly in ocean/remote areas.
- 8). A-RNP: Utilized in complex flight environments with stringent airspace management needs, applicable in busy airspace or complicated flight conditions.
- 9). RNP APCH: Specifically for navigation performance standards during the approach phase, aimed at providing more precise and safer approach paths.
- 10). RNP AR APCH: Used for the approach and landing phases, particularly suited for airports with complex terrain or restricted airspace.
- 11). RNP 0.3: Primarily targeted for helicopter operations.

2.1.2 TP based on trajectory-based operations

2.1.2.1 Trajectory-based operations concept

(1) Evolution of air traffic management

The current ATM system operates within sectorized airspace and relies on predefined flight routes, with airspace organization dictating the regulation of flight paths and segments. When traffic demand surpasses handling capacity, controllers may implement ground holding measures, in-flight adjustments (e.g., speed reduction or holding patterns), or rerouting options to manage congestion.

Modern operational improvements primarily originate from PBN-enabled automated flight route planning and execution, optimizing airspace organization, traffic management, and runway operations. Since ICAO's 1998 proposal for a global ATM modernization framework [3], major programs like the U.S. NextGen,

Europe's SESAR, and Japan's CARATS have been implemented to enhance aviation safety and efficiency. These initiatives adhere to the TBO framework.

The evolution of ATM from an ATC standpoint is categorized into three phases: procedural control (historically applied using estimated aircraft positions), surveillance-driven control (current practice using real-time positional data), and TBO, which anticipates future aircraft positions. TBO represents a foundational shift in next-generation ATM systems, addressing aviation's critical challenge of balancing capacity and demand by optimizing predictability and resource allocation [15].

(2) Air traffic management based on trajectory-based operations

The ICAO's ATM operational concept (Doc 9854) defines the framework for advancing global and regional ATM systems, establishing the TBO framework. Under TBO, every flight operates with a shared and managed trajectory, serving as a unified flight plan. Air traffic stakeholders—including controllers, airlines, airports, and aircraft operators—leverage the System-Wide Information Management (SWIM) platform to access real-time trajectory data, enabling collaborative decision-making and tactical adjustments across all phases of flight operations.

To ensure this transition is coherent, unified, and integrated, the global ATM community requires a common baseline and framework to guide the development and implementation of interconnected capabilities and processes. In 2012, the ICAO approved the fourth edition of the global air navigation plan, which introduced the aviation system block upgrade (ASBU) approach, proposing air navigation policies and establishing a technical roadmap to support necessary developments. Subsequently, the sixth edition of the global air navigation plan (GANP) was released in 2019, dividing improvement areas into information (weather, intelligence, system-wide information management, and collaborative flight and flow), operations (flow management, separation management, surface operations, runway sequencing), and technology (communication, navigation, surveillance). It proposed an evolutionary pathway toward comprehensive implementation of TBO, positioning TBO as the ultimate integration and realization goal across various operational domains, with plans for global

application beyond 2031.

TBO, as a critical technology of the ASBU, can significantly enhance the predictability of air traffic flow, reduce airborne waiting times, decrease carbon emissions, minimize flight delays, and ensure smoother flights, thereby considerably improving global air transportation efficiency. Research and modernization plans for ATMs are underway in various regions worldwide. Although these plans differ, they align with the singular global vision articulated by the ICAO's global ATM operational concept. Fundamental conceptual changes are being realized in different parts of the world at varying levels of maturity (for example, dynamic and flexible airspace management and support for mixed capabilities). However, the evolution and transition to a TBO environment have yet to be fully achieved. Central to TBO is TP and synchronization. The current state of TP information sharing has systemic consequences, reflected in the suboptimal performance of discrete individual decision support system functionalities and inadequate system-level performance.

The global ATM framework integrates seven core elements: airspace organization and management (AOM), demand-capacity balancing (DCB), airport operations (AO), traffic synchronization (TS), conflict management (CM), airspace user operations (AUO), and service delivery management (SDM). These components are linked via System-Wide Information Management (SWIM) and operate on a unified trajectory grounded in collaborative decision-making (CDM) principles. Through coordinated interaction, stakeholders establish a mutually agreed trajectory that aligns with collective operational objectives, ensuring safe and efficient flight execution across all phases.

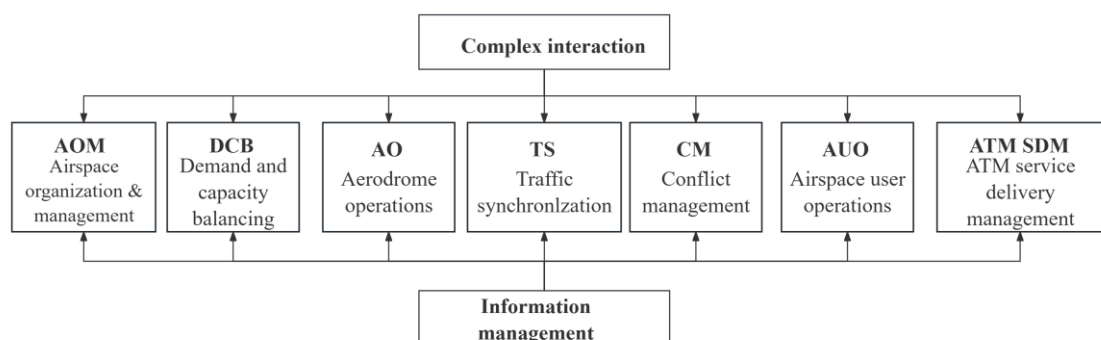


Figure 2.2 ATM concept components.

In the context of the global ATM operational concept:

AOM: Dynamically designs and allocates airspace structures to meet operational demands.

AO: Delivers essential services and infrastructure for air traffic management.

DCB: Assesses traffic demand and capacity across the system, applying measures to address mismatches. These actions, determined collaboratively, span strategic, pre-tactical, and tactical phases.

TS: Tactically organizes air traffic flows to ensure safety, order, and efficiency.

AUO: Encompasses flight planning, monitoring, and execution.

CM: Mitigates collision risks through three layers:

- 1). Strategic: Uses AOM, DCB, and TS to manage traffic flow and reduce separation needs preemptively.
- 2). Separation Provision: Tactically maintains minimum separation standards.
- 3). Collision Avoidance: Acts as a final safeguard if separation fails.

SDM coordinates and balances decisions related to ATM performance, services, and assets across the various conceptual components.

Figure 2.3 illustrates the TBO operational framework, highlighting that shared planning and consistent trajectories are essential information for interactions in a TBO environment, linking all stakeholders in air transportation and their decision support systems [15].

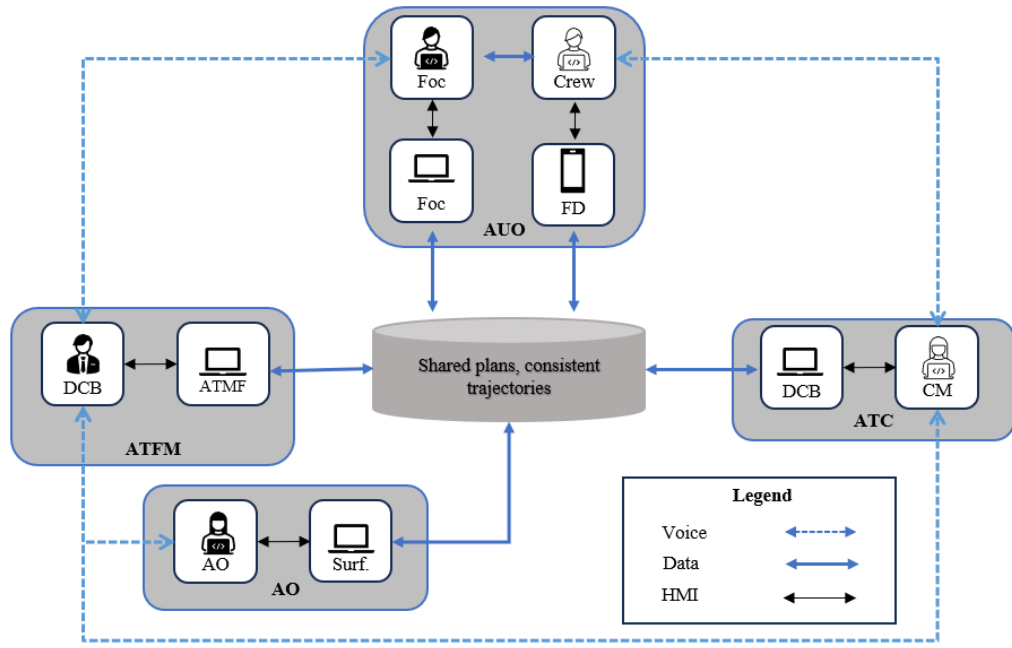


Figure 2.3 TBO operation diagram [8].

2.1.2.2 TP framework under trajectory-based operations

In the ATM system, various DSTs rely heavily on TP software. While all TP systems' design and implementation logic are similar, differences in operational environments, accuracy, uncertainty, response times, input data, and interaction modes lead to varying technological dependencies for each DST. This results in diverse TP systems coexisting within the ATM environment, posing potential challenges for comparative analysis, shared applications, and interoperability among automated systems.

To address these challenges, the FAA/Eurocontrol Action Plan 16 [9] proposed a universal TP model consisting of four modules: Preparation, computation, update, and output, as shown in Figure 2.4. The prediction process begins in the preparation module, which processes raw data collected from users to create the necessary input parameters for the TP model. The computation module then calculates the predicted trajectory based on these inputs and passes the results to the update module. This module checks whether the calculated trajectory meets the required constraints; if not, it restarts the process by activating the preparation or computation modules for further data processing or recalculation. Once the

predicted trajectory complies with the constraints, it is forwarded to the output module, which delivers the TP results to the user as needed.

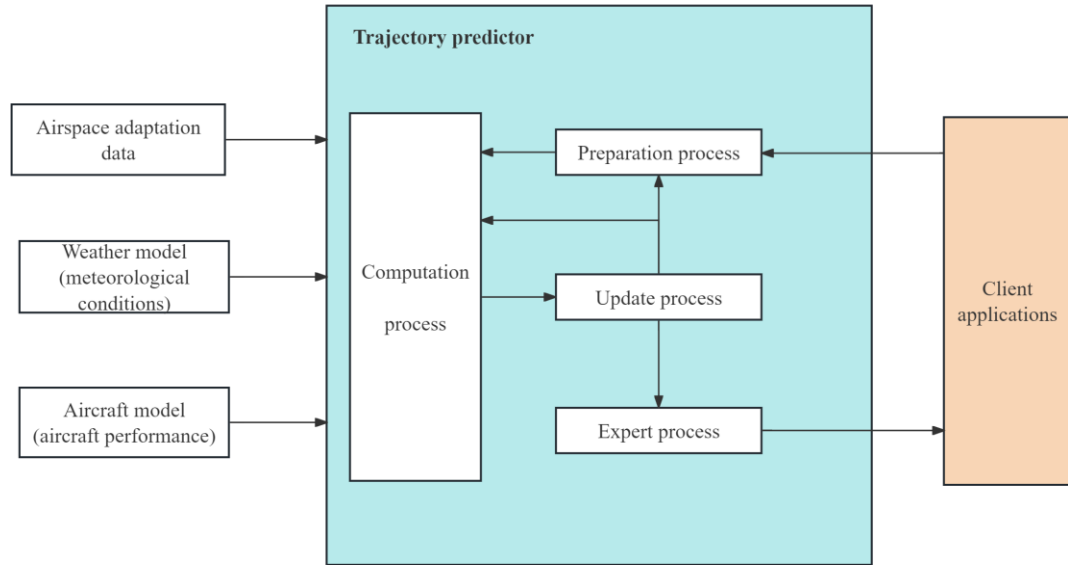


Figure 2.4 Trajectory predictor process flow.

The data flow for TP is illustrated in Figure 2.5. The TP process utilizes input parameters such as aircraft status, operational logic models, flight plans, airline protocols, weather-related ATC constraints, and performance metrics. These inputs are synthesized to create a behavioral logic framework and trajectory constraints. The behavioral logic defines sequential operational expectations for the aircraft, often termed "aircraft intent." The computational phase forms the core of TP, employing trajectory algorithms to integrate behavioral logic, weather conditions, performance data, and system adaptations for generating predicted trajectories. An iterative refinement mechanism adjusts behavioral models, constraints, and flight plans through periodic updates or by detecting deviations between predictions and real-time observations, potentially initiating reprocessing. Outputs include predicted trajectories, error margins, and data quality alerts for client systems [19]. TBO implementation requires precise 4D spatiotemporal coordinates (x, y, z, t) to represent aircraft positions, enabling synchronized sharing of real-time and planned flight states.

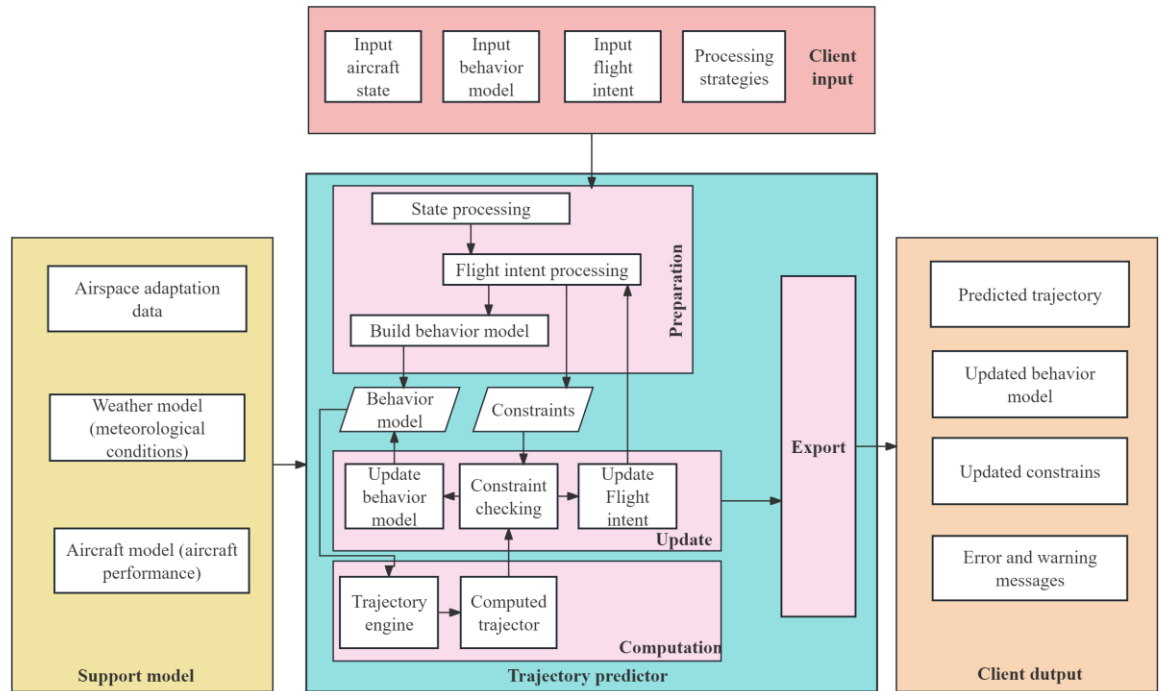


Figure 2.5 General TP data flow.

1). Support model

The support model incorporates three primary input categories:

- a). Adaptation data: Configuration parameters for system optimization, including airspace design specifications, aircraft performance calibrations, and geospatial models.
- b). Weather model: Generates forecasts of atmospheric conditions (e.g., wind, temperature) within the aircraft's operational airspace.
- c). Aircraft model: Defines performance characteristics (e.g., climb/descent rates, fuel efficiency) and operational limits.

2). Intermediate processing

The intermediate processing phase involves three key steps:

- a). Aircraft state processing generates initial parameters for trajectory calculations.
- b). Flight intent processing validates these parameters against flight intent constraints to detect discrepancies.
- c). Behavior model generation constructs a behavioral model by integrating initial

parameters and flight intent data.

These steps ensure accurate TP by aligning computational inputs with operational objectives.

3). Output content

The export process delivers TP results to client systems, including:

- a). Predicted trajectory: Forecasted 4D path (latitude, longitude, altitude, time) of the aircraft.
- b). Revised behavior model: Updated operational logic (if deviations between predictions and actual data are detected).
- c). Adjusted constraints: Modified operational limits (e.g., weather, airspace restrictions).
- d). Computation alerts: Errors or warnings generated during trajectory modeling.

This output lets clients synchronize decisions with real-time trajectory updates and system conditions.

2.1.2.3 New requirements of trajectory-based operations for TP

The global ATM operational concept under TBO is characterized by: CDM Based on trajectory information: This approach leads to a negotiated trajectory that embodies a shared intent among stakeholders. Dynamic and flexible characteristics: It enables a highly coordinated global ATM system to adapt to changing conditions. Redefined airspace: The airspace is structured according to traditional ATC routes and includes optimized flight paths users desire, often called free airspace. Comprehensive trajectory: The trajectory encompasses the entire journey from the departure gate to the arrival gate. Delegation of separation responsibility: Aircraft separation can be delegated to flight crews, alleviating the workload on air traffic controllers [15].

In the global TBO-based ATM system context, new requirements for TP have emerged, primarily manifesting in the following areas:

- 1). Higher accuracy requirements: The predicted trajectory describes the positions an aircraft must follow along its designated path at specified times. The accuracy

of trajectories generated by strategic predictions directly affects the rationality of flight plans within the entire air transportation system. Improving TP accuracy is crucial for enhancing airspace utilization and ensuring flight safety. In tactical predictions, various uncertainties may prevent the aircraft from flying precisely as planned, potentially leading to conflicts. Therefore, trajectory adjustments are essential, and increasing the predictability of trajectories is necessary to minimize these adjustments. Additionally, free airspace and delegated aircraft separation require higher TP accuracy.

2). Faster computation speed requirements: Trajectories are meant to be shared, which requires time. A faster TP speed can allow more time for sharing and managing trajectory information, thus promoting the rationality of negotiated trajectories.

3). Longer LAT requirements: maintaining a certain level of accuracy while extending the LAT for TPs enables the provision of future flight positions and status information over a broader time span. This supports the sharing and management of trajectory information, creating favorable conditions for conflict detection and resolution. Additionally, under TBO, users have the right to choose 4D trajectories, increasing traffic management's complexity. Longer prediction horizons are needed to reduce uncertainty, especially during the cruise phase [20].

4). More comprehensive trajectory requirements: In addition to the airborne phase, TP should also encompass the taxiing trajectory from the gate to the runway and from the runway to the gate. The TP should provide a complete trajectory from the departure gate to the arrival gate.

5). Higher stability requirements: TP relies on external conditions as inputs. When these external conditions change significantly, recalculating future trajectories is often necessary. Frequent trajectory recalculations are detrimental to the sharing and management of trajectory information. Therefore, there is a need to enhance the analysis and evaluation of how external factors impact trajectory accuracy to avoid unnecessary frequent TP calculations.

6). Higher uniformity and synchronization requirements: In a TBO environment, multiple TP software systems operate simultaneously. Maintaining consistency

among the predictions from different TPs and completing predictions within the same decision cycle aids in the sharing and managing of trajectory information. Improving the uniformity and synchronization of TPs is a new requirement for TP under TBO.

The requirements for TP in the global ATM system based on TBO are multifaceted. The importance of these requirements varies depending on the operational environment and the supported decision support systems. However, enhancing prediction accuracy remains a crucial objective in aircraft TP efforts.

2.2 Current status of aircraft TP research

As an essential function or component, aircraft TP emerged with the emergence of ATM decision support systems and has continued to develop with the deepening of research and new technologies. Aircraft TP has different classifications according to various classification principles. According to the time scale, it can be divided into short-term (less than 10 minutes time span), medium- and long-term predictions (equal to or more than 10 minutes time span). According to the form of the prediction, the results can be divided into deterministic and probabilistic predictions. Depending on whether the aircraft takes off, it can be divided into strategic and tactical predictions. The flight phase that is focused on can be divided into take-off, descent, and cruise phase predictions. According to the operating environment, it can be divided into airborne and ground systems. According to the technology it relies on, it can be divided into four methods: dynamics methods, state estimation methods, traditional ML, and deep neural networks [4], [15], [16], [21]. Since the prediction model that calculates and forms the trajectory is the core of TP, classifying according to the technical characteristics of the prediction model can better reflect the model's essential characteristics and development trend. Therefore, this study establishes a taxonomy of trajectory prediction methodologies through an analytical evaluation of model architectural attributes and operational paradigms. Since the state estimation method is based on the kinematic mechanism, it mainly studies objects' motion without considering the circumstances leading to the motion. It can be understood as a method based on kinematics [4]. For ease of

description, state estimation and dynamics methods are summarized as methods based on physical mechanisms, which are classic TP methods. This further divides TP methods into two categories: methods based on physical mechanisms and ML. For more complex application scenarios, combining and applying multiple TP models and methods to meet actual needs is usually necessary, so the integrated application of TP methods is also an essential part of TP research.

2.2.1 Dynamics methods

Dynamic TP methods analyze interactions between aerodynamic forces and aircraft motion. The core six-degree-of-freedom (6-DOF) differential equations mathematically define flight parameters. While model variants exist, all originate from the fundamental 6-DOF framework capturing positional and attitude dynamics. In ATM systems, critical parameters like control surfaces and inertial properties often lack measurement precision for detailed modeling. However, such precision proves unnecessary in practice. Simplified point-mass models (PMM) adequately meet operational requirements through controlled approximations [11-13].

The PMM represents a foundational TP approach. This methodology treats aircraft as concentrated masses, employing mechanical equilibrium analysis integrated with flight performance databases. Implementation requires validated aircraft performance data to characterize thrust-drag equilibrium and fuel consumption dynamics. EUROCONTROL-developed standards (BADA model) serve as key references for parameter integration. Given that thrust, drag, and mass exhibit altitude-dependent variations influenced by atmospheric conditions, dynamic parameter integration becomes essential to account for altitude-dependent variations in thrust and aerodynamic forces [25]. Synthesizing numerical frameworks, operational specifications, atmospheric variables, and flight intent enhances prediction precision [26], [27]. PMM predicts future aircraft trajectory over a time interval using integrodifferential equations [28].

Aircraft motion constitutes a stochastic hybrid system (SHS) with distinct operational phases, necessitating phase-specific dynamic formulations. Consequently, PMM emerges as the predominant SHS implementation framework [16]. Lympelopoulous et al. [29] developed PMM-based SHS

representations integrating physical motion continua with discrete navigational states from flight plans and flight management system (FMS). Lee et al. [16] formulated hybrid estimation algorithms for trajectory tracking and arrival prediction, establishing nonlinear dynamics for continuous flight phases and probabilistic state transitions between operational modes.

Intent information is a key factor in TP, and flying on the planned route is a basic principle that pilots should follow. Intent information is divided into flight intent and aircraft intent. For pilots, flight intent is ambiguous by definition and does not fully describe how the aircraft will operate during the flight. There are many different ways to perform a flight, and they all conform to a single rule of flight intent, each resulting in a different trajectory. Aircraft intent consists of structured instructions carefully processed by flight intent (such as planned route) and information such as flight procedure restrictions, aircraft performance, and airline operating preferences. It is a script that clearly describes how the aircraft is controlled [19]. Recent advancements focus on formalizing aircraft intent characterization through enhanced operational schemata [20-22]. Schuster et al. [17] developed an augmented flight scripting framework incorporating real-time telemetry data and dynamic intent parameters, replacing static BADA-derived configurations. This methodology achieved 12% accuracy improvements over required navigation performance standards across all flight phases. Concurrently, Felix et al. [16] devised a computational intent formalization system integrating compliance matrices, airline operational policies, and pilot decision protocols, ultimately synthesizing a unified trajectory prediction architecture with embedded performance-environment coupling.

The dynamics model relies on aircraft performance parameters as fundamental inputs. BADA serves as a performance parameter database for aircraft. Due to the large number of aircraft types stored and the continuous enrichment and improvement through crowdfunding, it is the most widely used aircraft performance parameter database, providing a comprehensive reference for aircraft TP [32]. Alligier et al. [33] improved the TP accuracy by learning some unknown PMM parameters from past observations and comparing them with the standard model based on BADA. Thippavong et al. [34] developed an adaptive algorithm that continuously adjusts the quality of the estimate by reducing the difference

between the observed and modeled energy rates to improve the climb TP accuracy. Baklacioglu et al. [35] developed an aviation propulsion model (APM) by optimizing transport aircraft flight data using a genetic algorithm for precise TP. Sun et al. [36] used various data mining methods and maximum likelihood estimation methods to build generative models of relevant performance parameters. Alligier et al. [37] employed stochastic gradient boosting trees to estimate unknown PMM parameters, thereby enhancing TP precision.

Weather is an important information that affects flight. Aircraft baseline operational characteristics exhibit altitude-dependent variations due to ambient meteorological parameters, particularly temperature/pressure gradients. Consequently, dynamic model calibration requires real-time atmospheric parameter assimilation to compensate for meteorological-induced performance deviations, thereby enhancing trajectory prediction fidelity. Porretta et al. [31] decomposed the wind field into nominal and random components. The nominal wind was simulated as a lookup table, and the random wind component represented deviations from the nominal wind field, calculated from measured or forecasted weather data samples. The participation of random wind as an influencing factor in the calculation improves the TP accuracy. Tang et al. [38] developed a meteorological calibration framework for extracting baseline operational profiles while performing route-specific forecast adjustments. By extracting the flight intention from the weather data, the randomness of the historical flight altitude distribution caused by meteorological factors is eliminated. Inter-altitudinal separation quantification was achieved through dynamic spatial warping (DSW) algorithms to derive baseline route-specific profiles. Meteorological grid (GRIB) assimilation implemented Cressman interpolation with aircraft-derived meteorological telemetry (AMDAR), achieving error reduction in forecast deviations. This integrated methodology enhanced strategic 4D TP resolution through synergetic profile-weather optimization.

AI progress has enabled wider application of ML techniques to enhance input parameter precision for point mass representations. For instance, Alligier et al. introduced ML methods to learn the best thrust law from the historical data and estimate the mass of a single aircraft from past trajectory points to predict speed

intentions [37], [39], [40]. Marinescu et al. [41] employed Gaussian process regression on automatic dependent surveillance–broadcast (ADS-B) data to decrease wind prediction inaccuracies by half at two designated flight waypoints. In addition, Lymperopoulos et al. [29] used probabilistic methods, Hadjaz et al. [42] applied online learning or adaptively adjusted some modeling parameters [43], [44], Tastambekov [45], Hrastovec and Solina, etc. [46], [47] used different ML techniques to estimate and predict aircraft performance. By improving the accuracy of PMM parameters, the TP accuracy is effectively improved.

While dynamic models play a vital role in TP, their predictive accuracy fundamentally depends on precise parameter inputs. These models must account for various elements, including aircraft specifications, operational states, meteorological factors, and flight intent, which involve proprietary data or face accessibility constraints. Parameters not classified as proprietary often rely on predefined configurations or database estimates, leading to potential inaccuracies in critical variables like wind patterns, aircraft weight, and ATC decisions. Insufficient data availability directly compromises parameter reliability, subsequently diminishing model effectiveness. Such input data variability amplifies uncertainty in the TP process [16]. Moreover, conventional TP approaches demonstrate reduced operational efficiency when handling high-volume real-time data streams in complex ATM environments [37].

2.2.2 State estimation methods

The state estimation approach employs kinematic principles and applies state-space theory to mathematically characterize moving objects' motion patterns. This methodology effectively models nonlinear systems describable by state-space formulations, finding extensive application in target tracking domains. Given that commercial airliners, particularly large aircraft, typically demonstrate gradual kinematic parameter variations, their motion states exhibit reliable short-term predictability. This characteristic makes it particularly suitable for TP research. Implementation requires establishing state transition matrices through motion-derived equations. Unlike performance-based models, aircraft TP focuses on future kinematic states rather than aircraft-specific parameters like mass and thrust, analyzing temporal relationships between positional, velocity, accelerative, and angular attributes and historical states. Based on operational mode assumptions

(single vs. multi-mode), two distinct TP frameworks—single-model and multi-model estimation—are developed to generate trajectory computations [16].

2.2.2.1 Single-mode estimation

Aircraft kinematic behavior encompasses distinct operational phases including departure, ascent, cruise maintenance, coordinated turns, and approach sequences. For computational tractability, model simplification typically disregards these multimodal characteristics, adopting single-mode assumptions for state estimation. The predominant single-mode analytical technique remains the Kalman Filter (KF) methodology [48], [49]. KF operates as a recursive computational framework for dynamic system state determination, predicated on Gaussian distribution assumptions for both system evolution disturbances and measurement uncertainties. Through iterative refinement of state vectors and covariance matrices, this algorithm delivers optimized real-time state approximations.

Contemporary TP methodologies extend beyond the foundational KF [41,42], incorporating advanced techniques like particle filtering [43], hidden Markov modeling (HMM) [44-47], and their derivative variants [54]. Chatterji's research [55] implemented KF for near-term trajectory projection, utilizing instantaneous ground speed and flight path angle data with kinematic equations to extrapolate positional states. This approach assumes fixed operational intent during brief intervals, potentially compromising accuracy during significant flight deviations. Addressing computational limitations in multi-aircraft scenarios, Lymperopoulos et al. [56] demonstrated conventional Monte Carlo methods' inefficacy in high-dimensional state estimation, subsequently developing an enhanced particle filter framework. HMM-based solutions have also gained prominence, as exemplified by Ayhan's work [52], which integrates meteorological variables with historical trajectory patterns to model the impact of environmental uncertainty.

2.2.2.2 Multi-model estimation

While unimodal approaches demonstrate proven efficacy in TP, their inherent limitations emerge when handling hybrid operational systems. These singular modeling frameworks typically treat aircraft dynamics as stochastic variables, projecting future states through current conditions while neglecting complex atmospheric interactions [21]. The conventional KF exemplifies this paradigm,

deriving posterior states through linear combinations of Kalman gains and innovation vectors. However, KF-based trackers exhibit heightened sensitivity to modeling assumptions—significant discrepancies between presumed and actual maneuvers generate substantial residual errors [9]. TP challenges essentially constitute stochastic linear hybrid system (SLHS) estimation problems necessitating multimodal solutions. SLHS frameworks decompose aircraft kinematics into discrete operational regimes, including constant velocity (CV), coordinated turn (CT), uniform acceleration (CA), and altitude-specific modes (CH/CHCA), each governed by simplified dynamic equations. Multimodal estimation enhances performance through parallel execution of mode-specific state calculators, though computational complexity escalates exponentially. This limitation drives the development of approximate solutions like interacting multiple models (IMMs), which synergistically combines multiple estimators (KF variants) through probabilistic weighting. IMM algorithms dynamically integrate divergent motion hypotheses, achieving robust state estimation with operational efficiency, thus becoming prevalent in modern TP implementations [16].

Scholarly advancements in IMM applications demonstrate marked performance enhancements. Li's team [57] achieved an order-of-magnitude reduction in positional root mean square (RMS) errors through IMM-optimized state estimation. Concurrently, Maeder's research [58] incorporated meteorological data into state equations to enhance trajectory forecasting precision. Song's innovative framework [59] developed a data-driven architecture combining historical trajectory analysis with flight intent recognition, enabling dynamic adjustment of mode transition probabilities for optimized state updates.

Multimodal algorithms traditionally conceptualize flight phase transitions through Markovian frameworks with fixed transition matrices decoupled from kinematic parameters [60]. Aviation practice reveals aircraft adhere to structured flight plans involving both discrete operational phase shifts (e.g., climb-to-cruise) and continuous kinematic states. This operational reality has driven scholarly consensus toward state-dependent transition models, where mode-switching probabilities correlate with real-time parameters like velocity and geospatial positioning, leveraging historical flight data repositories [53-56]. Challenging conventional Markov assumptions, Seah's team [62] developed a state-dependent

transition framework for multimodal KF, effectively addressing aircraft tracking challenges through continuous state-aware probability adjustments.

Current multimodal estimation frameworks face theoretical constraints in transition probability computation. Standard MM implementations presume zero-mean residuals for likelihood-based mode probability updates—an assumption frequently violated due to incomplete mode coverage [65]. This limitation has catalyzed methodological innovations across three dimensions: intention-aware prediction, uncertainty quantification, and probabilistic inference [58-61]. Yepes' team [67] developed an intent-driven TP framework that synergizes residual-mean IMM outputs with operational context analysis to determine optimal flight intent hypotheses. Parallel advancements by Thomas et al. [20] established stochastic TP methodologies incorporating atmospheric uncertainty parameters (wind vectors, thermal variations) through Bayesian probabilistic modeling, enabling probabilistic airspeed and performance parameter estimation via monitored flight data.

To achieve state propagation, conventional state estimation frameworks employ kinematic equations based on positional/velocity/accelerative parameters. While computationally efficient, these models exhibit diminishing reliability over extended time horizons due to maneuver uncertainty accumulation, restricting their application to short-term predictions. Even intention-enhanced variants demonstrate temporal limitations, as intent inference accuracy degrades with prediction horizon extension [16]. Emerging data-driven solutions leverage deep learning architectures to optimize state estimation parameters, exemplified by Guan's hybrid framework [70] that synergizes historical flight records (BADA database) with real-time sensor data through physics-informed neural networks, generating TPs via multi-source information fusion.

2.2.3 Machine learning methods

ML constitutes a computational paradigm focused on pattern extraction from empirical datasets, enabling predictive analytics through learned representations. This discipline seeks optimal mathematical mappings between input-output pairs via algorithmic optimization, with functional forms spanning regression models, support vector machines (SVMs), and deep neural architectures. The optimization

process employs gradient-based methods, including stochastic gradient descent (SGD), root mean square propagation (RMSprop), adaptive momentum estimation (Adam), and their variants, which iteratively refine model parameters through error backpropagation. The distinction between trainable parameters (automatically adjusted during learning) and hyperparameters (requiring empirical determination through systematic experimentation) is critical to this process, the latter significantly influencing model convergence and generalization capabilities.

ML architectures adopt a tripartite classification framework based on data supervision paradigms and feedback mechanisms:

- 1). Supervised learning: each sample in the training set has a label called supervised learning. According to the type of label, it can be divided into regression problems (label y is a continuous value), classification problems (label y is discrete), and structural learning problems (label y is a structured object, such as a sequence, tree, or graph). Emerging label-efficient methodologies (e.g., semi-supervised/weakly-supervised learning) have been developed to address the scarcity of annotated training data through partial supervision mechanisms.
- 2). Unsupervised learning: Unsupervised learning frameworks execute self-supervised discovery of latent data structures through intrinsic pattern extraction from unannotated datasets. They encompass core methodologies such as cluster identification via similarity analysis, probability distribution modeling, representation learning for feature abstraction, and manifold-based dimensionality compression.
- 3). Reinforcement learning: a ML algorithm that learns through interaction. The agent makes an action based on the state of the environment and receives an immediate or delayed reward. The agent continuously learns and adjusts its strategy in interacting with the environment to maximize the expected total return. Reinforcement learning is an online learning mechanism. At present, aircraft TP mainly involves two forms: supervised learning (aircraft model, season, and meteorological conditions are used as labels) and unsupervised learning (samples primarily include the location, time, ground speed, heading angle, and other information of historical trajectory points, without clear labels).

2.2.3.1 Neural networks methods

The contemporary ML landscape has undergone transformative evolution through deep neural networks, propelled by synergistic advancements in data scalability, computational architectures, algorithmic sophistication, and cross-domain applicability, with these models now dominating ML research through their unparalleled capabilities in hierarchical feature abstraction, distributed knowledge representation, and spatiotemporal pattern analysis. Their technical superiority has driven revolutionary applications across multimodal perception systems (computer vision/natural language processing/speech recognition), autonomous decision-making platforms (autonomous driving/medical diagnostics), and personalized service engines (recommender systems/gaming AI), while in aircraft TP research, four principal neural architectures prevail: recurrent neural networks (RNNs) for temporal dependency modeling via cyclic connectivity, GANs enabling probabilistic trajectory synthesis through discriminator-generator mechanisms, sequence-to-sequence models facilitating variable-length input-output mapping with encoder-decoder structures, and CNNs extracting spatial-temporal features via kernel-based hierarchical filtering.

(1) Recurrent neural networks

RNNs employ persistent hidden states to establish temporal memory mechanisms through cyclic neuronal connections, where self-referential information flow enables sequential pattern learning. This architectural innovation implements parameter sharing across time steps, effectively transforming temporal sequences into depth-expanded feedforward networks. Particularly effective in modeling long-range dependencies, RNN variants have demonstrated exceptional performance in aviation TP. In this quintessential spatiotemporal sequence modeling task, flight paths manifest as ordered 4D waypoints (latitude/longitude/altitude/timestamp). Within this domain, gate-regulated architectures like LSTM and Gated Recurrent Units (GRUs) dominate implementations, overcoming traditional RNNs' gradient limitations through sophisticated forget/update gate mechanisms while preserving critical temporal dynamics in aircraft motion patterns.

LSTM-based architectures have catalyzed significant advancements in aircraft TP

through multiple technical innovations: Shi et al. [71] pioneered temporal dependency modeling by capturing inter-state correlations within trajectory sequences through LSTM's memory cells, achieving enhanced predictive precision. Subsequent developments introduced architectural refinements - Yang et al. [72] implemented stochastic regularization techniques like dropout to prevent overfitting while maintaining temporal modeling capabilities, and Yang et al. [73] deployed bidirectional (Bi)-LSTM architectures to address ADS-B signal degradation scenarios through dual-directional context analysis. Technical enhancements extended to motion pattern modeling, where Ma et al. [74] integrated taxiway dynamics with decaying memory window mechanisms to optimize LSTM's hidden state transitions for ground movement prediction. Zhao et al. [75] systematically addressed multidimensional feature fusion through deep LSTM structures that encoded complex flight environment parameters. The sequence-to-sequence paradigm saw implementation in Zeng et al. [12] via deep encoder-decoder (SS-d) LSTM stacks, effectively resolving 4D TP through hierarchical temporal abstraction. Collaborative prediction frameworks emerged through Xu et al.'s [76] social (S)-LSTM architecture, employing aircraft-specific LSTM branches with shared pooling layers to model multi-agent spatial interactions through hidden state synchronization.

GRU-based models have demonstrated notable efficacy in aircraft TP through adaptive learning frameworks. Han et al. [77] developed a dual-phase GRU architecture combining offline pre-training and online fine-tuning, where historical trajectory data initializes network parameters for subsequent adaptive updates using real-time flight data, enabling dynamic prediction of spatiotemporal trajectory points. Huang et al. [78] advanced this paradigm through BiGRU integration, leveraging forward-backward temporal analysis to enhance short-term TP accuracy by simultaneously modeling past-future contextual dependencies in flight paths.

(2) Generative adversarial networks

GANs operate through adversarial optimization between a generator and a discriminator: the generator maps latent vectors (e.g., random noise) to approximate the target data space, while the discriminator evaluates the

authenticity of generated versus real samples. This minimax game drives both networks toward the Nash equilibrium, where synthetic data aligns with the real distribution. A critical limitation, mode collapse, occurs when the discriminator overfits to narrow data patterns, reducing output diversity. To mitigate this, domain-specific GAN variants integrate architectural adaptations (e.g., CNNs/RNNs for image/sequential data) and stabilization techniques (spectral normalization/gradient penalty). The framework's flexibility allows hybridization with traditional generative models (e.g., Variational Autoencoders (VAEs), autoregressive networks), enabling applications in trajectory synthesis, anomaly detection, and beyond.

The foundational GAN framework proposed by Goodfellow et al. [79] employs multilayer perceptrons (MLPs) for adversarial training. In aircraft TP, GANs reframe the task as probabilistic distribution learning: the generator synthesizes trajectories matching real-data distributions, and the discriminator identifies discriminative features between actual and predicted trajectories. This unsupervised paradigm eliminates labor-intensive data labeling. Recent advancements include Wu et al. [21] implementing Wasserstein GAN-gradient penalty (WGAN-GP) with hybrid architectures (one-dimensional convolutional neural network (Conv1D)/ two-dimensional convolutional neural network (Conv2D)/LSTM), converting 4D trajectory sequences into RGB images for cross-modal prediction. Pang et al. [80] developed conditional GANs (CGANs) incorporating weather and flight plan constraints, while Hu et al. [81] enhanced CGANs with multi-network fusion (Conv1D/Conv2D/LSTM) for short-term accuracy. Jarry et al. [82] extended GANs to joint TP and anomaly detection, using the discriminator to quantify trajectory normality through similarity scoring (1=typical, 0=anomalous).

(3) Sequence-to-sequence neural networks

The sequence-to-sequence architecture, a cornerstone of sequential data processing in deep learning, employs dual-component neural frameworks comprising an encoder and a decoder. While traditional implementations utilize paired RNNs for these components, the Transformer model introduced by Vaswani et al. [83] revolutionized the paradigm through its exclusive reliance on

attention mechanisms. This architecture eliminates recurrent connections, enabling full parallelization across modern GPU architectures while effectively capturing long-range dependencies. Compared to RNN-based sequence-to-sequence models, Transformers demonstrate superior computational efficiency and scalability, driving their widespread adoption in cross-domain applications, including but not limited to machine translation, speech processing, and multimodal tasks.

Since aircraft TP is a sequence-to-sequence conversion problem, the emergence of the Transformer model has attracted the attention of experts regarding aircraft TP. Luo et al. [84] designed a TP method based on the Transformer. This method grouped and trained the historical trajectory of the aircraft in a sliding window manner, obtained good prediction accuracy, and could predict the trend of changes in the next 30 steps. Tong et al. [85] also proposed a Transformer-based TP model for long-term TP and showed good accuracy performance. Tan et al. [86] used the Transformer encoder to extract global features of the flight trajectory and combined and densely processed the spatial and temporal local features extracted by CNN and RNN, respectively, to achieve aircraft TP.

(4) Convolutional neural networks

The CNN was first used to process image information. It can effectively overcome the two problems of fully connected feedforward networks: too many parameters and difficulty extracting local invariant features. CNNs are feedforward architectures constructed through interleaved stacking of convolutional, pooling, and fully-connected layers. Its structural features are local connection, weight sharing, and aggregation. CNN and its variants are widely used in aircraft TP, mainly for extracting spatial features [81], [87], [88], [89], [90] or weather features [14], [80], [91], [92], [93] of trajectories. Patrikar et al. [94] even use them to extract spatial and weather features, respectively. CNN and its variants are usually used with other networks, such as RNN, to achieve TP functions.

2.2.3.2 Traditional machine learning methods

This thesis's traditional ML methods refer to ML methods other than deep neural networks. Aircraft TP mainly includes regression models, random forests (RFs), gradient boosting machines (GBMs), Gaussian mixture models (GMMs), HMM,

clustering algorithms, and Attention mechanisms.

(1) Regression models

The regression model is a statistical method used to predict continuous variables. The model establishes predictive capabilities through statistical analysis of correlations between input (independent) and output (dependent) variables. Common regression models include linear regression model, polynomial regression model, ridge regression, logistic regression, etc.

TP frequently employs regression techniques such as locally weighted linear/polynomial regression. Kanneganti et al. [95] implemented a linear model utilizing aircraft heading and horizontal speed for 10-minute position forecasting. Lee et al. [96] integrated aircraft specifications, ground speed, altitude, and meteorological data with stepwise regression to optimize arrival time prediction inputs. Tastambekov et al. [36] developed wavelet-based linear regression models across X/Y/Z axes using historical radar data for 10-30 minute TP without aviation parameters. Hong et al. [97] combined clustering and pattern-specific multi-linear regression with ATC intention data for arrival time estimation. Hamed et al. [24] fused point mass models with neural networks and Loess regression, employing PCA for input optimization in altitude prediction.

(2) Random forest models

The RF is an ensemble learning method that improves the accuracy and stability of the model by building multiple decision trees and combining their prediction results. It can be used for both classification and regression tasks. It has the advantages of high accuracy, resistance to overfitting, and robustness. Its main disadvantages are long training time and slow prediction speed. RF is also a basic method commonly used for aircraft TP [93-95]. Recently, Rizvi et al. [100] proposed a TP method, which first uses LSTM, RF regression, and other models to predict the trajectory and then uses the K-nearest neighbors (K-NN) to combine the prediction results to obtain the final prediction result.

RF employs ensemble learning through constructing numerous decision trees and aggregating their outputs, enhancing model accuracy and stability for classification/regression tasks. Key strengths include robustness, overfitting

resistance, and high precision, though limitations involve computational intensity and slower inference speeds. As a foundational TP method [93-95], recent advancements include Rizvi et al. [100] integrating LSTM and RF regression with K-nearest neighbors (K-NN)-based result fusion for trajectory prediction.

(3) Gradient boosting machines models

The GBM is an ML technique employing an ensemble of decision trees for regression and classification tasks. It gradually fits residuals and builds a series of weak learners (usually decision trees) to form a strong learner. XGBoost, LightGBM, and CatBoost are its common variants. The GBM is also commonly used in aircraft TP [37]. For example, Alligier and Gianazza [37] used a GBM to build a prediction model for each aircraft type considered in the TP of the aircraft climbing phase. They performed ML and estimation of mass and speed intentions.

(4) Gaussian mixture models

The GMM is a probabilistic model composed of multiple Gaussian distributions. The Gaussian distribution of each component defines the likelihood of data points being assigned to that component. GMM is used to represent data sets with multiple Gaussian distribution components. This unsupervised ML technique is commonly employed for core applications, such as clustering, density estimation, and pattern discovery. It is also used in aircraft TP.

Liu and Hansen [91] developed a deep generative framework for aircraft TP, employing an LSTM-based sequence-to-sequence architecture to derive Gaussian Mixture Model (GMM) parameters. The methodology incorporates adaptive KF, beam search optimization, and Rauch-Tung-Striebel (RTS) smoothing for enhanced GMM prediction accuracy. It synthesizes three-dimensional (3D) flight plans, 4D trajectories, and meteorological parameters (wind patterns, temperature gradients, convective weather data).

(5) Hidden Markov models

The HMM serves as a foundational probabilistic framework in ML applications. It effectively models time-dependent random processes by defining hidden states, observation sequences, state transition probabilities, and observation probabilities. HMM is particularly good at processing time series data and sequence labeling

tasks by modeling the probabilistic relationship between the hidden states and observations of sequence data.

Ayhan and Samet [52] introduced a HMM framework for flight TP, leveraging historical flight trajectories and contextual meteorological data to identify weather patterns linked to specific flight positions. The aircraft's position is defined as a hidden state, and the weather cube observed around the trajectory point is the realization of this hidden state. The transition matrix is derived through HMM training on historical flight trajectory and meteorological datasets, enabling TP by applying observed or inferred weather sequences. Qiao et al. [101] developed an HMM framework for TP incorporating adaptive parameter optimization. The framework dynamically tunes parameters in response to real-time motion dynamics. The approach integrates density-driven trajectory segmentation to enhance computational efficiency.

(6) Clustering algorithms

Clustering algorithms constitute a core unsupervised ML technique that groups data points into clusters with maximal intra-cluster homogeneity and inter-cluster distinction. In aircraft TP, prevalent clustering techniques include K-means, K-medoids, and Density-Based Spatial Clustering (DBSCAN), identifying noise-resilient spatial patterns. Clustering can improve the accuracy of TP, so they are widely used in TP [11], [13], [98], [99], [102], [103]. In terms of usage, clustering algorithms are generally used as components of hybrid models to complete TP and other components are trained and predicted by category based on clustering.

The key to aircraft trajectory clustering is to use a suitable trajectory distance calculation method. Euclidean, Hausdorff, Frechet, and warp distance are commonly used distance calculation methods, but they have their usage scenario limitations. For trajectories with different lengths, Euclidean distance clustering methods do not work well. If the geometry of the trajectory is concerned, shape distances such as Hausdorff and Frechet distances can adapt to the trajectory but cannot compare the trajectory as a whole. The warping-based distance technique computes optimal trajectory alignments by minimizing position-matching costs through predefined metrics. It calculates the distance by the sum of the expenses between all position pairs. It is beneficial when processing time series data. The

representative algorithm of this type of method is dynamic time warping (DTW). A primary limitation lies in its computationally intensive nature, necessitating substantial resource allocation for implementation. It may not apply to long-sequence large-scale trajectory data sets. Besse et al. [104] introduced the symmetrized segment-path distance (SSPD), a shape-sensitive metric designed for holistic trajectory analysis. The appropriate distance calculation method should be selected according to the application scenario to improve the clustering quality and create conditions for strengthening TP performance.

(7) Attention mechanisms

The Attention mechanism enables neural networks to dynamically prioritize salient input segments during sequential data processing, enhancing model efficacy and decision transparency. AM supplements existing architectures as a computational enhancement strategy rather than operating as an independent framework. Soft Attention, Hard Attention, Multi-head Attention, Self-Attention (SA), etc., are common forms of expression. In aircraft TP, it is used as a component of a hybrid model to determine the different roles of each trajectory point, which helps to identify prominent points in the trajectory and assign higher weights to them, capture the dynamic impact of different spatiotemporal positions on the current position, make up for the shortcomings of RNN in capturing long-term global features, and improve TP accuracy [87], [88], [92], [105].

2.2.3.3 Hybrid model methods

Neither neural networks nor traditional ML methods are designed and implemented for the characteristics of the aircraft TP field. Even aircraft TP has various application scenarios and different prediction requirements. Prediction frameworks employing isolated neural network architectures or conventional ML models exhibit insufficient capability to satisfy most TP operational requirements. The current ML TP method is usually implemented using a hybrid model. The hybrid model is generated by combining two or more neural networks and/or traditional ML basic models to leverage the complementary capabilities of each constituent framework, achieve complementary advantages, and effectively improve the comprehensive ability of TP. Common hybrid model methods for aircraft TP methods include RNN and CNN hybrid, Attention and RNN hybrid,

hybrid with a clustering algorithm, and complex hybrid.

(1) Hybrid recurrent neural networks and convolutional neural networks

This hybrid framework synergistically integrates CNNs for hierarchical extraction of spatial and meteorological features, with RNNs capturing temporal dependencies in trajectory sequences. The fusion of CNN-RNN architectures enables robust spatiotemporal feature fusion, establishing it as the dominant paradigm in TP due to its state-of-the-art multimodal data processing efficacy.

Schimpf [92] designed a generalized hybrid-recurrent prediction method to improve the route generalization ability and flight TP accuracy based on the limitation that TP requires separate training for different routes and sample data. Building upon this architectural paradigm, the researchers systematically integrated constituent modules to develop CNN-LSTM, CNN-GRU, SA-LSTM, and SA-GRU architectural variants, followed by empirical validation through comparative performance evaluations.

Pang et al. [106] developed a CNN-LSTM hybrid architecture for aircraft trajectory forecasting, leveraging hierarchical convolutional and dense layer configurations to process and encode meteorological features. This framework synergistically integrates abstract weather representations with historical positional data via LSTM recursion, enabling spatiotemporal feature fusion for positional forecasting. The LSTM component was specifically adopted to mitigate the vanishing gradient phenomenon inherent in conventional RNNs. Building on their prior 3D TP framework [93], the authors advanced the methodology to incorporate 4D TP.

Ma and Tian [90] presented a methodological analysis of 4D trajectory forecasting utilizing ADS-B datasets. Their framework investigated spatiotemporal forecasting paradigms by processing historical 4D trajectory sequences augmented with flight state parameters (ground speed, heading). Spatial feature extraction was implemented via One-dimensional CNNs to hierarchically encode geospatial attributes, while LSTM networks modeled temporal dependencies across trajectory sequences. The synergistic fusion of these spatiotemporal representations enabled high-fidelity 4D TP through multimodal feature fusion,

achieving enhanced operational precision in aviation navigation systems.

Hafienya et al. [89] introduced a long-term TP method, which repeatedly used RNN and CNN models to extract flight trajectory features.

(2) Hybrid Attention and recurrent neural networks

In aircraft TP, Attention is often used as a component of TP hybrid models and is often used in combination with RNN [87], [88], [92], [105].

For example, Schimpf [92] proposed a generalized hybrid-recurrent prediction method. Based on this method framework, they formed a hybrid model including CNN-LSTM and CNN-GRU by combining and applying related components, and a comparative experiment was conducted. Recently, Jia et al. [105] developed a dual-component TP framework integrating LSTM networks with attention mechanisms. The LSTM module processes flight trajectory data to model hierarchical temporal dependencies, while adaptive attention mechanisms dynamically weight the significance of individual trajectory points based on contextual relevance. This architecture enables prioritized feature aggregation, optimizing prediction accuracy through discriminative spatiotemporal pattern recognition.

(3) Hybrid with clustering algorithms

The trajectories after clustering have more precise and stable characteristics, which is convenient for model training and TP. The idea of clustering the trajectories first and then training and predicting the models for each cluster is widely adopted. Empirical evaluations demonstrate that the proposed architecture significantly enhances TP accuracy through discriminative feature weighting and temporal-spatial pattern recognition [11], [13], [88], [98], [102], [103], [107].

Barratt et al. [102] investigated a probabilistic model for generating aircraft trajectories in terminal airspace operations. Their methodology involved clustering flight paths using K-means algorithms, followed by building a GMM from the categorized clusters to enable precise TP.

Lui [11] proposed an arrival time prediction model based on clustering and RF regression. The model uses K-medoids to cluster historical trajectories, trains a

random forest regression model based on the clustering results, and uses it to predict arrival transit time. The framework enhances prediction precision for arrival transit time estimation.

Wang et al. [13] introduced a short-term TP method for terminal maneuvering areas (TMA), combining cluster-based preprocessing with neural network-driven ML components. DBSCAN clusters the trajectories into components and noise, and multiple convolutional neural networks (MCNNs) is used to find a good prediction model for each cluster unit based on the classification.

Liu et al. [98] employed the DBSCAN clustering algorithm to categorize historical flight trajectories into multiple categories, extracted the centerlines of each category, and formed each historical trajectory sample based on the matching of the centerlines with environmental features. Four ML models, logistic regression, SVMs, RFs, and gradient boosting, were selected for comparative experiments. After the model was trained, they used the environmental conditions of the predicted flight to determine its category. Then, the centerline of the category was selected as its predicted trajectory. The results showed that the random forest method performed best for most test airport pairs.

Le et al. [99] developed a short-term TP approach by applying K-means clustering to historical flight paths, training RF models per cluster, and dynamically selecting prediction models based on detected trajectory patterns in the operational sector.

Gui et al. [103] introduced a TMA trajectory optimization method. The approach applies spectral clustering to historical trajectory data, generating candidate 4D trajectories using parameters such as entry position, entry speed, weather conditions, and wake turbulence constraints. An optimal trajectory is selected via conflict detection and Pareto frontier analysis.

(4) Complex hybrid

In addition to the hybrid models of RNN and CNN, Attention and RNN, and Attention and clustering algorithms, these hybrid models usually involve the participation of components such as RNN, CNN, Attention, and clustering algorithms. They also include more complex hybrid models formed by multiple applications of these components in different combinations. This type of model is

also a common TP model.

Rizvi et al. [100] proposed a TP method, which first used LSTM, RF regression, and other models to predict the trajectory. Subsequently, the K-NN algorithm was employed to aggregate prediction outputs for final trajectory determination.

Tan et al. [86] proposed a new TP model that integrates spatiotemporal features with local and global attributes. The model is a hybrid computational model that incorporates residual network (ResNet) with convolutional block Attention module (for extracting local spatial features), DSC-BiLSTM-Multihead-Attention (for extracting local temporal features), and Transformer encoder (for extracting global features). The spatiotemporal feature information extracted by the above components is concatenated and processed by dense layers to output the predicted trajectory. Empirical evaluations demonstrate the framework's robust predictive precision and generalization performance across diverse operational scenarios.

Wang et al. [87] developed a TP framework integrating temporal convolutional networks (TCNs), LSTM, and attention mechanisms. The TCN extracts spatial dependencies, LSTM models temporal dynamics, and attention identifies multi-scale periodic patterns. This hybrid architecture outperforms standalone LSTM or TCN models in prediction accuracy.

Zhang et al. [108] proposed a TP method for small sample data. WGAN-GP is used to increase the number of small sample data. LSTM is trained based on the enhanced samples and outputs the predicted trajectory.

Huang and Ding [78] introduced a TP framework utilizing a Dual Self-Attention (DSA)-TCN-BiGRU architecture. The TCN component captures high-order spatial-temporal trajectory features while enhancing computational efficiency for long-sequence processing. The BiGRU module, integrated with DSA, enables bidirectional analysis of trajectory point dependencies within hidden states to refine prediction outcomes.

Wu and Yu [88] developed a TP framework combining clustering and spatiotemporal feature extraction. The method employs K-medoids clustering with Hausdorff distance metrics, followed by cluster-specific prediction model construction. A CNN-BiLSTM network extracts spatial-temporal trajectory

patterns, augmented by joint attention mechanisms to prioritize critical trajectory point features.

Hafienya et al. [89] introduced a long-term TP approach. After dividing the dataset into spatial and temporal parts, they used CNN and GRU to extract the flight trajectory's spatial and temporal features, respectively, and 3D CNN was used to predict future flight status. They merged these features and used a linear activation function to predict the 4D flight trajectory. The uncertainty of the model prediction was achieved through Mont Carlo dropout.

2.2.4 Application of TP methods

The TP framework serves as the computational core of TP systems. It comprises two fundamental components: model architecture and its parameterization. The TP method refers to selecting a suitable model and determining appropriate model parameters in combination with specific application scenarios. For ML, the model itself is manifested as a model architecture. Model parameters include parameters and hyperparameters. Model training is determining parameters and hyperparameters based on sample data. It is challenging to optimize and adjust hyperparameters automatically through training, and they are mainly achieved through manual settings. In practical deployments, even minor operational variations (e.g., route adjustments or seasonal patterns) typically necessitate model retraining to update parameter configurations. The application of TP methods is also an essential part of TP research.

2.2.4.1 Uncertainty expression

TP model efficacy is contingent upon input data accuracy (e.g., aircraft performance metrics, intent signals, and weather data). Inherent uncertainties in these parameters propagate into deterministic prediction inaccuracies. Consequently, probabilistic TP—predicting trajectory spatiotemporal distributions—demonstrates enhanced operational utility compared to deterministic approaches [16]. Recently, many scholars have researched probabilistic TP methods and achieved corresponding results.

Probabilistic TP employs Bayesian neural networks (BNNs) to quantify inherent model uncertainties by representing network weights as probabilistic distributions. BNN usually uses Bayesian approximate variational reasoning to make

calculations feasible while capturing model uncertainty, or use dropout regularization technology to implicitly define a variational distribution by randomly discarding neurons.

For example, Pang and Liu [109] introduced a probabilistic aircraft TP framework leveraging Bayesian neural networks (BNNs). The method integrates flight plan data and weather tensors to parameterize trajectory uncertainty, generating spatiotemporal distributions rather than deterministic outputs. Uncertainty quantification is achieved via dropout-enabled Bayesian variational inference (VI) within a CNN architecture.

Thomas et al. [20] introduced a method to quantify uncertainty factors and infer unknown state variables from monitoring data. This method considers the uncertainty of wind speed, wind direction, and temperature. It uses Bayesian reasoning to infer the true airspeed, Mach number, or cost index variables expressed as probability density for random TP.

Feng et al. [110] introduced a TP method based on a hybrid LSTM and BNN. LSTM is used to extract the scene feature vector, combining flight status and weather information. BNN is used to solve the uncertainty problem. The BNN architecture integrates a fully connected output layer to optimize the network and generate uncertainty-aware predictions, producing aircraft position forecasts with associated confidence intervals.

Hafienya et al. [89] developed a hybrid CNN-RNN architecture to capture spatiotemporal trajectory dependencies for future state forecasting jointly. The framework employs a linear activation function in its output layer to regress continuous 4D trajectory parameters (latitude, longitude, altitude, and timestamp). They achieved uncertainty in the model prediction through the Monte Carlo dropout.

2.2.4.2 Weather information processing

Weather is crucial environmental information that affects flights and is the leading natural factor causing flight delays. Convective weather and strong turbulence in meteorological conditions seriously affect flight safety and are the main reasons for changes in aircraft trajectories. Weather information has long been used for

aircraft TP. The dynamic and stochastic nature of meteorological conditions introduces nonlinear interactions with aircraft trajectories, posing persistent challenges in operationalizing meteorological influences to enhance probabilistic prediction fidelity.

In methods according to physical mechanisms, weather, an important input parameter of traditional ballistic prediction methods, significantly impacts TP accuracy. In response, researchers have developed mitigation strategies to minimize the operational effects of meteorological uncertainty through probabilistic modeling and ensemble-based assimilation [31], [38]. The main factor of wind is usually decomposed into scalar and random variables or transverse and longitudinal wind. On this basis, relevant parameters such as thrust, speed, and state are corrected. However, improvements to traditional TP methods still struggle to meet the high-precision, multi-dimensional prediction requirements of the TBO environment. In recent years, a paradigm shift has been seen toward ML-driven approaches, where models autonomously extract spatiotemporal patterns from historical flight trajectories and meteorological datasets to enhance aircraft TP accuracy. The effectiveness of prediction improves as the volume of sample data used for learning increases. Currently, proposed prediction models include GMM, HMM, MLP, CNN, LSTM, and GAN et al. [52], [91].

2.2.4.3 Multi-machine factor processing

Air traffic congestion along planned routes introduces spatiotemporal perturbations to flight trajectories, compounded by interdependencies between aircraft sharing common airspace segments. Integrating system-wide congestion dynamics and aircraft-to-aircraft interaction models into TP frameworks has been shown to enhance predictive fidelity. Current research in multi-agent TP [108-110] addresses these challenges through advanced methodologies: Olympopoulos et al. [56] identified computational inefficiencies in sequential Monte Carlo methods for high-dimensional multi-aircraft state estimation and subsequently developed a scalable particle filter variant. Dai et al. [113] formulated a Bézier curve-based 4D trajectory optimization model that concurrently minimizes flight duration, velocity adjustments, and conflict probabilities through parametric spatial smoothing.

Raul et al. [114] proposed a framework for generating 4D optimized trajectories in a multi-aircraft environment. The trajectory optimization module is based on the PMM of the aircraft. By setting the look-ahead time (LAT), calculating the activity area based on the flight plan, and using updated information such as weather and trajectory-sensitive regions, several alternative trajectories are formed to avoid lateral and longitudinal wake-sensitive areas and weather no-fly zones, thereby achieving trajectory optimization.

Chang et al. [115] introduced a trajectory multi-objective optimization approach based on robust optimal control and flexible horizontal route selection. This approach integrates sector busyness as a proxy metric to account for multi-aircraft environmental constraints, formulates a multi-objective cost function for trajectory optimization, and resolves the problem via an elitist non-dominant sorting genetic algorithm II (NSGA-II) framework to derive Pareto-optimal trajectory solutions.

The above multi-machine environment TP and optimization adopts the method based on the physical mechanism model. With the advancement and maturity of ML technology, more and more related research results based on ML models have emerged:

Xu et al. [76] developed a coordinated multi-agent trajectory prediction framework leveraging an S-LSTM architecture. The system architected individual LSTM submodules for distinct aircraft and aggregated their hidden states via a shared pooling layer, enabling explicit modeling of inter-aircraft interaction dynamics across spatiotemporal scales.

Patrikar et al. [94] proposed a baseline socially aware TP algorithm based on the characteristics of aircraft control and TP in terminal airspace without towers. This method encodes the 3D trajectory using a TCN. The dynamic weather environment (wind vector) is encoded using a CNN and attached to the encoded trajectory. The architecture employs a graph attention network (GAT) to model inter-aircraft relational dynamics, utilizing attention weights for context-aware feature aggregation across heterogeneous flight data streams. Finally, conditional variational autoencoders (CVAEs) and forward Verlet algorithms generate future trajectories for all aircraft.

2.2.4.4 Model integration application

Integrated application mainly combines multiple TP models to output a more complete or accurate prediction trajectory. The framework supports full or partial trajectory generation, spanning from departure gate to arrival gate, while demonstrating superior predictive accuracy relative to single-model baselines. The TP models here include both basic model-based and hybrid model-based TP models. The integrated application models include traditional dynamic and ML models.

Soler et al. [116] developed a phase-segmented trajectory optimization framework grounded in phase-specific dynamics and constraints. Each flight segment (e.g., climb, cruise) employs tailored dynamic models. The approach unifies these phases into a sequential optimal control formulation by parameterizing phase transition points, thereby converting multi-phase optimization into a unified control problem to generate eco-optimal multi-phase trajectories.

Recent advancements in TP have leveraged ensemble machine learning techniques to address temporal and operational complexities:

Rizvi et al. [100] developed an ensemble framework combining RF regression and LSTM networks, trained on commercial flight data. A K-NN-based trajectory fusion mechanism synthesizes predictions from both models, ensuring worst-case performance parity with individual models and superior accuracy in typical scenarios.

Zhou et al. [117] introduced a temporal-adaptive ensemble architecture that dynamically selects optimal models (short/medium/long-term) based on prediction horizons, enhancing both temporal adaptability and holistic performance.

Hu et al. [81] pioneered a CGAN approach, reformulating TP as probabilistic distribution estimation. The adversarial training framework (generator-discriminator) incorporates hybrid architectures (Conv1D/2D, LSTM) to enable single-step multi-horizon predictions, eliminating iterative prediction errors.

Given the limitation that current TP requires separate training for different routes and sample data, Schimpf et al. [92] designed a general method (framework) for

aircraft TP based on deep learning. It can combine and apply components such as CNN, SA, LSTM, and GRU. It extracts weather features through convolutional or SA networks and connects them with the expected positions obtained from the flight plan. The predicted trajectory output by the recursive network is regarded as the actual trajectory estimate, which realizes route generalization and improves the accuracy of aircraft TP.

2.2.5 Overview of research status

Based on the above data analysis, it can be found that the research on aircraft TP mainly revolves around two routes. Two primary approaches exist: enhancements to physics-based models and exploratory investigations leveraging ML techniques. The TP methods currently used in ATM business operations are based on the physical mechanism method, including dynamic and state estimation methods. The physical mechanism method is based on a precise mechanism, and the model logic is rigorous and accurate. The critical problem is that the input parameters are uncertain, which leads to uncertainty in the TP results, and the TP accuracy is difficult to guarantee. Therefore, improving the accuracy of the relevant input parameters has always been the research focus in this direction [15]. Although ML has been used in aircraft TP for more than 20 years [37], ML-based methods are still in the exploratory research stage, with minimal applications in actual business operations and no apparent benefits for end users. The slow progress of AI business applications in the ATM field, including TP, can be explained by the fact that the ATM field is a critical field that endangers life and safety, which are the most important [7].

2.2.5.1 Improvement of physical mechanism-based methods

Research and construction work has been continuously carried out to improve the accuracy of relevant input parameters. Through crowdfunding, the Base of Aircraft Data (BADA) data is continually improved and maintained. For uncertain factors such as aircraft performance, aircraft status, and aircraft intention, the estimation accuracy is improved by optimizing the estimation method. Since these parameters are closely related to meteorological conditions, multi-source data is used to enhance the reality and accuracy of meteorological data. On the other hand, relevant parameters are corrected according to weather conditions. Recently, data-driven technologies such as deep learning, as the latest general basic technology,

have also been widely used in the reasoning and optimization of model input parameters and operating status. Different ML techniques are used to learn thrust laws from historical data, estimate aircraft mass that changes over time, predict speed intentions, etc., and the TP accuracy is improved by improving the accuracy of model parameters or state parameters. Current physics-based models face dual challenges: dependency on extensive operational parameters (e.g., aircraft mass, wind patterns) that often involve proprietary data or rely on legacy database approximations, resulting in compromised prediction accuracy. Meanwhile, state estimation frameworks derive kinematic equations from positional/velocity measurements but fail to account for behavioral uncertainties during extended maneuvers, leading to progressive error accumulation. Although the industry has effectively improved TP accuracy through long-term efforts, it still cannot meet the higher requirements for TP accuracy in response to aviation transportation capacity challenges in the TBO environment.

2.2.5.2 Exploratory research based on machine learning methods

ML approach autonomously learns latent trajectory patterns, environmental influences, and pilot preference profiles from historical data, eliminating the need to explicitly model operational parameters (e.g., aircraft specifications, real-time states, meteorological conditions) and their direct causal relationships with trajectory outcomes. They have unique advantages that physical mechanism-based methods do not have. Of course, the comprehensive use of environmental information such as terrain and weather and flight-related information such as aircraft intentions and airline preferences will help improve the pertinence of ML model design and construction and improve performance such as model training efficiency and model TP accuracy.

Like other fields, the field of aircraft TP has been exploring the possibility of applying new theories and technologies in this field. Corresponding to the development of AI, TP research based on ML has experienced development from conventional ML models to deep neural networks, from directly using model algorithms or primary neural networks for prediction to building complex composite models based on typical model algorithms and neural networks, from using trajectory data for prediction to comprehensively using historical trajectories, flight plans, aircraft intentions, weather and other related data, from

single-machine prediction to considering the mutual influence of multiple machines, from single-stage TP to cross-stage prediction, and from short-term prediction to medium- and long-term prediction. The current main research content is to build a complex hybrid model based on deep neural networks, comprehensively use flight trajectories and various related information, and realize multi-stage or even full-route TP. Begin to explore the model's application method on different routes, the transfer learning problem of TP [92], and the automatic optimization learning problem of TP model hyperparameters [78]. Improving TP accuracy remains the primary goal of aircraft TP research.

2.2.6 Comparative analysis of mainstream TP models

Building upon the overview in Section 2.2.5, this section provides a systematic comparative analysis of mainstream TP methodologies. The summary in Table 2.2 aims to contextualize the research landscape and highlight the specific niches and evolution of different approaches.

Table 2.2 Comparative analysis of aircraft trajectory prediction methods.

Method category	Core principle	Advantages	Limitations
Physics-based	Uses aircraft performance and kinematic equations.	Physically interpretable; model logic is rigorous.	Highly sensitive to uncertain inputs (e.g., mass, wind); cannot capture pilot/controller intent.
Classic machine learning (e.g., SVM)	Learns mapping from hand-crafted input features to trajectory parameters.	Superior accuracy to physics-models; more interpretable than deep learning.	Relies heavily on manual feature engineering; struggles with long-sequence data and temporal dependencies.
Deep learning (deterministic) (e.g.,	Uses deep networks to automatically	Powerful sequence modeling; automatic feature learning;	Models are typically deterministic , unable to quantify

Method category	Core principle	Advantages	Limitations
LSTM, Transformer)	learn spatio-temporal patterns from raw data.	captures complex non-linearities.	prediction uncertainty or capture multi-modal future outcomes.
Deep learning (generative) (e.g., GAN, VAE)	Learns the underlying data distribution to generate multiple, plausible futures.	Capable of modeling uncertainty and multi-modality ; produces probabilistic outputs.	GAN : Training instability, mode collapse. VAE : Tends to generate overly conservative or "blurry" average trajectories.
Proposed Improved CTGAN	A conditional GAN optimized and stabilized for tabular ADS-B trajectory data.	• Data efficiency: Robust performance with limited data. • Probabilistic output: Natively models prediction uncertainty. • High fidelity: Generates accurate and diverse trajectories.	Requires careful domain-specific tuning.

The evolution in Table 2.2 shows a clear trajectory: from physically rigorous models to flexible data-driven approaches, with a recent paradigm shift towards capturing the stochastic and multi-modal nature of flight paths.

While the field is actively developing complex hybrid models, a critical gap remains between the deterministic nature of most DL models and the operational need for uncertainty-aware predictions in safety-critical ATM.

Our development of the improved CTGAN is a direct response to this gap. It is

strategically designed to leverage the strengths of generative models (i.e., uncertainty quantification) while specifically mitigating their primary weakness of training instability. This work aims to provide a robust, data-efficient, and probabilistic solution, addressing a fundamental limitation in current data-driven TP approaches.

2.3 Typical deep learning network

2.3.1 Generative adversarial networks

First proposed by Goodfellow et al. [79] in 2014, GANs have emerged as a prominent research focus in deep learning. The combination of a discriminator network and a generator network, along with the adversarial game theory used for model training, not only enhances the quality of generated outputs but also allows for the integration of various traditional ML methods, leading to numerous new research directions and pushing the study of generative models into a new phase. Notable AI expert Yann LeCun even referred to GANs as one of the most exciting technologies in artificial intelligence development over the past decade.

GAN technology is not limited to image generation; it has also been applied to image translation, text generation, video generation, and 3D model generation, among other fields. Researchers continue to propose new variants and improvements, expanding the applicability of GANs. The ongoing evolution of GANs has played a significant role in image generation, data augmentation, and creative design, contributing to advancements in AI.

2.3.1.1 Evolution of generative adversarial networks

The original GAN framework proposed by Goodfellow et al. in 2014 used fully connected layers for both the generator and discriminator, facing issues like training instability and mode collapse. To overcome these limitations, Radford et al. [118] proposed the deep convolutional GAN (DCGAN) in 2015, replacing fully connected layers with convolutional architectures. This architectural change, along with empirical design choices, significantly improved the stability of adversarial training. The innovations and contributions of DCGAN have notably advanced the field, making it an essential reference for researchers and practitioners using GANs.

In addition to DCGAN, researchers have proposed numerous new GAN models to address the limitations of the original GAN and adapt to specific application areas. For instance, WGAN addresses the issue of gradient vanishing when the generated and real distributions do not overlap by using Wasserstein distance instead of the traditional Kullback-Leibler (KL) and Jensen-Shannon (JS) divergence measures. This change enhances training stability, and various methods have been developed to constrain the discriminator network's weight range, resulting in multiple WGAN variants [121-123]. CycleGAN [122] employs a dual-domain, multi-cycle mapping approach to transfer knowledge extracted from one domain to a related domain, achieving domain adaptation. By utilizing two generative adversarial networks that learn to translate images between domains while preserving the underlying content, CycleGAN facilitates knowledge transfer without requiring paired training data. This method enables high-quality image generation in the target domain, reflecting the characteristics of the source domain while maintaining consistency in the overall structure. InfoGAN splits the input of the generator G into random variables and latent codes. By using the mutual information between the latent codes and feature vectors as a constraint, InfoGAN [123], [124] enables the latent codes to represent different feature dimensions of the generated samples. This approach enhances the interpretability of the network, allowing for a clearer understanding of how specific features influence the generation process. Consequently, InfoGAN provides a more structured way to control the characteristics of the generated outputs. Self-Attention GAN (SAGAN) [125] addresses the challenge GANs face in capturing complex patterns by incorporating a SA mechanism into the network. This enables the model to prioritize critical data components, improving its ability to synthesize intricate structures. StyleGAN [126] introduces a controllable generation process by manipulating the style attributes of images in the dataset. Its style-based generator enables the creation of high-quality, high-resolution images with varied styles. BigGAN [127] leverages large batch sizes during training and enhances the number of convolutional channels and network parameters. It employs techniques like the "truncation trick" and stability controls to generate high-fidelity images with minimal mode collapse. CGAN enhances the stability of GAN training by incorporating conditional input parameters,

allowing for more directed and controlled output data. CTGAN [128] is designed explicitly for tabular data, utilizing reversible data transformations and a modified architecture to improve the training process, effectively addressing the challenges of generating tabular data.

Additionally, some researchers have focused on improving the loss functions of GANs based on mathematical principles, enabling better alignment between the distributions of generated data and training data [120], [129], [130]. These advancements have driven progress in GAN technology, enabling diverse computer vision and image processing applications.

(1) Original generative adversarial networks

In ML, training paradigms are broadly classified into supervised and unsupervised learning, depending on data annotation. Functionally, models are categorized as generative or discriminative. GANs unify these concepts by merging supervised/unsupervised learning and generative/discriminative objectives into a single iterative framework. Here, the discriminator employs supervised learning (self-labeled data), while the generator operates unsupervised, providing a novel methodology for synthetic data generation.

Unlike conventional generative models built around a single network, GANs comprise two components: a generator and a discriminator. The generator accepts input from a predefined distribution (e.g., random noise or latent variables) and transforms it into the target distribution. The discriminator, in contrast, analyzes both synthetic and real training data to classify their authenticity, determining whether inputs originate from the generator or the training dataset. The model architecture is illustrated in Figure 2.6.

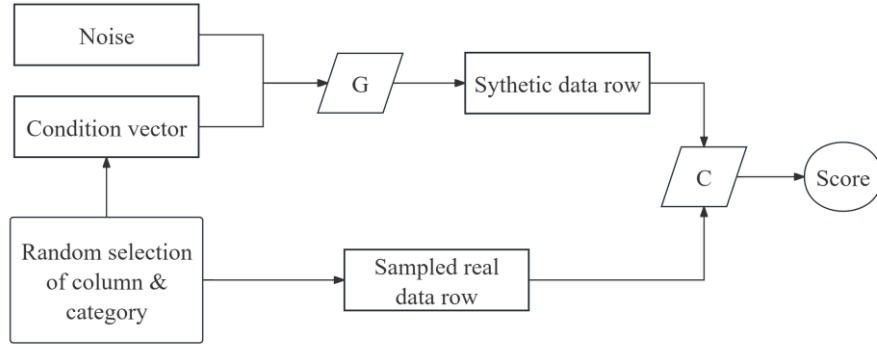


Figure 2.6 The model architecture of the original generative adversarial networks.

The original GAN employs a relatively simple, fully connected MLP. The loss function is expressed in the form of binary cross-entropy. During training, the generator and the discriminator rely on each other's output results to optimize their parameters, aiming to minimize the overall loss function. If the input noise is defined as z , the real data is set to x , then generator G and the discriminator D are playing a mini-max game, the cross-entropy loss $V(D, G)$ can be expressed as follows:

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim P_d} [\log (D(x))] + \mathbb{E}_{x \sim P_z} [\log (1 - D(G(z)))] \quad (1)$$

The generator (G) aims to minimize the term $\log (1 - D(G(z)))$, meaning it tries to generate samples so convincing that the discriminator (D) assigns a high probability ($D(G(z))$ close to 1).

The discriminator (D) aims to maximize the entire function, which involves correctly identifying real data $\log (D(x))$ and correctly flagging generated data ($\log (1 - D(G(z)))$).

This creates an adversarial game where both networks are pitted against each other, driving mutual improvement.

The training process can be directly mapped onto the architectural components and data flows in Figure 2.6:

1). Forward pass

A batch of random noise vectors z is sampled and fed into the generator (G).

G transforms z into fake data $G(z)$. A concurrent batch of real data x is sampled from the true data distribution $p_{\text{data}}(x)$. Both x and $G(z)$ are presented as inputs to the discriminator (D). D processes both batches, outputting scalar probabilities $D(x)$ (for real data) and $D(G(z))$ (for fake data).

2). Loss computation & backward pass

The outputs $D(x)$ and $D(G(z))$ are used to compute the two components of the loss in Equation (1): $\log(D(x))$ and $\log(1 - D(G(z)))$. The discriminator's loss is computed as $-\log(D(x)) + \log(1 - D(G(z)))$ (since it aims to maximize the sum, gradient ascent is equivalent to minimizing this negative sum). The generator's loss is computed as $\log(1 - D(G(z)))$ (or, more commonly in practice, $-\log(D(G(z)))$ for stronger gradients early in training). The gradients are then calculated via backpropagation and used to update the parameters of D and G alternately.

(2) Conditional generative adversarial networks

As noted in the future work section of the original GAN paper, adding extra inputs to both the generator (G) and discriminator (D) can lead to a conditional generative model. Following this idea, Mirza and Osindero [131] designed and implemented CGANs. The transformation from GAN to CGAN is shown in Figure 2.7.

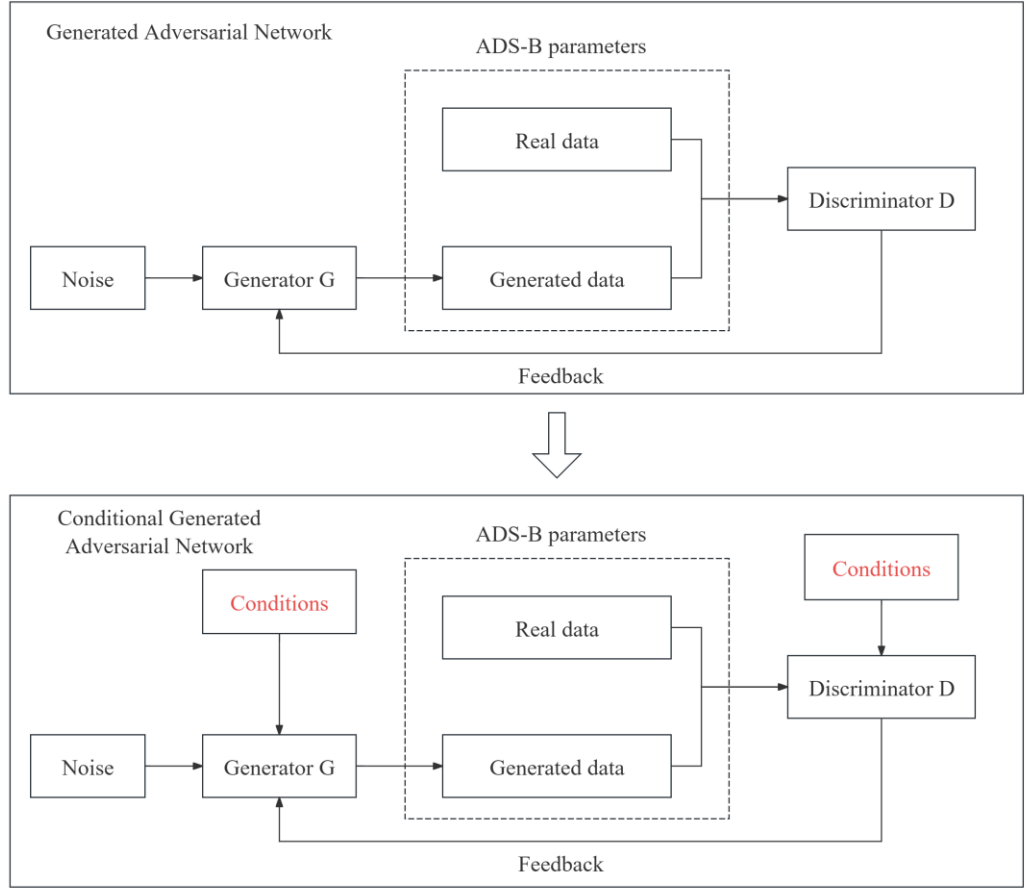


Figure 2.7 The transformation from GAN to CGAN.

By incorporating additional information or conditions, the conditional generative model can guide the data generation process and control the generated data. This extra information can take various forms, such as class labels, weather conditions, etc. The objective function will be modified to formula (2):

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim P_d} [\log (D(x|y))] + \mathbb{E}_{x \sim P_z} [\log (1 - D(G(z|y)))] \quad (2)$$

In addition to enhancing control over generated data, CGAN effectively addresses the training shortcomings of standard GANs. While traditional GANs struggle to learn multiple categories, making it challenging to construct a model that can handle various classes, CGANs can generate multi-modal distributions using different conditions to produce diverse data types, thereby mitigating the mode collapse issue commonly associated with GANs.

2.3.1.2 Characteristics of generative adversarial networks

(1) Advantages of generative adversarial networks

Compared to traditional generative models, GANs offer several notable advantages [132]:

- 1). Higher data generation quality: GANs significantly outperform existing generative models like VAEs in generation quality. This is because VAEs rely on approximated variational lower bounds and use loss metrics like cross-entropy or mean squared error (MSE) to calculate the distance between reconstructed and training samples. In contrast, GANs directly learn the target data distribution from the training set, resulting in superior performance, especially for high-frequency data.
- 2). Lower training complexity: GANs do not depend on Monte Carlo estimates or Markov chain predictions, and they make no prior assumptions about the target data distribution, making their computational approach relatively straightforward.
- 3). Faster generation speed: GANs generate complete samples at once, unlike methods that generate samples pixel by pixel (e.g., PixelCNN, WaveNet), leading to faster generation times.
- 4). Flexibility and scalability: The GAN framework, built on adversarial training mechanisms, imposes no formal constraints on the generator and discriminator beyond requiring differentiability, resulting in high flexibility and scalability.

(2) Shortcomings of generative adversarial networks

GANs have several inherent limitations [132]:

- 1). Weak control over generation: GANs use a unified adversarial loss function instead of various loss function designs found in traditional neural networks. Their training goal is not to optimize model convergence but to fit the target sample distribution through adversarial training. This approach allows a high degree of freedom in generated samples, meaning basic GANs lack the controllability necessary for conditional generation.
- 2). Training instability and mode collapse: The training process of GANs is essentially a competitive game between the generator and the discriminator, which can be highly unstable. If the generator captures a distribution that the discriminator misclassifies as real, it will focus on this mode for output, halting

further optimization. Training instability often causes the generator to converge to local optima, reducing output diversity and triggering mode collapse.

3). Insufficient constraint in learning modes: GANs' learning modes lack sufficient constraints. While GANs offer significant generative flexibility, their output quality remains constrained by the discriminator's binary evaluation mechanism. By operating via binary classification (real vs. synthetic), the discriminator prioritizes coarse feature discrimination over nuanced data understanding, limiting its capacity to capture comprehensive data patterns.

4). Poor performance on discrete data: The optimization guidance for the generator during training does not come directly from the information in the real dataset but rather from the discriminator's probability judgments of the generated data. The discriminator attempts to capture the differences between real and generated samples, feeding this information to the generator for iterative optimization. This feedback process is relatively friendly to continuous data but can struggle with discrete data, such as one-hot encoded forms. The discriminator easily captures the differences in representation between real and generated discrete data, rendering the generator's updates less meaningful. Consequently, GANs perform poorly on discrete data types (like speech, text, and music) compared to continuous spatial data.

2.3.1.3 Optimization methods of generative adversarial networks

(1) Optimization of model structure

The original generative adversarial model proposed by Goodfellow was relatively simple, consisting of a generator and a discriminator based on MLPs. The main contribution of the original GAN was to provide a framework embodying adversarial game theory. Improving the model structure is an effective way to enhance GANs. For example, DCGANs utilize deep convolutional networks to construct the generator and discriminator, employing continuous convolutional and deconvolutional layers to extract image features, which has become a benchmark structure for applying GANs in the graphics domain.

(2) Loss function optimization

The original GANs experienced issues related to their loss function during training,

such as gradient vanishing when the generated data distribution does not sufficiently overlap with the training data distribution. This issue cannot be resolved through model structure optimization alone, so improving the loss function is a feasible way to enhance the value of GAN-generated data. For instance, WGAN-GP replaces the original loss function with Wasserstein distance and employs gradient penalty to optimize Lipschitz constraints, thereby improving the training stability of GANs.

(3) Avoiding mode collapse

Mode collapse is a problem inherent to the adversarial mechanism of GANs, which results in the generator outputting data distributions that only cover part of the sample space, leading to many repetitive or similar samples. Adjusting training parameters, regularization, and using multiple generators and discriminators can help alleviate mode collapse and enhance sample diversity. Still, they have not entirely resolved the issue.

(4) Conditional output control

The generated sample data of GANs is constrained by adversarial training to fit the distribution of the real dataset. However, there is no conditional control to guide generation throughout the training process, resulting in a high degree of randomness. The quality of generated data can be improved by using conditional vectors to control attributes of the generated samples (e.g., CGANs), enhancing conditional constraints through sample classification, or indirectly manipulating latent variables from the perspective of mutual information to extract attribute relationships (e.g., InfoGAN).

2.3.2 Recurrent neural networks

To address the challenges of feature extraction in time-series data, Elman [133] developed the first fully connected RNN, known as the Elman network or simple RNN, in 1990, laying the foundation for subsequent developments in RNNs. RNNs are fundamental modules in deep learning, characterized by recurrent connections that inherently create deep structures. Even an RNN model with a single layer of hidden units can become very deep when unfolded over time. RNNs are primarily used to learn the relationships and patterns between sequential

inputs and outputs, making them advantageous for time-series forecasting. Their key feature is that the output of neurons at a given time can be fed back as input, enabling them to learn from the order of sequential data effectively.

Due to their excellent performance in time-series prediction, RNNs are widely applied across various temporal problems, including time-series forecasting [134], natural language processing [135], machine translation [136], and image captioning [137]. Aircraft TP is a typical time-series forecasting problem, and RNNs serve as the standard architecture for sequence modeling tasks due to their inherent capacity to process sequential data.

2.3.2.1 Evolution of recurrent neural networks

Elman's work sparked extensive research into RNNs, and many scholars have optimized and expanded RNNs from the perspective of network structures and combinations to suit various application scenarios. To address the challenge that basic RNNs face in processing long sequences, the LSTM network was developed, followed by the GRU, which further simplified the model structure and improved computational efficiency. LSTM and GRU brought RNN research and applications into a new stage of rapid development, with significant contributions from Graves in optimizing and popularizing LSTM [139-142].

Apart from LSTM and GRU, there are many other variants of RNNs, such as:

a). Bidirectional RNNs (BRNNs) [142]: BRNNs use two separate hidden layers to process data in both forward and backward directions along the time series, fully utilizing context information from both directions. b). Stacked LSTM [143]: In this variant, multiple LSTM layers form a stacked LSTM model, where each input at each time step corresponds to one output, rather than having all inputs correspond to a single output. This improves training efficiency and achieves higher accuracy. c). Graph RNNs (graph convolutional recurrent networks, GCRNs) [145]: GCRNs extend the standard RNN architecture from a hierarchical structure to a graph structure, enabling the network to learn spatiotemporal relationships from graph-structured and time-varying data, enhancing the model's learning speed. d). Memory RNNs [144]: These networks introduce external memory modules or enhanced memory mechanisms, allowing the network to capture and leverage long-term dependencies effectively.

These RNN variants have unique characteristics and are widely applied in different fields and scenarios.

(1) Basic recurrent neural networks

The simple recurrent network (SRN), developed by Elman in 1990, also known as the basic RNN, contains only a single layer of hidden units. Unlike a typical two-layer feedforward neural network, SRN introduces feedback connections between the hidden layer nodes. The SRN consists of four components: input, output, hidden, and context. The input, output, and hidden units belong to the input, output, and hidden layers, respectively, while the context units serve as links between the hidden layer nodes.

A combined illustration of the hidden layer neuron structure and the time-unfolded network structure can be shown in Figure 2.8 to visualize how the SRN processes sequences over time.

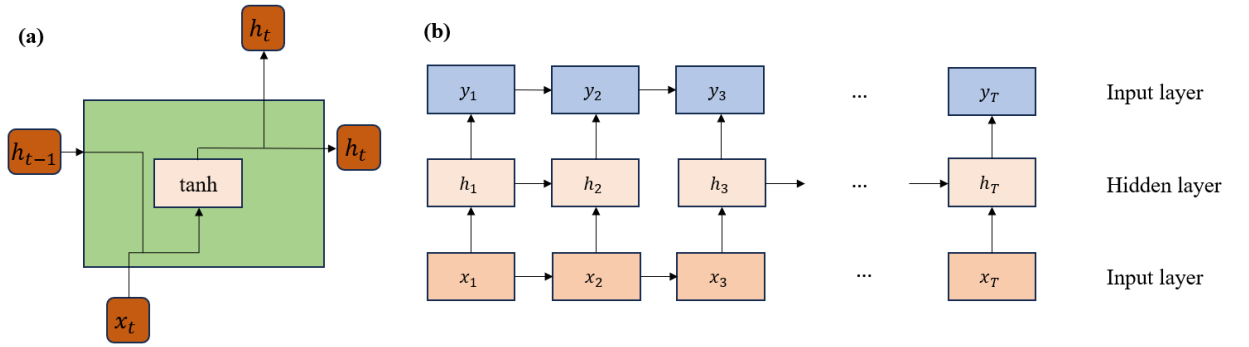


Figure 2.8 (a) SRN recurrent unit structure, (b) SRN network structure expanded over time.

SRN calculates hidden states and outputs through forward propagation and uses backpropagation through time (BPTT) to update weights and biases to achieve SRN training. Let vector x_t represent the input of the network at time t , and h_t represent the hidden layer state (i.e., the activity value of the hidden layer neurons). Then h_t is related to the input x_t at the current moment and to the hidden layer state h_{t-1} at the previous moment. The update formula of the hidden state of a simple RNN at time t is as follows:

$$h_t = \sigma(W_{hx} \cdot x_t + W_{hh} \cdot h_{t-1} + b_h) \quad (3)$$

where W_{hx} is the weight matrix input to the hidden layer state, W_{hh} is the weight matrix from the hidden layer state to the hidden layer state, b_h is the bias of the hidden layer, and σ is the activation function, usually a sigmoid function or a tanh function. The SRN output calculation formula is as follows:

$$y_t = W_{hy} \cdot x_t + b_y \quad (4)$$

where y_t is the output value, W_{hy} is the weight matrix from hidden layer state to output, and b_y is the bias of the output layer.

The calculation formula of SRN is relatively simple and is suitable for basic time series prediction and simple sequence processing tasks.

(2) LSTM networks

Although basic RNNs are highly effective for many simple time-series prediction and sequence processing tasks, they can face the issue of vanishing and exploding gradients when dealing with long sequence data. This problem arises because long sequences result in an intense RNN network, and during back propagation (BP), error signals either vanish or explode as they pass through the deep structure. Hochreiter and Schmidhuber [145] introduced the LSTM network to address this issue in 1997. The core innovation of LSTM is introducing a gating mechanism that controls the flow of information. The structure of an LSTM unit is illustrated in Figure 2.9.

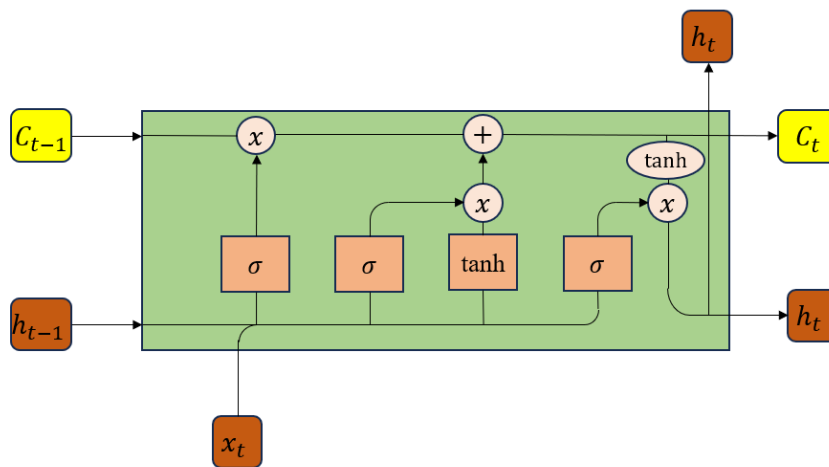


Figure 2.9 LSTM network unit structure.

LSTM has four structures: a). Forget gate (f_t) decides which information to discard from the cell state. b). Input gate (i_t) determines which values to update in the cell state. c). Cell state update ($\tilde{c}_t$) creates a new candidate cell state to potentially add to the current state. d). Output gate (o_t) controls the output based on the cell state. Let t be the current time step and $t - 1$ be the previous time step, then the relevant propagation equation can be expressed as:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \quad (5)$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \quad (6)$$

$$\tilde{c}_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \quad (7)$$

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{c}_t \quad (8)$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \quad (9)$$

$$h_t = o_t \cdot \tanh C_t \quad (10)$$

where W represents the weight matrix, b represents the bias, σ stands for the sigmoid activation function, $\tanh$ is the hyperbolic tangent activation function, C_t stands for the updated cell state.

These formulas describe the forward propagation process of an LSTM unit. The forget gate controls how much information from the previous memory cell state C_{t-1} is retained; the input gate controls how the current input x_t and the previous hidden state h_{t-1} influence the update of the memory cell state, and the output gate controls how much information from the memory cell state C_t affects the current hidden state h_t .

By introducing gates that decide whether to retain the existing memory, the LSTM unit can detect essential features from the input sequence and keep that information over long periods. This allows LSTM to capture potential long-range dependencies in sequences. These characteristics enable LSTM to excel in handling long sequence data and overcome the vanishing gradient problem that plagues basic RNNs.

(3) Gated recurrent units

In 2014, Cho et al. [146] introduced the GRU, a recurrent architecture that

allows adaptive learning of temporal dependencies across varying time horizons, making it versatile for sequence modeling. Similar to LSTM, GRU employs gating mechanisms (an update gate and a reset gate) to control information flow, though it omits LSTM's dedicated memory cell. The update gate modulates how much new information integrates into the hidden state, while the reset gate regulates retention of prior states, balancing memory and forgetting.

Due to its more straightforward structure, GRU is more suitable for building more extensive networks. From a computational perspective, with only two gates, GRU is more efficient and can save computational costs. The structure of a GRU unit is shown in Figure 2.10.

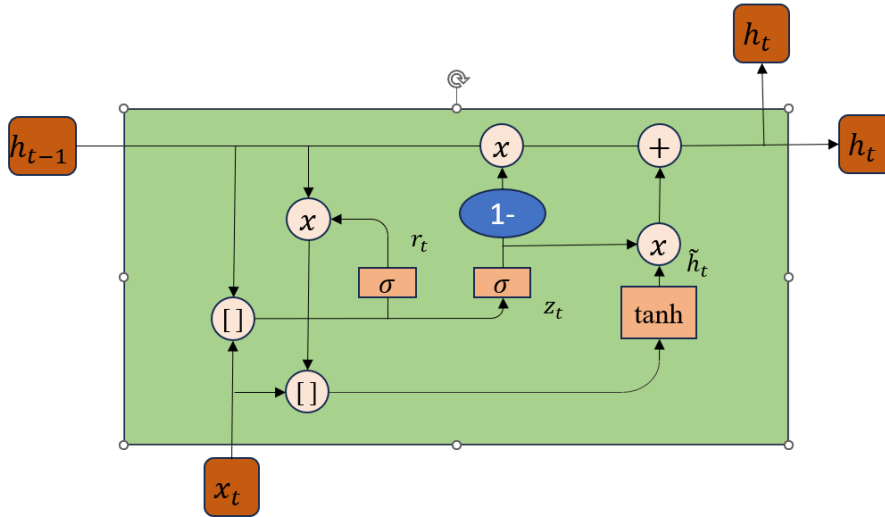


Figure 2.10 Structure of gated recurrent unit.

Below shows the calculation formula of the GRU forward propagation process:

$$z_t = \sigma(W_z \cdot x_t + U_z \cdot h_{t-1} + b_z) \quad (11)$$

$$r_t = \sigma(W_r \cdot x_t + U_r \cdot h_{t-1} + b_r) \quad (12)$$

$$\tilde{h}_t = \tanh(W_h \cdot x_t + r_t \cdot (U_h \cdot h_{t-1}) + b_h) \quad (13)$$

$$h_t = z_t \cdot h_{t-1} + (1 - z_t) \cdot \tilde{h}_t \quad (14)$$

where z_t represents update gate output, σ stands for the sigmoid activation

function, W represents weight matrix, U is the weight matrix from the previous hidden state, b represents bias, r_t is reset gate output, $\tilde{h}_t$ stands for the candidate hidden state, and $\tanh$ represents the hyperbolic tangent activation function.

2.3.2.2 Characteristics of recurrent neural networks

(1) Advantages of recurrent neural networks

Compared to feedforward neural networks and other deep learning models, RNNs have the following advantages:

- 1). Handling sequential data: RNNs are designed explicitly for processing sequential data and can capture temporal dependencies in input sequences. This makes them particularly suitable for time series forecasting, natural language processing, and speech recognition.
- 2). Parameter sharing: RNNs employ parameter sharing across time steps, enabling consistent transformation application at each stage. This design reduces parameter count, enhances training efficiency, and improves scalability.
- 3). Modeling temporal dependencies: RNNs can retain and use previous input information to compute the current time step. This is crucial for modeling temporal dependencies. For instance, in next-token prediction tasks, RNNs leverage their capacity to process sequential context, analyzing preceding tokens to estimate a probability distribution for the subsequent word, enabling autoregressive text generation.

(2) Shortcomings of recurrent neural networks

RNNs face several challenges apart from their advantages:

- 1). Vanishing and exploding gradients: RNNs suffer from vanishing/exploding gradient issues when modeling long-range dependencies. The vanishing gradient problem, for instance, results in ineffectual parameter updates, thereby hindering the model's ability to capture relationships across extended sequences. Conversely, the exploding gradient problem leads to overly rapid parameter updates, destabilizing the model. Although model variants like residual connections, grid LSTM [147], and layer trajectory LSTM [148] have

been proposed to mitigate these issues, they remain a significant obstacle, significantly when increasing model depth. Much progress is still needed to solve the gradient issues fully.

2). Training complexity: Due to the sequential nature of data, training RNNs is more complex than training neural networks for non-sequential data. BP in RNNs requires BPTT, which is computationally intensive and time-consuming.

3). Difficulty in parallelization: Because RNN computations depend on previous time steps, they are inherently sequential and difficult to parallelize. This limits the ability to fully leverage the powerful parallel computing resources available today, resulting in slower training, especially when handling long sequences.

4). Limited ability to capture long-term dependencies: While RNNs can model temporal dependencies, their ability to capture long-term dependencies is limited. Variants like LSTM and GRU significantly improve this, but they still struggle with long sequences. For tasks such as long-range TP, the accuracy drops sharply as the time span increases [149], [150].

2.3.2.3 Optimization methods for recurrent neural networks

When applying RNNs, it is essential to address their limitations by optimizing them according to specific application scenarios and objectives. The following are feasible optimization methods:

(1) Using advanced architectures

Advanced architectures like LSTM, GRU, Bidirectional RNNs, and Stacked RNNs emerged to overcome the shortcomings of vanilla RNNs. LSTM and GRU, for instance, explicitly tackle long-range dependency learning and gradient flow stabilization through gated mechanisms. Bidirectional RNNs consider both forward and backward sequences, fully utilizing contextual information. Stacked RNNs increase the model's expressiveness. Optimizing RNNs with these advanced architectures is an effective way to improve model performance.

(2) Choosing the right optimizer

The choice of optimizer significantly impacts model training efficiency and quality. Since vanishing and exploding gradients are common issues in RNNs, selecting the right optimizer for specific tasks is crucial. Several well-established optimizers, such as SGD, Adam, and RMSprop, can be used depending on the application needs.

(3) Gradient clipping

Gradient clipping—a widely adopted technique to mitigate exploding gradients in RNNs—caps gradient magnitudes during backpropagation, ensuring stable training dynamics. Any gradient that exceeds this threshold is clipped by setting a maximum norm for the gradients, helping maintain training stability.

(4) Handling sequence length

One of the primary causes of vanishing or exploding gradients is the sequence length. Managing sequence length appropriately for the specific application is critical. For very long sequences, truncated BPTT can be used to break long sequences into shorter sub-sequences for more manageable computations. Dynamic RNNs can be employed for variable-length sequences, which compute the sequences based on their actual length rather than padding them to a uniform length.

(5) Parallelization and acceleration

Parallelizing RNNs are inherently challenging due to their sequential nature. However, data and model parallelism can be leveraged to improve computational efficiency, especially on multi-GPU systems. Data parallelism involves splitting the data into batches and processing them parallel across multiple devices, sharing the same model parameters. Model parallelism distributes different parts of the model across various devices, each handling the computation of specific model parameters. This approach is beneficial when dealing with models with many parameters.

Additionally, hyperparameter tuning, choosing appropriate activation functions, regularization techniques, and preprocessing training samples are effective ways to enhance the adaptability of RNNs. Optimizing RNNs requires combining multiple methods and techniques to achieve the best results.

2.3.3 Sequence-to-sequence generative model

The sequence-to-sequence architecture maps variable-length input sequences to output sequences, forming the backbone of tasks like machine translation, text summarization, and dialogue generation. It employs an encoder-decoder framework: the encoder compresses the input into a fixed-dimensional latent representation (context vector), which the decoder autoregressively decodes into the target sequence.

Depending on the underlying neural network used, sequence-to-sequence models can be categorized into:

a). RNN-based sequence-to-sequence models: These rely on recurrent neural networks to process sequential data. b). Attention-based sequence-to-sequence models: Attention mechanisms allow the model to focus on different parts of the input sequence. c). SA-based sequence-to-sequence models: These models, like the Transformer, rely on SA mechanisms to capture dependencies in the input sequence without relying on RNNs. d). Hybrid sequence-to-sequence models: These combine multiple neural networks such as Attention, RNN, and CNN to enhance the performance of sequence-to-sequence tasks.

SA-based sequence-to-sequence models, such as the Transformer, are primarily used in aircraft TP. The Transformer architecture, with its self-attention mechanism, demonstrates superior efficacy in modeling long-range dependencies across sequential data, excelling in tasks such as machine translation, text summarization, and dialogue generation.

2.3.3.1 Evolution of Transformers

In 2017, Vaswani et al. introduced the Transformer architecture [83], a purely attention-based model that revolutionized sequence modeling. As illustrated in Figure 2.11, it employs stacked encoder-decoder blocks. Each encoder layer contains a multi-head self-attention (MHSA) mechanism and a position-wise feedforward network, each followed by residual connections and layer normalization to stabilize training and mitigate information degradation. The decoder mirrors this structure but incorporates two critical modifications: 1). a masked MHSA layer in its first sublayer to prevent future token leakage during autoregressive decoding, and 2). a cross-attention layer where the encoder's

output supplies keys (K) and values (V), enabling decoder queries (Q) to attend to the source sequence. To compensate for the lack of recurrence or convolution, positional encodings are injected into input embeddings, explicitly encoding sequential order.

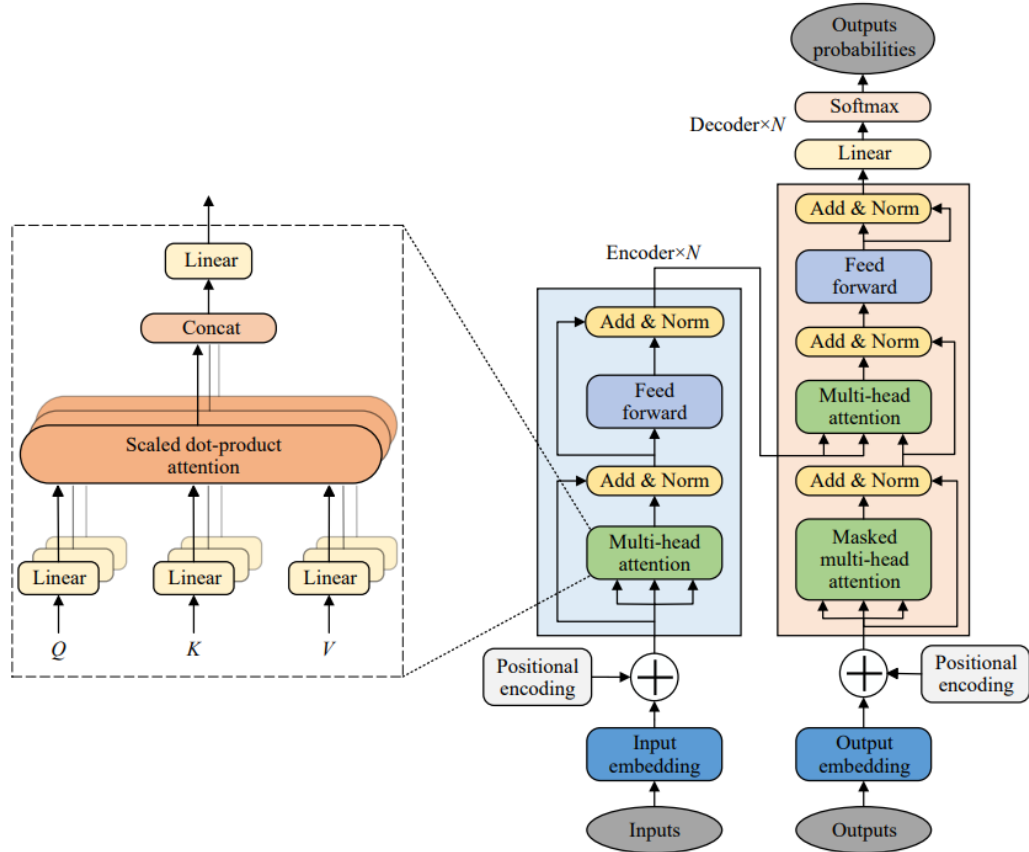


Figure 2.11 Transformer architecture.

As a sequence-to-sequence model, the emergence of Transformer has set off a wave of application research in natural language processing and has formed many variants. For example, BERT improves the performance of natural language understanding tasks by using a bidirectional SA mechanism for pre-training [151]. GPT, using a unidirectional SA mechanism for pre-training, is better adapted to generation tasks [152]. T5 (Text-To-Text Transfer Transformer) simplifies the conversion between different tasks by unifying text processing tasks into text-to-text conversion [153]. Transformer-XL enhances the ability to capture long sequence dependencies by introducing a memory mechanism [154]. XLNet, through autoregressive modeling (i.e. predicting the next word based on context), improves the performance of multiple natural language processing (NLP) tasks

such as question answering, natural language reasoning, and text classification [155]. The Transformer model has become more efficient and powerful in different tasks and application scenarios through these improvements and variants.

2.3.3.2 Characteristics of Transformers

(1) Advantages of the Transformer model

- 1). Strong parallelization: The SA mechanism of the Transformer allows simultaneous processing of all positions in the sequence during training, unlike RNNs, which require sequential processing. This significantly speeds up training.
- 2). Capturing long-range dependencies: The SA mechanism enables the Transformer to simultaneously attend to all positions in a sequence, inherently capturing dependencies across arbitrary distances without degradation, overcoming the distance limitations of recurrent architectures.
- 3). Good model interpretability: The SA mechanism's attention weights offer inherent interpretability. They reveal how the model prioritizes token-level interactions during decision-making, a critical advantage for model introspection, saliency analysis, and diagnosing biases or overfitting.
- 4). High flexibility: The Transformer model demonstrates broad applicability and is effectively employed in diverse domains such as natural language understanding, computer vision, and temporal forecasting.
- 5). Modular design: Since the Transformer comprises multiple identical encoder and decoder layers, its design and adjustments are more flexible and straightforward, making it easier to tailor the model for specific tasks.

(2) Shortcomings of the Transformer model

- 1). High computational resource requirements: Despite its strong parallelization capabilities, the total computational complexity of the SA mechanism is $O(n^2d)$, where n is the length of the input sequence, and d is the dimensionality of each vector. This quadratic relationship with sequence length means processing long sequences requires significant computational resources. The Transformer may underperform in resource-constrained environments compared to lighter models like smaller RNNs or CNNs.

2). Dependence on large-scale data: Transformers typically require large datasets to train and learn their parameters thoroughly. In scenarios where data is limited, the model may struggle to perform well and not generalize effectively.

3). Hyperparameter tuning complexity: The Transformer model has many hyperparameters (e.g., the number of layers, number of Attention heads, embedding dimensions), and understanding how these hyperparameters interact and affect the model's performance is complex. This increases the complexity of hyperparameter tuning, making it more challenging to optimize the model.

2.3.3.3 Optimization methods for Transformers

Since the Transformer model was proposed, many researchers have made various improvements to overcome its shortcomings and improve its performance. The following are some of the main enhancements that provide valuable references for optimizing the application of Transformer in different fields and scenarios.

(1) Reducing computational complexity

The complexity is reduced by linearizing the SA mechanism, making the model more suitable for handling long sequences. Techniques like sparse SA and focusing only on local windows and global content can lower computational complexity. Additionally, using locality-sensitive hashing (LSH) to approximate SA helps reduce the complexity further.

(2) Enhancing positional encoding

Relative positional encoding enhances a model's ability to learn relative token distances compared to absolute positional encoding. It improves long-range dependency modeling. Rotary positional embeddings (RoPE) also enhance the expressiveness of positional encoding.

(3) Hybrid architectures

Combining the Transformer with RNNs or CNNs can improve performance by leveraging the local properties of RNNs/CNNs and the global capabilities of the Transformer.

(4) Parameter sharing

Cross-layer parameter sharing reduces the number of model parameters, improving training and inference efficiency.

(5) Pretraining and fine-tuning

Using bidirectional pretraining followed by task-specific fine-tuning improves performance on natural language processing tasks. Unsupervised pretraining combined with supervised fine-tuning enhances the effectiveness of generative tasks.

(6) Improved loss functions and regularization

Label smoothing mitigates overfitting by regularizing confidence peaks, thereby enhancing generalization performance. Adversarial training fortifies model robustness by exposing it to perturbed inputs during training, increasing resistance to adversarial perturbations.

2.3.4 Convolutional neural networks

In 1998, LeCun et al. introduced the LeNet-5 architecture for handwritten digit recognition, pioneering using CNNs in practical ML applications [156]. LeNet-5 is a CNN based on the gradient BP algorithm, which laid the foundation for the subsequent development of CNNs. With features like local connections, weight sharing, and hierarchical feature extraction, CNNs have proven highly effective in processing complex images and data, becoming a fundamental network in deep neural networks.

Due to their unique properties, such as translation invariance and powerful feature extraction capabilities, CNNs are extensively employed in diverse domains, including image classification (e.g., handwritten digit recognition, object recognition), object detection (e.g., face detection, object detection in autonomous driving), image segmentation (e.g., tumor segmentation in medical imaging), video analysis (e.g., action recognition, video surveillance), NLP. CNNs also serve as a crucial deep learning foundation for aircraft TP, commonly used for tasks like extracting weather features and spatial trajectory characteristics.

2.3.4.1 Evolution of convolutional neural networks

LeNet-5, developed by LeCun et al., although it appeared at the earliest, is also a very successful CNN. Many banks in the United States used the handwritten digit

recognition system developed based on LeNet-5 in the 1990s to recognize the handwriting on checks. Since LeNet-5 was developed by LeCun and others in 1998, multiple CNN variants have been developed for different application scenarios and needs. For example, the squeeze-and-excitation network (SENet) improves the classification performance of the network by capturing the correlation between channels [157]. The ResNet network changes the content of network learning from complete output to learning residuals by introducing shortcut connections, which effectively alleviates the training problems of deep neural networks [158]. AlexNet introduces data enhancement and dropout, which significantly improves the classification ability of images [159]. GoogLeNet employs parallel multi-scale convolutional kernels to enhance image classification and detection performance [160]. The densely connected convolutional network (DenseNet) utilizes dense connections, where each layer receives concatenated feature maps from all preceding layers, enabling richer feature reuse [161].

(1) Typical convolutional neural networks

CNNs exhibit a structurally more straightforward design compared to other deep neural networks. The core architecture (Figure 2.12) comprises an input layer, alternating convolutional and pooling layers (interleaved and repeated as needed), fully connected layers, and an output layer.

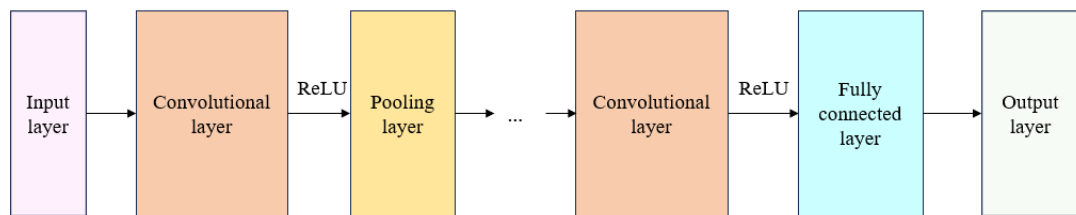


Figure 2.12 Basic structure of convolutional neural networks.

1). Convolutional layer

The convolutional layer is the most essential structural unit in the convolutional neural network model. The architecture employs multiple convolutional filters to process input data, extracting hierarchical features. Shallow convolutional layers usually only extract detailed features, while deep convolutional layers iterate and integrate these shallow features to obtain abstract features. The operation process

is shown in Figure 2.13.

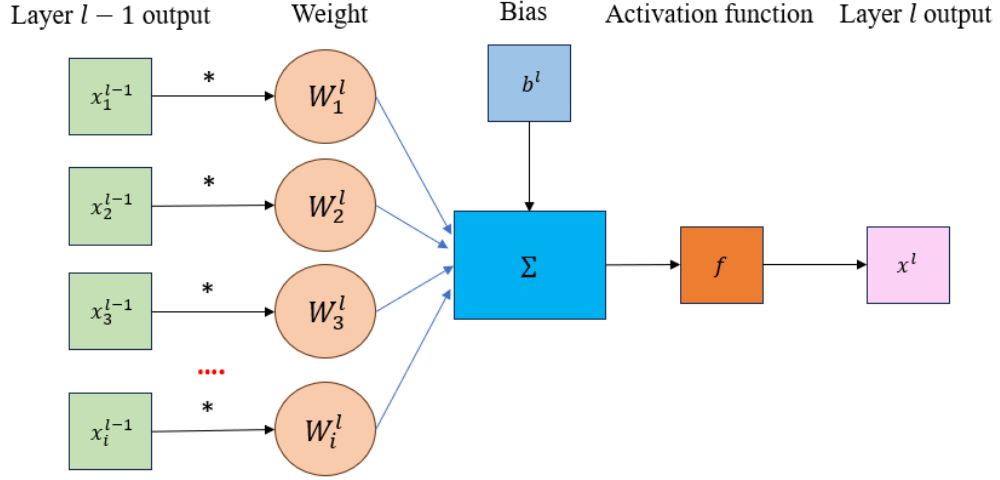


Figure 2.13 Convolutional layer operation process.

The mathematical expression of the convolutional layer is shown in formula (15):

$$x^l = f(\sum_{i \in M} x_i^{l-1} * W_i^l + b^l) \quad (15)$$

where W^l and b^l represent the weight and bias corresponding to the convolution filter at the i_{th} position of the input feature vector, respectively; x^{l-1} represents the feature map of the previous layer; x^{l-1} stands for the feature map of the current layer; f represents the activation function; M represents the set of feature maps.

2). Activation function

Following each convolutional layer, an activation function applies nonlinear transformations to the feature maps, enabling complex feature space transformations and enhancing model expressivity for improved label fitting. Typical activation functions include sigmoid, tanh, ReLU, etc. The activation function can be regarded as a component of the convolution layer. The operation process of the convolution layer is shown in Figure 2.13.

3). Pooling layer

Generally, the pooling layer is periodically embedded after the convolution layer. Using the pooling layer can reduce the dimension of the input features to a certain extent and further reduce the training parameters and computational cost of the

network model, which can not only accelerate the processing speed of the model but also prevent overfitting. The two most common pooling operations in CNNs are average and max pooling.

4). Fully connected layer

In a fully connected layer, all neurons interconnect with every neuron in the preceding layer. Following pooling operations, the features are flattened into a 1D vector and fed into the fully connected layer to capture high-level abstract patterns for classification. Typically positioned at the network's final stage, the fully connected layer optimizes classification accuracy by integrating global feature representations.

(2) Residual networks

Compared with shallow structures, deep CNN models have better performance. However, as the depth of the model grows, the gradient will disappear or explode, making the network training process more difficult and hindering the convergence of the network. To handle the above problems, He et al. proposed the ResNet in 2015 [158]. By introducing shortcut connections to the nonlinear convolutional layer, shallow features can be reused so that the content of network learning is changed from the complete output of the previous layer to the learning residual, which improves learning efficiency. ResNet comprises multiple basic residual units stacked together, forming an intense network. Each residual unit consists of numerous cascaded (equal width) convolutional layers and a cross-layer direct connection, and then output after activation by ReLU, as shown in Figure 2.14.

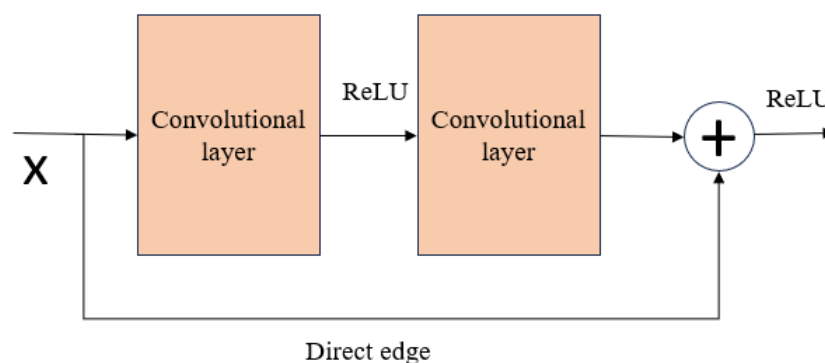


Figure 2.14 Schematic diagram of residual networks.

ResNet addresses the training challenges of deep neural networks through residual blocks with skip connections, which stabilize gradient flow and enable deeper architectures. This innovation has achieved state-of-the-art performance across computer vision tasks such as image classification, object detection, semantic segmentation, and visual tracking.

2.3.4.2 Characteristics of convolutional neural networks

CNN was developed from image processing (handwritten digit recognition) and has significant advantages in image recognition and processing. The characteristics of CNN can be intuitively described from the perspective of image data processing.

(1) Advantages of convolutional neural networks

1). Local connectivity: In CNNs, each neuron is connected only to a local region of the previous layer rather than to all neurons. This regional connectivity leverages the spatial locality of image data, reducing the number of parameters and improving computational efficiency.

2). Weight sharing: In convolutional layers, the same convolutional filter (kernel) is applied across the entire input image to extract features. This weight-sharing mechanism reduces the number of learning parameters, enhances training efficiency, and improves the model's generalization ability.

3). Translation invariance: Thanks to convolution and pooling operations, CNNs exhibit translation invariance. This means that even if the object in an image shifts position, the CNN can still effectively recognize and extract features.

4). Hierarchical feature extraction: CNNs hierarchically extract features via alternating convolutional and pooling layers, capturing low-level patterns (e.g., edges, textures) that progressively evolve into high-level abstractions (e.g., shapes, objects). This hierarchical feature learning mechanism—enabling multi-scale and spatially invariant pattern recognition—underpins their effectiveness in processing complex visual data.

5). Dimensionality reduction: Pooling layers down-sample feature maps by reducing their spatial resolution, thereby decreasing computational complexity and retaining critical spatial information. This dimensionality reduction mitigates

overfitting and improves generalization by emphasizing dominant features while discarding redundant details.

6). Versatility: While CNNs were pioneered for image-based tasks, their architectural principles have enabled successful adaptation to non-grid data, such as time series and text. With appropriate modifications, CNNs perform well across various tasks. In aircraft TP, trajectory data has both spatial and temporal features. Since spatial features are also fundamental to images, CNNs are frequently used as a core machine-learning model for TP.

(2) Shortcomings of convolutional neural networks

CNN has several limitations, as outlined below:

1). Large data requirements: CNNs rely heavily on extensive labeled datasets to achieve robust performance. A lack of sufficient data may lead to overfitting or suboptimal performance.

2). Sensitivity to noise: CNNs are sensitive to noise in the input images. Even small amounts of noise or perturbations can cause the model to make incorrect predictions.

3). Low sensitivity to position and orientation: CNNs have limited sensitivity to the positional and directional information within images. Small changes in the object's position within the image may result in the CNN failing to recognize it correctly.

4). Lack of robustness to rotation and scaling: CNNs are not inherently robust to transformations like rotation or scaling. Improving performance often requires data augmentation or specialized network structures, such as spatial transformer networks.

5). High computational resource requirements: The training and inference processes of CNNs demand significant computational resources, including GPUs and memory. This becomes incredibly challenging when handling high-resolution images or complex models, making hardware requirements critical.

2.3.4.3 Optimization methods for convolutional neural networks

(1) Data enhancement

Many shortcomings of CNN, such as large data demand, sensitivity to noise, insensitivity to position and direction, and non-robustness to rotation and scaling, are closely related to data. Data enhancement is an effective means of improving CNN's performance. For example, increasing the diversity and number of training data by rotating, translating, scaling, cropping, flipping, and adding noise can reduce overfitting and effectively improve the performance of CNN.

(2) Training process optimization

For specific application scenarios and goals, the performance of CNN can be effectively improved by setting appropriate model structures and hyperparameters, selecting appropriate model training methods and processes, and optimizing model parameters.

2.4 Deep Learning TP support data

The primary data resources for aircraft TP include aircraft performance, aircraft surveillance, meteorological, and flight plan data. Aircraft performance data is critical for TP methods based on dynamics. Deep learning-based TP methods focus on extracting trajectory features from relevant data to optimize model parameters and enhance prediction capabilities. The three main data types used for training deep learning models in this domain are aircraft surveillance, meteorological, and flight plan data.

Aircraft surveillance data records the actual flight trajectory and is the fundamental data source for training aircraft TP models. Due to the rich availability, which is easier to obtain and has higher accuracy and broadcast frequency than traditional radar surveillance data, ADS-B data have become the preferred data source for deep learning-based TP. Meteorological data is a crucial factor affecting aircraft trajectories and is essential for improving TP accuracy. Weather parameters—including wind speed/direction, temperature gradients, and atmospheric turbulence—are critical determinants of aircraft flight trajectory planning. Flight plan data reflects flight intent and is valuable for improving medium- and long-term TP accuracy. Flight plans are primarily held by national and regional aviation authorities, their subordinate agencies, and relevant airlines and operators. ATC, ATFM, and airline operations centers possess these data.

Access to flight plan data is generally obtained through collaboration with these organizations.

Below is an introduction to two essential data resources: ADS-B and meteorological data.

2.4.1 Automatic dependent surveillance–broadcast data

The ADS-B is a contemporary aviation technology for real-time aircraft monitoring. It consists of four parts: an airborne ADS-B output transmitter, an airborne ADS-B in receiver, and an ADS-B ground station. It monitors aircraft and perceives the flight situation in the air by broadcasting and receiving relevant aircraft flight status information. The system composition is shown in Figure 2.15. ADS-B data is the main data transmitted and exchanged between system users, recording the aircraft's location, heading, speed, and other flight status information.

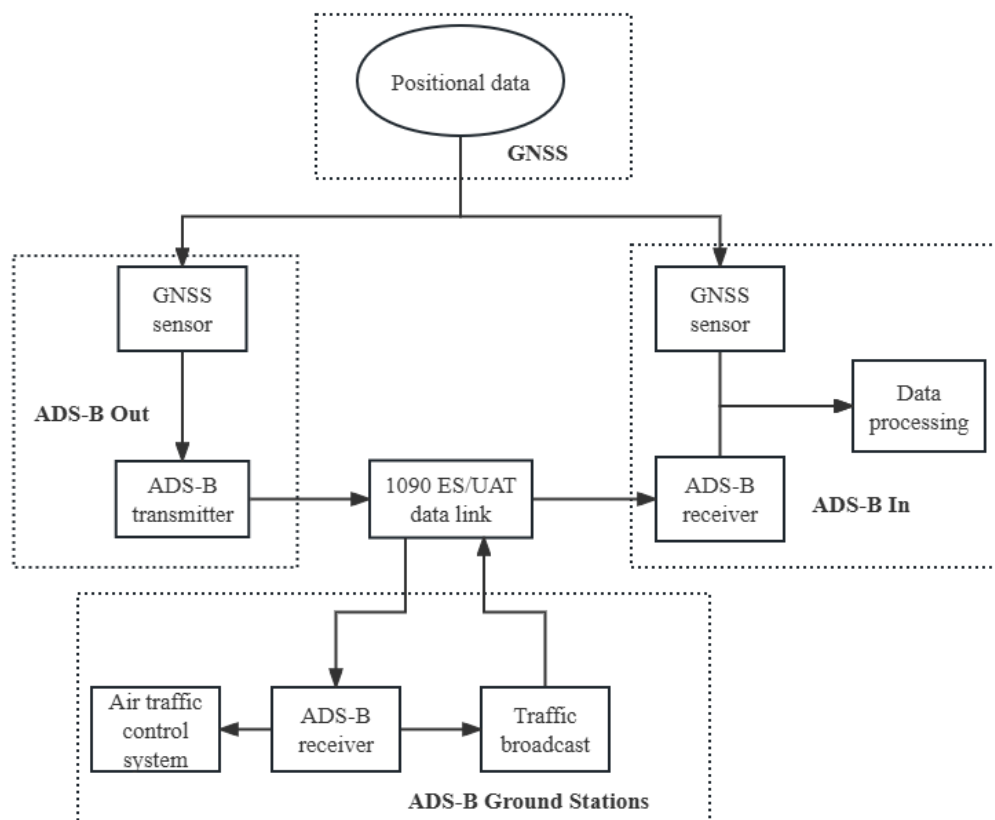


Figure 2.15 Composition of the automatic dependent surveillance–broadcast system.

2.4.1.1 Data content and format

The broadcast ADS-B data includes ICAO code, aircraft position, time, ground speed, track angle, data integrity, and other information. After receiving the transmitted data, the ADS-B ground station encodes all the data into Asterix cat021 format and forwards it to the ADS-B IN component of the ATC system. This structure aligns with the standard schema of ADS-B historical trajectory datasets, which comprise nine core fields detailing flight parameters (e.g., position, altitude, velocity, and identification metadata) [162], shown in Table 2.3.

Table 2.3 ASTERIX category 021 message fields.

Category	Description	Data example
Position in WGS-84 co-ordinates	Latitude and longitude in WGS-84 Co-ordinates.	Lat: 39.9042° Lon: 116.4074°
Target address	24 bit target address (emitter identifier) assigned uniquely to each target.	3C65B9 (ICAO number)
Time of reception for position\velocity	Time of reception of the latest position\velocity squitter in the ground station, in the form of elapsed time since last midnight, expressed as UTC.	45982.15 (UTC time)
Geometric height	Minimum height from a plane tangent to the earth's ellipsoid, defined by WGS-84, in two's complement form.	3025 (feet)
Quality indicators	Navigation integrity category (NIC) and other quality indicators.	NIC=8
Flight level	Flight level from barometric measurements in two's complement form.	350 (FL)

Geometric vertical rate	Geometric vertical rate, in two's complement form, with reference to WGS-84.	-12.8 (feet/minute)
Airborne ground vector	Ground speed and track angle elements of airborne ground vector.	GS: 480 kt, Track: 125.6°
Target identification	Target (aircraft or vehicle) identification in 8 characters, as reported by the target.	"CPA475" (callsign)

2.4.1.2 Data characteristics

(1) Update frequency and accuracy

ADS-B transmits position updates every 0.5 seconds, while other information, such as aircraft address codes and flight numbers, are updated every 1 second. Under normal circumstances, two ADS-B messages are expected every second. However, because the time information in these messages is recorded in 1/128-second units, rounding errors can occur when converting to conventional time units (hours, minutes, seconds). As a result, some timestamps may have three messages while others might only have one. Theoretically, there should always be at least one message per second. The update frequency of ADS-B is significantly higher than traditional radar systems, which typically update every 4 to 15 seconds [163]. This higher update rate allows for a reduction in the minimum required separation between aircraft, thereby increasing airspace capacity.

Leveraging the GNSS, the ADS-B system improves location accuracy to better than 0.1 nautical miles [164]. Additionally, by broadcasting data, ADS-B overcomes the limitations of radar surveillance systems, such as significant signal loss during extreme weather conditions and the inability to cover remote, oceanic, and low-altitude areas. As a result, ADS-B has become a vital tool in surveillance operations, playing an essential role in ATM by providing higher accuracy, continuity, and broader coverage than traditional radar systems.

(2) Data quality

ADS-B reports incorporate quality assessment metrics—such as Navigation Integrity Category (NIC), Navigation Uncertainty Category for Position (NUCp), and Navigation Accuracy Category for Position (NACp)—to evaluate the reliability of transmitted navigation data. NIC and NUCp values, which relate to the horizontal protection level (HPL), are crucial for ensuring the integrity of the ADS-B report. The data is considered precision condition data when $NIC > 8$ or $NUCp > 7$. However, these parameters may sometimes fall below the expected values or even be absent. NACp, which measures accuracy, is defined by the estimated position uncertainty (EPU), using a 95% confidence constraint to estimate position uncertainty. The FAA requires a minimum NACp value of 8 ($EPU < 92.6$ m). Generally, when $NIC \geq 7$, the NACp value is 8; when $NIC < 7$, the NACp value drops to 0.

The 1090 MHz frequency used by ADS-B is highly congested due to spectrum sharing by multiple systems, leading to same-frequency interference that can disrupt the system's regular operation. Complex electromagnetic environments are the main factor contributing to reduced ADS-B surveillance performance. This reduction is primarily reflected in data loss and incomplete data. Data duplication may also occur due to retransmissions by ground-receiving stations. In a study analyzing ADS-B data collected by the Hong Kong Observatory (HKO) from December 23 to December 27, 2016, covering 2,100 flights (both arrivals and departures), it was found that 95% of trajectory points had complete data parameters. Most trajectory points had a time interval of approximately 1 second, with a maximum interval of less than 10 seconds. This falls short of the theoretical expectation of one message per second, confirming the commonality of abnormal data phenomena. In practice, ADS-B data update intervals may vary based on flight conditions such as flight phase, GNSS receiver performance, and airspace density [163]. Therefore, these discrepancies must be analyzed and processed according to specific situations.

2.4.1.3 Main data sources

Currently, ADS-B data resources are widely available, and several organizations and companies provide comprehensive ADS-B data services:

(1) Flightradar24

Hosted and operated by the Swedish company Svenska Resenätverket AB, Flightradar24 is a comprehensive flight tracking platform offering real-time global air traffic information. It has over 40,000 ADS-B receivers worldwide. The primary channel for collecting information is ADS-B data received via ground stations or satellites. It also supplements its data collection with MLAT (for non-ADS-B aircraft), radar data from specific regions, and FLARM (for small aircraft like gliders). Flightradar24's data is accessible via its website and mobile apps, providing services like real-time flight tracking, historical flight data, and data subscriptions.

(2) FlightAware

FlightAware, run by a U.S.-based multinational tech company, is one of the most comprehensive flight-tracking platforms globally in terms of data resources. It operates a network of over 35,000 ADS-B ground stations in more than 200 countries and territories. FlightAware offers real-time flight tracking, flight trajectory data (real-time, historical, and predictive), data application tools, and industry solutions.

(3) OpenSky Network

The OpenSky Network is a community-based, open, non-profit platform comprising over 6,000 sensors operated by volunteers, industry supporters, and academic/government organizations. It archives all unfiltered raw data collected, including ADS-B data, in one of the most significant historical databases with more than 30 trillion messages. The platform provides data access and application tools like OpenSky Explorer for browsing historical air traffic, a detailed aircraft metadata database, and curated datasets for specific research needs. It supports researchers in improving ATC technologies and processes.

(4) ADS-B Exchange

ADS-B Exchange is a community-driven, open platform that provides free access to its data. The platform relies on a global network of ADS-B ground receivers operated by enthusiasts and institutions. These receivers capture raw, unfiltered data directly from aircraft, including all types of flights, such as private flights. ADS-B Exchange offers several API options for accessing flight data, including a

commercial-grade API for enterprises and a lightweight API for hobbyists. These APIs support the retrieval of real-time airspace information and historical data.

(5) Plane Finder

Operated by Pinkfroot Ltd. in the UK, Plane Finder collects flight trajectory data through ADS-B receivers, MLAT (which calculates aircraft position using signals received by multiple ground stations), FAA feeds, and FLARM (for monitoring small planes and gliders). The platform offers real-time flight tracking based on trajectory data, historical data playback, data sharing, mobile apps, and commercial-grade enterprise solutions.

2.4.2 Meteorological data

There are two main uses for meteorological data required for aircraft TP. One is to be used for model training, historical tracks, and related data (such as track, flight intention, etc.). The other is to be used as a condition for predicting the track, usually using weather forecast data or historical statistical data. Meteorological data related to aircraft TP include temperature, humidity, wind speed, direction, clouds, precipitation, etc. When meteorological data is not available, estimated values are sometimes used instead. The European Centre for Medium-Range Weather Forecasts (ECMWF) and the North American Mesoscale Forecast System (NAM) are widely utilized meteorological datasets, which support medium-range meteorological forecasting in aviation and climatology, among other fields.

2.4.2.1 Data types and exchange formats

(1) Data types

When using ML for aircraft TP, various weather data are crucial in improving model performance and prediction accuracy. These weather data types may represent different stages of similar information or products containing multiple data types. The selection of weather data depends on specific requirements, compatibility with trajectory information, and the availability of resources. Below are commonly used weather data types in TP:

- 1). Wind speed and wind direction: Wind speed and direction at different altitude levels (e.g., upper-level winds) affect horizontal and vertical aircraft movement.

- 2). Temperature: Temperature at various altitude levels influences aircraft performance and fuel efficiency. Unique temperature phenomena, such as temperature inversions, can affect flight level selection.
- 3). Air pressure: Atmospheric pressure at different levels is used to correct flight altitude and assess aircraft performance.
- 4). Humidity: Relative humidity affects cloud formation and the risk of icing.
- 5). Precipitation: Precipitation intensity and type (e.g., rain, snow, hail) influence visibility and flight safety. The location and movement speed of precipitation areas are also crucial factors.
- 6). Cloud cover: Cloud types, coverage, and height affect flight visibility and turbulence risk. Cumulonimbus clouds, in particular, are associated with strong turbulence and thunderstorms.
- 7). Turbulence: Predictions of turbulence intensity and location are critical for flight comfort and safety, including clear air turbulence (CAT) and terrain-induced turbulence.
- 8). Thunderstorms: The location, movement path, and intensity of thunderstorms significantly impact route selection and flight safety.
- 9). Visibility: Horizontal and vertical visibility data are vital for takeoff and landing operations.
- 10). Echo top (ET): ET is a meteorological radar product representing the top height of radar echoes. It is used to characterize the distribution and intensity of thunderstorms and strong convection, helping to analyze the vertical structure and strength of thunderstorms. Research indicates that ET is one of TP's most suitable overall weather products [92].

(2) Data format

Standard data exchange formats for weather data include six types, each suitable for different kinds of data handling and applications:

- 1). NetCDF (network common data form): NetCDF is a self-describing and portable file format well-suited for storing multi-dimensional array data, such as

meteorological data. It is widely used in climate and weather research due to its ability to store large, complex datasets.

2). GRIB (gridded binary): GRIB is a compressed binary format specifically designed for storing and transmitting weather forecast data. It is beneficial for handling large datasets in numerical weather prediction models.

3). HDF5 (hierarchical data format version 5): HDF5 is a versatile format for storing and managing complex data, supporting multi-dimensional arrays and rich metadata. It is used for a variety of scientific data, including weather datasets.

4). CSV (comma-separated values): CSV files store structured tabular data in a simple, human-readable format. While CSV is not typically used for large-scale meteorological data due to its limitations in handling complex datasets, it is helpful for more straightforward applications or smaller datasets.

5). JSON (Javascript object notation): JSON is a lightweight data exchange format widely used for Web-based applications to transfer and store structured data. It is helpful for representing hierarchical data structures and can be used to transmit weather data over networks.

6). GeoTIFF: GeoTIFF is an image file format that includes geographic information, making it ideal for storing raster data. It is commonly used to represent the spatial distribution of meteorological elements such as precipitation, temperature, humidity, wind speed, and ET data, providing geographic context for the data.

These formats are chosen based on the specific needs of the weather data application, such as data complexity, size, and ease of integration with other systems.

2.4.2.2 Data sources

(1) European Centre for Medium-Range Weather Forecasts

ECMWF is an independent intergovernmental organization with over 30 members and cooperating countries. The ECMWF specializes in meteorological data reanalysis, producing medium-range to seasonal forecasts, and advancing numerical modeling and data assimilation methodologies. Additionally, it

operates the EU's Copernicus Atmosphere Monitoring Service and Climate Change Service, supporting global environmental and climate policy initiatives. Some of its notable products include real-time weather forecasts, reanalysis datasets (such as ERA-Interim and ERA5), seasonal forecasts, and climate monitoring products, primarily available in GRIB and NetCDF formats.

(2) AMDAR

AMDAR is a key component of the World Meteorological Organization (WMO)'s Integrated Global Observing System and uses sensors and equipment on commercial aircraft to collect meteorological data. The data collected includes temperature, wind speed, wind direction, pressure, and, increasingly, humidity and turbulence measurements. These data are transmitted in real-time to ground meteorological stations for aggregation, processing, and storage. Over 700,000 high-quality observations are collected daily, providing essential high-altitude data, particularly in remote and oceanic areas where traditional meteorological networks fall short. Some AMDAR data are available through public channels, though the publicly accessible data may be processed and aggregated, excluding detailed raw observations.

(3) WorldClim

WorldClim is a high-resolution global climatology database curated by an international consortium, offering 19 bioclimatic variables and monthly baseline climate layers for ecological modeling and species distribution studies. These datasets include temperature (e.g., annual mean temperature, monthly mean temperature, maximum and minimum temperatures) and precipitation data (including seasonal variation). WorldClim offers past climate data (1970-2000 averages), current and future climate predictions based on different climate models and scenarios. The spatial resolution ranges from 30 arc seconds (~1 km) to 10 arc minutes (~18 km). WorldClim is a free online platform that provides high-resolution climate datasets for ecologists, geographers, climatologists, and other researcher.

(4) National Centers for Environmental Information

The National Centers for Environmental Information (NCEI), a division of the

U.S. National Oceanic and Atmospheric Administration (NOAA), serves as a primary repository for archiving and disseminating global environmental datasets, including weather, climate, hydrology, and space weather observations. Managing over 37 petabytes of data, NCEI integrates operational forecast models such as the North American Mesoscale (NAM) and Global Forecast System (GFS), while providing open-access historical climate records and real-time meteorological variables (e.g., temperature, precipitation, wind speed) via its public digital platform.

(5) China Meteorological Data Service Center

The China Meteorological Data Service Center (CMDC), a division of the China Meteorological Administration (CMA), serves as a critical node within the WMO's global data exchange framework. It provides various meteorological and climate data services, including real-time weather data (e.g., observations, weather forecasts, radar data, and satellite cloud imagery) and historical weather data. CMDC also offers long-term climate data and statistical analyses that support climate change research. While the primary coverage is for China, it also provides some global data through international collaboration and satellite observations. Users can access the data through Web services, API interfaces, or mobile apps.

2.5 Chapter summary

This chapter primarily focuses on the theoretical and technical foundations necessary for the subsequent deep learning-based TP methods research. The specific contents include the following:

1). Summary of the fundamentals and developmental needs of aircraft TP

The chapter outlines critical concepts of aircraft TP, the division of trajectory phases, and different classification methods for TP. It then analyzes the basic framework of TP under the new TBO operational model, emphasizing the latest requirements for higher accuracy, faster computation speed, longer LAT, more complete trajectories, and more excellent stability.

2). Summary of current status of aircraft TP research

This section outlines the current research status of aircraft TP. It is structured to

review the field through three key methodological approaches: Dynamics methods, state estimation methods, and machine learning methods. It then introduces the application of above TP methods, and last gives an overview of the research status.

3). Summary of the evolution, technical characteristics, and optimization methods of typical deep learning networks

This section reviews and summarizes the formation, development, advantages, limitations, and optimization methods of various deep learning models, such as GAN, RNN, CNN, and sequence-to-sequence networks. The goal is to provide a foundation for effectively applying aircraft TP's latest deep learning advancements.

4). Overview of critical data resources for deep learning-based TP

The chapter highlights two essential data resources for aircraft TP: ADS-B and meteorological data. The contents, exchange formats, update frequencies, quality characteristics, and key data sources are summarized for ADS-B data. The types of data, exchange formats, and significant data sources are outlined for meteorological data, giving a comprehensive understanding of the dataset resources crucial for TP tasks.

Chapter 3.

Trajectory prediction technology framework based on deep learning

Deep learning has emerged as a transformative force in big data applications due to several key attributes that align well with the demands of processing vast and complex datasets. Exploratory research in the field of aircraft TP also demonstrates the promising potential of this technology in this domain. Studying the characteristics, application methods, and system features of deep learning in aircraft TP helps to understand its intrinsic attributes better and promotes the field's development.

3.1 TP system composition

As a critical component of ATM DSTs, TP can be delivered in various software forms, such as modules, plugins, components, or services. The TP system consists of multiple software units, including data input, data management, data processing, and data output, which work together to solve aircraft TP problems and exhibit the fundamental characteristics of an information system. The TP systems used in real-world ATM operations primarily rely on physics-based models. However, prediction models based on ML have unique advantages that physics-based models lack. The coexistence and even integration of both approaches is a practical need. Therefore, the system architecture design should accommodate both types of TP models, allowing for the flexibility to incorporate physics-based models and machine-learning models in a cohesive framework.

3.1.1 Basic system architecture

To promote the exchange and sharing of research findings in TP and to build a TBO-based ATM operational environment, FAA/Eurocontrol AP16 proposed a general TP model. This model defines a TP system composed of four key modules:

preparation, computation, update, and output, along with the corresponding TP data processing workflow. This model is a conceptual framework that aims to be concise for ease of understanding the core content. From the perspective of operational applications, additional auxiliary modules must be incorporated to form a complete information system capable of delivering the necessary TP functionalities. As a specialized subset of TP systems, deep learning-integrated TP frameworks must conform to the core structure of conventional TP models, preserving identical logical dependencies and essential interface partitioning.

Figure 3.1 illustrates the basic architecture of a TP system embodying this approach. It logically describes the essential components and their relationships within the TP system, ensuring alignment with the general TP model while accommodating the unique requirements of deep learning-based prediction systems. This structure is a foundation for developing a comprehensive TP system that seamlessly integrates with existing ATM operations, enabling efficient and accurate TP in a TBO environment.

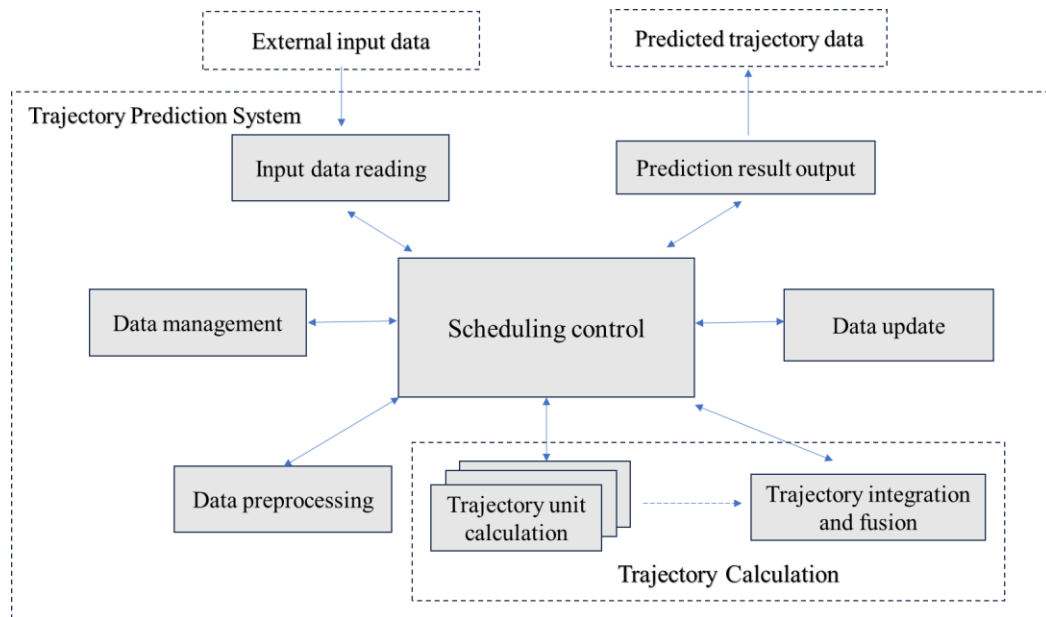


Figure 3.1 Basic architecture of TP system.

In this architecture, the data preprocessing, data update, and prediction result output units correspond to the preparation, data update, and data output functions of the AP16 general TP model. The trajectory unit computation and trajectory integration/fusion unit handle the trajectory computation functionality of the

general TP model. Along with the scheduling control, data management, and input data reading units, these components form a complete TP system that executes TP tasks. The specific roles of each unit are described as follows:

1). Scheduling control: This unit monitors external system requests and internal unit operation status in real-time. It triggers the execution of relevant software units based on preset conditions and coordinates their operations to complete the TP task.

2). Data management: This unit provides storage, organization, and retrieval capabilities for data required in TP. Physics-based TP includes storing data such as trigger conditions for data updates, flight intent, aircraft performance models, weather models, adaptation data, preprocessed input data, trajectory data from prediction units, and the final fused trajectory data. For ML-based methods, it may store data such as model parameters, preprocessing results, and similar trajectory data.

3). Input data reading: This unit reads external data required for TP, including aircraft initial state data, flight intent, weather conditions, and airspace information, all of which are time-sensitive.

4). Data preprocessing: This unit cleans, reorganizes, and normalizes input data to ensure consistency. Depending on the needs of the prediction unit, it may also perform specialized tasks like extracting aircraft states and flight intent and constructing behavioral models.

5). Trajectory unit computation: This unit predicts trajectory segments or complete trajectories. This involves software-based TP models with predefined parameters for ML methods. These models can include single neural networks or hybrid models combining multiple neural networks and traditional ML algorithms.

6). Trajectory integration/fusion: This unit integrates or fuses the output trajectory segments from various prediction models into a final =TP result. This process is handled by a trajectory integration/fusion model implemented as a software module.

7). Data update: This unit checks the conditions that trigger a recalculation of the trajectory. When the conditions are met, a new round of data reading,

preprocessing, and TP, generating updated trajectory data, is initiated.

8). Prediction result output: The system outputs the predicted trajectory data, including the aircraft's predicted trajectory and any associated errors or warnings. The users of this data may be human operators or, more often, other software systems such as ATM DSTs.

The system's basic architecture logically describes the key elements and their interrelationships within the TP system. The logical structure remains the same whether using physics-based or machine-learning models or even a hybrid of both. Both models are considered part of the trajectory computation unit, meaning this architecture is equally applicable to TP systems based on physics and ML models.

3.1.2 Trajectory calculation

Trajectory computation encompasses two components within the system's foundational architecture: trajectory unit calculation and integration fusion. As the central component of the TP system, it produces the final predicted trajectory.

3.1.2.1 Integration framework

For deep learning and other iterative optimization algorithms, no single algorithm is efficient or effective for all problems. Similarly, each deep learning-based TP model has an optimal range of applications. Using a single TP model or method can often lead to inefficiency and complexity in complex scenarios, such as gate-to-gate complete TP or variable LAT TP. Therefore, integrating multiple TP models or methods is essential.

The integrated framework for applying multiple TP models is illustrated in Figure 3.2. Among them, the TP model is a prediction unit calculation module that has been materialized as a software module. The design of the trajectory integration fusion unit mainly determines the integrated fusion algorithm of the calculation output results of the relevant prediction units, including selection, staged integration, and data fusion. Selection means selecting one of the multiple prediction results as the final predicted trajectory. Staged integration, if the output result of the prediction unit calculation module is the expected trajectory of different route stages, it is necessary to splice the trajectories of each stage to form a multi-segment trajectory or a full-segment trajectory from gate to gate. Data

fusion fuses the output results of different prediction unit calculation modules (such as weighted average based on spatial distance) to form the final predicted trajectory.

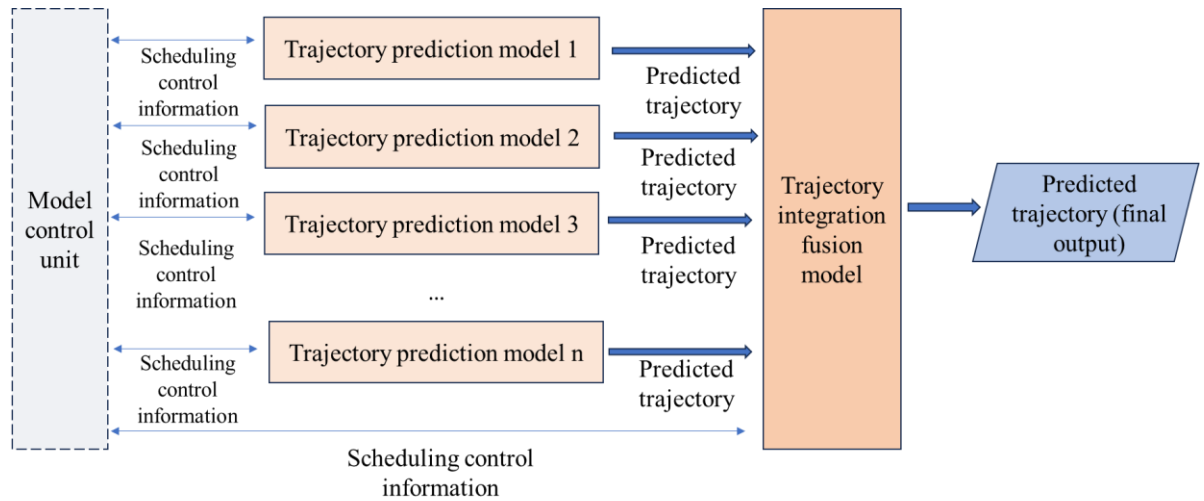


Figure 3.2 Integrated application framework of TP models.

3.1.2.2 Model description

To facilitate the sharing and application of the TP models and the trajectory integration and fusion models that have been materialized into software units, it is essential to organize them according to relevant software engineering standards. In addition to the software code, detailed documentation is crucial as it is the foundation for correctly using the TP models. The software unit description for TP models should include the following key elements:

- 1). Name: A descriptive name for the TP model.
- 2). Scope of application: Defines the scenarios where the model is applicable, such as short-term or long-term TP, terminal area arrival predictions, etc.
- 3). Functionality and performance: Describes the functions and performance metrics of the model, with particular emphasis on the TP accuracy metrics.
- 4). Invocation method: Specifies the process to execute the TP model, such as an APP call, Web service call, or other APIs.
- 5). Runtime environment requirements: Specifies the necessary infrastructure, including hardware, software, network resources, and other dependencies such as database systems, Web servers, or runtime environments.

- 6). Parameter configuration details: Describes all parameters, including their names, meanings, purposes, types, formats, ranges of values, and how they affect the prediction outcomes. This part focuses on model parameters and hyperparameters for ML-based TP models.
- 7). Interface definition: Provides detailed information about the input and output data, including their names, meanings, types, formats, and acceptable ranges of values.
- 8). Model training and testing documentation: This part should describe the sample data used for training and testing, along with ML-based models' training and testing methodologies. It should also include details about the sample data's composition, size, accuracy, and resolution.
- 9). Additional notes: Includes any specific information, particularly regarding factors that may impact the accuracy and confidence of the TPs.

The TP model software unit documentation should be accessible to human users and machines. Therefore, it is recommended to use appropriate standards and formats for description, such as Representational State Transfer (REST), Web Services Description Language (WSDL), Universal Description, Discovery, and Integration (UDDI), JSON-RPC, OpenAPI, and other Web service-related standards. This approach ensures that the software units are well-documented and can be easily integrated into broader systems, enabling seamless interoperability and use.

3.2 Construction and application of TP system

The construction and application of the TP system based on deep learning includes the rules of general information system construction and reflects the characteristics of deep learning technology application. Figure 3.3 shows the system's construction and application process based on deep learning.

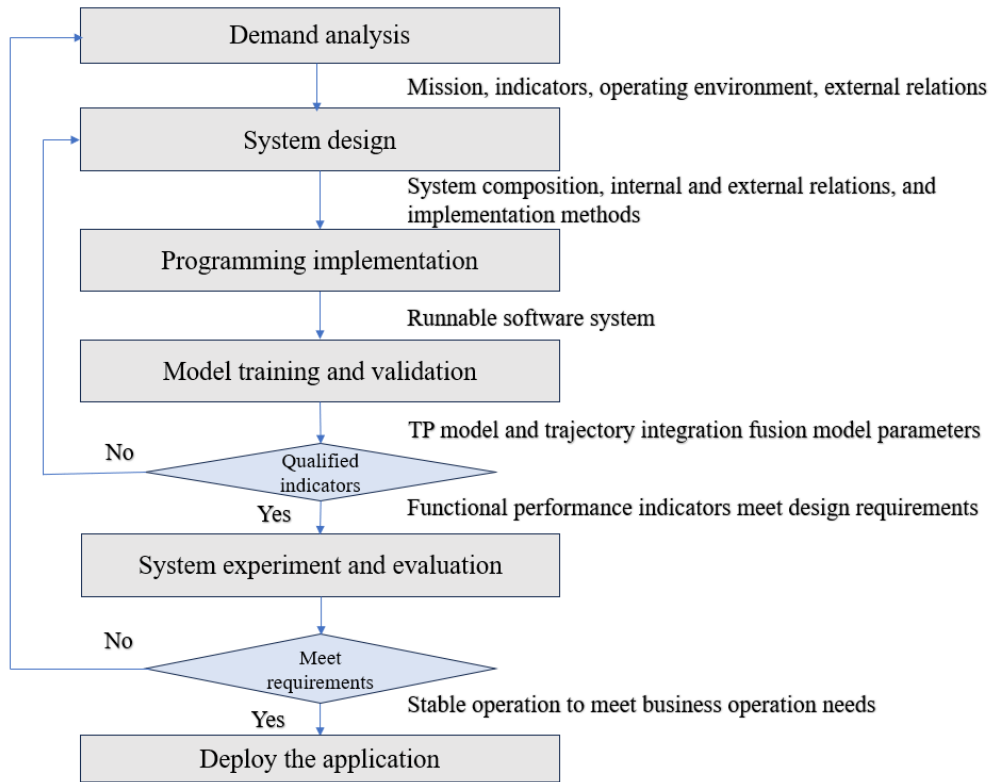


Figure 3.3 Construction and application process of TP system based on deep learning.

3.2.1 Demand analysis

Based on the specific conditions of the TP system application service objects, application deployment environment, usage requirements, and available resources, the business requirements for the TP system are sorted out and further converted into more specific and clear technical demands for the design and usage of the traction system.

3.2.2 System design

Based on the proposed technical requirements, design the system components, the internal and external relationships of the system, and the system shared database, and then design each system unit on this basis. The design of the trajectory calculation model is the core of the system design, including the design of two types of units: trajectory unit calculation and trajectory integration fusion. The trajectory unit calculation unit's design is essentially the TP model's design, including which essential deep neural networks to use and how to apply and connect, which data to use for model training and how to train, which optimization method to use for training, etc. The typical structure of the deep learning TP model

is shown in Section 3.4.1 of this chapter. The design of the trajectory integration fusion unit mainly determines the integrated fusion algorithm of the calculation output results of the relevant prediction units.

3.2.3 Programming implementation

The process of programming each unit of the designed system separately and combining them to form a complete system. According to the characteristics of each unit of the TP system, a suitable software framework can be selected for development. Common software frameworks for TP models based on ML include TensorFlow, PyTorch, and Keras.

3.2.4 Model training and validation

This task is unique for data-driven TP methods such as ML. It mainly involves training and verifying the trajectory calculation model composed of the TP and the trajectory integration fusion models. The objective involves identifying parameters and hyperparameters for the ML-based trajectory computation model, assessing its generalizability and predictive accuracy, and validating compliance with specified design criteria. For details on the setting and optimization of model parameters and hyperparameters, see Section 3.4.2 of this chapter.

An essential task in the model training and verification stage is sample data preparation, closely related to model design. Sample data preparation should be completed according to the sample data content, attributes, format, scale, and quality requirements designed in the system design link and separated into training sets, verification sets, and test sets based on design requirements. The sample data sources include ADS-B data, flight plans, weather, ATC instructions, airport facilities, and other flight-related historical data, which can directly or indirectly reflect information such as flight intentions, flight environment, flight restrictions, and airline preferences.

3.2.5 System testing and evaluation

Based on the results of the development of each system unit, the complete TP system is constructed. The system is then tested and evaluated either in a test environment or in a real operational environment to verify the correctness and stability of its operation. This process involves assessing the system's overall capabilities to determine whether it meets the designed functionality and

performance metrics, as well as its ability to satisfy the business's operational requirements. If the business operation requirements are unmet, corresponding improvements and supplements must be made according to the deficiencies.

3.2.6 Deploy application

The TP system that has passed the test evaluation is installed and deployed in the actual business operation environment and runs independently or integrated with relevant information systems to undertake TP tasks.

3.3 Evaluation of TP system

TP is an essential function of most (if not all) ATM DSTs. Each DST relies on its customized trajectory predictor, which has unique requirements for its software and hardware operating environment, input and output interfaces, TP accuracy, calculation time, prediction time span, and usage. Therefore, the evaluation of TP systems should be comprehensive, objective, and focused to characterize system characteristics accurately.

3.3.1 TP system evaluation framework

3.3.1.1 Framework analysis

The aircraft TP system is a professional information system that should follow the evaluation standards of general information systems while highlighting the characteristics of professional information systems to objectively, accurately, and comprehensively evaluate particular systems' capabilities and quality characteristics and facilitate user understanding and application. The evaluation indicators of general information systems should cover multiple aspects such as function, performance, efficiency, safety, maintainability, cost-effectiveness, and user satisfaction. Software-based information systems should also include portability indicators to evaluate the software's platform independence and environmental adaptability. Among them, function and performance indicators are the most fundamental indicators of information systems and are also the focus of system evaluation and assessment.

Some research results have been obtained on the evaluation framework of aircraft TP methods. Olive et al. [165] proposed two expectations for TP based on analyzing various typical application scenarios of TP, namely operational realism

and physical realism. Operational realism requires that the generated trajectory should be familiar to experts, pilots, and air traffic controllers; physical realism means that the generated trajectory should be consistent with the performance of the aircraft, such as limited climb rate. The authors believe that in most cases, the generated trajectory is intended to serve realistic simulation, but the focus of realism is not static but depends on the application scenario. Therefore, there is no "best" TP method (model), only "the most suitable". The author of this paper proposes an evaluation framework for TP methods from the aspects of realism, operability, and accuracy of prediction results, including expert evaluation, similarity evaluation, statistical evaluation, and physical evaluation:

- 1). Expert evaluation: From the perspective of professionals such as air traffic controllers or pilots, judge whether the generated trajectory realistically reproduces the characteristics of the actual trajectory in real life and evaluate the realism of the predicted trajectory.
- 2). Similarity evaluation: The realism of generated trajectories and actual trajectories is assessed by measuring their proximity to class-specific sample trajectories, while the accuracy of predicted trajectories is evaluated.
- 3). Statistical evaluation: From the perspective of trajectory distribution, judge the similarity between the distribution of the generated trajectory dataset and the real trajectory set, and evaluate whether the TP model accurately reflects the essential characteristics of the real trajectory set.
- 4). Physical evaluation: From the perspective of physical laws, evaluate the rationality of the predicted trajectory while considering the aircraft's performance, such as climbing ability.

Mondoloni et al. [166] considered the evaluation framework from the TP role and impact perspective. They believed that only TP performance indicators that affect the higher-level system goals supported by TP are meaningful. TP is an important supporting component of DSTs, and its performance directly affects the performance of related DSTs. When DST is combined with other systems, the interoperability of TP, which is its essential support, also needs to be considered. On this basis, a set of TP performance evaluation indicators, including accuracy,

confidence, speed, stability, reliability, uniformity, and synchronization, is proposed, among which uniformity and synchronization are only required for the interoperability evaluation between TPs.

1). Accuracy: The ability of a system to accurately predict future flight trajectories. Accuracy can be evaluated across multiple dimensions, such as vertical, lateral, longitudinal, and temporal aspects.

2). Confidence: A high-confidence TP signifies a high probability of attaining the defined accuracy level. Distinct confidence values may apply to individual accuracy metrics.

3). Speed: The speed of a TP is defined as the duration required to generate a prediction following a request. These measurements may vary depending on the request type.

4). Stability: When the input conditions do not change, the results of multiple predictions remain consistent, and it is called a stable TP.

5). Reliability: Reliability is the ability of the system to complete TPs under specified conditions and within a specified time.

6). Uniformity: This metric is only applicable when considering interoperability. If the predictions of two trajectory predictors are consistent, they are defined as uniform. This property is similar to accuracy, and the comparison is made between predictors rather than between predictors and "true" data.

7). Synchrony: This property is only applicable when studying interoperability. Two predictors are synchronized when the predictions are made at the same time.

In addition, Schuster et al. [17] believe that the LAT is a crucial parameter of interest in TP. Ideally, the TP model should be able to accurately predict the aircraft's trajectory from the departure gate to the arrival gate in advance. This will maximize the benefits provided by TP by identifying conflicts before they occur, thereby achieving strategic optimization of air traffic capacity. At the same time, long-duration TP also supports short- and medium-duration conflict detection and resolution.

3.3.1.2 Basic evaluation indicators

The above authors have studied the evaluation of TP systems from different aspects. The relevant research results are expressed differently, the content is the same, and they complement each other in detail. Some indicators have significant correlations, such as the uniformity indicator and accuracy, which are closely related. The synchronization indicator mainly depends on the control of driving time of different prediction methods and is related to the speed indicator. Based on the above research results, combined with the evaluation content of general information systems and based on the perspective of TP system users, the TP system evaluation framework should include the following aspects:

- 1). System function: Describes the business areas and capabilities of the system service, such as take-off TP, terminal arrival TP, cruise TP, full route TP, TP considering multiple aircraft, etc.
- 2). TP accuracy: Describes the accuracy of the predicted trajectory, usually expressed by multiple error indicators, such as mean absolute error (MAE), MSE, root mean square error (RMSE), mean absolute percentage error (MAPE) and time-of-overfly errors (TOE). For absolute errors, they should be expressed in units of commonly used physical quantities, such as spatial errors in nautical miles or meters, to facilitate user understanding. Accuracy usually has prerequisites, such as the prediction time span (the so-called LAT), so the prerequisites are crucial additional information for accurately describing the accuracy.
- 3). Confidence of prediction results: Describe the credibility of each predicted trajectory point, which is generally expressed in the form of confidence interval, confidence ellipse, PDF, and Bayesian confidence.
- 4). Response speed: Describes the speed of TP service, usually refers to the time required from receiving the TP service request to outputting the TP result.
- 5). Stability: Describes the stability of the prediction result. The ability to maintain consistency in multiple prediction results when the input conditions remain unchanged or change slightly.
- 6). Reliability: Describes the reliability of TP capability, manifested as the system's ability to complete TP within a specified time under specified conditions.

- 7). Input information requirements: Describe the input parameters required for TP, such as the type, number, and input parameter quality requirements.
- 8). Environmental adaptability: Describes the operating environment that the system adapts to and the minimum requirements for the operating environment for regular operation. The adapted environment includes a cloud environment, distributed computing environment, etc. The minimum requirements for the operating environment include software, hardware, and network elements, such as the number and performance of processors, storage space requirements, etc.
- 9). System architecture: Describes the software architecture form adopted by the system, such as C/S, B/S, or stand-alone form.
- 10). User experience: Describes the convenience of using the system, such as system deployment, interface settings, user learning, and operation convenience.
- 11). System maintainability: Describe the system's operating stability and its recovery capabilities when failures occur, such as mean repair time, mean time between failures, etc.

The above are several factors that need to be considered when evaluating a TP system. For specific application backgrounds, the primary evaluation criteria should be to meet the expected application, add or refine relevant factors, and fully reflect the content of particular concerns in the application background.

3.3.1.3 Machine learning evaluation metrics

In TP methods based on ML, the evaluation metrics should account for the unique characteristics of these approaches compared to traditional physics-based models. Unlike physics-based models, which rely on precise operational mechanisms and universal applicability across similar scenarios, ML models extract hidden knowledge from historical data and apply it to predict unknown situations. Given this distinction, ML models are highly dependent on the quality and relevance of the training data, and the model trained on one scenario (e.g., a specific route or airport) may not easily generalize to others. Therefore, the evaluation metrics for ML-based TP methods should reflect these characteristics, and the following indicators are recommended:

- 1). Sample data requirements

This metric evaluates the model's dependency on training data, including a). Data types: Specifies the data types required for model training, such as ADS-B data, meteorological data, or flight plans. b). Data volume: Describes the scale of data required for practical model training. c). Data quality: Assesses the quality standards of the input data (e.g., accuracy, completeness) and their impact on model performance. d). Data source: Identifies the origin of the training data and outlines the processes or methods used to transform raw data into a usable dataset for training.

2). Training complexity

This indicator assesses the difficulty of training the model, including a). Stability of training: Measures how stable the training process is and whether it converges consistently. b). Time requirements: Estimates the time required to train the model under different data and computing resource conditions. c). Computational resources: Evaluate the resource intensity of the training process, such as CPU/GPU usage and memory requirements.

3). Route generalization ability

This metric evaluates the model's capacity to generalize beyond its training conditions: a). Transfer learning capability: This metric evaluates whether the model can utilize transfer learning to adapt knowledge from one route or airport to other routes or airports. b). Reinforcement learning capability: This metric assesses the model's capacity to enhance and adapt its performance in novel scenarios via reinforcement learning. c). Cross-scenario application: Assesses how effectively model parameters learned in one scenario (e.g., a specific airport or route) can be applied to others and how well this transfer reduces the training time and enhances the performance for new scenarios.

4). Loss curve characteristics

The loss curve provides insights into the model's training process: a). Loss function behavior: The shape and behavior of the loss curve during training, which indicates how well the model is learning. b). Optimization guidance: The loss curve can guide hyperparameter tuning for future applications, helping to select or adjust initial values in new scenarios. c). Overfitting/underfitting detection:

Provides an early indication of whether the model is overfitting or underfitting, allowing for adjustments in model complexity or data usage.

5). Hyperparameter Tuning

This metric evaluates the model's capacity for optimizing hyperparameters: a). Automatic hyperparameter tuning: Determines whether the model can automatically adjust and optimize hyperparameters during training. b). Hyperparameter robustness: Assesses whether the hyperparameters are adaptable and provide strong generalization across different scenarios, ensuring the model remains effective without frequent manual tuning.

These specialized evaluation metrics ensure that ML-based TP models are assessed on traditional performance criteria and how well they adapt to real-world complexities such as data dependency, scenario generalization, and model tuning. This structured approach holistically assesses the model's strengths, weaknesses, and operational boundaries across diverse environmental conditions.

3.3.2 TP accuracy evaluation

3.3.2.1 Framework analysis

TP accuracy serves as the primary performance metric, quantified by analyzing deviations between predicted trajectories and real-world aircraft trajectories. Spatial and temporal errors measure it. The spatial error can be divided into distance errors in the horizontal and height directions. Horizontal error can be disaggregated into two components: lateral (cross-track) error and longitudinal (along-track) error [12], [17], [92], [166], shown in Figure 3.4.

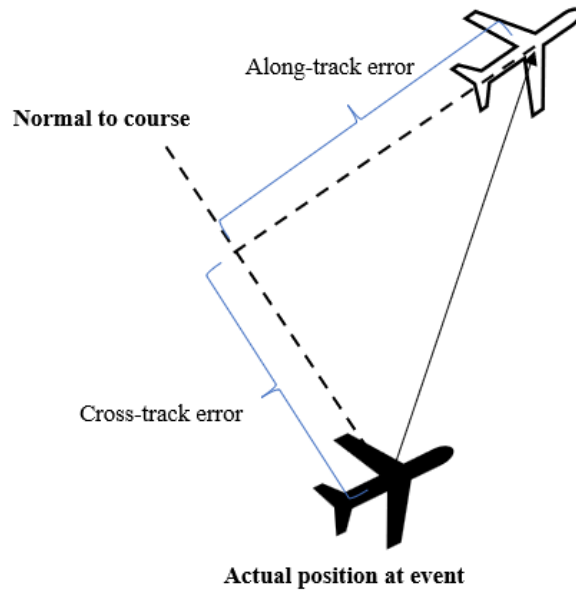


Figure 3.4 Horizontal error of the aircraft.

During the implementation of ATM services, the trajectory accuracy impact in different directions on different stages and different services is not equally essential and unchanging, so it is necessary to subdivide the spatial error:

- 1). Spatial error: This metric quantifies the three-dimensional Euclidean deviation between predicted and observed flight positions, providing a holistic assessment of positional accuracy. Used to indicate how accurate a forecast is.
- 2). Horizontal error: This metric quantifies the horizontal deviation between predicted and actual flight positions, calculated as the Euclidean distance in the horizontal plane. The error is decomposed into across-track error (perpendicular to the planned flight path) and along-track error (parallel to the planned path), enabling targeted analysis of directional deviations.
- 3). Cross-track error: Measures the position difference between a flight's predicted position and the flight's actual position, projected onto a vector perpendicular to the actual flight path for a specific event. Tools that use aircraft position forecasts for separation or load balancing rely on accurate position information.
- 4). Along-track error: Measures the difference between a flight's predicted position and the flight's actual position, projected onto the actual flight path at the time of a specified event. Tools that use aircraft position forecasts for separation

or load balancing rely on accurate position information.

5). Altitude error: This metric quantifies the vertical error (altitude discrepancy) between predicted and observed flight altitudes at a defined event. Aviation tools leveraging aircraft TPs for separation assurance, load balancing, or altitude clearance depend on precise altitude data to ensure safe and efficient airspace management.

6). Temporal error: This metric calculates temporal deviations between forecasted and observed event timestamps (e.g., arrival/departure) along flight trajectories. ATM systems requiring time-synchronized operations (e.g., aircraft sequencing, airspace merging) critically depend on precise temporal predictions at designated waypoints.

The key to TP accuracy assessment is error measurement. Given that 4D trajectories have spatiotemporal characteristics, trajectories are usually defined as discrete point samples representing time and position. Spatial error is generally generated by measuring the distance error between predicted and actual trajectory points that coincide in time (i.e., at the same moment). Temporal error is usually defined by measuring the difference in time attributes (the time interval between the aircraft passing through the point) between predicted and actual trajectory points that coincide in position (i.e., at the same location). It can be seen that the essence of error measurement is the process of determining the corresponding point pairs between the predicted and the actual trajectory. Since the sampling points of the predicted and actual trajectory may not be at the same time and position, interpolation between points is required for comparison, which may introduce interpolation errors and further lead to errors in TP accuracy assessment.

There are many methods for measuring the error between trajectory points. Various methods have different adaptability to different situations (such as straight and turning segments), and different methods will also lead to different calculation results. Therefore, the appropriate error measurement method should be selected for specific application scenarios. Standard trajectory error measurement methods include co-temporal, co-spatial, and nearest point methods, which determine the corresponding points between the predicted and actual trajectory [15], [167].

1). The synchronous error measurement method, as shown in Figure 3.5, usually uses the time interpolation method to simultaneously calculate the position of the predicted and actual trajectory. Vertical/horizontal deviations are quantified and divided into cross-track (perpendicular to the path) and along-track (parallel to the path) components. These components may be derived from observed or forecasted trajectories, computed along the path's curvature, or mapped onto local tangent planes at interpolated points. This synchronized approach ensures accurate 3D positional error analysis. In our study, we primarily used this error calculation method.

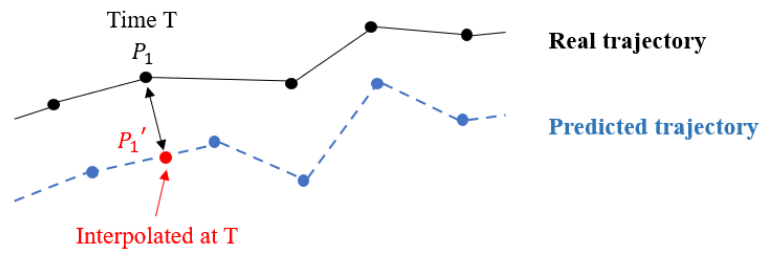


Figure 3.5 The synchronous error measurement.

2). The spatial error measurement method (Figure 3.6) aligns observed and predicted positions at shared geographic reference points (e.g., waypoints). By propagating both trajectories equidistantly from these anchor points, temporal, vertical, cross-track, and along-track deviations are computed. This approach enables integrated spatiotemporal error analysis.

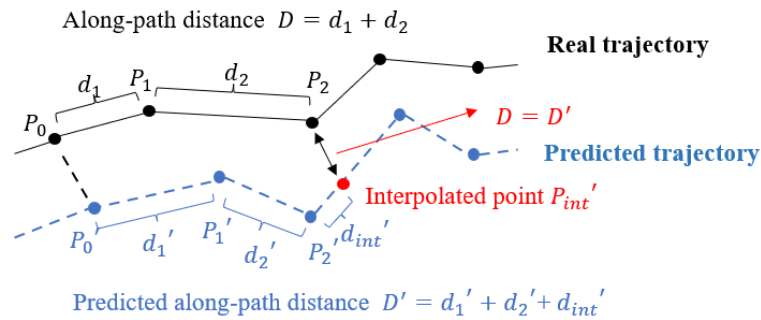


Figure 3.6 The common space error measurement method.

3). The closest point error measurement method, as shown in Figure 3.7, calculates the closest point between the actual trajectory point and the point on the interpolated predicted trajectory and regards it as the corresponding point. The

difference between these points in height, time, and cross-track can be calculated, but the along-track component cannot be strictly defined. This method can be used to estimate time and space errors.

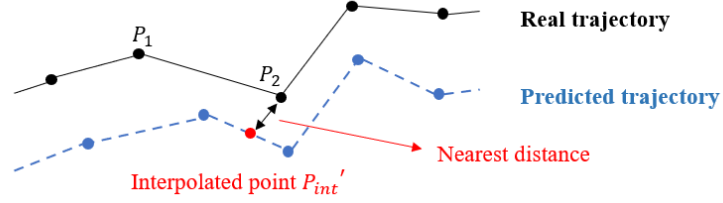


Figure 3.7 The closest point error measurement method.

3.3.2.2 Calculation method

Combined with the specific application scenarios and requirements of TP, the horizontal error and its component errors (cross error and along-course error), altitude error, and time error, which are indicators describing TP accuracy, can be characterized by their statistical properties in the form of maximum value, average value, standard deviation, RMS and N_{th} percentile. The quantification of TP accuracy first requires the definition of the meaning of prediction accuracy and a detailed description of the operating environment and conditions for accuracy measurement, such as the variation of accuracy as a complex function of the forward-looking time span, which is an additional condition that must be stated when describing TP accuracy. All spatial error indicators will rely on the definition of distance. Although there are many definitions of trajectory spatial distance, such as Euclidean, Hausdorff, Frechet, warp distance, and distance based entirely on trajectory shape [104], the Euclidean distance is still the most widely used and basic one for TP accuracy. The commonly used TP accuracy evaluation indicators and their definitions are as follows [73], [87], [108], [168]:

1). Mean deviation error (MDE) / MAE

$$MDE = \frac{1}{n} \sum_{i=1}^n |\hat{y}_i - y_i| \quad (16)$$

where $\hat{y}_i$ is the predicted value, y_i is the actual value, n is the number of data points.

2). Euclidean error (EE)

$$EE = \sqrt{(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 + (z_i - \hat{z}_i)^2} \quad (17)$$

where x_i, y_i, z_i are the actual spatial coordinates, and $\hat{x}_i, \hat{y}_i, \hat{z}_i$ are the predicted coordinates.

3). MSE

$$MSE = \frac{1}{n} \sum_{i=1}^n (\hat{y}_i - y_i)^2 \quad (18)$$

where $\hat{y}_i$ represents the predicted value, y_i stands for the actual value, n is the number of data points.

4). RMSE

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (\hat{y}_i - y_i)^2} \quad (19)$$

5). MAE

$$MAE = \max(|\hat{y}_1 - y_1|, |\hat{y}_2 - y_2|, \dots, |\hat{y}_n - y_n|) \quad (20)$$

6). MAPE

$$MAPE = \frac{1}{n} \sum_{i=1}^n \left| \frac{y_i - \hat{y}_i}{y_i} \right| \times 100\% \quad (21)$$

where $y_i \neq 0$, to avoid division by zero.

3.4 Deep learning TP model

The core of TP is the TP model. Other software modules of the TP system exist to ensure the regular operation of the TP model and to give full play to its service capabilities.

3.4.1 Typical structural model

With the deepening of research, at present, the TP model based on ML is usually

a hybrid model, which is generated by combining two or more neural networks and/or traditional ML basic models to fully develop the advantages of each basic model, achieve complementary advantages, and effectively improve the comprehensive ability of TP. The combination of basic models reflects a certain regularity. Standard aircraft TP hybrid models can be summarized as GAN framework, RNN + CNN, Attention + RNN, clustering algorithm +, and other modes [16]. These modes can flexibly add related components in practical applications, such as dense layers, Dropout regularization measures, etc. According to task requirements, more complex TP models can also be constructed. It can combine ML networks such as RNN, CNN, Attention and clustering algorithms, and even models based on the GAN framework, RNN + CNN, Attention + RNN, clustering algorithm + pattern formation to build new complex models.

3.4.1.1 The framework mode of generative adversarial networks

Since GAN supports unsupervised training, it avoids the time-consuming and laborious work of producing many sample labels and is widely used in many fields. GAN is also a commonly used deep learning model for aircraft TP [16]. TP can be performed based on this model and can also be regarded as a framework for generative adversarial training strategies. Each network can be any neural network. CNN, RNN, Attention, and clustering algorithms, as well as their learning models and algorithms, can be integrated into it to build an aircraft TP model.

Figure 3.8 shows the application model based on the GAN framework in aircraft TP. The generator and discriminator have different functions and goals, and the network structures they use are also different. The sample data contains historical trajectories, planned routes, weather, and other information. Historical trajectories are often used as actual trajectories [21], [81].

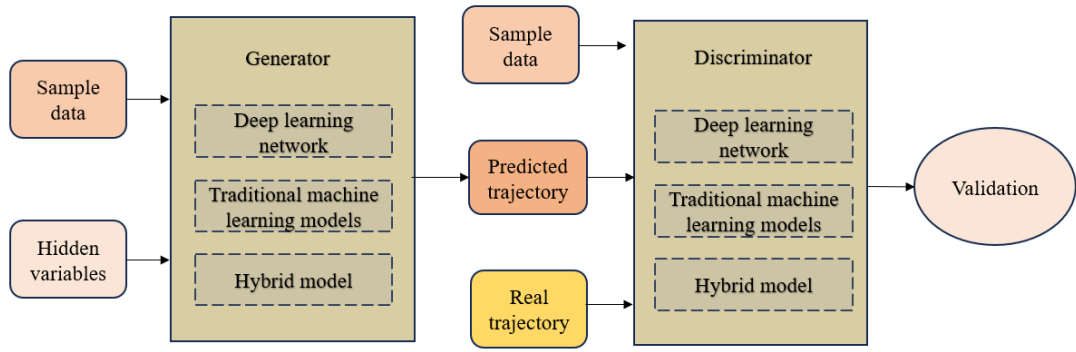


Figure 3.8 The framework mode of generative adversarial networks.

3.4.1.2 The framework mode combining convolutional neural networks and recurrent neural networks

CNNs utilize localized connectivity and parameter sharing to minimize model complexity, enhance computational performance, and simplify training processes. Their translation-invariant nature allows robust extraction of spatially invariant patterns. In contrast, RNNs process sequential data through cyclic connections that retain temporal state information, enabling effective modeling of long-range dependencies by unfolding time steps into feedforward-like layers. The spatial attributes of trajectory data align with CNN's feature extraction capabilities, while its temporal dynamics match RNN's sequential modeling strengths. A hybrid CNN-RNN architecture has become prevalent in aircraft TP, where CNNs first extract spatial and weather-related features, followed by RNNs capturing temporal patterns. The integrated features are then processed through dense layers to generate trajectory forecasts. [14], [87], [88], [90], [92], [93].

Weather data can be divided into two modes depending on whether it is used. Figure 3.9 shows the mode of using weather and historical trajectory data to train the model. CNN first extracts weather feature data from the weather cube, concatenates the weather feature data with the trajectory data based on the trajectory point, and then inputs it into the RNN, and finally generates the predicted trajectory through the output layer.

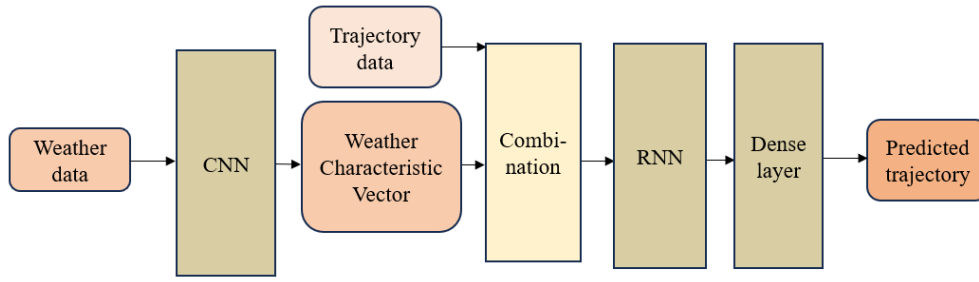


Figure 3.9 The framework mode combining convolutional neural networks and recurrent neural networks (using weather data and trajectory data).

Figure 3.10 is a model without using weather data. CNN is used to dig out the spatial features of historical trajectory data, which is then input into RNN to obtain the trajectory time series features. Then, the predicted trajectory is generated through the output layer.

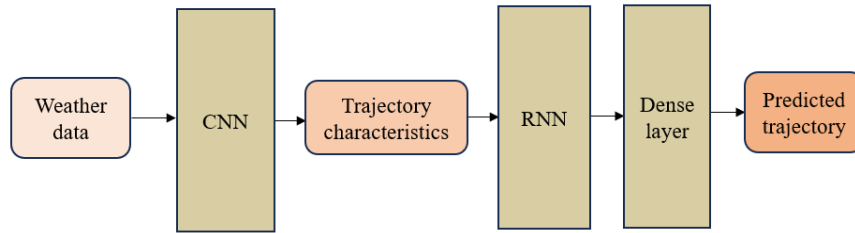


Figure 3.10 The framework mode combining convolutional neural networks and recurrent neural networks (using only trajectory data).

3.4.1.3 The framework mode combining Attention mechanism and recurrent neural networks

The Attention mechanism enables models to dynamically prioritize specific segments of input sequences during information processing, enhancing both predictive accuracy and interpretability. Soft Attention, Hard Attention, Multi-head Attention, SA, etc. are common variants. Although RNN can effectively extract the long-term correlation of sequences, it is still insufficient in capturing long-term global features. In aircraft TP, Attention is often used in combination with RNN to determine the different roles of each trajectory point of the aircraft trajectory, assign different weights, and determine the dynamic impact of different trajectory points on the current trajectory point, making up for the shortcomings of RNN in capturing long-term global features and improving TP accuracy [87], [88], [105]. Its commonly used combined architecture is shown in Figure 3.11

Attention follows the RNN to determine the dynamic impact of different trajectory points on the current trajectory point and generates a predicted trajectory through the output layer.

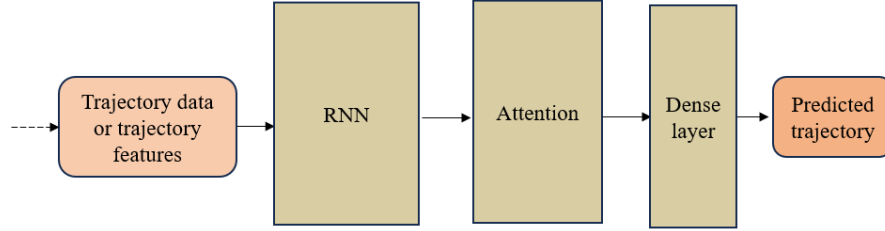


Figure 3.11 The framework mode combining Attention mechanism and recurrent neural networks.

3.4.1.4 The framework mode combining clustering

Clustering techniques, as unsupervised machine learning approaches, organize unlabeled datasets into distinct groups (clusters) by maximizing intra-cluster similarity while minimizing inter-cluster resemblance. In contrast, data points in different clusters have low similarity. K-means, K-medoids, and DBSCAN are specific algorithms commonly used in aircraft TP. Clustering can simplify the characteristics of each group or cluster, facilitate model training, and improve the accuracy of TP, especially for data generation models commonly used in TP (such as GAN), which capture the potential structure and characteristics of data through hidden variables. Clustering grouping helps to reduce the complexity of the multi-peak distribution of hidden variables, effectively alleviates the phenomenon of mode collapse, and improves the generation quality and stability of the model. The application mode of the clustering algorithm in TP is clustering first and then training, as shown in Figure 3.12; that is, first cluster the sample data and then train the model for each category to determine the best model parameters for each category. During inference, TP requires an initial cluster assignment to activate the corresponding cluster-specific predictor [11], [13], [88], [98], [99].

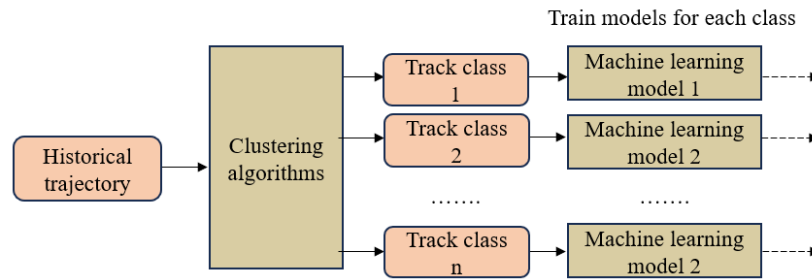


Figure 3.12 The framework mode combining clustering.

3.4.2 Model parameter determination method

The essence of ML is to let the computer automatically find an optimal function from a set of functions to approximate the true mapping relationship between each sample feature vector and the label (or output). The function set is equivalent to the structure of the ML model. The optimal function is a specific function that can reflect the true mapping relationship. It is uniquely determined from the function set through appropriate parameters. "Automatic search" means that it is completed through the learning algorithm. ML models involve two types of parameters: parameters and hyperparameters. Parameters are model components and can be automatically updated through optimization algorithms during model training. Hyperparameters are parameters used to define model structure and optimization strategies. Manual settings or some search methods mainly determine them.

3.4.2.1 Hyperparameter determination

Hyperparameters are parameters that need to be set in advance during the training process of ML algorithms. These parameters are not updated or learned through training data but are set manually by developers or determined through some search method before training begins. The setting of hyperparameters affects the model's training process and final performance.

(1) Hyperparameter content

Common hyperparameters include:

- 1). Learning rate: This critical hyperparameter regulates the magnitude of adjustments made to model weights during training iterations, directly influencing convergence speed and stability in gradient-based optimization processes.
- 2). Batch size: This configuration parameter specifies the quantity of input

examples processed during a single optimization step, directly affecting memory usage and gradient estimation stability.

3). Epochs: This configuration parameter defines how many complete passes the algorithm makes over the training dataset during optimization, directly impacting learning thoroughness and convergence stability.

4). Network structure parameters: These architectural hyperparameters define the network's structural configuration, including its depth (layer count), node density per layer, and activation operations, collectively shaping model complexity and hierarchical feature learning capabilities.

5). Regularization parameters: Such as the L1 or L2 regularization coefficient, are used to prevent overfitting.

6). Optimizer parameters: Momentum, Adam optimizer's beta value, etc.

(2) Hyperparameter tuning

Correctly selecting and adjusting hyperparameters can significantly improve model performance and generalization, preventing overfitting or underfitting. For example, a learning rate that is too high may cause the model to fail to converge, while a learning rate that is too low could result in excessively long training times. Similarly, too many hidden layers may lead to overfitting, while too few layers may prevent the model from capturing complex data patterns. In practice, hyperparameter tuning is a time-consuming and computationally intensive task. High-performance ML models can be trained through iterative experimentation and tuning combined with data-driven parameter optimization. Standard hyperparameter tuning methods include:

1). Grid search: This optimization technique performs a systematic evaluation of the entire hyperparameter space by testing every possible parameter combination within predefined ranges, prioritizing search completeness over computational efficiency to identify the highest-performing configuration. It is suitable when the number of hyperparameters is relatively tiny.

2). Random search: Randomly selects combinations of hyperparameters for evaluation. This approach demonstrates superior computational efficiency

compared to exhaustive parameter exploration methods, particularly within high-dimensional parameter configurations.

3). Bayesian optimization: This method uses Bayesian inference to construct a probabilistic model to select the optimal combination of hyperparameters.

4). Cross-validation: This technique assesses hyperparameter configurations by iteratively partitioning the dataset into complementary subsets under systematic rotation protocols. Integrated with optimization strategies like grid or random search, it identifies optimal configurations through repeated validation on rotated data splits, enhancing the generalization capability evaluation.

5). Loss visualization: A graphical method to display the change in the loss function during training, helping to determine and adjust hyperparameters. By plotting the loss curve, one can intuitively see how the loss on the training and validation sets changes over time. The model enters an overfitting state when the training loss decreases, but the validation loss rises. When both training and validation losses are high and show no significant decrease, the model is underfitting. Loss visualization helps to understand the dynamics of model training, detect possible overfitting or underfitting, and adjust hyperparameters to optimize model performance.

3.4.2.2 Parameter optimization

Parameters are numerical values learned from data during model training. They directly affect the model's predictive power. In neural networks, parameters include weights and biases. The model minimizes the loss function by optimizing these parameters to produce accurate outputs given inputs. The parameter optimization process of the deep learning TP model is shown in Figure 3.13.

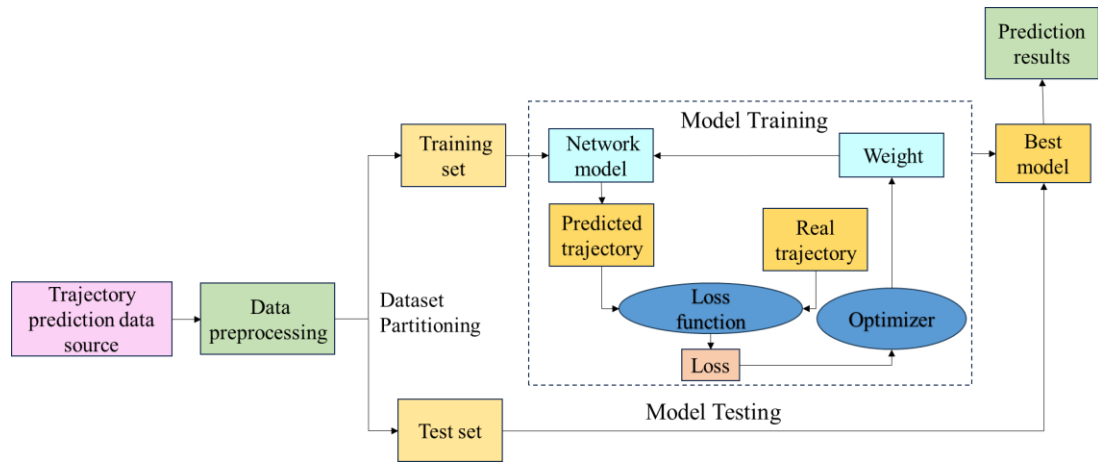


Figure 3.13 TP model parameter optimization process based on deep learning.

Parameter optimization is achieved through optimization algorithms. They are used to minimize or maximize the objective function and promote the model's performance by adjusting the model parameters. Commonly used parameter optimization algorithms include:

1). GD (gradient descent)

This includes batch GD, stochastic gradient descent (SGD), and mini-batch GD.

- a). Batch GD uses the entire training set to calculate the gradient each time.
- b). SGD uses one training sample to calculate the gradient each time.
- c). Mini-batch GD uses a small batch of training samples to calculate the gradient each time.

2). Momentum method

Add momentum term to GD to accelerate convergence.

3). Adaptive learning rate method

Mainly including AdaGrad, RMSprop, Adam, AdaDelta and Nadam, etc. These adaptive learning rate methods have their advantages and disadvantages in different application scenarios. Choosing the appropriate method can significantly promote the efficiency and effect of model training.

3.5 Chapter summary

This chapter studies the technical characteristics of a deep learning-based TP system from the perspective of business operations and applications, specifically

addressing the following aspects:

1). Proposed composition of the deep learning-based TP system

Considering general information system components, the architecture of the TBO-based general TP model, and the integration needs of ML TP models, this chapter proposes a TP system composed of crucial business processing units such as data preprocessing, data updating, prediction result output, trajectory unit calculation (TP model), and trajectory integration and fusion. Additionally, auxiliary system units such as scheduling control, data management, and input data reading are included. A framework for the integrated application of TP models is presented. From the perspective of shared application, the chapter outlines the essential descriptive information needed to characterize ML TP model software units accurately.

2). Proposed construction and application process of the deep learning-based TP system

The process includes essential stages such as requirement analysis, system design, programming implementation, model training and validation, system testing and evaluation, and deployment and application. Model training and validation are unique stages specific to ML-based TP systems.

3). Proposed evaluation metric system for deep learning-based TP systems

Based on an analysis of related research findings, this chapter proposes a basic evaluation metric system composed of elements such as system functionality, TP accuracy, confidence in prediction results, response speed, stability, reliability, input information requirements, environmental adaptability, system architecture, user experience, and system maintainability. Particular emphasis is placed on metrics specific to ML-based TP methods. For TP accuracy, a core element, commonly used evaluation metrics, and their calculation methods are proposed.

4). Proposed typical structure patterns and parameter determination methods for deep learning-based TP models

TP often uses hybrid models of multiple typical ML models to enhance prediction performance. Based on an analysis of hybrid models, this chapter proposes typical

structural patterns such as the GAN framework, RNN+CNN, Attention+RNN, and clustering algorithms. Additionally, methods for determining and optimizing the parameters and hyperparameters of deep learning-based TP models are introduced.

Chapter 4.

Aircraft trajectory prediction based on conditional tabular generative adversarial networks

The basic TP model is the basis for forming a hybrid model, and it largely determines the characteristics and quality of the TP method. As a new achievement of the GAN architecture, CTGAN has many outstanding features that can be used for TP. Based on the analysis of the characteristics and its relationship with aircraft TP, this chapter designs an aircraft TP method based on the improved CTGAN. It conducts comparative experiments with typical deep learning models such as LSTM and GAN. The results show that the improved CTGAN is a basic model suitable for 4D aircraft TP and has good application potential.

4.1 Research background

Aircraft TP is the process of using a trajectory model to predict the future flight status of an aircraft. Its core is to establish a trajectory model that can accurately reflect the comprehensive influence of various factors, such as flight intention, environmental conditions, aircraft performance, flight rules, and procedures [169]. TP involves many deterministic and uncertain factors, making it challenging to construct a trajectory model accurately. Because of the importance of TP, experts and scholars have long been committed to the work to improve the accuracy and reliability of TP. Various TP methods have been proposed such as estimation methods, kinetic methods, and ML methods.

The estimation and kinetic methods are considered traditional approaches that have been widely used in the ATM field. The estimation method primarily establishes motion equations based on parameters such as the aircraft's position, velocity, and acceleration. This allows for TP based on the aircraft's current state.

This method is relatively simple, and large aircraft typically have tiny variations in motion parameters. Therefore, it exhibits good short-term predictive performance and is commonly used in short-term (several minutes or less) TP scenarios. Its main drawback is that prediction errors increase rapidly with longer prediction time spans. The kinetic method builds a TP model based on aerodynamic principles. Modeling requires consideration of various factors, including aircraft performance, aircraft state, environmental conditions, and aircraft intent. This involves a large number of parameters. Some parameters may be commercially sensitive and not easily accessible. Additionally, specific parameters exhibit dynamic characteristics, continuously changing during flight, making accurate estimation challenging. To simplify the model, assumptions are often necessary. The uncertainty associated with a large amount of input parameter data introduces a more significant delay in TP, affecting its accuracy and reliability.

ML methods have some unique advantages not present in traditional modeling approaches. They can extract trajectory features, identify hidden patterns, and implement TP from historical trajectory data without prior knowledge or assumptions. This has become a new research direction [16]. In this thesis, prior knowledge refers to the relevant factors that need to be understood before conducting TP, such as aircraft performance, aircraft state, environmental conditions, and aircraft intent. It also encompasses their specific impact on flight trajectories. This knowledge supports the construction of mathematical models and other aspects of analysis. LSTM neural networks can effectively capture the long-term correlation of a sequence, which is consistent with the temporal characteristics of the trajectory data. They are the most commonly used neural network for short-term TP [67-70]. Although the LSTM model has an excellent time series memorization ability, its prediction ability is still limited by the chaotic phenomenon in the TP process [150]. As the predicted period becomes longer, the output probability density of the predicted trajectory position gradually becomes uniform. This phenomenon caused by flight uncertainty makes it difficult for most LSTM models to support TP over a longer time span, such as medium- and long-term TP.

Recently, a GAN, as the latest neural network technique, has been widely used in various fields and has achieved remarkable results, such as image generation [170], data generation [171], image resolution enhancement [172], trajectory synthesis [173], framework design [174] and other fields. A GAN can learn the data distribution from the competition between a generator and discriminator and find the characteristics of the dataset without a priori knowledge or assumptions.

The outstanding features and performance of GANs have been explored in their application in aircraft TP, aiming to address some of the limitations of LSTM-based algorithms. Jarry et al. [82] used a traditional GAN to model a trajectory based on the take-off and landing data of aircraft near an airport. They proved that the trajectory generated by a GAN is consistent with the actual flight trajectory and can be used for aircraft TP. Pang and Liu [80] used CGANs to predict flight trajectories using weather as a conditional parameter. Their research mainly predicts 2D (longitude, latitude) trajectories. In the most recent research, Wu et al. [21] converted aircraft time series data into RGB images and then used an image-based GAN for 4D TP. The study significantly increased the amount of calculation due to converting aviation data into images. To ensure the calculation efficiency, each prediction only selects a small number of (200 in the study) 4D trajectory points from one route as input, which cannot fully express the subtle trajectory characteristics. The studies above based on GANs demonstrate the feasibility of applying this technology to aircraft TP. However, it's worth noting that these studies are still preliminary, and further exploration is needed.

4.2 Features analysis of CTGANs

CTGAN is a deep learning network proposed for the characteristics of tabular data and is the latest development in the GAN family [128]. Although GAN has the potential to simulate arbitrary distributions, for tabular data, due to several unique properties of tabular data, including non-Gaussian distribution of continuous columns, unbalanced distribution of discrete columns, and mixture of discrete and continuous columns, GAN-based models cannot outperform simple statistical methods. CTGAN improves GAN's tabular data processing performance by introducing several technical methods, including strengthening the training

process with reversible data transformation and architecture changes and solving the data imbalance problem by using conditional generators.

CTGAN directly uses tabular data as input and organizes the sample data into a large table, where each row is randomly sampled as an independent unit. CTGAN can solve many problems of single-table non-sequential data well, including:

- 1). Mixed data types. Tabular data in real-world applications typically comprises mixed data types (e.g., continuous, ordinal, categorical), and is further complicated by the presence of complex, inter-column correlations.
- 2). Non-Gaussian distribution: Pixel values follow a Gaussian-like distribution in illuminated images. Continuous variables in tabular data are usually non-Gaussian and have a long-tailed distribution.
- 3). Multimodal distribution: Ordinary GAN cannot model all the modes on a simple 2D dataset, so it cannot model the multimodal distribution of continuous columns. Suppose the generator finds a realistic example and tries to repeat it every time. In that case, ordinary GAN only makes a real/fake decision on one example, and the discriminator does not have enough information to determine the problem.
- 4). Learning from sparse one-hot vectors: To handle non-sequential categorical columns, they are typically encoded as one-hot vectors. During generation, the model outputs a probability distribution over categories using the softmax function. However, this approach poses a challenge in GANs, as a discriminator can easily exploit the sparsity difference—real data exhibit perfectly sparse one-hot vectors, while generated samples produce smoother probability distributions—to distinguish real from fake, bypassing the need to assess the holistic authenticity of the row.
- 5). Highly imbalanced categorical columns: Most categorical columns have highly imbalanced distributions in real-world datasets.
- 6). High dimensionality: The high dimensionality of tabular data increases the complexity exponentially.

Compared to GANs, CTGAN allows the model to better learn the distribution of

tabular data by introducing mode-specific normalization and training-by-sampling mechanisms.

Mode-specific normalization is a mechanism used in CTGAN to handle the relationship between continuous features and categorical features. Continuous features (numerical features) in tabular data are often associated with categorical features (discrete features), such as age (continuous feature) and occupation (categorical feature). Numerical features of different categories may have different distributions, such as the age distribution of different occupations is different. In CTGAN, to better capture the distribution of continuous features under different category features, the model normalizes the continuous features of each category separately, that is, the continuous features in each category are normalized using the specific mean and variance in the category. This makes it easier for the model to learn the particular distribution pattern of continuous features for each category, thereby avoiding confusion caused by the difference in distribution between categories.

Training-by-sampling is a mechanism used to solve the class imbalance problem during training, especially for scenarios where the class distribution in tabular data is severely imbalanced. In real-world tabular data, different class features may be distributed very unevenly, for example, the number of samples in some categories is much larger than in others. In CTGAN, to solve this class imbalance problem, the model samples rare classes more frequently during training to ensure that the model can learn enough patterns from these minority classes. Training-by-sampling adjusts the sampling frequency according to the number of samples in each class. For minority classes, the probability of sampling will increase accordingly so that the features of these classes can be fully learned during training.

In summary, mode-specific normalization helps the model handle feature distribution issues, including mixed data types, non-Gaussian, and multi-mode distribution. Training-by-sampling solves the problem of class imbalance.

The architecture of CTGAN can be simplified, as shown in Figure 4.1.

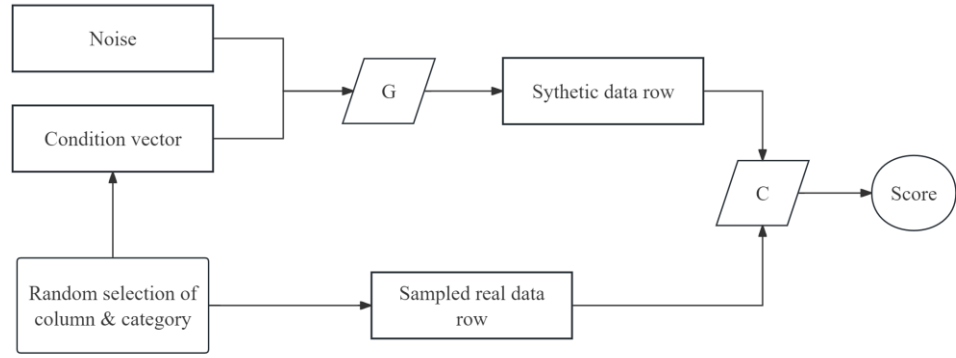


Figure 4.1 Simplified architecture for CTGANs.

Recently, Afonja et al. [175], based on the marginal distribution, column pair correlation, joint distribution, and downstream task utility performance analysis and evaluation from high sample to low sample state, to address the shortcomings of CTGAN in the case of low sample number, the method of adding decorrelated edge feature matching was adopted to improve the downstream utility and synthetic data statistical property performance of CTGAN.

Many studies in multiple fields, such as privacy protection, network intrusion, fault diagnosis, etc., have preliminarily confirmed the excellent performance of CTGAN [177-181]. It has achieved higher stability and model training efficiency while improving classification accuracy [181]. Given that aircraft TP involves sample data suitable for tabular representation, non-Gaussian distribution of data, and an unbalanced distribution of sample data, which is consistent with several advantages of CTGAN, the application of CTGAN in the field of aircraft TP is worth exploring.

4.3 TP method based on CTGANs

Regarding data representation, aircraft TP primarily utilizes two data types: secondary radar surveillance data and Automatic Dependent Surveillance–Broadcast (ADS-B) data. Among them, ADS-B data is expected to become the mainstream and future trend in civil aviation [182]. ADS-B data is typically stored in ASTERIX Category 021 format. It organizes received aircraft broadcast messages chronologically, with each message as a unit. Each message includes

crucial information such as aircraft identification, transmission time, longitude, latitude, altitude, heading, ground speed, and data quality. Additional information, such as indicated airspeed, wind speed, wind direction, and air temperature, may also be included. This data can be easily transformed into tabular form, with each data point corresponding to a row in the table. Regarding data distribution, each flight trajectory has spatial positions different from other routes. Even for flights on the same route, the spatial positions of the trajectory points are not the same. ADS-B data lack symmetry and do not conform to Gaussian distribution characteristics. In terms of sample data composition, samples of aircraft trajectory data under extreme weather conditions are scarce, and they are not proportional to the number of samples under typical meteorological conditions. This leads to an imbalanced distribution of sample data, which can be easily overlooked during ML model training. Furthermore, uncertain factors such as space weather and route congestion influence actual aircraft flight trajectories. To accurately predict trajectories, these uncertain factors need to be considered, and the "C" (conditional) in CTGAN is suitable for expressing these uncertainties.

The correlation between the advantages of CTGAN and the characteristics of aircraft TP is shown in Figure 4.1. The advantages of CTGAN in supporting tabular data, mixed data types, non-Gaussian distribution, multi-modal distribution, and highly unbalanced distribution match the characteristics of aircraft TP, which are important issues that must be faced in aircraft TP. It can be seen that CTGAN may become a suitable machine-learning method for aircraft TP.

Table 4.1 Correlation between CTGAN advantages and aircraft TP characteristics.

CTGAN advantages		Aircraft TP features	Relevance
Mixed data types		TP sample data may include discrete and continuous data types: 1). Discrete data, such as flight code, aircraft type, airline branch, etc.; 2). Continuous data, such as the location coordinates of trajectory points and the time elapsed, etc.	Direct support

Non-Gaussian distribution	The TP sample data is non-Gaussian distribution data. 1). Symmetry: the aircraft trajectory is linear, and its shape is mainly determined by the location of the take-off and target airports and the geographical environment along the way. There is no symmetry in the position of the trajectory points; 2). Concentration: the trajectory points are relatively evenly distributed along the route, and no concentration exists.	Direct support
Multi-mode distribution	There are many types of trajectories in TP sample data. Due to different aircraft models, airline flight strategies, and environmental conditions in different seasons and periods, the same route historical trajectory has different trajectory patterns, and the aircraft trajectory is distributed in multiple modes.	Direct support
Sparse One-hot vectors	Some dimensions of the feature vector in the TP sample data can be expressed in the form of classification and grading, such as using graded cloud, rain, fog, and wind factors to express the meteorological environment, but these categories may appear at the same time, and the sparsity characteristics of the TP feature vector are not obvious.	Not much demand
High imbalance	Some dimensions of the feature vector in the TP sample data can be expressed by classification and grading, such as using graded clouds, rain, fog, and wind factors to express the meteorological environment. Still, these categories may appear simultaneously, and the sparsity characteristics of the TP feature vector are not prominent.	Direct support
High	The dimension of TP sample data is not high.	Not much

dimension	Although the aircraft TP feature vector may demand include information such as airline, aircraft type, flight season and time, airport environment, meteorological environment, trajectory point location, elapsed time, ground speed, and heading, the dimension is usually insignificant.
-----------	---

Figure 4.2 shows the overall framework of the proposed 4D aircraft TP method, which includes three major parts, i.e. the improvement of the CTGAN framework, the design of model evaluation metrics, and 4D aircraft TP. The 4D aircraft TP consists of data preparation, normalization of flight time parameters, model training and TP generation.

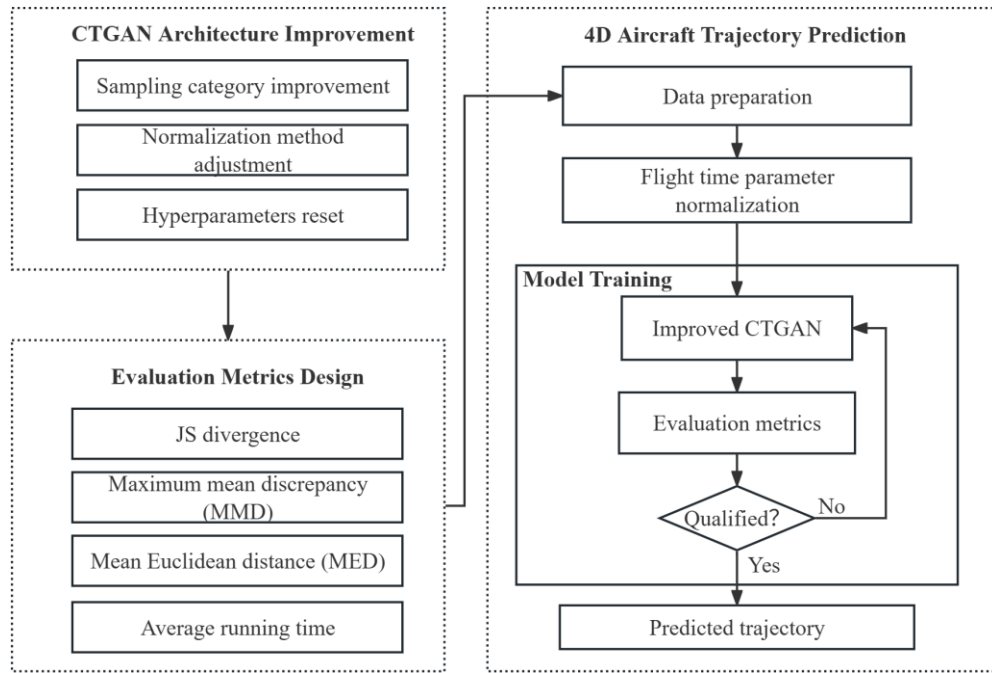


Figure 4.2 The overall framework of the 4D aircraft TP approach based on an improved CTGAN architecture.

4.3.1 Improvements for CTGANs

We made two key improvements to the original CTGAN architecture to accommodate the ADS-B data characteristics. The first improvement focuses on the CTGAN sampling category, replacing the logarithmic frequency with the actual data frequency. Given that the original CTGAN is primarily designed for

data augmentation, it requires a deliberate effort to expand the proportion of less-represented categories when generating data. This is done to maintain diversity in the augmented data types, often achieved through logarithmic frequency sampling. However, using the ADS-B dataset, there is no need to alter the proportion of data categories for TP tasks. Logarithmic frequency sampling in this context will make the generator oversample rare types unrealistically, significantly impacting model convergence [183]. The second improvement involves replacing the original CTGAN's one-hot encoding with leave-one-out encoding. For 4D aircraft TP, the ADS-B parameters we input are all continuous parameters. There is no need to use a condition class label for classification in the CTGAN framework. Compared with leave-one-out encoding, the one-hot encoding adds many dimensions to the dataset but does not provide much information. These dimensions can lead to very sparse results from the generator, making it challenging to optimize, especially for neural networks. What is worse is that each pair of information-sparse columns has a linear relationship between them. This linear relationship means that one variable can be easily predicted using other variables, resulting in problems of parallelism and multicollinearity in high-dimensional cases. Instead, we use randomly generated noise as input to increase data augmentation. Thereby it can improve the operating efficiency of the entire framework. It should be noted that when the input ADS-B data contains categorical data (such as aircraft type, ICAO code, etc.), it is better to choose one-hot encoding instead. Figure 4.3 shows the improved CTGAN architecture, where we determine the hyperparameters through multiple experiments.

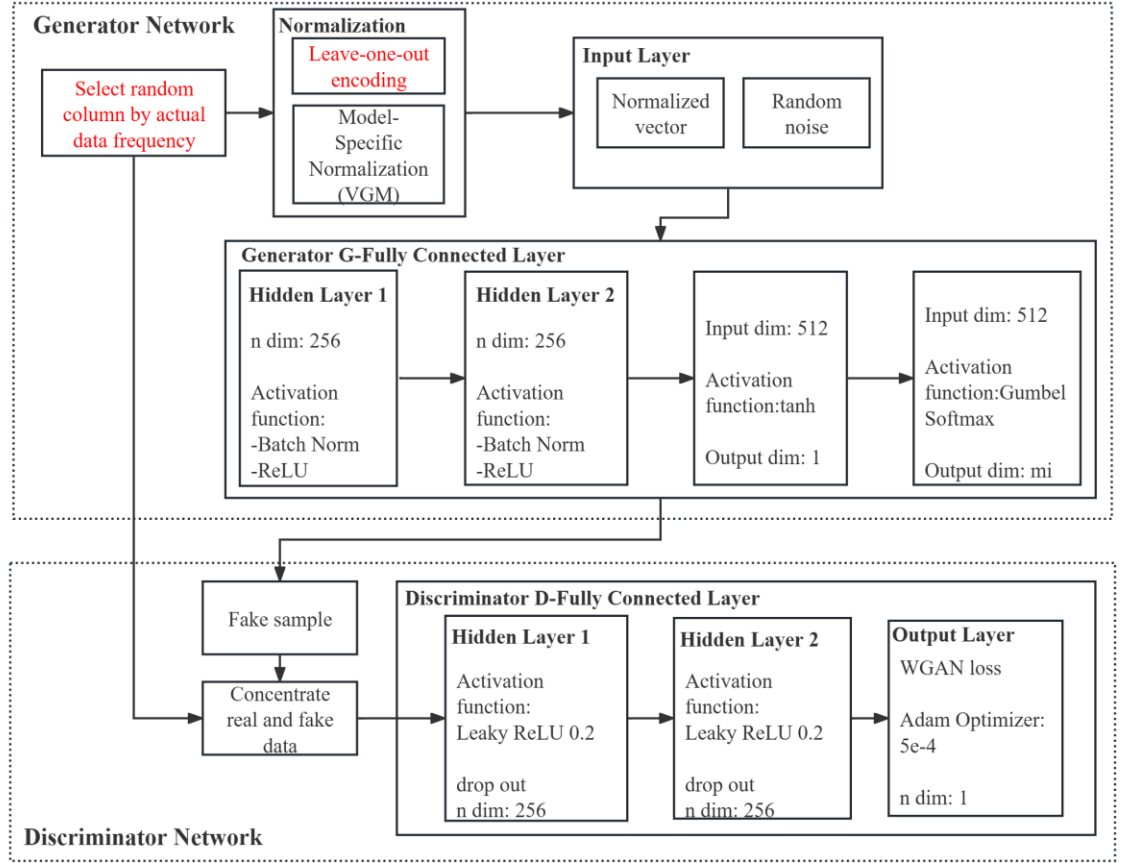


Figure 4.3 Improved CTGAN architecture for ADS-B data.

The new CTGAN architecture includes two networks, a generator network and a discriminator network. In the generator network, we normalize each column of the ADS-B data using a variational Gaussian mixture (VGM) model. This vector will be used as an input with randomly generated noise for data augmentation. The first hidden layer is a fully connected layer with 256 nodes. Here, we carry out a linear transformation process of vector with noise. We also apply batch normalization to the rectified linear unit (ReLU) activation function of this layer and set it to 750 [184]. The second hidden layer is also a fully connected layer with 512 output nodes. We use the same linear transformation process. The last layer converts neurons from 512 dimensions to 1 output neuron. In the Gumbel Softmax activation function [185], we set the vector alpha to 0.2. In this way, the generator network is interpreted as Equation (22):

$$\begin{cases} h_0 = z \oplus C \\ h_1 = h_0 \oplus \text{ReLU}\left(\text{BN}\left(\text{FC}_{|C|+|z|\rightarrow 256}(h_0)\right)\right) \\ h_2 = h_1 \oplus \text{ReLU}\left(\text{BN}\left(\text{FC}_{|C|+|z|+256\rightarrow 256}(h_1)\right)\right) \\ \hat{\alpha}_i = \tanh\left(\text{FC}_{|C|+|z|+512\rightarrow 1}(h_2)\right) \quad 1 \leq i \leq N_c \\ \hat{\beta}_i = \text{gumbel}_{0.2}\left(\text{FC}_{|C|+|z|+512\rightarrow m_i}(h_2)\right) \quad 1 \leq i \leq N_c \end{cases} \quad (22)$$

where $\oplus$ represents the concatenate vectors; C stands for the normalized input vector; m_i is the mask vector; z is random noise; $\text{FC}_{u\rightarrow v}(x)$ represents applying a linear transformation on a u -dimensional input to obtain a v -dimensional output; and $\text{gumbel}_\tau(x)$ stands for applying the Gumbel Softmax activation function with parameter τ to a vector x . BN, tanh, and ReLU functions are used for batch normalization.

In the discriminator network, we use a PacGAN framework [186] with 10 samples in each Pac to avoid model loss. In the first hidden layer, the input dimensions are linearly transformed into fully connected layers to produce 256 dimensions at the output. We apply a leaky ReLU activation function with a parameter of 0.2 and utilize a dropout function [187] with a parameter of 0.3 on each input layer. In the second hidden layer, the same linear transformation is also performed in the fully connected layer to produce 256 dimensions at the output. The leaky ReLU activation function is set to 0.2 together with a dropout function of 0.3 on each input layer. Starting from the second hidden layer with 256 dimensions, a WGAN is used with a penalty on the gradient [188] for linear transformation and we adapte the Adam optimization with a learning rate of $1 \cdot 10^{-6}$ to produce 1 output dimension. Thus, the discriminator network is interpreted as follows:

$$\begin{cases} h_0 = r_1 \oplus \dots \oplus r_{10} \oplus C_1 \oplus \dots \oplus C_{10} \\ h_1 = \text{drop}(\text{leaky}_{0.2}(\text{FC}_{10|r|+10|C|\rightarrow 256}(h_0))) \\ h_2 = \text{drop}(\text{leaky}_{0.2}(\text{FC}_{256\rightarrow 256}(h_1))) \\ C(\cdot) = \text{FC}_{256\rightarrow 1}(h_2) \end{cases} \quad (23)$$

where drop indicates the dropout; $\text{leaky}_\gamma(x)$ represents applying a leaky ReLU activation on x with leaky ratio γ ; $C(\cdot)$ stands for the critic (with 10 samples) in discriminator.

The project description is shown as follows:

The project involves the implementation of an improved CTGAN architecture for flight TP. The CTGAN is a type of GAN explicitly designed for tabular data. This project aims to train a GAN on given tabular ADS-B data to generate synthetic samples that closely resemble the statistical properties of the original ADS-B data to do the flight TP. The key components include:

- 1). Data transformation: The project employs a transformer to normalize the training ADS-B data.
- 2). Model architecture: The CTGAN consists of a generator and a discriminator. The generator aims to create synthetic samples, while the discriminator distinguishes between real and synthetic samples.
- 3). Training process: The training process involves an iterative loop over epochs. The discriminator is trained for multiple steps within each epoch, followed by the generator. The generator generates synthetic samples, and the discriminator is trained to distinguish between real and synthetic samples.
- 4). Optimization: Adam optimizers are used to update the generator and discriminator parameters during training.
- 5). Logging and deprecation warning: The project includes logging training loss information and issuing a deprecation warning about the epochs argument.

The input and outputs are described as:

Input: the ADS-B data. Include time, latitude, longitude, geometric height, ground speed, and track angle. Output: the trained model. It can be used to generate the predicted trajectory.

4.3.2 Evaluation metrics design

For TP modeling, the CTGAN generator input ADS-B data are usually real trajectory time series data, and the output generated data are non-time series random data. Although the generated data distribution range is in the same range as the input data, there is no one-to-one correspondence between the two datasets. Thus they cannot be directly compared. If the distribution of two datasets is similar, the mean difference of their trajectories will also be small, which can approximately reflect the gap between the real distribution and the generated

distribution. Therefore, the mean is used to reflect the similarity between the generated data and the input data.

We consider both data distribution and trajectory distance to evaluate this similarity. There are three commonly used GAN model evaluation metrics: the inception score (IS) [189], Frechet inception distance (FID) [190], and maximum mean discrepancy (MMD) [191]. They are mainly designed for image evaluation needs and evaluated from the data distribution perspective.

The IS is not selected as an evaluation metric with the following considerations. The IS needs to train a neural network for the classifier and then calculates the conditional distribution characteristics of samples according to this neural network. The IS is an indirect method to evaluate image quality. It is sensitive to the constructed network weights and relies too much on the classifier. It cannot directly reflect the differences between real and generated data and often cannot truly reflect the distribution of ADS-B data trajectories.

The FID judges the distribution similarity of two datasets by calculating the Wasserstein-2 distance or Frechet distance between the real data and the generated data characteristic distribution. It comprehensively evaluates the characteristic distance between the real distribution and the generated distribution and can well reflect the TP effect. However, the FID mainly aims at image GAN model problems and needs to be evaluated on a pretrained Inception V3 model, which may only be able to capture some features. Furthermore, the FID requires many samples, with a suggested minimum sample size of 10,000 [192]. These sample-converted images are computationally expensive and slow to run. At the same time, altering the tabular data generated by the CTGAN into image data and then evaluating it is more computationally expensive, so we did not choose the FID.

The MMD is a statistical method used to measure the distance between two probability distributions by comparing the means of their respective feature spaces. MMD is a non-parametric technique that, unlike other evaluation metrics, does not rely on data distribution assumptions. One of the advantages of using MMD as an evaluation metric for GANs is its simplicity and ease of computation. Due to these properties, we choose MMD as one of the evaluation metrics in our study.

As an alternative, we choose the most commonly used metrics to measure the similarity of two distributions in ML, Jensen–Shannon (JS) divergence. This metric can analyze the distribution similarity of different parameters separately to find the parameter with the worst-fitting degree. After testing, we find that the agreement of the real distribution and the generated distribution is good. The JS divergence can explain the similarity between the two distributions. In addition, due to the spatial properties of the flight trajectories of the experimental subjects, it is necessary to use the Euclidean distance to calculate the spatial distance between the two trajectories. Since there is no one-to-one correspondence between the data generated by the CTGAN and the input data, we use the mean Euclidean distance (MED) instead of the commonly used Euclidean distance to calculate the distance between two trajectory data points.

In summary, we used the JS divergence, MMD, MED, and average running time to evaluate the model.

4.3.2.1 JS divergence

Considering the randomness of the data generated by the CTGAN generator, we took the mean value after multiple generators run to calculate the JS divergence.

The JS-divergence is a variant of the Kullback–Leibler (KL) divergence, can be derived from it. The KL divergence can measure the distribution similarity between the pair columns in both the input and generated datasets. This value quantifies the information lost when the CTGAN model generates a new dataset, with the equation shown as follows:

$$D_{KL}(P \parallel Q) = -\sum_{x \in X} P(x) \log \frac{1}{P(x)} + \sum_{x \in X} P(x) \log \frac{1}{Q(x)} = H(P, Q) - H(P) \quad (24)$$

where $H(P, Q)$ refers to the cross-entropy of P and Q ; and $H(P)$ is the entropy of P (which is the same as the cross-entropy of P with itself). From Equation (24), when the similarity between $P(x)$ and $Q(x)$ is higher, the KL divergence is smaller.

$$JSD(P \parallel Q) = \frac{1}{2}D(P \parallel M) + \frac{1}{2}D(Q \parallel M) \quad (25)$$

where $M = \frac{1}{2}(P + Q)$. The JS value is between 0 to 1. The smaller the value, the more substantial the similarity is. When they are the same, it is 0.

4.3.2.2 Maximum mean discrepancy (distribution similarity metric)

Once the continuous function f in the sample space is selected, the MMD can calculate the mean value of the f function with different distributions. By calculating the difference between the mean values of the two f functions from the ADS-B test set and the generated dataset, we can obtain the average discrepancy of different distributions to determine whether the two distributions are similar. We also use the mean value of multiple generated data for calculation using the Equation (26):

$$MMD^2(X, Y) = \left\| \frac{1}{n} \sum_{i=1}^n f(x_i) - \frac{1}{m} \sum_{j=1}^m f(x_j) \right\|_H^2 \quad (26)$$

where H refers to the reproducing kernel Hilbert space (RKHS). If a Gaussian kernel function is used to replace f , the formula can generally be interpreted as:

$$MMD^2(X, Y) = \left\| \frac{1}{n^2} \sum_i^n \sum_{i'}^n k(x_i, x_{i'}) - \frac{2}{nm} \sum_i^n \sum_j^m k(x_i, y_j) + \frac{1}{m^2} \sum_j^m \sum_{j'}^m k(y_j, y_{j'}) \right\|_H \quad (27)$$

$$k(u, v) = e^{-\frac{\|u-v\|^2}{2\sigma^2}} \quad (28)$$

where X represents a random variable in the generated distribution; Y represents a random variable in the real distribution; n represents the maximum value of X ; m represents the maximum value of Y ; k stands for the Gaussian kernel function; and σ is the width parameter of the kernel function. The smaller the MMD values, the more accurate the model is.

4.3.2.3 Mean Euclidean distance (sample-level accuracy metric)

After first standardizing the parameters with a z score, based on the point-by-point calculation of the Euclidean distance from the two datasets, the MED is obtained by taking the average as Equation (29) shows:

$$MED = \frac{1}{n} \sum_{i=1}^n |s_i - \hat{s}_i| \quad (29)$$

where s_i is the average trajectory point of the generated dataset; and $\hat{s}_i$ is the average trajectory point of the real dataset.

4.3.2.4 Average running time

The prediction time is an important metric to measure the performance of TP model, and the TP time can reflect the training efficiency of the model. It affects

the subsequent decision-making efficiency of air traffic controllers. It also reflects the ability to respond to airspace situations and apply high update frequency data. To ensure model convergence, we run multiple epochs for model training. Considering that different input flight data will significantly affect each epoch's running time, the model's average running time for each epoch of a single flight is taken as an evaluation metric.

4.3.3 Aircraft TP process

4.3.3.1 Data preprocessing

Data preprocessing mainly includes two tasks: data cleaning and reorganization and time parameter normalization.

(1) Data cleaning and reorganization

During the broadcasting process, ADS-B data may cause data errors due to system failures and sensor errors, so it needs to be preprocessed before use. Common ADS-B data errors include data duplication, anomalies, data missing, etc. The data preprocessing process used in this thesis is shown in Figure 4.4.

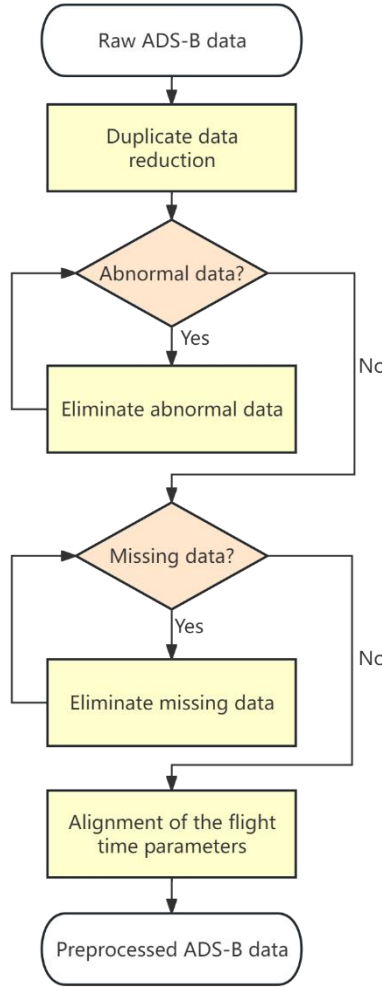


Figure 4.4 The ADS-B data preprocessing flow chart.

(2) Time parameter normalization

Unlike common ADS-B data preprocessing workflows, this study introduces another step of aligning aircraft time parameters. Since CTGAN does not require equal-length trajectory samples for data augmentation, there is no need to interpolate the original data, preserving the original data distribution and improving the accuracy of the CTGAN-generated data. Different flights on the same route typically have varying departure, arrival, and waypoint times. To facilitate the training of the CTGAN model with more flight data, the time parameters of the trajectory data need to be aligned.

Given the weather, equipment, and other reasons, some track points of the received ADS-B data cannot be used due to parameter defects [163]. This

phenomenon will cause an increase of the time intervals of some trajectory points. Therefore, it is first necessary to interpolate the relevant flight data track point near the designated waypoint when determining the relative time origin of an inbound flight. Then, the time of the track points within a certain radius from the waypoint and the closest to the waypoint is selected as the relative time origin. Finally, the standardized processing of the time parameters of the flight data of the same route is realized. Time parameters need to be normalized for both the training dataset and test dataset. The process is shown in Figure 4.5.

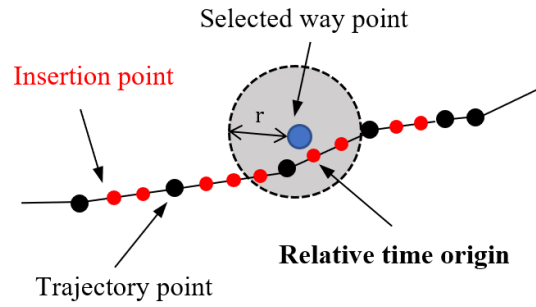


Figure 4.5 The diagram of alignment of arrival flight time parameters.

4.3.3.2 Model training

The processed flight data are input into the improved CTGAN and processed according to the set hyperparameters, a trained TP model is obtained. The input flight data include the time, longitude, latitude, geometric altitude, ground speed, and track angle parameters. After continuous training of the mini-max game using the CTGAN generator and discriminator, we can accomplish a training model. Based on this model, trajectory points that conform to the distribution of input flight data can be generated. Next, the designed model evaluation metrics are used to evaluate the generated trajectory points. If the evaluation results show that the matching degree between the generated trajectory points and the test set is low, the hyperparameters are modified and model training is repeated. Given that the trajectory points in the CTGAN TP results do not have a strict one-to-one correspondence with the test set data's trajectory points, evaluating the prediction results on a point-by-point basis is not feasible. Therefore, a segmented evaluation method can be employed. This involves dividing the flight route into equally spaced segments. Each segment calculates and compares the average of the predicted trajectory points and the test data's trajectory points. When the density

of trajectory points is sufficiently high, this method can objectively reflect the degree of conformity between the prediction results and the test data. Considering that the intervals between trajectory points in the test data set are uneven and can differ significantly (with experimental data having differences up to tenfold), it is necessary to perform trajectory point interpolation on the test data to improve the accuracy of the comparison assessment. The trajectory points of ADS-B raw data are represented in the WGS-84 coordinate system, with geographic positions indicated by latitude and longitude and geometric altitude in feet. Due to the inconsistent coordinate units, performing a coordinate transformation from the WGS-84 coordinate system to the Cartesian coordinate system is necessary. This will standardize the three-dimensional coordinate units, facilitating the generation of a comparative analysis between trajectories and the test set. Figure 4.6 illustrates the segmented comparison method.

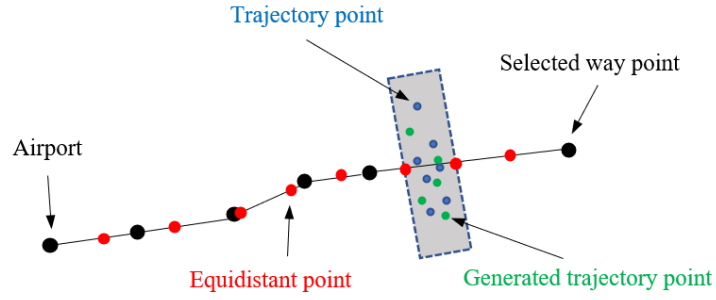


Figure 4.6 Schematic diagram of data alignment between the test dataset and generated data.

4.3.3.3 TP using the trained model

The goal of 4D TP is to predict the spatiotemporal state sequence of the aircraft through the trained model. For every track point S_i , it is represented by time t , longitude x , latitude y , geometric height z , ground speed v_{xy} , and track angle h , as shown below:

$$S_i = \{t, x, y, z, v_{xy}, h\} \quad (30)$$

If the 4D trajectory is regarded as a random variable P , the historical trajectory set is expressed as $T = \{T_1 = s_{1:N}^1, \dots, T_k = s_{1:N}^k, \dots, T_M = s_{1:N}^M\}$, where N denotes the number of points in the track; M denotes the number of historical tracks. Assuming that T follows the distribution of $P = P(s_{1:N}; \theta)$, where θ is a

parameter, then the 4D TP problem can be expressed as solving $P(s_{1:N}; \theta)$. According to the maximum likelihood estimation (MLE), the 4D trajectory distribution is expressed as:

$$\underset{\theta}{\operatorname{argmax}} p(D|\theta) = \underset{\theta}{\operatorname{argmax}} \prod_{i=1}^M P(s_{1:N}^i; \theta) \quad (31)$$

For this article, the essence of model training is to learn the distribution of $P(s_{1:N}^i; \theta)$. The trajectory generated by the trained model is the predicted trajectory for the selected route. Given that CTGAN outputs trajectory point data that is not arranged in chronological order, it is necessary to reorganize it in ascending order based on its time variable to form the predicted trajectory.

4.4 Experiment and analysis

4.4.1 Data preprocessing

4.4.1.1 Data selection

Hong Kong is one of the top three regions in terms of air traffic volume in China, and the nearby airspace has an extensive flight flow and complex airspace structure. We used the ADS-B data around the Hong Kong region for aircraft TP. In particular, the dataset collected by the HKO from December 23, 2016, to December 27, 2016, was selected. After data processing, we obtained 2100 complete flight data (including inbound and outbound flights) from the ADS-B dataset, and each flight data contains at least 5000 trajectory points. The time interval of track points recorded in the ADS-B dataset is approximately 1 s, with a maximum interval of less than 10 s. The parameter integrity is above 95% and the farthest waypoint is 400 km from the airport.

Inbound flights are subject to more air traffic control than outbound flights. Inbound flight trajectories have more circling track points and they more often deviate from flight plan waypoints. Thus they are more complicated than departure flights and worthy of more attention. Considering that flights between the KADLO and VHHH (Hong Kong International Airport) waypoints account for 1/5 of the total data in our dataset, we focused on selecting this route to test the TP model in the experiment. The flight trajectory from KADLO to VHHH is shown in Figure 4.7.

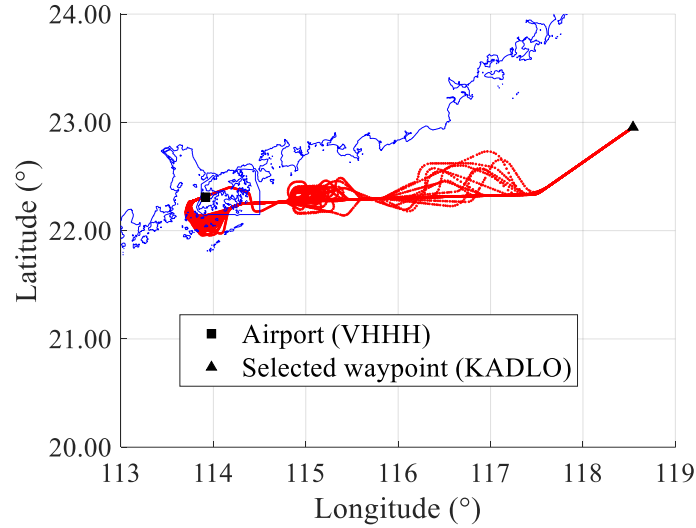


Figure 4.7 Flight trajectory from KADLO to VHHH on December 23, 2016.

4.4.1.2 Data preprocessing

According to the method outlined in Section 4.3.3.1, the ADS-B data is first converted from ASTERIX Category 021 format to tabular form. Following this, coordinate transformation and standardization of time parameters are applied. Finally, the sample data is split into a training set and a test set, with proportions of 0.9 and 0.1, respectively. The path between the KADLO and VHHH waypoints is divided into 200 segments on average to facilitate the comparison between generated data and the test dataset. The average of the predicted trajectory points and the test data's trajectory points are calculated and compared for each segment.

We selected 30 flight trajectories from waypoint KADLO to waypoint VHHH as our experimental data. The training and testing data were divided in a ratio of 0.9 to 0.1. This created two sets of sample data, one tailored to meet the data requirements of CGAN and the other for LSTM.

For comparison with CGAN, considering CGAN's requirement for samples to have equal length, the preprocessed ADS-B data needs to be evenly divided by distance for each trajectory, ensuring that all courses on the same route have the same number of trajectory points. In this experiment, each trajectory consists of 4336 trajectory points.

For comparison with the LSTM model, since the LSTM model describes the length of time using a sequence of trajectory points, the sample data needs to be

processed into data with equally spaced time intervals. This means transforming each trajectory into a series of trajectory points with uniform time intervals. In this experiment, we set the time interval to be one second.

4.4.2 Model hyperparameter confirmation and training

For CTGAN, we built an improved CTGAN using Keras and TensorFlow in PyCharm.

For the determination of hyperparameters, we used the grid search algorithm. Referring to the default settings of CTGAN, the fixed category embedding dimension is 128, the number of hidden layers of the generator and discriminator is set to 256*256, and the number of steps of the discriminator is 1 to save training time. The number of samples designed by Pac is 10 to reduce the risk of mode collapse, and 50,000 epochs are uniformly used as the training cycle to ensure convergence. The learning rates used by the grid search algorithm are 0.0005, 0.0002, 0.001, 0.001, 0.01, and the batch sizes are 500, 750, and 1000. After training, MED is used as the evaluation indicator. The results are shown in Table 4.2.

Table 4.2 MED calculation under different hyperparameters of the CTGAN model.

Batch size\learning rate	0.0005	0.0002	0.0001	0.001	0.01
500	0.03827	0.03818	0.03831	0.03833	0.03841
750	0.03816	0.03829	0.03831	0.03833	0.03840
1000	0.03821	0.03822	0.03834	0.03828	0.03841

After comparison, the training parameters of the final improved CTGAN are shown in Table 4.3.

Table 4.3 Training parameters for the improved CTGAN.

Parameter	Description	Value
Embedding dim	Size of the random sample passed to the Generator.	128

Generator dim	Size of the output samples for each one of the residuals.	(256,256)
Discriminator dim	Size of the output samples for each one of the Discriminator layers.	(256,256)
Generator lr	Learning rate for the generator.	$5 \cdot 10^{-4}$
Generator decay	Generator weight decay for the Adam Optimizer.	$1 \cdot 10^{-6}$
Discriminator lr	Learning rate for the discriminator.	$5 \cdot 10^{-4}$
Discriminator decay	Discriminator weight decay for the Adam Optimizer.	$1 \cdot 10^{-6}$
Batch size	Number of data samples to process in each step.	750
Step ratio	Number of discriminator updates to do for each generator update.	1
Epochs	Number of training epochs.	50,000
Pac	Number of samples to group together when applying the discriminator.	10

For the CGAN model, the running environment is the same as CTGAN model. We set the model training to 50,000 epochs, the ReLU to 0.2, batch size to default. Both the generator and discriminator dimensions were set to (256×256).

For the LSTM model, we used MATLAB to run the prediction algorithm for comparison purposes. The remaining parameters were set as follows: 100 hidden layers, 2000 epochs, a learning rate of 0.005, a learning drop factor of 0.9, and a learning rate drop period of 1600.

4.4.3 Improved CTGAN model testing and analysis

To test the accuracy and stability of the TP model, we conducted three types of experiments. The first type of experiment uses the trajectory data of multiple flights to train and build a prediction model. The probability density function of the generated and real datasets is used to express the distribution's essential characteristics to verify the model's accuracy. The second type of experiment includes two experiments: one is to test the stability of the model by reducing the

number of input flights, and the other is to artificially thin out track data to further test the stability of the model. The third type of experiment compares the CTGAN model's effect before and after improvement.

4.4.3.1 Evaluation metric testing and analysis

Experiment datasets were selected from the 4D trajectory data of the inbound flights from KADLO to VHHH, which includes 80 flights. Table 4.4 shows the evaluation results of the training model of multiple flights according to the evaluation metrics. Overall, all evaluation metric results are within a reasonable range, indicating that the model's generated and real data have a high degree of fitting. The JS divergence evaluation results show that the fitting degree of the trajectory angle parameter is the worst compared with other parameters, because there is a sudden change in this parameter, that is, when the track angle exceeds 360 degrees, it will be recalculated from 0 degrees.

Table 4.4 Evaluation results of different parameters for 80 flights from KADLO to VHHH from December 23, 2016, to December 27, 2016, in the Hong Kong region.

Parameter	JS	MMD	MED	Running time (S)
Timestamp (UTC seconds)	0.0032			
Longitude (°)	0.0479			
Latitude (°)	0.0060			
Geometric height (ft)	0.0181	0.0152	0.0083	4.1702
Ground speed (NM/s)	0.0131			
Track angle (°)	0.2348			

To observe the similarity of the distribution more intuitively, the PDFs (probability density functions) of all parameters in the 4D trajectory of the test dataset and the generated data are plotted based on the kernel function. The difference in the PDFs intuitively reflects the distance between the predicted distribution and the real distribution. Figure 4.8 shows the different distributions of the real and generated 4D trajectory data, containing six characteristic parameters: time, longitude, latitude, geometric height, ground speed, and track

angle. The vertical axis is the probability density of each characteristic parameter, and the horizontal axis is the value of each parameter. Overall, for each parameter, the red line (representing the generated data distribution) agrees well with the real data distribution (i.e. blue line). This implies that the distribution generated by the improved CTGAN model is close to the real 4D trajectory distribution. Figure 4.9 shows the value of the loss function for the generator and discriminator with every 100 epochs for training. It shows that the model has converged.

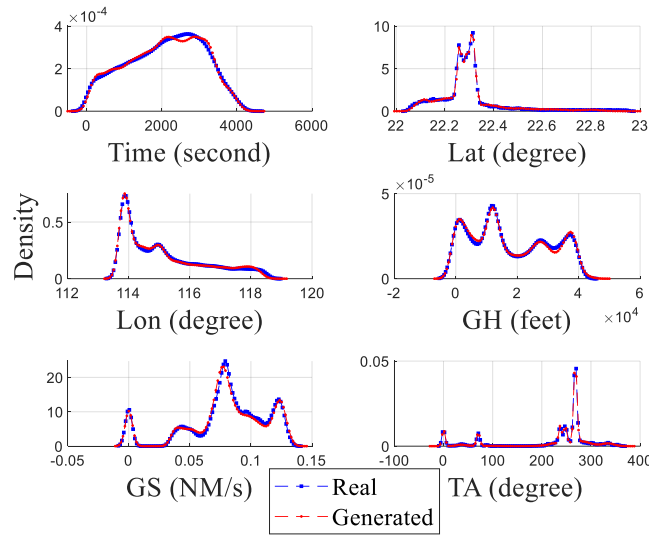


Figure 4.8 Distribution comparison of generated and real data for 80 flights from way KADLO to VHHH during December 23-27, 2016, in the Hong Kong region.

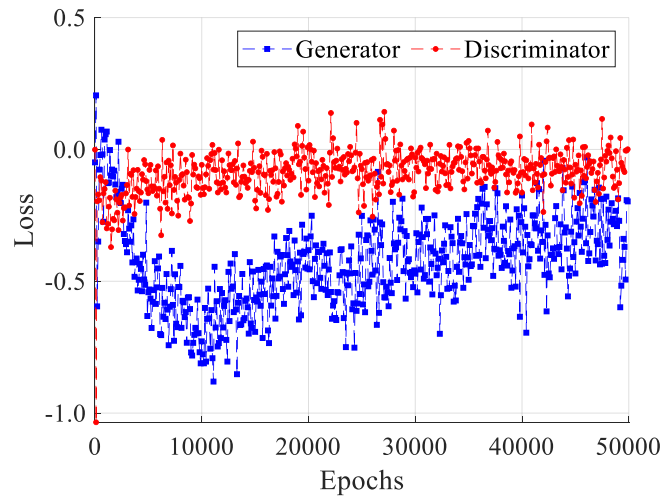


Figure 4.9 Loss function value of the generator and discriminator for training.

4.4.3.2 Stability testing and analysis

(1) Experiment with different numbers of flights

We used different input flight data to analyze the impact of data volume on the improved CTGAN model training, as shown in Figure 4.10. The qualitative trends observed in the figure are quantitatively validated by the evaluation metrics in Table 4.5. Specifically:

1). For Figure 4.10 (a) (single flight), the most significant difference occurs in the track angle parameter, indicating overfitting. This is supported by the high JS divergence (0.1271) and MMD (0.0522) values in Table 4.5, which are the highest among all configurations. The time and longitude parameters also show distributional discrepancies, consistent with these metrics.

2). For Figure 4.10 (b) (5 flights), the distribution of the track angle parameter is better learned, as evidenced by the reduced JS divergence (0.0396) and MMD (0.0316). This indicates that increasing input data reduces overfitting. However, the geometric height distribution learning is worse than in (a), which may be due to random sampling in the generator when data is limited.

3). For Figure 4.10 (c) (10 flights), the distributions of most parameters show good fitting, except for the time parameter. The MMD value decreases to 0.0152, demonstrating improved distribution learning, and the JS divergence is 0.0465. The MED value (0.0100) remains low, indicating consistent geometric accuracy.

4). For Figure 4.10 (d) (80 flights), the time parameter distribution difference is significantly improved compared to (c), with a JS divergence of 0.0539 and an MMD of 0.0150. The MED value (0.0083) further confirms consistency.

Overall, the visualization and metrics suggest that the improved CTGAN model can capture the real data distribution well with even a small number of inputs (5 flights), as shown by the low JS divergence (0.0396) and MMD (0.0316). With more input data, distribution learning quality improves further, and the model shows stability, as the MMD value stabilizes at around 0.015 for 10 and 80 flights. The MED metric remains consistently low (ranging from 0.0048 to 0.0100) across all input sizes, demonstrating robust performance. Finally, the running time increases linearly from 0.5041 seconds (1 flight) to 4.1702 seconds (80 flights)

per epoch, as expected.

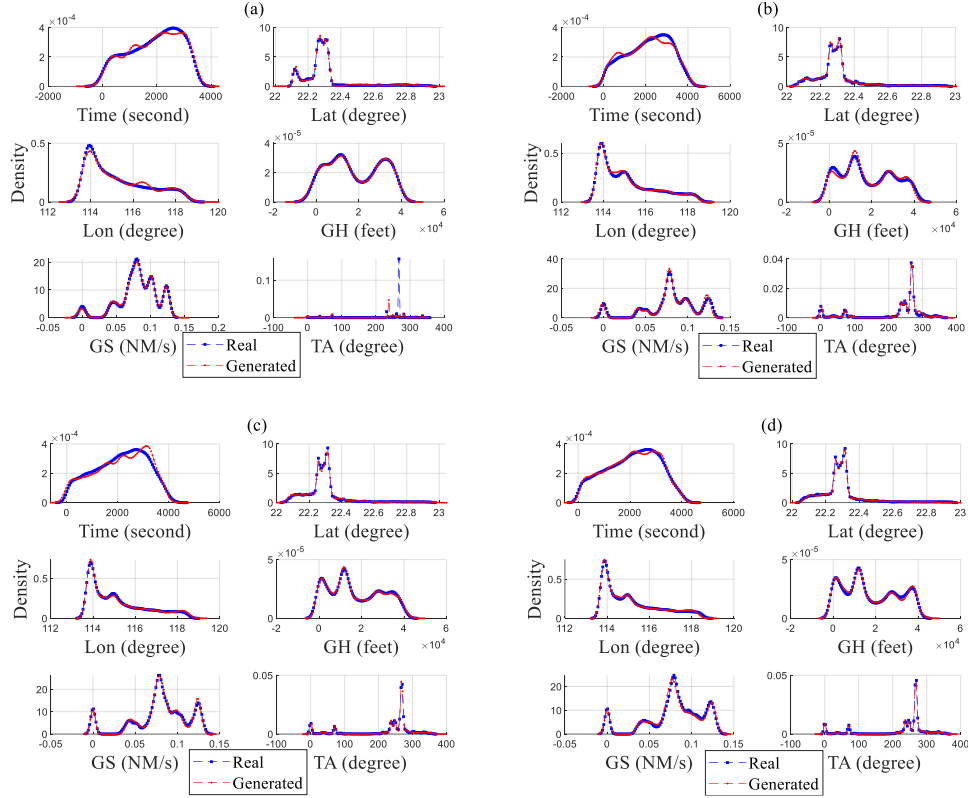


Figure 4.10 Distribution comparison of real and generated data for different numbers of input flights. (a) includes a single flight, (b) includes 5 flights, (c) includes 10 flights, and (d) includes 80 flights.

Table 4.5 Multiflight model evaluation results from KADLO to VHHH during December 23-27, 2016, in the Hong Kong region.

Number of flights	JS	MMD	MED	Running time (s)
1	0.1271	0.0522	0.0066	0.5041
5	0.0396	0.0316	0.0048	0.7661
10	0.0465	0.0152	0.0100	0.9156
80	0.0539	0.0150	0.0083	4.1702

(2) Experiment with thinned data

We also artificially reduced the number of input trajectory points to simulate a situation with insufficient data integrity to further verify the model's stability. During December 23-27, 2016, the 4D trajectory data of 80 inbound flights from KADLO to VHHH were thinned at a given time interval, and the trajectory data

with different data percentages were used as input. Table 4.6 shows the evaluation results for different thinning percentages. The JS divergences still represent the mean value of time stamp, latitude, longitude, geometric altitude, ground speed, and track angle divergence.

The results indicate that equal step-size thinning has little effect on the model's ability to learn the overall data distribution, as evidenced by the relatively close and stable values of JS divergence and MMD across all thinning percentages (e.g., MMD ranges narrowly from 0.0140 to 0.0162).

The most sensitive metric to data thinning is the MED. As the thinning percentage decreases (i.e., more data is removed), the MED generally increases, indicating a gradual degradation in the point-wise geometric fitting of the generated trajectories. Crucially, when the thinning percentage is 80%, the MED value is 0.0097, which remains very close to the MED value of 0.0085 for the full dataset (100% data). This small difference supports the conclusion that an 80% thinning percentage retains sufficient information for the model to generate trajectories with a high degree of geometric fidelity.

This finding is visually corroborated by Figure 4.11. The generated data distribution using an 80% thinning percentage (sub-figure d) maintains a high degree of visual fitting with the real data distribution, closely resembling the results achieved with the full dataset. Therefore, based on the quantitative evidence from Table 4.6 (specifically the MED, JS, and MMD values) and the qualitative evidence from Figure 4.11, we conclude that the model exhibits good stability and can maintain high performance even with a significant portion (80%) of the data removed.

Table 4.6 Evaluation results of models with different levels of thinning from KADLO to VHHH during December 23-27, 2016 in the Hong Kong region.

Data percentage remained after thinning	JS	MMD	MED	Running time (s)
100%	0.0539	0.0150	0.0085	4.1702
80%	0.0414	0.0155	0.0097	3.8102
60%	0.0516	0.0162	0.0114	2.5944

40%	0.0371	0.0140	0.0148	1.5533
20%	0.0540	0.0158	0.0254	1.2183

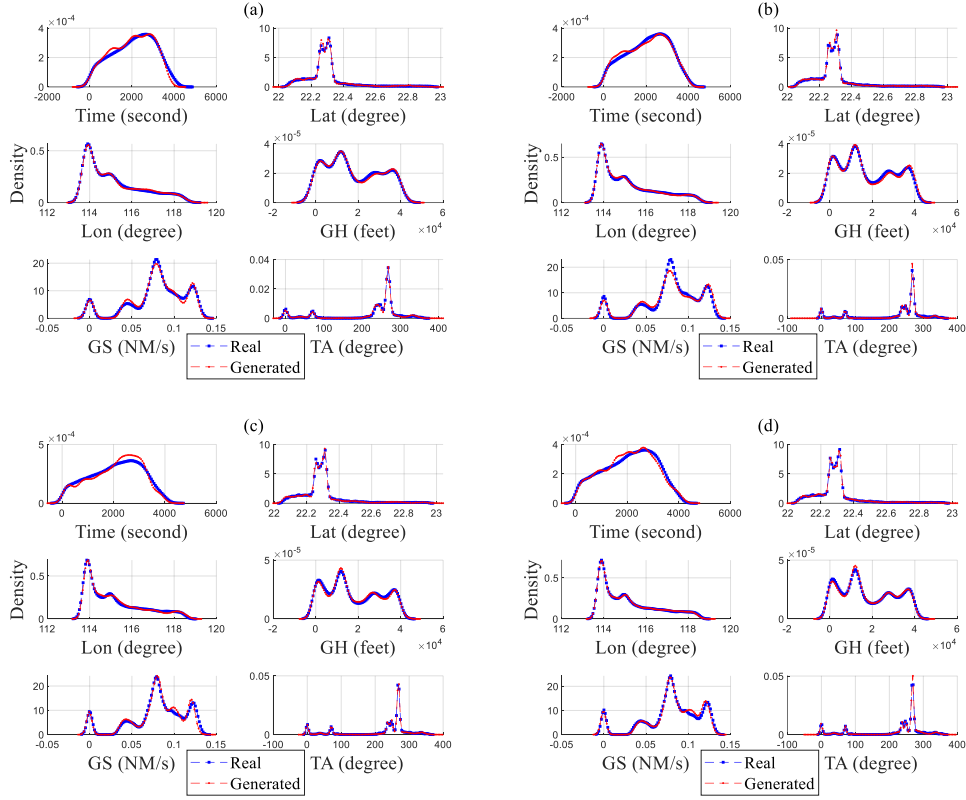


Figure 4.11 Distribution comparison of real and generated data for different levels of thinning from 80 flights. (a) 20% data remaining, (b) 40% data remaining, (c) 60% data remaining, and (d) 80% data remaining.

4.4.3.3 Comparative experiment before and after CTGAN improvement

The hyperparameters of the original CTGAN adopt the original default settings, in which the leaky ReLU function, alpha, and dropout function are all set to 0.2, the Adam optimization learning rate being $2 \cdot 10^{-4}$, the batch size being 500, and the generator dimension and the discriminator dimension being set to (256×256).

The dataset selected in the previous experiment was input into both the original and improved CTGAN. The evaluation results are presented in Table 4.7, which includes the calculated improvement rates for the key metrics.

An analysis of the improvement rates reveals the specific advantages of our proposed enhancements:

1). Accuracy enhancement (MED): The improved CTGAN achieves a dramatic

reduction in trajectory distance error. The improvement is most pronounced with smaller datasets, showing a 79.8% and 67.1% improvement for 1 and 5 input flights, respectively. Even with larger datasets (10 and 80 flights), a consistent and significant accuracy gain of over 17% is maintained. This demonstrates that our model is particularly effective and data-efficient.

2). Computational efficiency (Running Time): The improved architecture and hyperparameter tuning also lead to consistent reductions in running time across all data sizes, with improvements ranging from 9.8% to 26.1%. The most substantial relative speedup is observed for small to medium-sized inputs (5 and 10 flights).

3). Distributional metrics (JS and MMD): As noted by the reviewer, the differences in JS divergence and MMD are less pronounced. This is expected, as these metrics measure the overall distribution match, which the original CTGAN was already designed to optimize. Our improvements primarily target sample-level geometric accuracy (MED) and computational efficiency, areas where the quantitative results now clearly demonstrate superior performance.

In conclusion, by introducing the improvement rate analysis, we can definitively state that our proposed modifications successfully enhance the CTGAN model, delivering the best performance results in terms of a significant boost in prediction accuracy (especially for limited data) and a consistent improvement in computational speed.

Table 4.7 Comparison of CTGAN model evaluation metrics before and after improvement.

Metric	Number of Flights	Original CTGAN	Improved CTGAN	Improvement Rate
JS Divergence	1	0.1507	0.1271	+15.7%
	5	0.0270	0.0396	-46.7%
	10	0.0463	0.0465	-0.4%
	80	0.0620	0.0539	+13.1%
MMD	1	0.0325	0.0522	-60.6%
	5	0.0287	0.0316	-10.1%

	10	0.0155	0.0152	+1.9%
	80	0.0152	0.0150	+1.3%
	1	0.0327	0.0066	+79.8%
	5	0.0146	0.0048	+67.1%
MED	10	0.0121	0.0100	+17.4%
	80	0.0104	0.0085	+18.3%
	1	0.6229	0.5041	+19.1%
Running	5	1.0365	0.7661	+26.1%
Time (s)	10	1.1948	0.9156	+23.4%
	80	4.6240	4.1702	+9.8%

4.4.4 Comparison experiment and analysis with CGANs

Given that CGAN builds upon GAN by incorporating the 'Condition' component, and as this paper primarily explores the potential application of CTGAN in aircraft TP, factors such as weather, holidays, aircraft type, and pilot characteristics—suitable for representation with 'Condition'—have not yet been integrated. Without using 'Condition,' CGAN reverts to GAN. Hence, this paper does not conduct a separate comparative experiment for GAN.

As CTGAN is an extension of CGAN, comparative experiments can be assessed using the model evaluation metrics outlined in Section 4.3.2. We selected flights from KADLO to VHHH for comparison and computed the average JS divergence, MMD, and MED. The results are presented in Table 4.8. Among them, JS and MMD stand for the degree of distribution characteristics of the input data learned by the model. MED represents the error between the data generated by the model and the input data, which can better reflect the prediction accuracy. Compared with CGAN, the JS divergence and MMD of the improved CTGAN remain at similar levels, which means that both the CGAN and improved CTGAN models have learned the distribution characteristics of the input data. The MED has reduced by 81.55%, highlighting a significant improvement in CTGAN's prediction accuracy. This is attributed to the fact that CTGAN treats each row of the table data as an independent sample. In contrast to CGAN, which processes the entire trajectory as a sample unit, this increases the ability to use sample data

fully. Therefore, the improved CTGAN exhibits a more prominent advantage when dealing with small sample data. Figure 4.12 shows the distribution comparison between CGAN and improved CTGAN models. This result is consistent with the MED findings.

Table 4.8 Comparison of CGAN and improved CTGAN models.

Model	JS	MMD	MED
CGAN	0.1363	0.0420	0.0374
Improved CTGAN	0.1595	0.0415	0.0069

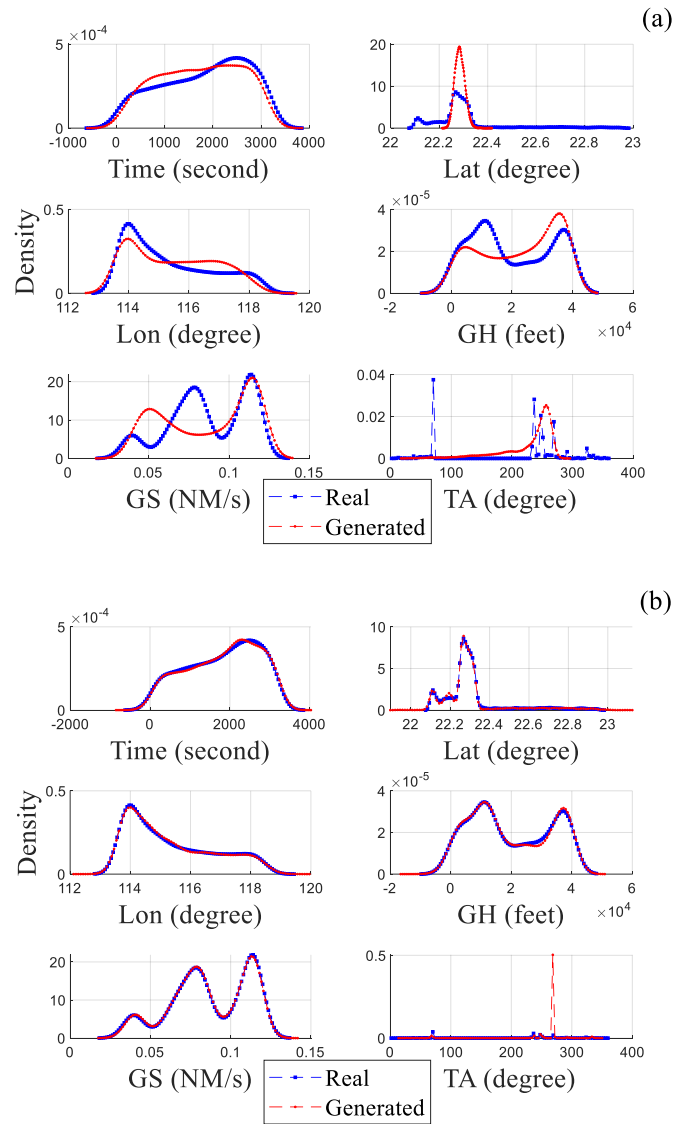


Figure 4.12 Distribution comparison of real and generated data based on CGAN and improved CTGAN models from 30 flights. (a) CGAN, (b) CTGAN.

4.4.5 Comparison experiment and analysis with Transformers

The Transformer model is a prominent representative of the series-to-series model. We selected 30 flights from KADLO to VHHH for comparison and calculated the average JS divergence, MMD, and MED. The results are shown in Table 4.8. Compared with the Transformer, the JS divergence and MMD of the improved CTGAN remain at a similar level, with a slightly higher JS and a marginally lower MMD, which means that both the Transformer and the improved CTGAN models have learned the distribution characteristics of the input data, but the improved CTGAN is better. The MED is reduced by 22.56%, indicating that the prediction accuracy of CTGAN has been greatly improved. Figure 4.13 shows the distribution comparison of the Transformer and improved CTGAN models. This result shows that the Transformer and improved CTGAN data distribution are similar. The improved CTGAN model is more suitable than the Transformer model for aircraft TP using ADS-B data.

Table 4.9 Comparison of Transformer and improved CTGAN models.

Model	JS	MMD	MED
Transformer	0.1001	0.0367	0.0421
Improved CTGAN	0.1026	0.0365	0.0326

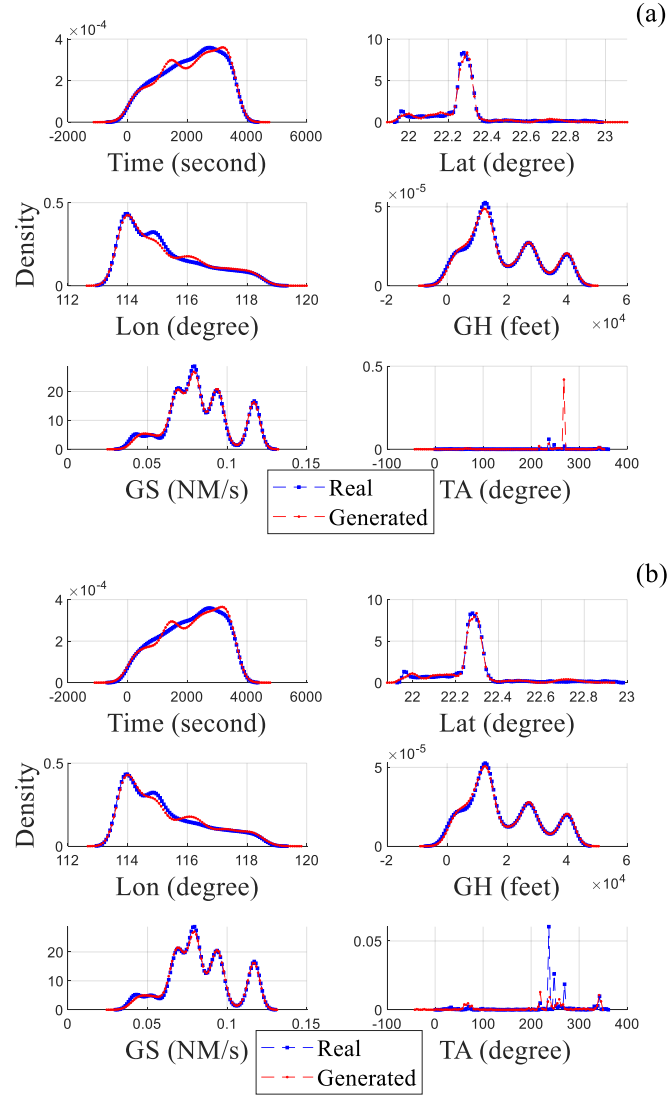


Figure 4.13 Comparison of distribution of real data and generated data from 30 flights based on Transformer and improved CTGAN models. (a) Transformer, (b) CTGAN.

4.4.6 Comparison experiment and analysis with LSTMs

Currently, for TP problems, the most widely used ML method is the model based on LSTM. Various hybrid models based on LSTM have been developed for different scenarios. Similarly, multiple hybrid models based on CTGAN can be designed for different application scenarios. For simplicity, this paper only conducted comparative experiments between the basic LSTM model and improved CTGAN model. The MAE of the LSTM and improved CTGAN models was calculated based on experimental data. For the LSTM model, experiments were conducted with time spans of 1, 3, 5, 7, 10, 15 minutes. The experimental results are presented in Table 4.10, where Lat represents latitude, Lon represents

longitude, GH stands for geometric height, GS for ground speed, and TA for track angle.

Table 4.10 Comparison of MAE between LSTM and improved CTGAN Models.

Model	MAE				
	Lat (°)	Lon (°)	GH (ft)	GS (NM/s)	TA (°)
LSTM-1 min	0.0032	0.0229	220.25	0.0008	2.9328
LSTM-3 min	0.0052	0.0340	393.08	0.0012	4.2577
LSTM-5 min	0.0116	0.0357	498.87	0.0041	5.0389
LSTM-7 min	0.0108	0.0498	630.25	0.0052	6.4220
LSTM-10 min	0.0116	0.0580	814.94	0.0063	8.7585
LSTM-15 min	0.0148	0.0719	1148.80	0.0050	12.3732
Improved CTGAN	0.0107	0.0090	146.89	0.0040	10.2938

For the LSTM model, as the prediction time span increases, the accuracy of predicted trajectory parameters, except for Lat, rapidly decreases. This aligns with the conclusions drawn by other authors, namely that due to inherent limitations in the LSTM algorithm, its predictive capability is constrained by chaotic phenomena [150]. For the Lat parameter, its prediction accuracy decreases relatively slowly. This is because the experimental trajectories are distributed from east to west and are basically parallel to the lines of latitude. The latitude values do not undergo significant changes, making them easier to learn and predict. Therefore, the impact on prediction accuracy within a specific time span is minimal.

Based on the comparison of MAE, the improved CTGAN model exhibits superior predictive capabilities for Lon and GH compared to the LSTM model. When the time span is 5 minutes, the LSTM model's prediction error for Lat and GS is comparable to that of the improved CTGAN model. The disparity between the two models grows as the time span increases. This aligns with CTGAN's ability to utilize PDFs to learn the distribution patterns of sample data, making it less susceptible to the influence of prediction time span.

As a representative example, we selected the LSTM experimental results with a

time span of 10 minutes for comparative analysis. This time span falls within the medium to long-term prediction category. Compared to LSTM, the improved CTGAN model shows reductions in the average MAE for parameters Lat, Lon, GH, and GS by 7.76%, 84.48%, 81.98%, and 36.51%, respectively. The modest improvement in Lat prediction accuracy is attributed to the distribution characteristics of the experimental data, which are roughly parallel to lines of latitude. However, there is an increase of 17.52% in MAE for TA. This is attributed to the sudden transition of track angle values from 360 degrees to 0 degrees, which significantly impacts the improved CTGAN model that relies on probability density for prediction. It is only after a time span of 15 minutes that the TA prediction accuracy of improved CTGAN surpasses that of the LSTM model. For TP, Lat, Lon, GH, and GS are core parameters, while TA is an auxiliary parameter that can be derived from the core parameters of adjacent trajectory points.

In our experiments, we applied equal time interval interpolation to the original ADS-B dataset to enhance the accuracy of LSTM TP. However, for CTGAN, equal time interval interpolation introduced data errors and reduced prediction accuracy. Therefore, if uninterpolated ADS-B datasets are used, the performance advantage of the improved CTGAN model will be more pronounced. CTGAN demonstrates a significant advantage over the LSTM model in medium to long-term TP. Additionally, in short-term predictions at the minute level, improved CTGAN performs on par with LSTM.

For a more intuitive comparison of the TP performance, we took flight AAR723 as an example and presented the TP results based on both the LSTM and improved CTGAN models. It is important to note that the trajectory parameters based on the improved CTGAN model have been smoothed for the ease of visualization. Figure 4.14 displays the prediction results for latitude, longitude, geometric height, ground speed, and track angle. It can be observed that the improved CTGAN model provides a notably higher level of accuracy in predicting latitude, longitude, geometric height, and ground speed parameters, closely matching the actual trajectory. This aligns with the findings presented in Table 4.10. As for the track angle parameter, both improved CTGAN and LSTM models exhibit very similar prediction performance.

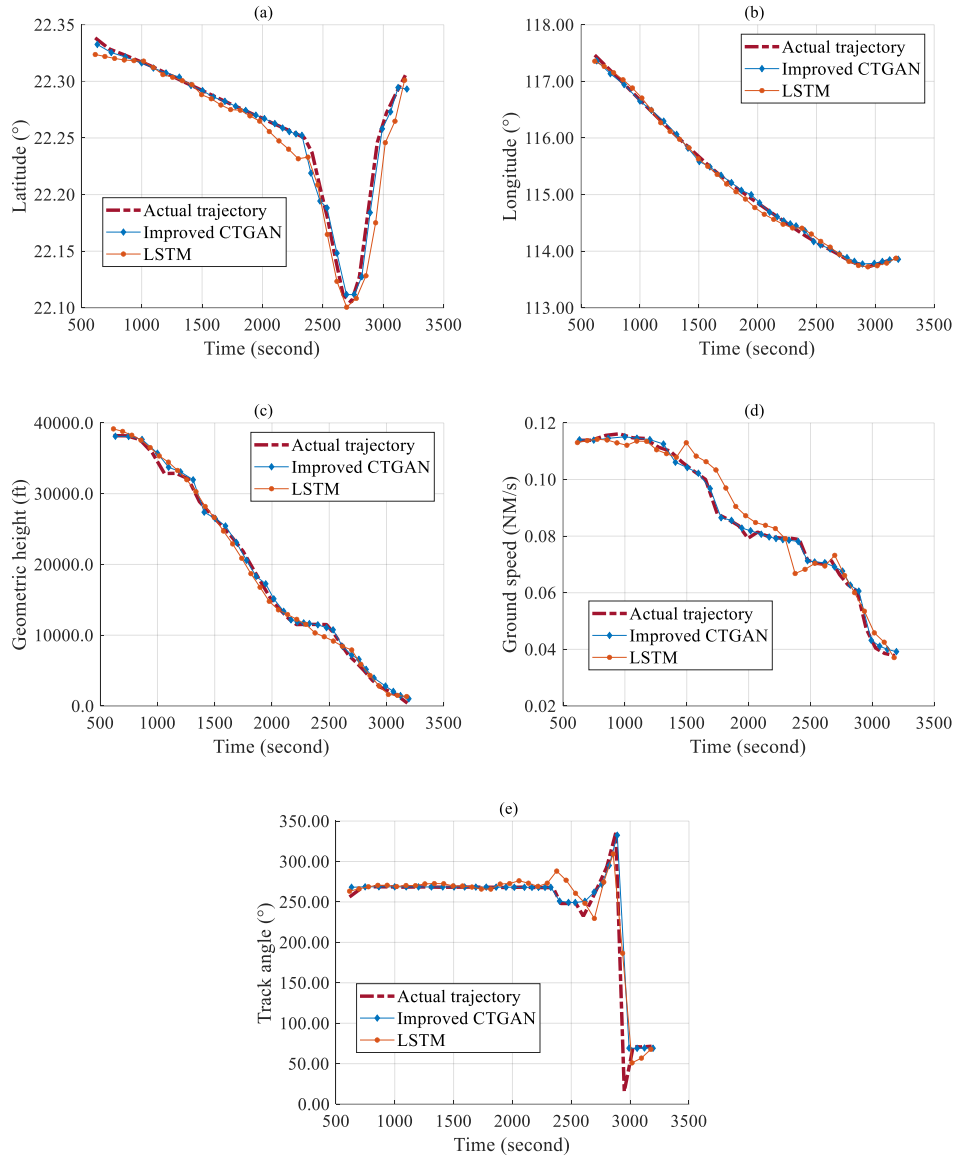


Figure 4.14 Comparison among the prediction results of improved CTGAN and LSTM models for flight AAR723. (a) latitude, (b) longitude, (c) geometric height, (d) ground speed, and (e) track angle.

4.4.7 Experimental conclusion

1). In this section, the average value reflects the gap between the real and generated distributions. We use JS divergence, MMD, MED, and average running time indicators to evaluate the model's performance. Different data amounts and data refinement schemes are used to verify the stability of the model. The JS divergence, MMD, and MED of the model evaluation indicators of the selected route TP are 0.0539, 0.0150, and 0.0085, respectively, proving that the model-predicted trajectory is highly similar to the actual 4D trajectory.

2). The model evaluation indicators are similar to the unreduced case. When reducing the number of input aircraft trajectories, except for the single flight case. The experimental results of trajectory point refinement show that moderate refinement does not affect the model's ability to learn the input data distribution. When the number of training flights is 5, and the trajectory point refinement percentage is 80%, model training can be achieved, indicating that the model has high stability and applicability, especially when the requirement for the number of training trajectories is low.

3). The improved CTGAN was compared with the original CTGAN structure. The results show that the average running time of the improved CTGAN is about 10% shorter than that of TP, and the MED is 20% shorter than that of TP. Compared with CGAN, the improved CTGAN has a higher accuracy, and the MED is reduced by 81.55%. Compared with Transformer, the MED is reduced by 22.56%. Compared with LSTM, when the prediction time span is 10 minutes, the average MAE of Lat, Lon, GH, and GS are reduced by 7.76%, 84.48%, 81.98%, and 36.51%, respectively. As the prediction time span increases, the advantages of the CTGAN model become more apparent.

In summary, CTGAN is a promising new method for four-dimensional aircraft TP with practical application potential.

4.5 Chapter summary

To meet the higher accuracy requirements for TP under TBO and in response to the current state of ML-based TP, this chapter primarily undertakes the following tasks:

1). Analysis of CTGAN characteristics

CTGAN is a deep learning network designed explicitly for tabular data and represents the latest development in the GAN family. CTGAN effectively addresses specific challenges in tabular data, including handling non-Gaussian distributions, imbalanced data, and mixed discrete and continuous columns, with significant advantages in these areas.

2). Introduction of the CTGAN Model into 4D aircraft TP

Based on an analysis of ADS-B data and the characteristics of aircraft TP, it is believed that CTGAN's strengths align well with the nature of ADS-B data and the requirements of aircraft TP. CTGAN has the potential to become a suitable machine-learning method for aircraft TP.

3). Design of a 4D aircraft TP method

Two essential modifications were made to the original CTGAN architecture. The sampling categories were changed from logarithmic frequency to actual data frequency, and one-hot encoding in the generator was replaced by leave-one-out encoding. The TP process included data preprocessing, model training, and TP. Additionally, a performance evaluation system was developed specifically for TP, using metrics such as JS divergence, MMD, MED, and average runtime to assess the overall model performance.

4). Comprehensive comparative experiments

Based on experimental data, tests, and analyses were conducted on JS divergence, MMD, MED, and average runtime. Stability tests and analyses were performed using varying sample sizes and trajectory points with different degrees of sparsity. Comparative experiments were conducted with the original CTGAN, CGAN, Transformer, and LSTM models. The results demonstrate that the improved CTGAN shows good stability and TP accuracy, indicating that CTGAN has promising application potential in 4D aircraft TP.

Chapter 5.

Medium- and long-term aircraft trajectory prediction based on clustering and conditional tabular generative adversarial networks

According to the selected time scale, aircraft TP can be divided into short-term and medium- and long-term prediction tasks [193]. Generally, short-term prediction refers to TP that occurs within a few minutes or less [55] and is mainly used for conflict detection and resolution [89], [194]. The medium- and long-term prediction tasks refer to TPs with a time scale exceeding ten minutes. These predictions are primarily used for coordinating flight missions and formulating flight plans. The accuracy of flight information decreases with the increase of prediction time interval [16]. Compared with short-term predictions, medium- and long-term predictions can effectively improve airspace utilization and aviation operation efficiency and are crucial for improving situational airspace awareness of air traffic controllers and flight flow management [195]. Based on the analysis of trajectory clustering and medium- and long-term TP methods, this section focuses on medium- and long-term aircraft TP. A 4D aircraft TP model based on clustering and conditional tabular generative adversarial network (C-CTGAN) is proposed to address this challenge. Comparative experiments with TP methods based on LSTMs and original CTGAN show that C-CTGAN can effectively improve the accuracy of medium- and long-term aircraft TP.

5.1 Research background

Aircraft TP involves many uncertainties, such as wind, temperature, pressure, and

weight. When the prediction time span increases, especially for medium- and long-term TPs of more than 10 minutes, these uncertainties will strongly affect the quality of the prediction. This is because such uncertainties usually accumulate over time, making the estimated variables increasingly inaccurate and decreasing TP accuracy. The accuracy of medium- and long-term TPs has always been a complex problem in the ATM field.

Medium- and long-term aircraft TP methods can generally be categorized into aircraft planning- and ML-based models [196]. Aircraft planning-based models focus on pre-planned routes and schedules, leveraging deterministic approaches to predict future trajectories based on current and historical flight data. ML-based models leverage complex algorithms and large datasets to capture and predict intricate trajectories that account for various variables and uncertainties. Both models have unique strengths and applications, depending on the prediction task's complexity and data availability.

Aircraft planning is a preformulated trajectory planning task that integrates multiple factors, including relationships among flights, air traffic control, and airports [197]. These models aim to align an aircraft's required time of arrival (RTA) with its flight plan, considering a predefined set of parameters and conditions. While aircraft planning-based models are widely used for medium- and long-term TP [198], they inherently assume ideal conditions and do not account for flight uncertainties or dynamic external influences, such as temporary airspace restrictions and weather variations [199]. Consequently, these models may produce TPs that significantly deviate from actual flight paths under non-ideal circumstances, highlighting their limitations in adapting to real-time operational variability [117]. Therefore, reducing or limiting uncertainty is usually used to improve TP accuracy [109].

For example, Thomas et al. [20] proposed a method to quantify uncertainty factors and infer unknown state variables from observed data. This method considers the uncertainty of wind speed, wind direction, and temperature and uses Bayesian reasoning to infer the true airspeed, Mach number, or cost index variables expressed in probability density, and uses them as input parameters of the dynamic TP model, effectively improving the accuracy of medium-term TP.

Karim [4] optimizes TP by establishing a wind and temperature network. Precisely, the aircraft in flight measures its local atmospheric data, namely temperature, and pressure, calculates the local wind vector and local air density, and broadcasts it to other aircraft and ground stations. The ground station or aircraft quickly fuses these received raw measurement data, promptly updates the wind/temperature field data products, and shares them between aircraft and ground stations through the information network. Each aircraft can use the improved local wind/temperature to correct the position (3D) and time, significantly improving its TP.

Tang et al. [38] proposed a method to improve the accuracy of strategic TP. This method estimates the trajectory points based on the corrected meteorological information based on the extraction of nominal flight profiles and corrected route weather forecast data to improve the accuracy of strategic TP. The nominal flight profile is the flight altitude line closest to the multiple trajectories in the route extracted from the historical trajectory data. The distance between the two flight altitude lines is measured using the author's DSW algorithm. The nominal flight profile eliminates the influence of meteorological factors and can reflect the flight intention. The correction of route weather forecast data is based on GRIB data, combined with the latest meteorological observation data forwarded by another reference aircraft (the aircraft in the previous flight), and the Cressman interpolation method is used to correct the GRIB data to improve the accuracy of the weather forecast. Then, by calculating the influence of weather on flight speed, the correction amount of trajectory point parameters is obtained, and finally, the TP accuracy is improved.

Zhou et al. [117] proposed a framework for multi-model integration based on the time span of TP. This method integrates the best prediction performance models of different time spans to form a TP integration model suitable for short, medium, and longtime spans, thereby improving the adaptability of the prediction time span and the overall performance of TP. The test data is the ADS-B trajectory data of the A320 aircraft in three segments of Guangzhou during the cruising phase. The models involved in the test include the speed trend extrapolation method, LSTM, stateful LSTM, BP neural network, Conv1D, KF method, and flight plan interpolation method. The results show that after 22 minutes, the flight plan

interpolation method achieves the best TP performance.

Compared with flight plan-based models, ML models have unique advantages. Their ability to extract trajectory features, identify hidden patterns, and obtain TPs from historical trajectory data has become a new research area [21], [98], [117].

Zhou et al. [117] conducted a model comparison experiment based on the TP time span and found that within the prediction time span of 25 s-22 min, the stateful LSTM has the best TP performance and can be used as an alternative model for medium-term TP.

Wu et al. [21] designed three TP models based on the WGAN-GP network, utilizing Conv1D, Conv2D, and LSTM networks. They transformed the 4D trajectory sequence data into RGB images and applied image processing techniques to address the TP problem. The experimental dataset consisted of historical 4D trajectory data from flights between Beijing and Chengdu in China from June 1 to August 4, 2019. Each 4D trajectory was aligned over time, consisting of 200 4D trajectory points. These points included six features: latitude, longitude, altitude, heading, horizontal velocity, and vertical velocity. The trajectory position data was converted into 3D Cartesian coordinates, and after normalization, the feature values were multiplied by 255 to convert them into RGB images. These images were then processed using the three GAN models. After training, the generated images were converted back into time series data. The experimental results indicated that Conv1D-GAN was the most suitable generative adversarial network for long-term aircraft TP.

Liu et al. [98] proposed a strategic TP method. This method uses the DBSCAN clustering algorithm to divide historical flight trajectories into multiple clusters, where the trajectories within each cluster are similar to each other. The centerlines of these clusters, called nominal paths, are extracted, and historical trajectory samples are generated by matching the centerlines with environmental features. Four ML models—logistic regression, SVM, RF, and gradient boosting—were selected for comparative experiments. After training the models, the environmental conditions of the flight to be predicted are used as input parameters to determine its category, and the centerline of the corresponding category is selected as its predicted trajectory. Each trajectory's environmental feature vector

includes information such as airline (major airline), local (departure time: morning or afternoon), season (spring, summer, autumn, or winter), convective weather, wind, and MIT (miles-in-trail restriction). The experiments used data from five airports in the United States, along with related convective weather, wind, and MIT data, covering the flight trajectories from 2013, 2014, and 2015. This method transforms TP into a classification problem based on environmental feature vectors. Instead of directly predicting individual flight trajectories, it selects the centerline corresponding to the environmental feature class as the predicted trajectory.

In summary, most TP models based on aircraft flight planning are grounded in physical mechanism models, primarily relying on methods to reduce or constrain uncertainties to improve TP accuracy. One of the fundamental approaches to enhancing prediction accuracy is improving the accuracy and timeliness of weather forecast data. This involves using more precise weather data to adjust planned flight paths, enhancing TP accuracy. In contrast, research and results in ML-based TP models remain limited, especially for medium- and long-term TP. Achieving high precision and high reliability in medium- and long-term TP remains a significant challenge.

5.2 Trajectory clustering algorithm

Common clustering algorithms for aircraft TP include K-means, K-medoids, DBSCAN, etc. Clustering can improve the accuracy of TP, so it is widely used in TP.

5.2.1 K-means

K-means is a well-known clustering algorithm that divides a dataset into k distinct and non-overlapping clusters based on the proximity between data points. The proximity between data points is typically measured using Euclidean distance, although other distance metrics can be used. The input parameters for K-means are the number of clusters (k) and the centroids of the k clusters. The centroids can be randomly selected or determined through calculation methods or prior knowledge to accelerate convergence and reduce the number of iterations.

For each iteration, each data point is assigned to the nearest cluster based on the

current centroids, and then the centroids are recalculated based on the points assigned to the clusters. This process is repeated until either the data point assignments no longer change or the centroids stabilize, at which point the algorithm converges.

The mathematical formulation of K-means can be expressed as:

Given a set of data points $\{x_1, x_2, \dots, x_n\}$, the goal of the K-means algorithm is to divide the data into k clusters $C = \{C_1, C_2, \dots, C_k\}$ to minimize the sum of squares of intra-cluster differences. Its objective function is:

$$\sum_{i=1}^k \sum_{x \in C_i} \|x - \mu_i\|^2 \quad (32)$$

where μ_i is the centroid of cluster C_i .

The algorithm has the advantages of simplicity, high efficiency, and good scalability. In terms of convergence, although it may converge to a local optimum rather than a global optimum, it usually converges quickly. Its shortcomings are mainly reflected in its sensitivity to initialization. First, the final cluster may depend on the initial position of the centroid. Secondly, the number of clusters is fixed, and the pre-specified number of clusters k may not be ideal in some cases. Furthermore, dividing a large cluster into two or more clusters is possible [200], [201]. Many improvement schemes have emerged for practical application needs [201]. The K-means algorithm has been widely used in many fields, such as image segmentation, document clustering, customer clustering, anomaly detection, etc. It is also a commonly used TP algorithm for aircraft TP.

The calculation method of K-means is as follows:

Input: Number of clusters K_y ; dataset D_y containing n objects

Output: Partition of D_y into K_y clusters

Procedure:

- 1). Begin with the dataset and the specified number of clusters K_y .
- 2). If $K_y = 1$, terminate and return the entire dataset as a single cluster.
- 3). Otherwise, randomly select K_y objects from D_y to serve as initial cluster

centroids.

- 4). Assign each object to the cluster whose centroid is closest, typically based on Euclidean distance.
- 5). Recompute the centroid of each cluster by calculating the mean of all objects assigned to that cluster.
- 6). Repeat steps 4 and 5 until convergence is achieved.

Convergence can be determined by a maximum number of iterations or when cluster centroids no longer change significantly between consecutive iterations.

5.2.2 K-medoids

K-medoids is a clustering algorithm related to the k-means algorithm. K-means and k-medoid algorithms divide a dataset into groups and try to minimize the squared error, which is the distance between the points marked as being in a cluster and the point designated as the center of that cluster. Unlike the k-means algorithm, k-medoids uses the actual points in the dataset (called medoids) as the center points instead of using the average. K-medoids cluster the objects in the dataset into k clusters, where k is known a priori, i.e., set in advance. Compared with k-means, it is more robust to noise and outliers. Still, its computational complexity is higher, and the initial selection of medoids significantly impacts the final result [200], [201]. K-medoids are more robust to noise and outliers because their objective function minimizes the sum of general pairwise dissimilarity rather than the sum of squared Euclidean distances. The k-medoids algorithm aims to minimize the sum of the distances of all points to their respective medoids. The formula is as follows:

$$J = \sum_{i=1}^k \sum_{x \in C_i} \|x - m_i\| \quad (33)$$

where: k is the number of clusters, C_i is the i_{th} cluster, m_i is the medoid of the i_{th} cluster (an actual point in the cluster), and x is a data point.

There are many possible choices for the objective function. Medoid is a data point in a finite data set, and its average dissimilarity with all data points is the smallest; it is the most central point in the set. The most common implementation of k-medoid clustering is the partitioning around medoids (PAM) algorithm, which

includes parameter initialization, association of data points with medoids, and selection of new medoids. The specific algorithm is:

Input: K_y (number of clusters), D_y (dataset with n objects).

Output: A partition of the data into K_y clusters.

Procedure:

- 1). Initialize by randomly selecting K_y objects from D_y as the initial medoids.
- 2). Assign each data point to the nearest medoid based on a chosen distance metric.
- 3). For each medoid m and each associated data point o , evaluate the effect of swapping m with o by calculating the total cost of the resulting configuration.
- 4). Retain the swap that yields the lowest total cost, and update the medoid accordingly.
- 5). Repeat steps 2 through 4 until the cluster assignments no longer change, indicating convergence.

5.2.3 DBSCAN

As a classic density-based clustering method, DBSCAN identifies clusters based on the density around core points. It classifies data instances as core points, density-reachable points, or noise. The algorithm initiates cluster formation from a core point and iteratively incorporates density-connected regions until the largest possible area containing both core and reachable points is formed. There are two main parameters in the DBSCAN algorithm: neighborhood radius and the minimum number of points required to form clusters. The value of the neighborhood radius affects the size of the clusters. The size of the minimum point value will affect noise identification and the significance of clustering. The settings of these two parameters directly affect the clustering effect [13]. The main advantages of DBSCAN are that it does not need to specify the number of clusters in advance, can discover clusters of arbitrary shapes, can divide the data set into multiple clusters and noise, and has good robustness to noise and outliers. Its main disadvantages are that the parameters are relatively sensitive, the distance calculation efficiency in high-dimensional space is low, and the clustering effect is not good when the density of the data sets is significantly different. DBSCAN

is very effective in practical applications such as processing geographical and image data and is especially suitable for discovering clusters of arbitrary shapes and identifying noise points.

DBSCAN algorithm steps:

- 1). Create a new cluster. Randomly select an unvisited point. If the number of points in the neighborhood of this point is greater than or equal to the set minimum number of cluster points, it will be used as the core point, and a new cluster will be created.
- 2). Add neighbor points to the cluster. Add all neighbor points of the core point to the cluster, and then recursively add all points in the neighborhood of these neighbor points to the cluster until the cluster no longer grows.
- 3). Mark all visited points, including core and neighborhood points.
- 4). Repeat the above steps until all points have been visited.
- 5). Output all clusters and noise points.

5.3 Typical medium- and long-term aircraft TP methods

There are few studies on the medium- and long-term aircraft TP. The following are several main prediction methods used in related papers: flight plan interpolation, stateful-LSTM, Conv1D-GAN, and nominal path method.

5.3.1 Flight plan interpolation

The flight plan trajectory is linearly interpolated according to the estimated time and waypoints of the original flight plan to obtain the trajectory points corresponding to each second. The TP accuracy of the flight plan interpolation method is independent of the predicted time span and will not change with the change of the time span. The linear interpolation calculation formula is as follows:

$$p_t = \hat{p}_{t_k} + \frac{\hat{p}_{t_{k+1}} - \hat{p}_{t_k}}{t_{k+1} - t_k} (t - t_k) \quad (34)$$

where p_t is the vector of the predicted trajectory position at t s, t_k is the estimated passing time of the k_{th} flight plan waypoint, and $\hat{p}_{t_k}$ is the vector consisting of the longitude, latitude, and altitude of time t_k in the flight plan.

Flight plan-based interpolation facilitates TP across varying temporal horizons without necessitating inputs such as aircraft speed or wind direction. Because the flight plan captures the aircraft's long-term motion behavior under predetermined environmental conditions—ignoring uncertain factors like wind and temperature—its prediction accuracy is independent of the forecast time span. However, this approach produces considerable trajectory deviations, making it impractical for short-term TP. In contrast, for long-term TP, it demonstrates superior performance compared to methods such as speed trend extrapolation, LSTM, stateful LSTM, BP neural network, Conv1D Net, and Kalman filtering [117].

5.3.2 Stateful LSTM

Stateful LSTM retains the hidden state and memory unit from one batch as the initial state for the subsequent batch, thereby maintaining the temporal dependencies inherent in sequential trajectory data. Existing research indicates that this architecture is particularly effective for modeling long-range time series, especially when strong correlations exist between consecutive data segments. The predictive accuracy of stateful LSTM is notably influenced by the number of training epochs, contributing to its superior trajectory prediction performance compared to standard LSTM. Nevertheless, this enhanced capability also makes stateful LSTM more susceptible to overfitting relative to its stateless counterpart [117]. This is because inheriting the last state of the previous batch training is equivalent to adding the samples selected in the last batch to the current batch training, making the weight of the samples in the last batch higher than the weight of the samples selected in the later batch, thus leading to overfitting.

5.3.3 Conv1D-GAN

One-dimensional convolution, with its ability to extract translation-invariant features along a specific direction, offers a natural advantage in processing time-series data such as 4D trajectories. Its core mechanism lies in progressively reconstructing the final predicted trajectory from noise sampling through the iterative alternation of feature extraction and upsampling modules. After converting the 4D trajectory time series data into RGB images, the Conv1D-GAN network is used to generate a synthetic image (the predicted trajectory represented in the form of an image). Then, the predicted trajectory is obtained after the inverse

transformation from RGB image to time series data. The Conv1D-GAN network structure used in the literature [21] is shown in Figure 5.1.

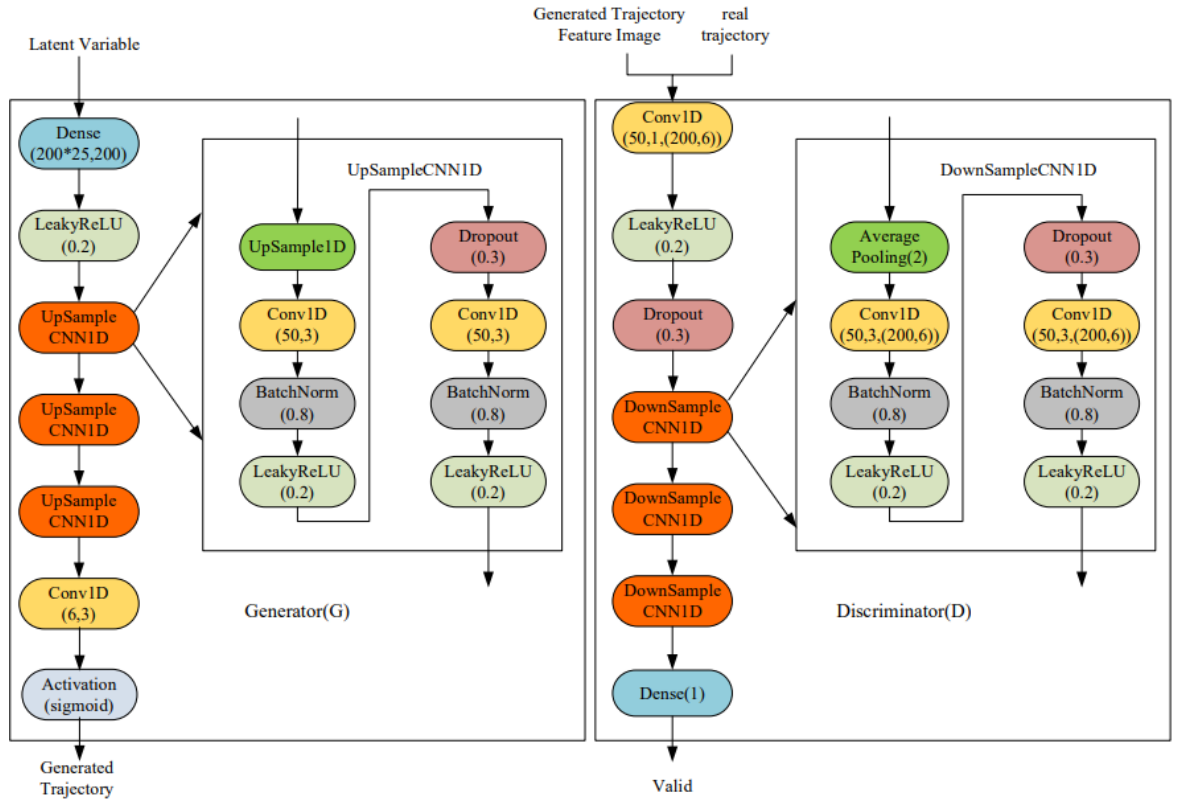


Figure 5.1 Conv1D-GAN structure [21].

The Conv1D-GAN architecture consists of a generator and a discriminator. In the generator, latent variables with a shape of 200×25 are sampled from a standard normal distribution and subsequently transformed into a dimension suitable for convolution via a fully connected layer. These variables are then fed into a one-dimensional upsampling convolutional subnetwork (UpSampleCNN1D), where the final trajectory is progressively generated through three upsampling operations. In the discriminator network, the generated trajectory is first passed through a Conv1D layer, followed by a one-dimensional downsampling convolutional subnetwork (DownSampleCNN1D) that progressively reduces the resolution. The final output score is produced through a fully connected layer.

All networks employ the leaky rectified linear unit (LeakyReLU) as the activation function. This function inherits the computational efficiency of ReLU while mitigating its limitation by allowing gradient flow for negative input values, thus preventing neurons from becoming inactive during training. The dropout

mechanism is added to both the upsampling and downsampling networks to limit the interaction of hidden layer nodes and prevent the model from overfitting. Batch normalization enhances robustness in learning rate selection and weight initialization, significantly streamlining the parameter tuning process. Additionally, it helps mitigate overfitting by stabilizing the distribution of layer inputs during training.

5.3.4 Nominal route method

The nominal route method [98] can be used for long-term TP, and its basic process is shown in Figure 5.2.

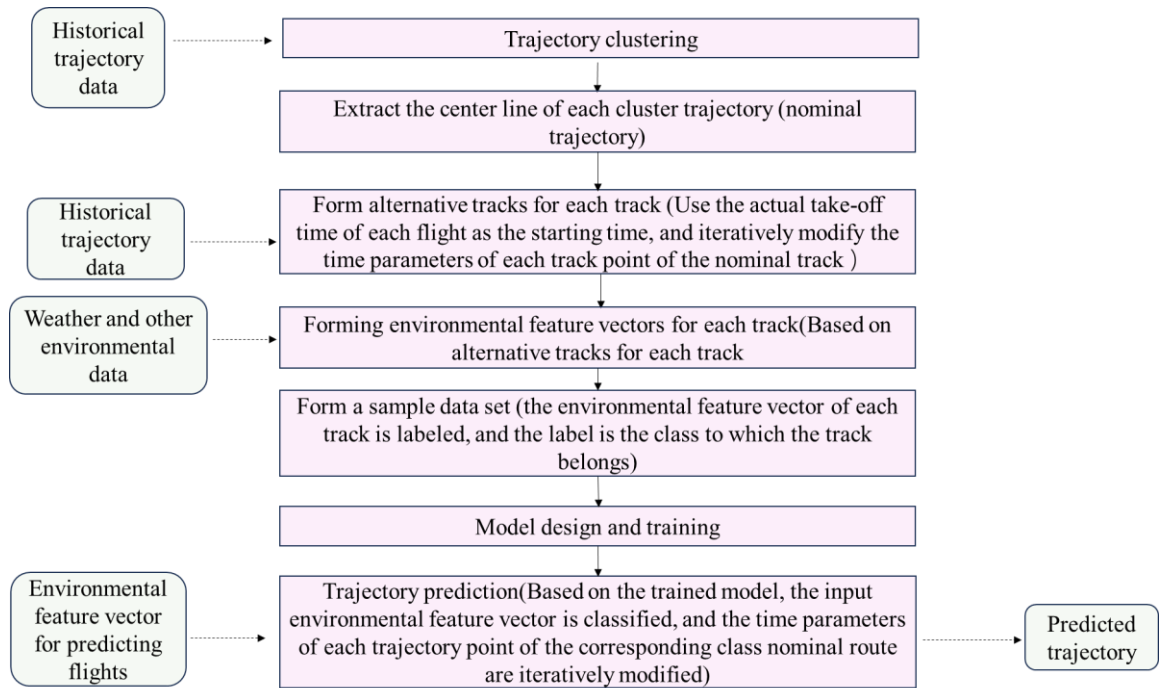


Figure 5.2 Basic process of TP using the nominal route method.

This method describes the fundamental framework of a TP approach, where the trajectory clustering model and TP model can be flexibly determined according to the specific application scenario. Since each aircraft trajectory cluster usually covers a wide distribution range, the nominal route method, which uses the cluster's central path as the predicted trajectory, often struggles to achieve high prediction accuracy.

5.4 TP method based on C-CTGANs

The flight plan interpolation method is often used with trajectory optimization algorithms. These optimization algorithms are primarily based on dynamic and kinematic models, and they adjust planned flight paths by incorporating or improving the evaluation of uncertainty factors, thus falling under the category of physical-mechanism-based models. RNNs (such as LSTM and GRU) have demonstrated excellent time-series memory capabilities regarding deep learning models. They have been widely used in many fields, including aircraft TP, where they are one of the most commonly used neural network models. However, the computational logic of RNNs, which predicts future states based on current and historical states, inherently leads to a decline in prediction accuracy as the time horizon increases. This makes RNNs more suitable for medium-term prediction (time spans of 10 to 20 minutes) but less effective for long-term prediction.

The nominal trajectory method uses the cluster's centerline as the predicted trajectory for that class. However, since each cluster usually has a specific distribution range, this method often struggles to achieve high accuracy for TP. In contrast, GAN models learn the distribution of target data directly from the training set, and their prediction accuracy does not deteriorate with more extended time spans, making them particularly suitable for medium- to long-term TP. The research and comparative experiments in Chapter 4 show that CTGAN is well-suited for aircraft TP applications, especially for medium- to long-term predictions.

That being said, CTGAN, as a GAN-based model, inevitably inherits some of the inherent drawbacks of GANs, such as training instability and mode collapse. The primary cause of these issues is often the multi-modal nature of the data distribution. Clustering can segment data into multiple clusters, reducing the data distribution's complexity and mitigating the multi-modal distribution problem. Thus, a hybrid model combining clustering algorithms with the CTGAN architecture holds promise for improving the performance of medium- to long-term aircraft TP. This combined model could enhance stability and accuracy by leveraging the advantages of clustering to reduce data complexity while retaining CTGAN's strong ability for long-range prediction.

The overall framework of the 4D aircraft TP method based on trajectory clustering

and CTGAN (referred to as C-CTGAN in this thesis) proposed in this section is shown in Figure 5.3. It consists of four parts: data preprocessing, trajectory clustering, model training, and TP.

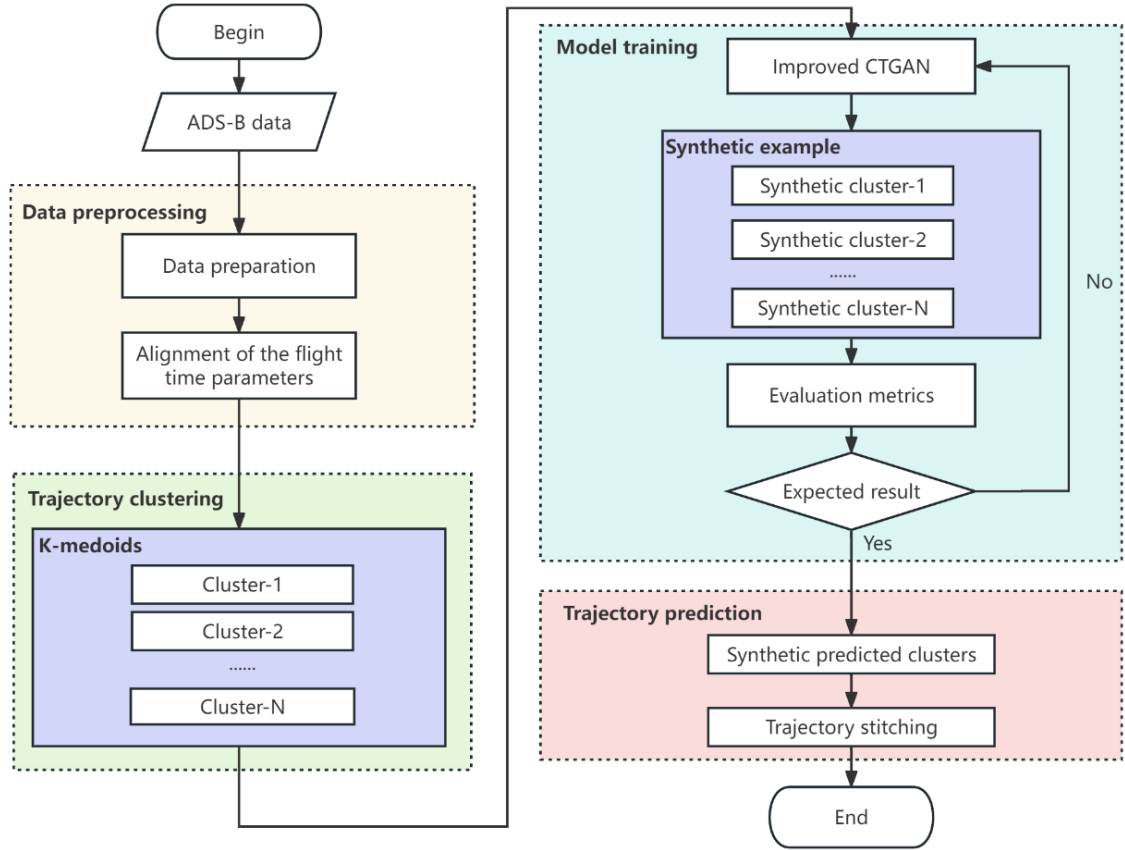


Figure 5.3 The general framework of the 4D aircraft TP method based on clustering and an improved CTGAN architecture.

5.4.1 Data preprocessing

The content and methods of data preprocessing in this chapter are the same as those in Chapter 4, mainly including data cleaning, reorganization, and time parameter normalization. For details, see Section 4.3.3.1.

5.4.2 Trajectory clustering

Clustering, an unsupervised ML method, has been widely used in various fields [202], such as data mining, pattern recognition [203], image analysis, bioinformatics, and recommendation systems. Trajectory clustering can be used to discover hidden trajectory patterns in an airspace model that are not easy for

humans to identify. The use of such hidden patterns helps improve the accuracy of TP [202]. For the CTGAN architecture, existing studies have shown that the CTGAN model integrated with the K-means method can effectively learn the distribution of data and then address imbalance issues in tabular data [178]. This is because clustering can be used to divide and classify data, decompose complex problems into relatively simple ones, and reduce the difficulty of problem processing. Based on this classification approach, the CTGAN can effectively learn the distribution characteristics of the data. For flights on the same route, an aircraft's trajectory parameters are primarily related to the aircraft's planar position (latitude and longitude). Considering this property, clustering is used to extract the plane position features from ADS-B data. The clustered trajectories are input into the CTGAN model as additional parameters to improve the TP accuracy.

5.4.2.1 K-medoids clustering algorithm

For trajectory clustering, we adopt the K-medoids algorithm. Its primary advantages are twofold: it robustly identifies planar features while being insensitive to noise and outliers, a clear improvement over K-means [204]; and it simplifies the clustering process compared to DBSCAN, as it only necessitates tuning the number of clusters to achieve optimal results. Considering that outliers inevitably exist in ADS-B data and that aircraft trajectory clustering is more suitable than DBSCAN clustering, we selected the K-medoids clustering method.

Outliers in trajectory data can significantly affect the performance of clustering and prediction models. To handle outliers, we implemented the following steps: 1). Initial data cleaning: We first performed an initial cleaning to remove any apparent errors and anomalies in the data, such as missing values, erroneous coordinates, and unrealistic speed or altitude changes. 2). Statistical analysis: We used statistical methods to identify potential outliers. Specifically, we calculated the z-scores for various trajectory parameters (e.g., speed, altitude) and removed data points that fell beyond a certain threshold (e.g., $|z| > 3$). This statistical approach helps detect data points that deviate significantly from the mean. 3). Visual inspection: For critical points regarding outliers, we visually inspected the trajectories to ensure that no significant flight patterns were wrongly identified as outliers. This step helps verify that the outliers identified by statistical methods are anomalies.

5.4.2.2 Selection of the number of clusters

The key to K-medoids clustering is to choose an appropriate number of clusters K . In this section, the elbow method and silhouette coefficient (SC) [205] are used to select the expected number of clusters. The elbow method generally uses the sum of squared errors (SSE) to calculate the overall loss of each cluster, and the formula is as follows:

$$SSE = \sum_{i=1}^k \sum_{p \in C_i} |p - m_i|^2 \quad (35)$$

where C_i refers to the i_{th} cluster, p represents a sample point in C_i , and m_i is the centroid of C_i . When the appropriate number of clusters is reached for the given data, the changes in the SSE tend to be stable. K is the expected number of groups, which are generally located at the elbow of the curve, as shown in Figure 5.4.

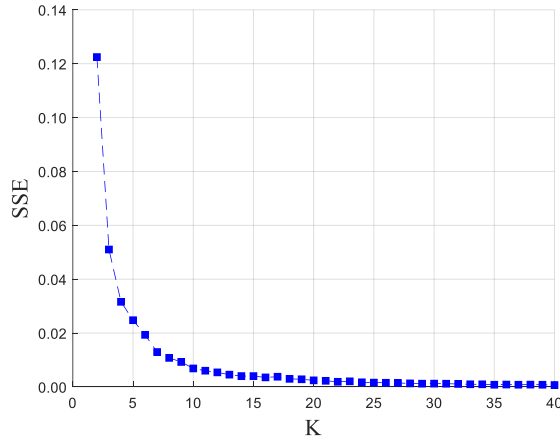


Figure 5.4 The SSE curve for determining the appropriate K value via the elbow method.

The silhouette coefficient reflects the quality of trajectory clustering based on cluster closeness and the separation between clusters; it is expressed as:

$$\text{effect.SC}(x) = \frac{b(x) - a(x)}{\max\{a(x), b(x)\}} \quad (36)$$

where $a(x)$ is the average distance between x and the other trajectories in the cluster and $b(x)$ is the average distance between x and the other trajectories in the nearest-neighbor cluster. Generally, the closer SC is to 1, the better the clustering effect.

In this section, we first calculate the SSE of the data set and plot the SSE as K changes. Next, we select three adjacent K values at the elbow of the curve and calculate the SC for different K values. Finally, we choose the K value with the SC closest to 1, which is the optimal number of clusters.

5.4.3 Model training

5.4.3.1 Improved CTGAN architecture

In this work, three improvements are made to the original CTGAN to improve the accuracy of TP by combining the characteristics of the ADS-B data and the results of aircraft trajectory clustering.

The first improvement addresses the aircraft characteristics contained in the ADS-B data. We perform sampling with the actual data frequencies instead of logarithmic frequencies. The purpose of this strategy is to prevent the synthesizer from unrealistically oversampling rare classes. In this context, rare classes refer to specific categories or attributes within the ADS-B data that occur relatively infrequently. Oversampling these occasional classes could disrupt model convergence [183]. Therefore, actual data frequencies are adopted to maintain a balanced and representative dataset for improved model training and performance.

In addition, we use target encoding instead of one-hot encoding to cluster the trajectory results. When the number of cluster centers increases, one-hot encoding adds many dimensions to the dataset but does not provide much information. These dimensions can lead to very sparse results from the generator, making optimization challenging, especially for neural networks. Even worse, each pair of information-sparse columns is linearly related. This linear relationship means that one variable can be easily predicted using other variables, resulting in problems of parallelism and multicollinearity in high-dimensional cases. In contrast, target encoding only occupies one feature space for processing trajectory clusters, so it can represent the relationships between categorical and target variables more directly. Therefore, we choose target encoding.

The third improvement is that we input clustering data in the generator as additional parameters to improve the TP accuracy. Figure 5.5 shows the architecture of the improved CTGAN. Differences from the original CTGAN are marked in red.

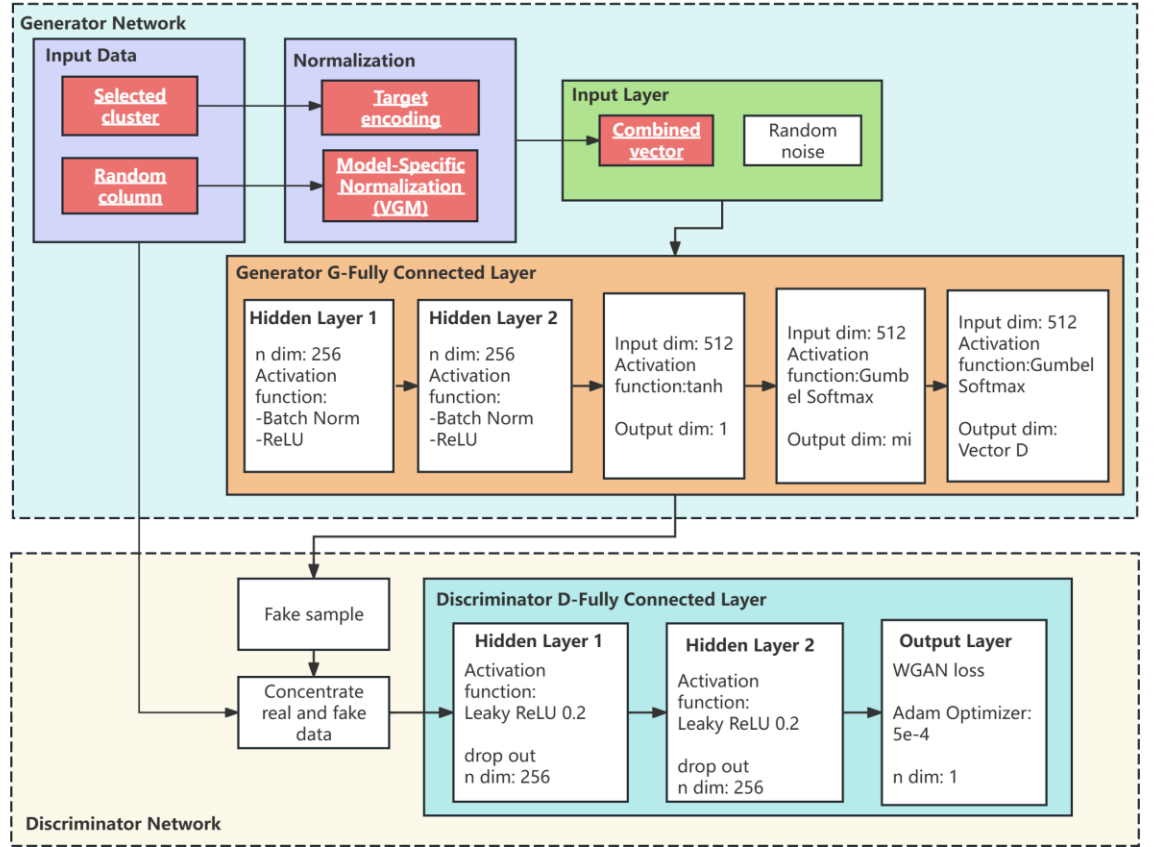


Figure 5.5 Improved CTGAN architecture based on the K-medoids clustering method.

The improved CTGAN architecture consists of a generator network and a discriminator network. In the generator network, the columns of the ADS-B data are normalized using a variational GMM. The selected clusters are converted to binary columns via target encoding and then concatenated with the normalized column vectors to form a combined vector. We input this vector and random noise into the fully connected generator layer as inputs. The first and second hidden layers are fully connected with 256 nodes, and the ReLU function (set to 750 dimensions to produce stable and efficient updates) is applied for batch normalization [206]. The last layer converts the neurons from 512 dimensions to 1 output dimension. The alpha values of the Gumbel softmax [207] function and the tanh function are set to 0.2, the same as in the original CTGAN.

The discriminator network adopts the same settings as those of the original CTGAN architecture. The PacGAN framework [186] is employed, and 10 samples are used in each Pac to prevent model loss. We uniformly set the leaky

ReLU parameter to 0.2 and the dropout function [187] parameter to 0.3. Finally, the loss is determined based on a WGAN and used to linearly transform the gradient [53]. The learning rate of adaptive moment estimation (Adam) optimization is set to $5 \cdot 10^{-4}$.

5.4.3.2 Model evaluation metrics

During model training, the CTGAN treats each row of tabular input data as an independent sample for random sampling without considering the order between rows. It learns to extract the overall distribution characteristics of the data through these samples. Based on the input data and the learned distribution characteristics, it outputs generated data in tabular form, where the rows are randomly arranged. For aircraft TP models, the input to the CTGAN generator is typically actual time series trajectory data. The generated output data consist of nontime-series random data that reflect the distribution characteristics of the trajectories. Therefore, there is no direct one-to-one correspondence between them. For the evaluation of model training results, if the distributions of two datasets are very close, the means of their trajectory parameters will also be very close. The mean values can approximately reflect the gap between the actual distribution and the generated distribution, so the mean values can be used to reflect the similarity between the generated data and the input dataset. To accurately evaluate the model, we use the MMD (maximum mean discrepancy) from the data distribution perspective and the MED (mean Euclidean distance) from the trajectory distance perspective. The formula can be found in Section 4.3.2.

5.4.3.3 Training process

To ensure the robustness and reliability of our model, we divided our dataset into three subsets: training, validation, and testing data. The following steps outline our process:

1). Data splitting

Training data: We used 80% of the dataset to train the model. This subset was used to learn the underlying patterns and relationships in the flight trajectory data.

Validation data: 10% of the dataset was allocated for validation. The validation data was used to tune hyperparameters and prevent overfitting by evaluating the

model's performance during training.

Testing data: The remaining 10% of the dataset was reserved for testing. This subset was not exposed to the model during training and was used to assess the model's generalization capability and final performance.

2). Normalization

We applied a Model-Specific-normalization technique to the input features to ensure all variables had a similar scale. This step was crucial for enhancing the convergence and stability of the training process.

3). Model training

K-medoids clustering: The K-medoids clustering was integrated with CTGAN to enhance the model's ability to handle mixed and complex trajectory patterns.

CTGAN model: We trained the CTGAN model using the training data, ensuring that the model could capture the complex distributions and dependencies in the trajectory data. The training process involved multiple iterations, adjusting the model parameters to minimize the loss function.

4). Validation and hyperparameter tuning

We monitored the model's performance on the validation data throughout the training process. Hyperparameters such as the number of epochs, learning rate, and batch size were tuned based on the validation results to optimize the model's performance. This iterative process helped in selecting the best model configuration.

5). Evaluation

After training, we evaluated the model's performance on the testing data. MMD and MED metrics were used to quantify the model's accuracy and robustness. The testing phase ensured that the model's performance was consistent and reliable on unseen data.

By providing this detailed explanation, we aim to clarify the training process and demonstrate the rigor applied to ensure the model's reliability and accuracy.

5.4.4 Trajectory prediction

After the C-CTGAN model is trained, it can be used for TP. The planned route is input into the trained model when predicting the trajectory. C-CTGAN will output the prediction results of each trajectory segment according to the classification and form a complete predicted trajectory after splicing, which is the final prediction result. Since the aircraft-planned route data corresponding to the historical trajectory cannot be obtained, the planned route in the experiment is replaced by the historical trajectory of the test data set.

5.5 Experiment and analysis

5.5.1 Data preparation

5.5.1.1 Experimental data

This chapter uses the same data source as Chapter 4, namely the ADS-B dataset around HKIA (Hong Kong International Airport) collected by the HKO from December 23 to December 27, 2016.

5.5.1.2 Data processing and grouping

First, following the method outlined in Section 4.3.3.1, ADS-B data are initially converted from the ASTERIX category 021 format to tabular form, followed by data cleaning, trajectory organization, and time alignment processing. Next, to validate the accuracy, adaptability, and stability of the TP model, two groups of data are selected from the preprocessed trajectory dataset, each consisting of 30 trajectories. The first sample data group consists of trajectories for flights that directly arrive and land, with relatively simple trajectory forms and no holding pattern data. The second sample data group consists of trajectories for flights with holding, with more complex trajectory forms. The trajectory conditions for each group are shown in Figure 5.6. Furthermore, the LSTM model indirectly estimates the time parameter through the arrangement of trajectory points. Therefore, the sample data must be processed into a trajectory composed of a series of trajectory points with the same time interval. To compare with those of LSTM-based models, the trajectory points of both data groups are recalculated and organized with equal time intervals (set as 1 second). Finally, for metric evaluation, each sample data group is further divided into training and testing sets at a ratio of 0.8 to 0.2.

For the C-CTGAN model, we express the basic characteristics of the data distributions using PDFs (probability density functions) of the generated dataset and the real dataset. We use the proposed model evaluation metrics, MMD and MED, to validate the model's accuracy. In a comparative experiment, we used the MAE to uniformly measure the accuracy of the prediction results of the C-CTGAN, an LSTM-based model, and the original CTGAN method.

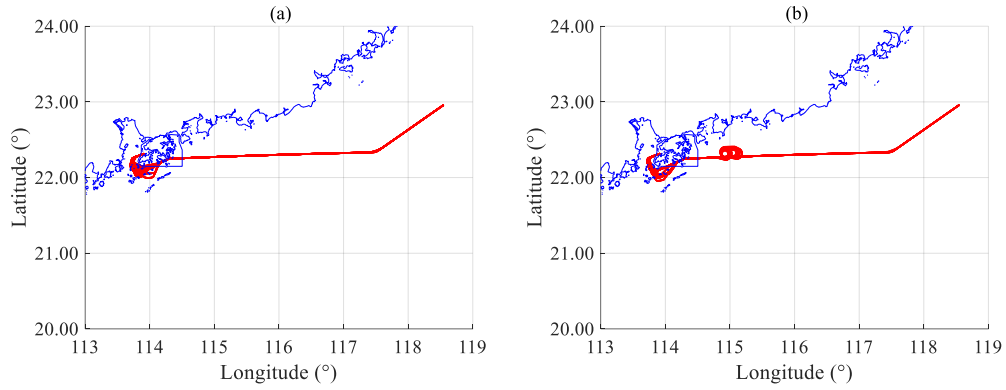


Figure 5.6 The selected flight trajectories from waypoints KADLO to VHHH: (a) without holding, (b) with holding.

5.5.2 Test environment and model parameter settings

The experiment is performed on a PC with Microsoft Windows 10 (CPU: 12th-Gen Intel(R) Core i9-12900KF at 3.20 GHz, RAM: 64.0 GB, GPU: NVIDIA GTX 3080). The CTGAN-based models were built using Keras and TensorFlow. The detailed training parameters are shown in Table 5.1.

Table 5.1 Training parameters for the C-CTGAN.

Parameter	Description	Value
Embedding dim.	Size of the random sample passed to the generator	128
Generator dim.	Size of the output samples for each one of the residuals	(256, 256)
Discriminator dim.	Size of the output samples for each one of the discriminator layers	(256, 256)
Generator lr	Learning rate for the generator	$5 \cdot 10^{-4}$

Generator decay	Generator weight decay for the Adam optimizer	$1 \cdot 10^{-6}$
Discriminator lr	Learning rate for the discriminator	$5 \cdot 10^{-4}$
Discriminator decay	Discriminator weight decay for the Adam optimizer	$1 \cdot 10^{-6}$
Batch size	Number of data samples to process in each step	750
Step ratio	Number of discriminator updates to perform for each generator update	1
Epochs	Number of training epochs	50,000
Pac	Number of samples to group together when applying the discriminator	10

For the LSTM-based models, we use MATLAB to run the prediction algorithms for comparison purposes, and the proportions of the training set and test set are also 0.8 and 0.2, respectively. Previous studies [117] have shown that the accuracy of BP neural networks for medium- and long-term aircraft TP is far inferior to that of LSTM neural networks. Therefore, we selected four commonly used neural networks—LSTM, CNN-LSTM, CNN-LSTM-Attention, and CNN-BiLSTM—for comparison analysis. The remaining parameters were set as follows: 100 hidden layers, 2000 epochs, a learning rate of 0.9, and a learning rate reduction period of 1600 for LSTM; a 2-layer convolutional network and a convolution kernel size of [3,1], where the first convolutional layer has 32 feature maps, the second layer has 64 feature maps, and both apply ReLU activation, 100 hidden layers, 1000 epochs, a minibatch size of 200, a learning rate of 0.005, a learning rate reduction factor of 0.9, and a learning rate reduction period of 800 for CNN-LSTM; the same as CNN-LSTM, with 100 hidden layers, 80 epochs, a minibatch size of 100, a learning rate of 0.005, a learning rate reduction factor of 0.9, and a validation frequency of 10 for CNN-LSTM-Attention. The parameters of CNN-BiLSTM are the same as those of CNN-LSTM, except that the BiLSTM layer is used instead of the LSTM layer, the learning rate is changed to 0.001, and the learning rate reduction factor is set to 0.1.

5.5.3 Model testing and analysis

5.5.3.1 C-CTGAN model testing and analysis

First, we cluster the two sample datasets according to the method outlined in Section 5.4.2. Based on Equations (35) and (36), the optimal number of clusters for both datasets is 6. The clustered trajectories are shown in Figure 5.7, where different colors represent different clusters.

Next, following the method described in Section 5.4.3, we train the model using the two sample datasets and calculate their respective MMD and MED based on Equations (26) and (29). For the first group of samples shown in Figure 5.7(a), the MMD is 0.0415, and the MED is 0.0049. For the second group of examples shown in Figure 5.7(b), the MMD is 0.0422, and the MED is 0.0069. Small values of MMD and MED indicate high accuracy in generating trajectories. When they are both equal to 0, the generated data are identical to the actual data. The parameter calculation results show that the predicted trajectory can simulate the actual trajectory well and is characterized by high prediction accuracy.

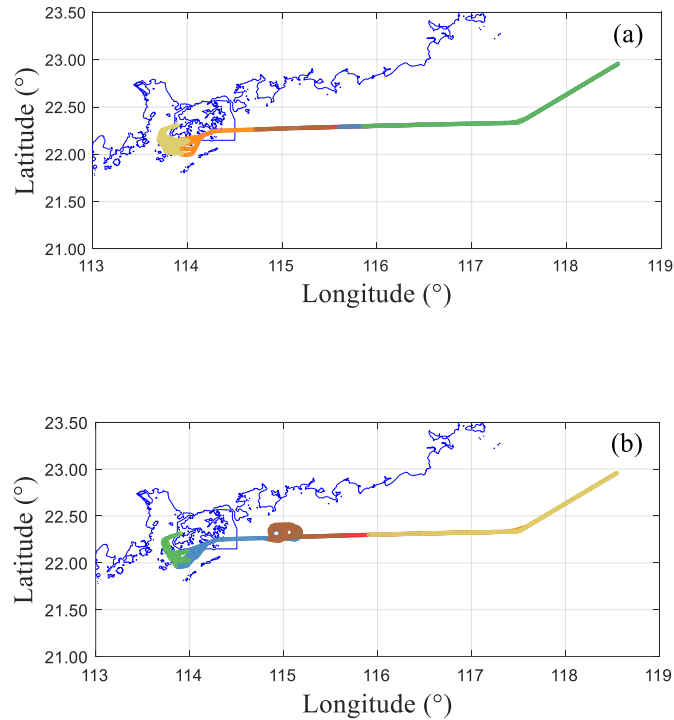


Figure 5.7 The clustered flight trajectories from waypoints KADLO to VHHH: (a) without holding, (b) with holding.

To intuitively observe the similarity between the model-generated data and the real data distribution, we calculate the probability density distribution with the kernel function as the PDF for all parameters of the two sets of data. This probability density distribution difference intuitively reflects the distance between the generated and real data distributions. Figure 5.8 shows the PDFs of the time, latitude, longitude, geometric height, ground speed, and track angle variables for the two datasets.

We found that there is a good fit between the blue line representing the real data distribution and the red line representing the generated data distribution except for the TA parameter. This result indicates that the real and generated data have a high degree of similarity in their distributions. This observation aligns well with the calculated evaluation metrics.

Comparing Figure 5.8 (b) to Figure 5.8 (a), we notice a significantly poorer fit in the case of Figure 5.8 (b), indicating that trajectories with flight holding can impact the accuracy of the C-CTGAN model. Regarding the TA parameter, there is a situation in which the fit is poor at a particular position, regardless of the presence of flight holding data. This is because the trajectory angle values undergo a bidirectional mutation from 360 degrees to 0 degrees, significantly affecting the output of the C-CTGAN model, which relies on probability density for prediction. Fortunately, in long-term TP, parameters such as Lat, Lon, GH, and GS are core parameters, while TA is an auxiliary parameter that can be derived from adjacent trajectory points. As a result, this approach has a very limited impact on TP.

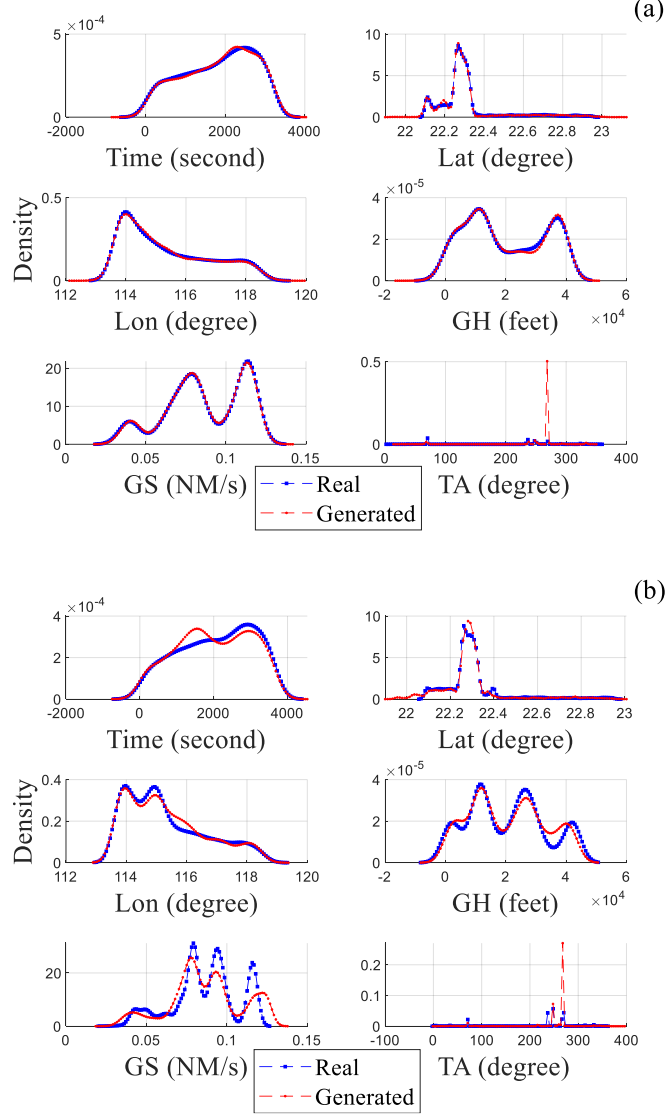


Figure 5.8 Distribution comparison between the generated and real data: (a) Flight trajectories without holding, (b) flight trajectories with holding.

The experimental results demonstrate that the trajectories generated by the C-CTGAN model exhibit a high degree of similarity with the actual trajectories. It should also be noted that compared to the commonly used LSTM-based models, CTGAN, being a probabilistic model, performs better in mixed scenarios due to its probabilistic nature.

5.5.3.2 Comparative experiments and analysis

To further demonstrate the TP accuracy of the C-CTGAN, we conducted comparative experiments with two sets of sample data using the LSTM, CNN-LSTM, CNN-LSTM-Attention, CNN-BiLSTM, and original CTGAN models.

Since the MAE provides a direct indication of the degree of conformity between the generated and actual trajectories, in the comparative experiments, we calculated the MAE for each of the above methods to analyze and compare the accuracy of different TP methods.

(1) Determination of the time span for the LSTM-based models

Previous research has shown that due to the inherent algorithmic limitations of LSTM models, their prediction accuracy decreases rapidly with increasing prediction time [107]. To address this issue, we conducted experiments using LSTM to predict trajectories with time spans of 1, 3, 5, 7, 10, and 15 minutes, as shown in Figure 5.9. This illustrates the trend of the MAE values for various relevant parameters; notably, the error rapidly increases with the extension of the prediction time span after z-score standardization, consistent with previous research findings. A z-score, also called a standard score, is the difference between a number and the mean divided by the standard deviation. In statistics, a z-score is a signed number of standard deviations indicating if an observation or data point is above the mean of the observed or measured values. Scores larger than the mean receive a positive standard score, and scores smaller than the mean receive a negative standard score. Since medium- to long-term prediction refers to predictions with a time span of more than 10 minutes, 10 minutes is selected for the LSTM prediction models for comparative analysis.

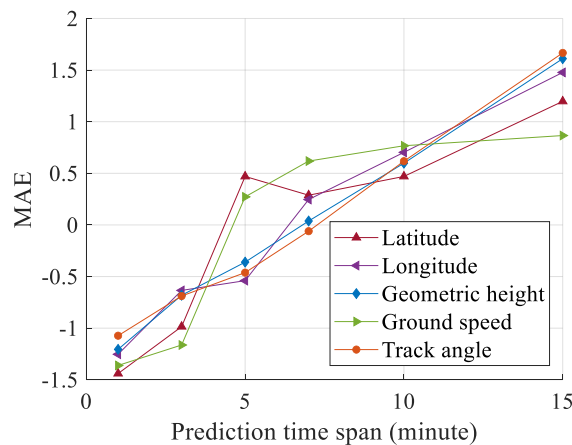


Figure 5.9 The relationship between the prediction error of the LSTM model and the prediction time span after z-score standardization. The ordinate represents the MAEs of different parameters.

(2) Comparison of TP models

Table 5.2 and Table 5.3 present the MAEs for TP using the models based on the data with and without holding, respectively. In these tables, Lat represents the latitude, Lon represents the longitude, GH represents the geometric height, GS represents the ground speed, and TA represents the track angle.

Table 5.2 MAE comparison among the LSTM, CNN-LSTM, CNN-LSTM-Attention, CNN-BiLSTM, original CTGAN, and C-CTGAN models based on flight trajectories without holding.

Model	MAE				
	Lat (°)	Lon (°)	GH (ft)	GS (NM/s)	TA (°)
LSTM	0.0116	0.0357	684.3974	0.0045	5.0389
CNN-LSTM	0.0093	0.0340	789.4817	0.0038	8.7183
CNN-LSTM-Attention	0.0101	0.0580	814.9366	0.0044	8.7585
CNN-BiLSTM	0.0106	0.0346	700.9499	0.0047	7.6823
Original CTGAN	0.0043	0.0297	266.3775	0.0023	3.2114
C-CTGAN	0.0028	0.0289	204.6852	0.0006	2.0433

Table 5.3 MAE comparison among the LSTM, CNN-LSTM, CNN-LSTM-Attention, CNN-BiLSTM, original CTGAN, and C-CTGAN models based on the flight trajectories with holding.

Model	MAE				
	Lat (°)	Lon (°)	GH (ft)	GS (NM/s)	TA (°)
LSTM	0.0216	0.0782	2025.96	0.0054	15.9191
CNN-LSTM	0.0213	0.0929	1449.65	0.0035	13.8093
CNN-LSTM-Attention	0.0339	0.2300	3250.20	0.0068	31.2471
CNN-BiLSTM	0.0299	0.1810	2972.91	0.0072	27.3524
Original CTGAN	0.0230	0.1164	1460.80	0.0041	17.5870
C-CTGAN	0.0183	0.0709	993.61	0.0034	18.5150

1). Comparative analysis of LSTM-based models

Table 5.2 and Table 5.3 indicate that the CNN-LSTM model performs the best.

Based on the sample without holding data, the MAEs of various parameters (Lat, Lon, GH, GS, TA) for the LSTM-based models are generally small, and the MAEs for the same parameters are similar overall. This indicates that all models yield a high level of TP accuracy. Based on the sample with holding data, the MAEs of trajectory-related parameters for the LSTM-based models increase significantly, and the differences in the MAEs for the same parameter are notably different among different models. This indicates a significant difference in TP accuracy for different models. The results also suggest that different models have varying degrees of adaptability to the sample data.

2). Comparative analysis between the C-CTGAN, original CTGAN and CNN-LSTM models

Since CNN-LSTM is the best-performing model of the LSTM family, we use it as the representative to perform a comparative analysis with other models. The data in Table 5.2 and Table 5.3 show that the MAEs of the trajectory-related parameters, except TA, in the C-CTGAN model are smaller than the corresponding values of the original CTGAN and CNN-LSTM models. The C-CTGAN model significantly outperforms the original CTGAN and CNN-LSTM models in terms of key TP parameters. For the data without holding, the TP accuracy of the C-CTGAN is considerably better than that for the data with holding.

When using the sample dataset without holding trajectories, the C-CTGAN model reduces the MAEs of core trajectory parameters such as latitude, longitude, geometric altitude, and ground speed by 69.89%, 15.00%, 74.07%, and 84.21%, respectively, compared to those of the CNN-LSTM model. Compared to those of the original CTGAN, the MAEs of the core parameters are reduced by 20.43%, 39.09%, 31.98%, and 17.07%, respectively.

When using the sample dataset with holding trajectories, the C-CTGAN model shows significant improvements in predicting parameters such as latitude, longitude, geometric altitude, and ground speed, with MAE reductions of 14.08%, 23.68%, 31.46%, and 2.86%, respectively, when compared to those of the CNN-LSTM model. Compared to those of the original CTGAN, the MAEs of the core parameters are reduced by 34.88%, 2.69%, 23.16%, and 73.91%, respectively.

Table 5.2 and Table 5.3 also indicate that for nonholding sample data, the TP accuracy of the original CTGAN is significantly better than that of the CNN-LSTM model. However, in the case of the holding dataset, the TP accuracies of the two models are essentially equivalent.

3). Typical TP visualization

To compare the medium- and long-term TP effects of different models in an intuitive manner, we select flight CPA405 (trajectory without holding) and flight ANA859 (trajectory with holding) as examples to examine the differences between the TP parameters among the CNN-LSTM, CTGAN, and C-CTGAN models. We smoothed the trajectory parameters based on the CTGAN and the C-CTGAN models for easy visualization.

Figure 5.10 displays the prediction results of the original CTGAN, C-CTGAN, and CNN-LSTM models based on the CPA405 dataset. According to the visualized experimental results, the C-CTGAN model can fit the actual trajectories and predict the overall trajectory trend effectively. Compared to those of the C-CTGAN model, the predictions of the original CTGAN and CNN-LSTM models roughly align with the direction of the actual trajectory. However, there is still a particular gap in fitting accuracy. This result is consistent with the results in Table 5.2. Compared to the original CTGAN, C-CTGAN exhibits a smaller average distance deviation, better meeting the accuracy requirements for TP.

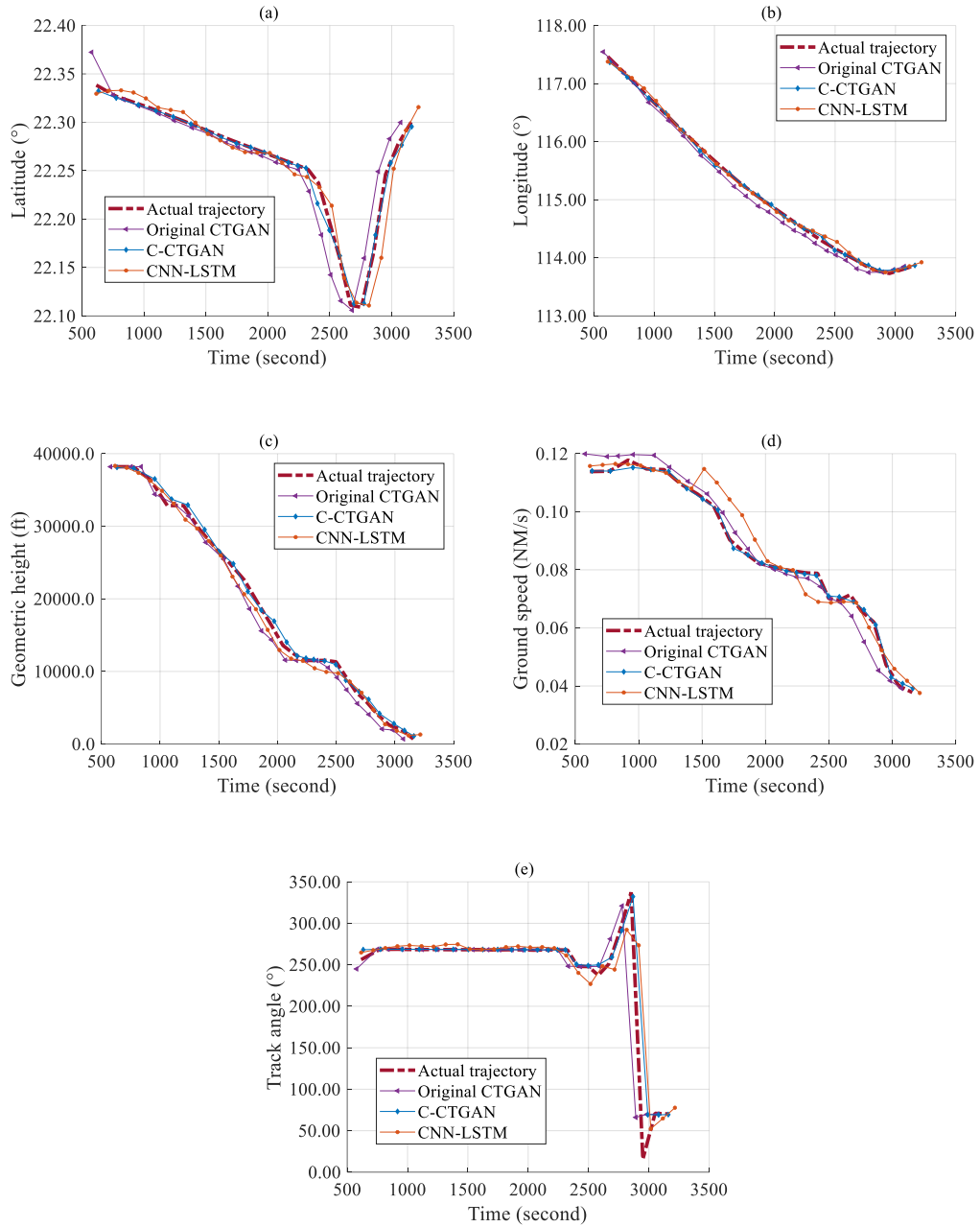


Figure 5.10 Comparison of the prediction results of the CNN-LSTM, CTGAN, and C-CTGAN models for flight CPA405 (trajectory without holding).

Using the actual flight trajectory of ANA859 as a reference, we examined the differences in the predicted trajectory parameters among the original CTGAN, C-CTGAN, and CNN-LSTM models, as shown in Figure 5.11. With the introduction of clustering as an additional step, C-CTGAN is better equipped to handle parameter fluctuations caused by holding and learn the data distribution compared to other models. Overall, the C-CTGAN model achieves the highest prediction accuracy, which is consistent with the results in Table 5.3.

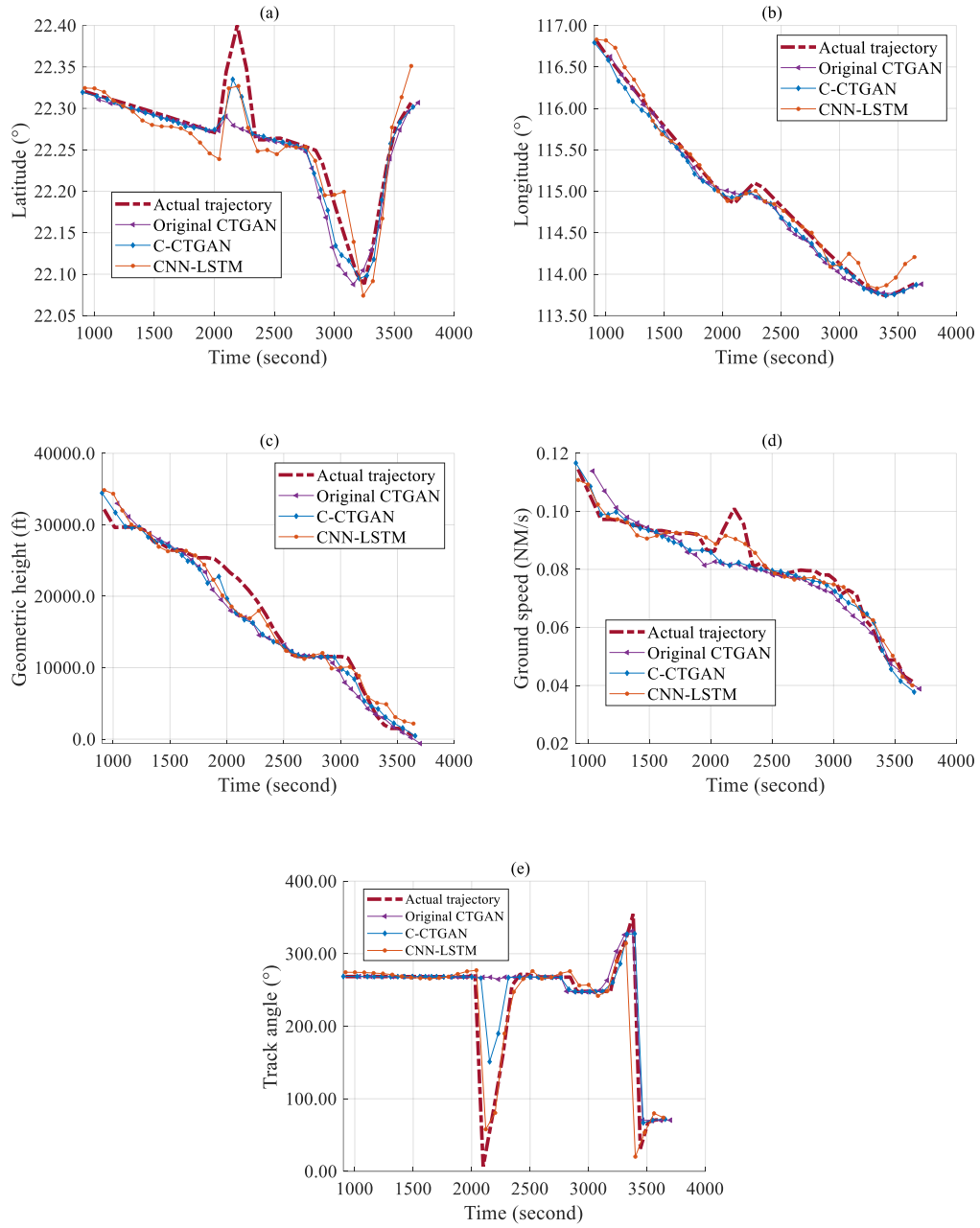


Figure 5.11 Comparison of the prediction results of the CNN-LSTM, CTGAN, and C-CTGAN models for flight ANA859 (trajectory with holding).

By comparing Figure 5.10 and Figure 5.11, it can be observed that the predicted trajectory parameters in Figure 5.10 generally better fit the real trajectory parameters than those in Figure 5.11. This finding aligns with the findings in Table 5.2 and Table 5.3, indicating that the holding trajectory data have a significant impact on the TP accuracy.

5.5.3.3 Comparison and analysis of model complexity

Analyzing model complexity is fundamental for efficient resource utilization, scalability, performance optimization, practical deployment, understanding trade-offs, and informed decision-making in developing and applying ML models. So in this section, we compared the model complexity between the LSTM-based models and CTGAN models.

1). LSTM-based models

The computational complexity of an LSTM model primarily depends on the dimensions of its inputs, outputs, and hidden states.

The computational complexity of an LSTM model for a sequence of length T is: $O(T(dh + h^2))$, where d is the length of the sequence, h is the input vector dimension, and T is the number of hidden units.

This complexity describes the amount of computation needed for the LSTM to process an input sequence of T length. For large-scale data and complex models, this complexity can significantly impact the training and prediction efficiency. Optimizations and hardware acceleration (like using GPUs) are often employed in practice to handle the computational demands of LSTM models effectively.

2). CTGAN-based models

The computational complexity of training a CTGAN model involves understanding the complexities of both the generator and the discriminator, as well as the overall training process.

The computational complexity of training a CTGAN model is: $O(T \times m \times h \times (L \times d + K \times n))$, where T is the number of training epochs, m is the batch size, h is the number of hidden units in the layers, d is the input vector dimension for the generator, n is the number of features in the tabular data, and L and K are the number of layers in the generator and discriminator, respectively.

This complexity estimate provides insight into the computational resources required for training a CTGAN model, highlighting the importance of efficient implementation and potential hardware acceleration for handling large datasets and complex models.

3). Comparison between the LSTM-based models and CTGAN models

LSTM models are particularly computationally intensive for long sequences and sizeable hidden state sizes. The quadratic dependency on the number of hidden units h makes them more sensitive to changes in h . CTGAN models depend on the depth and width of both the generator and discriminator networks. The complexity grows linearly with the number of features n and the dimension of the input vector d , as well as with the number of layers and hidden units. Regarding the ADS-B dataset, the CTGAN model is more complex than the LSTM model. In our experiments, CTGAN model, being a probabilistic model, requires significantly more time to run a complete cycle compared to LSTM-based models. Specifically, the CTGAN-based models take approximately 5 to 6 hours per model in the training process, while the LSTM-based models typically take between 40 minutes to 1.5 hours per model. Using the actual flight trajectory of ANA859 as a reference, the CNN-LSTM model takes 1.097 seconds for prediction, while the C-CTGAN model takes 2.273 seconds.

While the CTGAN-based model requires more training time and computational resources, it offers significant advantages in terms of accuracy and robustness in medium- and long-term TPs. The probabilistic nature of CTGAN allows it to perform better in mixed and complex scenarios than the LSTM-based model.

5.5.4 Experimental conclusion

1). In this section, the accuracy of the C-CTGAN model is evaluated using the MMD and MED metrics, which assess the model from the perspectives of data distribution and ballistic distance. While MMD and MED are adequate for evaluating GAN-based models, they are unsuitable for evaluating LSTM-based models. In contrast, MAE can be used for both types of models. Therefore, MAE is employed as an evaluation metric to compare the C-CTGAN model with LSTM, CNN-LSTM, CNN-LSTM-attention, CNN-BiLSTM, and the original CTGAN model. The results demonstrate that our model significantly outperforms the LSTM-based and original CTGAN models in predicting medium- and long-term trajectories. The clustering method effectively enhances the accuracy of the CTGAN model in TP.

2). Compared to the best-performing model in the LSTM family, the CNN-LSTM

model, the C-CTGAN model reduces the MAEs of key trajectory parameters such as latitude, longitude, geometric altitude, and ground speed by 69.89%, 15.00%, 74.07%, and 84.21%, respectively, when using a dataset of unseen trajectory samples. Compared to the original CTGAN method, this approach reduces MAEs in latitude, longitude, geometric altitude, and ground speed by 20.43%, 39.09%, 31.98%, and 17.07%, respectively. Furthermore, compared to the CNN-LSTM model, the C-CTGAN model shows significant improvement in predicting latitude, longitude, geometric altitude, and ground speed, with MAE reductions of 14.08%, 23.68%, 31.46%, and 2.86%, respectively. Compared to the original CTGAN algorithm, the MAEs of this method decrease by 34.88%, 2.69%, 23.16%, and 73.91%, respectively, for these parameters.

3). In the comparative experiments, the time span of predictions for LSTM-based models was 10 minutes. As the prediction time span increases, the advantages of the C-CTGAN model over LSTM-based TP models become more pronounced. The experiments also show that the trajectory segments used significantly impact the accuracy of the model's TP.

4). The experiments demonstrate that distinguishing trajectory types is crucial for maintaining TP accuracy. While the C-CTGAN model accurately predicted specific aircraft trajectories, its performance changed significantly when introducing holding patterns. It is essential to differentiate between scenarios with and without holding patterns. Applying a TP model without determining the trajectory mode, especially when holding patterns are involved, leads to decreased prediction accuracy. Therefore, future work must improve the model's handling of mixed data patterns to ensure robust performance across all trajectory types. One standard solution is to determine the trajectory mode based on information such as weather and flight plans, then select the pre-trained TP model corresponding to the identified mode.

In conclusion, the C-CTGAN model demonstrates significant advantages in TP, particularly in medium- and long-term prediction scenarios. Clustering improves TP accuracy, making this model a superior choice for complex medium- and long-term TP tasks.

5.6 Chapter summary

This chapter focuses on medium- and long-term TP and undertakes the following tasks:

1). Analysis of the current research on medium- and long-term TP

The primary models for medium- and long-term TP can be categorized into aircraft planning- and ML-based models. Most aircraft planning-based TP models are physics-based and aim to reduce or limit uncertainty to improve TP accuracy. One common approach to achieve this is by improving the accuracy of weather forecast data (including real-time weather conditions), which is used to adjust planned flight paths and enhance prediction accuracy. In contrast, there is limited research and few results in ML-based TP for medium- and long-term time spans. Achieving high accuracy and reliability in medium- and long-term TP remains a significant challenge.

2). Analysis and summary of common clustering methods and medium- and long-term TP models

This section provides a detailed introduction to commonly used clustering methods such as K-means, K-medoids, and DBSCAN, along with an analysis of their characteristics. Additionally, it introduces TP models such as flight plan interpolation, stateful LSTM, Conv1D-GAN, and nominal flight path models for medium- and long-term prediction. The composition and features of these models are discussed in detail.

3). Proposal of a 4D TP method based on K-medoids clustering and the CTGAN (C-CTGAN) framework

The proposed method begins by segmenting historical trajectories using a clustering model to reduce trajectory complexity. Each segment is then trained with its model. During TP, the trained models produce segment-wise predictions based on the input flight plan, stitched together to form the final predicted trajectory. The method was designed with optimizations to the original CTGAN model, considering the characteristics of ADS-B data and the requirements of medium- and long-term TP to improve performance. Additionally, evaluation

methods and metrics were proposed to assess and verify the accuracy and stability of the models.

4). In-depth comparative experiments to validate the proposed C-CTGAN model's performance in medium- and long-term TP

Based on the actual conditions of the experimental area, flight trajectories were classified into two types: with and without holding patterns. The proposed model was then compared against LSTM, CNN-LSTM, CNN-LSTM-Attention, CNN-BiLSTM, and the original CTGAN models using data from both types of trajectories to evaluate the model's performance. The analysis showed that the proposed C-CTGAN model significantly improved accuracy in medium- and long-term TPs. Finally, the complexity of the models was compared and analyzed.

Chapter 6.

Aircraft trajectory prediction considering weather factors and minority-class sample types

Weather is a critical factor influencing flight, with convective weather and severe turbulence significantly impacting flight safety. These conditions are often the primary reasons for changes in aircraft trajectories. Typically, aircraft follow planned flight paths outlined by ATC. However, when encountering convective weather and severe turbulence, ATC may direct flying aircraft to alter their flight paths (either horizontally or vertically) and adjust their arrival times at designated locations to avoid adverse weather conditions. Even when aircraft follow their planned routes, significant deviations may occur due to poor weather. Therefore, accounting for weather factors, especially the impact of severe weather on TP, is crucial for enhancing the safety and efficiency of air transport [71].

This chapter focuses on the technical methods for incorporating weather factors into TP. Specifically, it addresses the challenge of generating minority-class sample data for weather events, such as severe weather, which significantly impact TP but are underrepresented in datasets. The effectiveness of the proposed data augmentation methods is validated through experiments.

6.1 Research background

Commonly used aircraft TP methods that consider weather effects can be summarized into two categories: traditional TP and ML.

As an essential input parameter for traditional ballistic prediction methods, weather significantly affects TP accuracy. Therefore, many authors have proposed various methods to mitigate the impact of weather uncertainty. For example, Porretta et al. [31] used a standard civil aircraft 3D point mass model to simulate

aircraft dynamics and predict aircraft trajectories. The paper decomposed the wind field into two components: nominal and random. The nominal wind was simulated as a lookup table, and the random wind was the deviation between the wind field and the nominal wind, which was derived from samples obtained from measurements or weather forecasts. The participation of random wind as an influencing factor in the calculation improved the accuracy of TP. Tang et al. [38] proposed a method for extracting nominal flight profiles and correcting route meteorological forecasts. Extracting flight intentions from historical trajectory data eliminated the randomness of historical flight altitude distribution caused by meteorological factors. The DSW algorithm was used to measure the distance between two flight altitude lines and extract a nominal flight profile representative of the route. The Cressman interpolation method was used to correct the GRIB data in combination with the new AMDAR data forwarded by another reference aircraft to eliminate meteorological forecast errors. Based on the nominal flight profile and higher-precision meteorological forecast data, the accuracy of strategic 4D trajectory forecasts is improved.

However, improvements to traditional TP methods still struggle to meet the high-precision, multi-dimensional prediction requirements of the TBO environment. Therefore, in recent years, more and more scholars have turned to ML methods to learn from historical flight trajectories and meteorological data for aircraft TP. Since ML methods do not require explicit modeling of aircraft performance, procedures, and airspace but rather learn patterns from massive data, they can effectively distinguish the influence of weather factors on aircraft trajectories to a large extent. Therefore, they are very suitable for weather-related aircraft TP tasks. The effectiveness of prediction improves as the volume of sample data used for learning increases. Currently, proposed prediction models include GMM, HMM, MLP, CNN, LSTM, and GAN [52], [91].

Liu and Hansen proposed a deep generative network framework for aircraft TP [208], which consists of an LSTM encoder network and a mixture density LSTM decoder network. The decoder network learns spatiotemporal correlations from historical flight trajectories and outputs parameters of a GMM. This model uses beam search, adaptive Kalman filtering, and the Rauch-Tung-Striebel smoothing algorithm during inference. It can take high-dimensional weather features (wind,

temperature, and convective weather) and the final flight plan as input conditions to predict the aircraft's 4D trajectory.

Inspired by [208], Pang et al. proposed a network architecture that embeds convolutional layers into LSTM units to predict trajectories based on convective weather conditions before aircraft takeoff and flight plans [106]. In this model, ET (echo top) data is used as weather condition data and flight plan input to the LSTM layer to predict the aircraft's position. The results showed an improvement in the accuracy of 47% of flight plans, with an average increase of 12.3%. However, this result focuses on 3D trajectory prediction (ignoring altitude factors). Subsequently, Pang et al. replaced the previous LSTM units with CGAN architecture to predict trajectories based on ET data and flight plans [209]. This improved flight plan accuracy to 55.2%, with an overall variance reduction of 22.1%. Building upon this, Pang et al. extended their research to 4D TP [14], reporting errors in understandable forms of latitude, longitude, and feet altitude. Horizontal errors typically occurred within 1 degree of latitude and longitude, approximately 60 nautical miles (nmi), and 100 feet (ft) altitude. This study improved the accuracy of aircraft TP by explicitly considering weather uncertainty through weather feature extraction methods.

Recent research on weather-related aircraft TP is referenced in [92]. Schimpf et al. proposed a generalized hybrid-recurrent prediction model for aircraft TP to eliminate potential biases in model training across different routes and dates. This model combines CNN, SA, LSTM, and GRU, enabling route generalization and improving flight TP accuracy. In this approach, the weather is processed into cubes using conventional methods. After feature extraction, it is connected with relevant trajectory point information as feature inputs for model learning and prediction. The study also comprehensively compares weather products used in previous literature (includes ET, vertical integrated liquid (VIL), temperature, west wind speed and south wind speed) and their combined effectiveness, suggesting that the ET feature is the most suitable overall weather product for aircraft TP.

In summary, while these studies have made significant progress in considering weather-related aircraft TP tasks, most of their proposed models and datasets only

account for general convective weather and do not consider in-depth research on the impact of severe weather conditions. Challenges still exist in predicting aircraft trajectories under severe weather conditions. In this chapter, severe weather refers to meteorological conditions that significantly negatively affect aviation safety and operations, mainly strong convective weather, excluding extreme adverse weather conditions where aircraft cannot fly. The severe weather can be distinguished from general convective weather through ET data.

Predicting aircraft trajectories under severe weather conditions presents two main challenges. First, datasets containing severe weather and the corresponding flight data are scarce, making these minority-class sample datasets. Deep neural networks commonly used today often employ mini-batch stochastic sampling methods to enhance training efficiency. Since minority-class sample data are less likely to be selected, extracting the relevant feature knowledge is challenging. The second challenge is that severe weather significantly impacts aircraft trajectories more than general convective weather, resulting in more significant trajectory deviations. These deviations are easily identified as outliers by ML algorithms and thus are often ignored. Therefore, conducting research on TP technology based on minority-class category sample data is the key to solving the problem of TP under severe weather conditions.

6.2 Weather factor representation and data processing methods

6.2.1 Weather factor representation

Many weather factors influence flight, such as temperature, humidity, wind speed and direction, precipitation, thunderstorms, turbulence, cloud cover, and visibility. These elements typically exist in the form of weather data products, and their use in practice depends on factors such as the degree to which they influence trajectory in a given application, their compatibility with trajectory information, and the ease of accessing these resources.

In deep learning-based TP, weather is an essential factor that affects prediction accuracy. Therefore, it must be included as input data during model training and used as a control variable during TP to enhance precision. Weather data is typically processed into a cube and, after feature extraction, is connected with

corresponding trajectory point information, which is then input into the model during learning and prediction. The concept of the weather cube was first introduced by Ayhan and Samet [52], and it has since had a significant influence on subsequent TP models that take weather into account. The core idea is to treat airspace as a 3D grid network and store weather information within each grid. By establishing connections between each trajectory point and the weather grids ahead of it, these grids form a joint 4D trajectory cube, as illustrated in Figure 6.1. Most research studies considering weather impacts on TP have since referenced this weather cube concept.

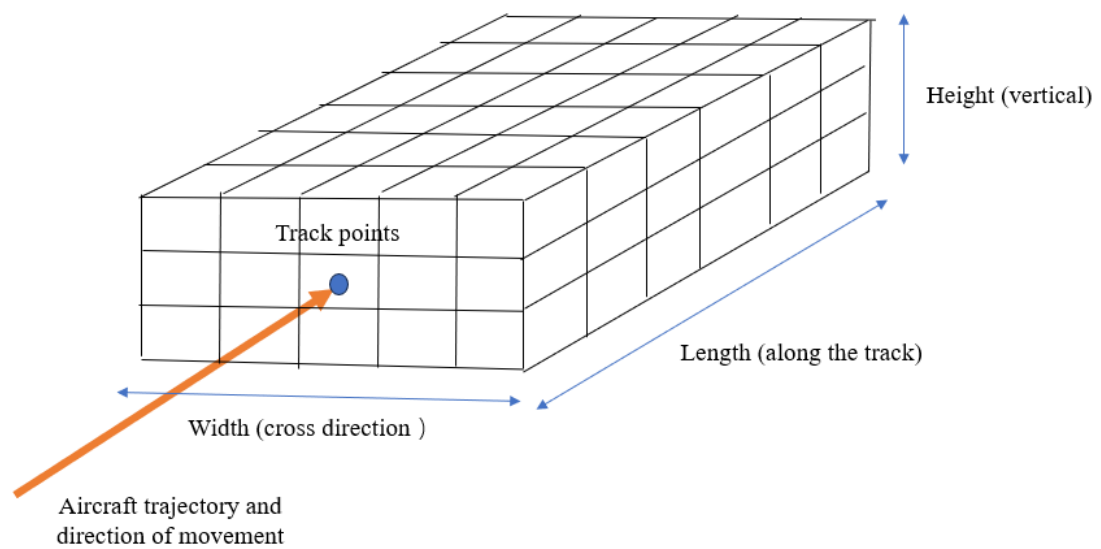


Figure 6.1 Weather cube concept diagram.

The three directions of the cube are consistent with the along-track direction, cross-direction, and vertical direction of the aircraft trajectory, and its size should cover the area of concern when considering weather impacts in actual flight. When considering the effect of multiple weather factors, multiple cubes can represent each cube corresponding to one weather factor.

6.2.2 Weather cube calculation

The weather cube calculation method usually adopts the method introduced in the literature [80], which is as follows:

Weather Cube Generation

Data: Actual trajectory data, weather data

Input: Trajectory points, cube size, weather

Output: Weather Cube Coord, Weather Cube

- 1). Convert the weather data coordinates to Mercator projection coordinates.
- 2). Convert the trajectory point coordinates to Mercator projection coordinates.

For each point in trajectory point, do:

- 3). Determine the flight direction.
- 4). Find a line perpendicular to the normal at the current position.
- 5). Generate 20 points along this perpendicular line.

For each step in Cube Size, do:

- 6). From each of these 20 points, move to the next point to the right, which lies along a new perpendicular line parallel to the previous line, at a distance equal to the step size.
- 7). Generate 20 points along the new perpendicular line.
- 8). From the rightmost of these new 20 points, move one more step to the right.

End of the loop over Cube Size steps.

End of the loop over trajectory points.

- 9). Save Weather Cube Coord and Weather Cube.

6.3 Minority-class sample data processing

6.3.1 Characteristics of minority-class sample types

Minority-class samples refer to one or several types of samples that constitute a tiny proportion of the overall dataset, resulting in data imbalance. The causes of imbalanced datasets can be categorized into two factors: one is the inherent imbalance of phenomena in nature, and the other is practical limitations during the data collection process, which leads to fewer examples of one or more categories. In many application scenarios, minority classes are often the main focus of attention. The absolute number of minority-class samples might still be

considerable, even in large datasets.

The appearance of the minority-class problem is closely related to how sample data is used during the training of deep learning models. Commonly used ML methods often adopt mini-batch stochastic sampling techniques to speed up the training process and improve the model's generalization ability. As a result, minority-class samples have a lower chance of being selected, making it challenging to extract relevant feature knowledge. This is particularly true for deep neural networks, which learn complex nonlinear relationships from data distributions—minority-class sample sizes make it difficult for the model to accurately capture hidden and complex relationships.

Minority-class samples differ from small samples, as the former pertains to the issue of data imbalance. This requires more consideration of the relationship between minority- and majority-class samples. When performing data augmentation, it is crucial to maintain a clear boundary between the minority and majority classes to avoid misclassification in future applications. Additionally, minority-class samples often include several smaller subclasses, and the relationships and distribution characteristics among these subclasses are crucial for enabling finer-grained learning and improving the performance of machine-learning models. This is an important aspect to consider while augmenting minority-class samples.

6.3.2 Methods for processing minority-class samples

The issue of minority-class samples is essentially a problem of data imbalance. There are four main approaches to addressing data imbalance: data-level methods, algorithm-level methods, cost-sensitive methods, and ensemble methods [176].

1). Data-level methods involve balancing the class distribution by removing some samples from the majority class or adding new samples to the minority class. Undersampling methods risk losing statistical information and reducing dataset size, while oversampling methods may introduce noise into the data and affect the learning ability of ML classifiers. In some cases, both methods can be combined to address class imbalance.

2). Algorithm-level methods aim to address the imbalance by modifying the

workflow of ML models. For example, in SVMs, the hyperplane can be guided by a bias term b to maximize the margin separating the two classes. Penalty constants can be introduced for each class, where the cost of misclassifying a minority-class instance is higher than that for the majority class. Additionally, designing hyperparameters that differentiate between majority and minority classes is another effective way to implement algorithm-level methods.

3). Cost-sensitive methods combine data-level and algorithm-level approaches to minimize the overall cost of misclassification. In real-world applications, minority classes are often assigned a higher cost compared to majority classes.

4). Ensemble methods involve training multiple ML classifiers to improve learning accuracy from imbalanced data. Ensemble methods increase accuracy by combining the results of various classifiers, making them more stable and reliable than single-classifier methods. In some cases, modifying the base learners' code can yield better results when dealing with imbalanced data.

All the abovementioned methods contribute to solving the imbalance problem in real-world learning scenarios. Except for data-level methods, which address imbalance by optimizing the data, the other approaches focus on modifying ML models to handle imbalance. A key advantage of data-level methods is their general applicability, as they do not involve modifying the ML model. This allows for the enhanced performance of existing models through data optimization, making it the preferred approach for handling data imbalance.

6.3.3 Synthetic minority over-sampling technique (SMOTE) data enhancement method

Data optimization (data-level) methods include oversampling and undersampling techniques. Undersampling reduces the size of the majority class. However, due to the reduction in the overall dataset size, it is often impractical in real-world scenarios. On the other hand, oversampling expands the minority class by adding more samples and is widely used to balance class distribution in datasets [210]. When dealing with minority-class samples, oversampling poses less risk of overfitting compared to undersampling [211].

The most popular oversampling method is the SMOTE, developed by Chawla et

al. [212]. It generates synthetic data by applying linear interpolation between minority-class data points and their K-nearest neighbors. SMOTE is a powerful oversampling method widely adopted in many applications. Another technique for generating synthetic minority-class samples involves estimating the potential probability distribution of the minority class and generating samples from it. Examples of this approach include PDF oversampling [213] and random walk oversampling [214]. However, in cases where data samples are scarce, density estimation is likely to be inaccurate.

The formalization of SMOTE is shown in Equation (37):

$$p_{new} = p_i + (p_l - p_i) \times w \quad (37)$$

where p_{new} is a new synthetic data point, p_i is a data point from the minority class, p_l is the k nearest neighbor of p_i , and w is a random variable, where $w \in [0,1]$.

SMOTE has several notable features [215]. Its ability to reduce potential overfitting risks, minimize information loss, and demonstrate strong adaptability are vital advantages. Unlike random oversampling, SMOTE generates new synthetic samples instead of simply duplicating existing ones, which helps mitigate overfitting risks and preserves more original data, thereby reducing information loss. Additionally, SMOTE's simplicity and ease of use make it highly flexible, as it allows users to adjust the number of synthetic samples and the number of nearest neighbors (K-value) as needed, offering a high level of adaptability. Due to these advantages, SMOTE has been widely applied across various industries.

However, SMOTE has several limitations, primarily related to assumptions about sample distribution, class boundary-blurring, and noise introduction. SMOTE assumes that minority-class samples are uniformly distributed in the feature space, overlooking the possibility that minority classes may consist of several small, isolated sub-concepts. In real-world datasets, minority-class samples may not be uniformly distributed. Additionally, SMOTE may generate synthetic samples in the boundary regions between different classes, which can blur the class boundaries. Finally, if noise samples exist within the minority class, SMOTE may generate synthetic samples based on these noisy data points, thus introducing even

more noise.

The primary cause of these limitations is that SMOTE assumes data distribution patterns that may not align with reality, as the synthetic samples are not generated from the original minority class distribution. These limitations can affect the quality of synthetic data and may lead to reduced training quality and performance of subsequent ML models.

Given the unique data characteristics and requirements of different fields, the original SMOTE may not fully meet the specific needs of all industries. As a result, SMOTE has undergone extensive improvements since its inception. Although these methods have made significant contributions to addressing data imbalance issues, several key challenges remain [176]:

- 1). By augmenting both classes indiscriminately, some SMOTE variants dilute data authenticity and lack the granularity to address class imbalance effectively, especially in large-scale settings.
- 2). Most SMOTE variants tend to add more records to the minority class without determining the appropriate range, which can increase data noise and reduce class separability.
- 3). Focusing primarily on inter-class imbalance, most SMOTE-based methods overlook the problem of intra-class imbalance, consequently resulting in diminished accuracy and over-generalization.
- 4). Most SMOTE variants fail to account for the quality of data generated by GANs, which can lead to suboptimal performance. This oversight ultimately hinders the accuracy of downstream deep learning models and limits improvements in other relevant metrics.

6.4 Aircraft TP methods under severe weather conditions

Severe weather data is considered a minority class in the imbalanced data problem for aircraft TP. Data augmentation methods can maximize the utilization of existing deep learning-based TP research results without requiring changes to the relevant software. The severe weather TP method is designed as a data augmentation-based TP approach, which includes data preprocessing, data

augmentation, model training, and TP.

6.4.1 Overall framework

In recent years, GANs, a novel neural network technology, have been widely applied in fields such as image generation [170], data generation [171], and image resolution enhancement [127] through the competition between generators and discriminators to learn the distribution of given data. GANs are also suitable for data augmentation and trajectory synthesis [173]. Zhang et al. [108] addressed the limited TP sample data issue by using WGAN-GP to augment the trajectory sample data. They trained an LSTM model for TP based on the augmented and original datasets. The experiments demonstrated that as the sample dataset expanded, the TP accuracy of the LSTM significantly improved. Although this model was designed for small sample scenarios rather than the imbalanced minority-class sample problem, it still represents a valuable attempt to apply data augmentation methods in TP.

CTGAN [128] is the latest network model developed based on the GAN framework, specializing in handling imbalanced data. Since weather parameters are continuous variables and severe weather is a vague concept with statistically defined boundaries, distinguishing individual trajectories under marginally flyable severe weather from normal weather trajectories can be challenging. Moreover, due to severe weather's varying degrees and factors, the minority-class samples may have different sub-class distributions. SMOTE is based on the assumption of uniform distribution rather than actual distribution for data augmentation, while CTGAN generates synthetic data by learning the distribution from the sample data. Additionally, CTGAN excels at handling imbalanced distribution by increasing the sampling frequency for minority-class samples and can further enhance the sub-classes within the minority class. This ability helps reduce the need for precise boundary definitions of severe weather.

Considering the characteristics of severe weather TP and the advantages and limitations of SMOTE and CTGAN, CTGAN is better suited for data augmentation in severe weather TP. To validate the effectiveness of this data augmentation approach, we designed a CTGAN-LSTM hybrid model for short-term TP experiments. In fact, LSTM could be replaced by other TP models

(including hybrid models) depending on the specific application scenario.

Taking LSTM as the TP model, the overall framework for TP under severe weather conditions based on data augmentation is illustrated in Figure 6.2. It consists of four parts: data preparation, data augmentation, model training, and trajectory prediction.

First, the processed weather data is used as a condition, combined with ADS-B data as input, and both datasets undergo preprocessing. Then, the improved CTGAN model is used to augment the minority-class severe weather trajectory data, controlling the number of synthetic data generated to ensure that the data distribution before and after combining the original trajectories remains consistent. This enables the LSTM-based model to learn more effectively.

Next, the synthetic data generated by CTGAN (augmented data) is combined with the preprocessed ADS-B data and weather cube feature data to form a new dataset, which is then input into the LSTM network for training. Finally, the trained LSTM-based model is applied to aircraft TP. Of course, in this framework, other data augmentation networks and prediction networks can replace CTGAN and LSTM, forming new hybrid models while maintaining the primary process.

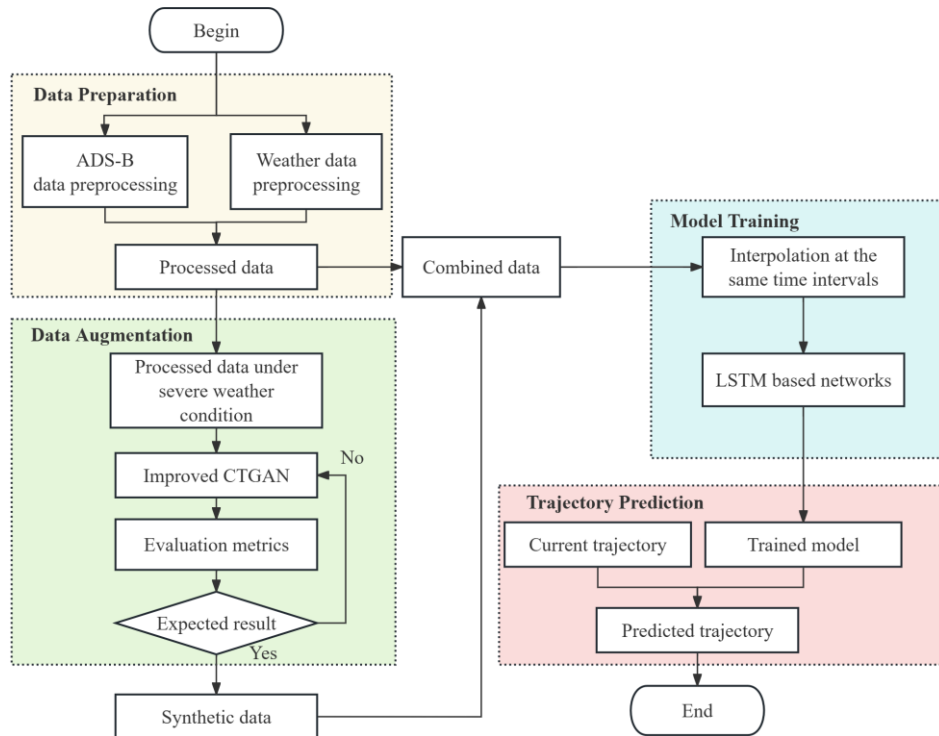


Figure 6.2 The general framework of the 4D aircraft TP method based on the improved CTGAN and LSTM-based architecture.

6.4.2 Data preprocessing

6.4.2.1 ADS-B data preprocessing

ADS-B data preprocessing includes filtering trajectory points according to parameter completeness, extracting data from the same flight, arranging them chronologically, and aligning time parameters. This is the same as the data preprocessing method in Chapter 4. For specific methods, see Section 4.3.3.1.

6.4.2.2 Severe weather data preprocessing and flight data selection

In current aircraft TP tasks, weather information commonly includes ET, VIL features, atmospheric temperature (TMP), and U/V wind components. According to the study in [92], ET data is one of the most indicative of weather's impact on trajectories. Therefore, this study employs ET data as the model input for preprocessing and generating the feature cube.

We generate the weather feature cubes using the method described in Section 6.2. We extract a weather cube ahead of the current position from the original weather file for each trajectory point and rotate it according to the aircraft's heading angle. Weather data is collected one layer above and below the current altitude. By scaling the original resolution of the weather file by a factor of 10, we can generate a $32 \times 32 \times 3$ feature cube for each trajectory point. In this way, the weather bandwidth can cover a sufficient area. Thus, we obtain a set of weather cubes that match the ADS-B trajectory data.

The feature extraction from the weather cubes follows the methods in [92] and [14]. Specifically, we use three convolutional layers and two dense layers, each followed by two dropout layers with a dropout rate of 1%. The convolution kernel sizes are $[6 \times 6, 3 \times 3, 1 \times 1]$, the convolution stride lengths are $[2, 2, 1]$, and the convolution filter sizes are $[1, 2, 4]$. The dense layer sizes are $[16, 3]$. This allows us to obtain weather cube feature data that matches the ADS-B trajectory data.

In this study, we selected flight data under severe weather conditions. First, based on historical weather forecasts from the HKO, we identified dates with unfavorable weather and extracted flight data and corresponding ET (convective

weather) data for those days. Previous studies on convective weather datasets indicate that flight trajectories are categorized as encountering weather deviations only when they experience convective weather at 25,000 feet or for at least 2 minutes [216]. Therefore, we filtered the flight data based on the maximum and average values of the corresponding ET data. The threshold for the ET maximum value is set to 25,000, and the average value threshold is set to 2,000. By this method, we screened flight data under severe weather conditions. Figure 6.3 shows a visualization of the data filtering process. The X-axis represents the data pairs, and the Y-axis represents the maximum and average values of each data pair. The red line shows the thresholds used in different indicators.

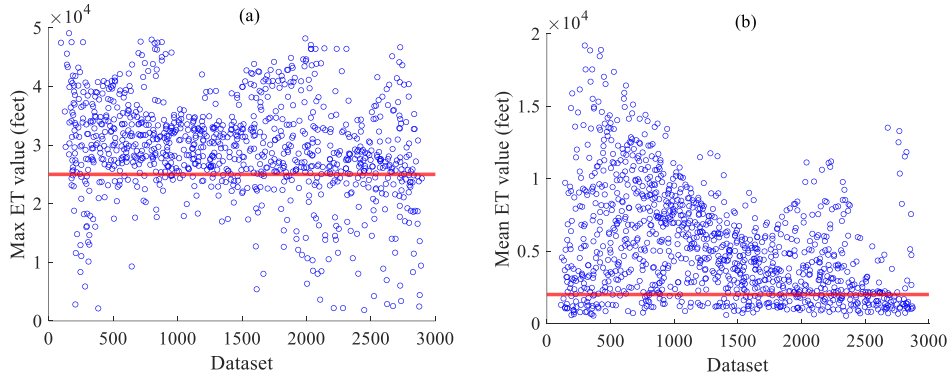


Figure 6.3 Visualization of the data filtering process using thresholds on ET data. (a) represents maximum ET value. (b) represents mean ET value.

6.4.3 Data augmentation methods

6.4.3.1 Improved CTGAN architecture

The architecture of the improved CTGAN for TP tasks under severe weather conditions referenced our previous study [217]. There are two significant differences from the original CTGAN to adapt to the features of ADS-B data.

The first improvement involves the aircraft characteristics contained in ADS-B data. We sample using actual data frequencies rather than logarithmic frequencies. This is done to maintain the diversity of the enhanced data types and prevent the generator from unrealistically oversampling trajectories under severe weather conditions, significantly affecting model convergence [183].

The second improvement is replacing the original CTGAN's one-hot encoding

with leave-one-out encoding to avoid problems of parallelism and multicollinearity in high-dimensional situations, where one-hot encoding can lead to too many and overly sparse generated features.

Unlike previous studies, this research introduces weather cube feature data as conditions. It aims to generate adverse weather data as much as possible during synthetic data generation to enhance data. Figure 6.4 illustrates the architecture of the improved CTGAN, with differences from the original CTGAN marked in red.

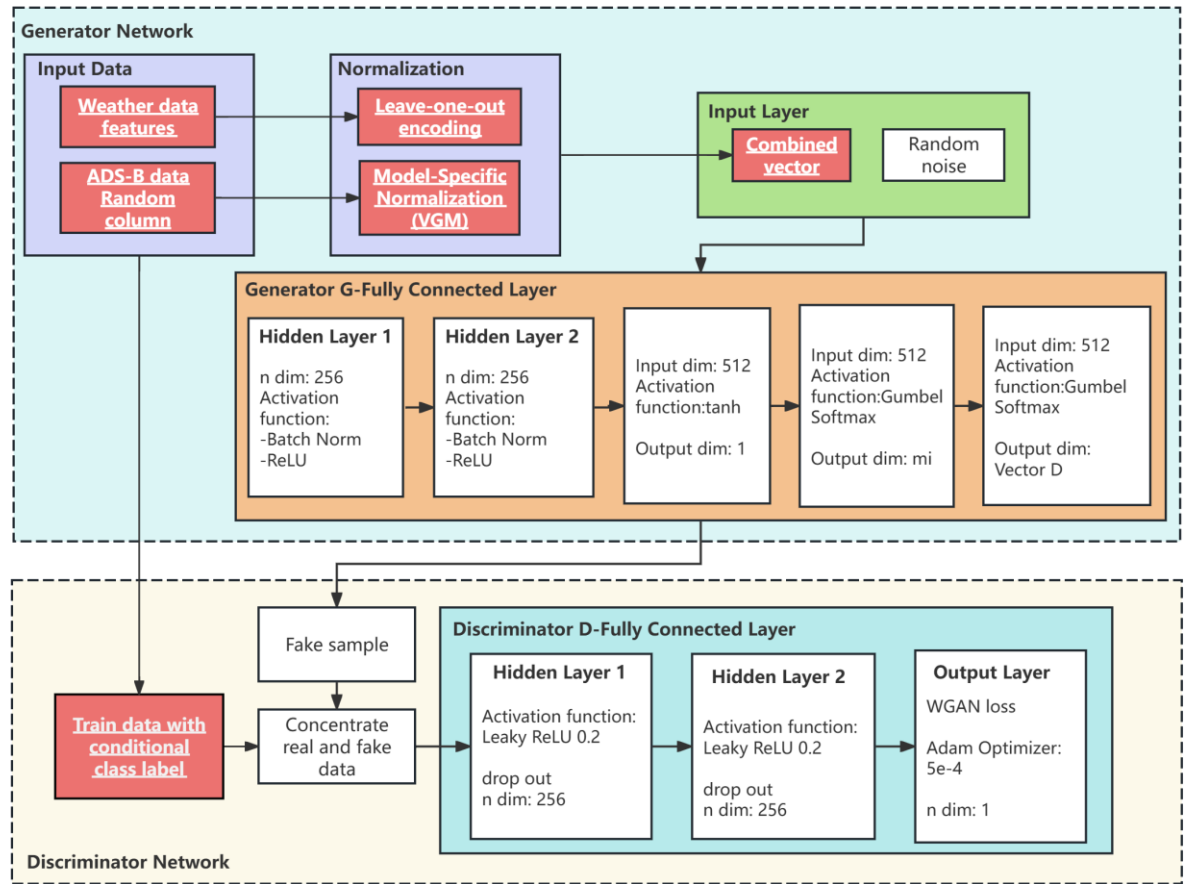


Figure 6.4 The CTGAN architecture considering severe weather conditions.

The improved CTGAN architecture consists of two parts: the generator network and the discriminator network. We perform data enhancement by adding features of aircraft trajectories under severe weather conditions and random noise. In the generator network, the random column vectors of ADS-B data are normalized using a variational GMM. After decoding the weather cube feature data using leave-one-out encoding, it is concatenated with the normalized column vectors to

form a combined vector. This vector and random noise are input into the fully connected generator layer.

In the discriminator network, we use weather cube feature data as a condition to bias the network towards generating aircraft trajectory data under severe weather conditions, achieving the effect of data enhancement. The parameter settings of the entire architecture are the same as in our previous study, with specific parameters shown in Table 6.1. Formulas (38) and (39) represent the generator and discriminator networks, respectively.

Table 6.1 Training parameters for the improved CTGAN.

Parameter	Description	Value
Embedding dimension	Size of the random sample passed to the Generator.	128
Generator dimension	Size of the output samples for each one of the residuals.	(256,256)
Discriminator dimension	Size of the output samples for each one of the Discriminator layers.	(256,256)
Generator learning rate	Learning rate for the generator.	$5 \cdot 10^{-4}$
Generator decay	Generator weight decay for the Adam Optimizer.	$1 \cdot 10^{-6}$
Discriminator learning rate	Learning rate for the discriminator.	$5 \cdot 10^{-4}$
Discriminator decay	Discriminator weight decay for the Adam Optimizer.	$1 \cdot 10^{-6}$
Batch size	Number of data samples to process in each step.	750
Step ratio	Number of discriminator updates to do for each generator update.	1
Epochs	Number of training epochs.	50,000
Pac	Number of samples to group together when applying the discriminator.	10

$$\begin{cases} h_0 = z \oplus \text{cond} \\ h_1 = h_0 \oplus \text{ReLU}\left(\text{BN}\left(\text{FC}_{|\text{cond}|+|z|\rightarrow 256}(h_0)\right)\right) \\ h_2 = h_1 \oplus \text{ReLU}\left(\text{BN}\left(\text{FC}_{|\text{cond}|+|z|+256\rightarrow 256}(h_1)\right)\right) \\ \hat{\alpha}_i = \tanh\left(\text{FC}_{|\text{cond}|+|z|+512\rightarrow 1}(h_2)\right) \quad 1 \leq i \leq N_c \\ \hat{\beta}_i = \text{gumbel}_{0.2}\left(\text{FC}_{|\text{cond}|+|z|+512\rightarrow m_i}(h_2)\right) \quad 1 \leq i \leq N_c \\ \hat{d}_i = \text{gumbel}_{0.2}\left(\text{FC}_{|\text{cond}|+|z|+512\rightarrow |D_i|}(h_2)\right) \quad 1 \leq i \leq N_d \end{cases} \quad (38)$$

$$\begin{cases} h_0 = r_1 \oplus \dots \oplus r_{10} \oplus \text{cond}_1 \oplus \dots \oplus \text{cond}_{10} \\ h_1 = \text{drop}(\text{leaky}_{0.2}(\text{FC}_{10|r|+10|\text{cond}|\rightarrow 256}(h_0))) \\ h_2 = \text{drop}(\text{leaky}_{0.2}(\text{FC}_{256\rightarrow 256}(h_1))) \\ \mathcal{C}(\cdot) = \text{FC}_{256\rightarrow 1}(h_2) \end{cases} \quad (39)$$

where $\oplus$ represents the vector concatenation operation; z represents random noise; $\text{FC}_{u\rightarrow v}(x)$ denotes the application of a linear transformation to a u -dimensional input to obtain a v -dimensional output; $\text{leaky}_\gamma(x)$ represents the application of the leaky ReLU activation function on x with leaky ratio γ ; and $\text{gumbel}_\tau(x)$ represents the application of the Gumbel softmax function with a parameter τ to a vector x . The $\tanh$, BN and ReLU functions are used for batch normalization.

6.4.3.2 Evaluation metrics

During the data augmentation process, the CTGAN treats each row of input tabular data as an independent sample for random sampling without considering the order between rows. It learns the overall distribution characteristics of the data through these samples. Based on the input data and learned distribution characteristics, it outputs generated data in tabular form. For the aircraft TP model, the input to the CTGAN generator is usually actual time series trajectory data. The generated output data consists of non-time-series random data reflecting the distribution characteristics of the trajectories. Therefore, there is no direct one-to-one correspondence between them. For evaluating model training results, if the distributions of two datasets are very close, then the means of their trajectory parameters will also be very close. The mean can approximately reflect the difference between actual and generated distribution. Therefore, we use the mean value to evaluate the model.

According to the requirements of the data augmentation task described in this

paper, evaluation metrics need to meet two conditions: first, to determine the quality of the generated data, ensuring that the data distribution of the new samples is similar to the original data; second, to eliminate the noise introduced by the generated data to avoid the fusion of generated data and original data leading to classification ambiguity. Therefore, to accurately evaluate the model, we use the MMD [191] from the perspective of data distribution and the MED from the standpoint of numerical prediction to calculate the trajectory distance and eliminate the noise introduced by the generated data. These two evaluation metrics are the same as those in Chapter 5; see Section 5.4.3.2 for details.

6.4.4 Model training

During the model training process, the data augmented by CTGAN will be mixed with the processed original ADS-B trajectory data, with a certain proportion of the augmented data representing trajectories under severe weather conditions. This mixture forms a new dataset. The latest mixed dataset will be interpolated at 1-second intervals for learning by LSTM-based models.

This paper uses four commonly used TP models, LSTM, CNN-LSTM, CNN-LSTM-Attention, and CNN-BiLSTM, to learn from the new mixed dataset. This is done to validate the impact of data augmentation on the accuracy of aircraft TP.

In this study, the training objective is to minimize the multi-task loss combining:

1). Trajectory prediction loss

$$\mathcal{L}_{traj} = \frac{1}{N} \sum_{t=1}^N \|\hat{y}_t - y_t\|_2^2 + \lambda \cdot KL(p(\hat{y}_t) \| p(y_t)) \quad (40)$$

where $\hat{y}_t$ and y_t are the predicted and ground-truth coordinates at time t , respectively. The second term introduces Kullback-Leibler divergence to capture uncertainty in dense traffic scenarios (weighted by $\lambda = 0.1$).

2). Auxiliary regularization

To prevent overfitting on short-term patterns, we incorporate a velocity constraint term:

$$\mathcal{L}_{vel} = \frac{1}{N} \sum_{t=2}^N \|(\hat{y}_t - \hat{y}_{t-1}) - (y_t - y_{t-1})\|_1 \quad (41)$$

The final loss is $\mathcal{L} = \mathcal{L}_{traj} + 0.3 \cdot \mathcal{L}_{vel}$.

6.4.5 Trajectory prediction

The goal of 4D TP is to predict the spatiotemporal state sequence of an aircraft through a trained model. If every track point is regarded as S_i , the 4D trajectory is regarded as a random variable P , and we assume that T follows the distribution of $P = P(s_{1:N}; \theta)$, the 4D TP problem can be expressed as $P(s_{1:N}; \theta)$. According to the maximum likelihood estimation, the 4D trajectory distribution is expressed as:

$$S_i = \{t, x, y, z, v_{xy}, h, w\} \quad (42)$$

$$T = \{T_1 = s_{1:N}^1, \dots, T_k = s_{1:N}^k, \dots, T_M = s_{1:N}^M\} \quad (43)$$

$$\underset{\theta}{\operatorname{argmax}} p(D|\theta) = \underset{\theta}{\operatorname{argmax}} \prod_{i=1}^M P(s_{1:N}^i; \theta) \quad (44)$$

where t represents time, x represents latitude, y represents longitude, z represents geometric height, v_{xy} represents ground speed, h represents track angle, and w represents the weather cube; N denotes the number of points in the track and M denotes the number of historical tracks; θ stands for a parameter.

The essence of model training is to learn the distribution of $P(s_{1:N}^i; \theta)$. In this paper, the trained LSTM-based models learn the distribution of $P(s_{1:N}^i; \theta)$. If we input the current aircraft trajectory through the model, we can get a predicted trajectory that corresponds to the input trajectory distribution and follows the time order.

6.5 Experiment and analysis

6.5.1 Data preparation

We chose the flight segment near the HKIA (Hong Kong International Airport) as our experimental data. HKIA is one of the busiest airports in the world, handling many flight takeoffs and landings daily and providing abundant data to verify the performance of prediction models. The airspace structure around Hong Kong Airport is complex, containing multiple inbound and outbound flight routes and corridors, which helps test the model's applicability and accuracy in complex environments. Additionally, located in the subtropical region near the South China Sea, Hong Kong is significantly affected by extreme weather, such as typhoons and heavy rain every year, which substantially impacts aviation traffic. Choosing

Hong Kong Airport allows effective research into the effects of weather factors on aircraft trajectories and optimizing prediction models to improve accuracy under adverse weather conditions. Therefore, selecting flight segments near Hong Kong Airport as experimental data has practical significance and provides rich and complex experimental conditions, contributing to prediction models' comprehensive and reliable performance.

The ADS-B data in this study comes from the HKO and the OpenSky website, covering flight data with severe weather conditions from 2016 to 2022. The corresponding ET data is obtained from the China Meteorological Administration.

In this dataset, flights from the KADLO waypoint to the VHHH waypoint (airport) account for nearly 1/4 of all data, and flights on this route have a flight time of over 1 hour. Therefore, we focused on experiments and analysis on this route. Due to severe weather conditions, many aircraft did not follow the usual flight routes. Their trajectory data deviated significantly from the expected path and sometimes did not pass through essential waypoints. These significantly deviated trajectories are extreme cases and were not included in this study. We chose to retain trajectory data passing through the ABBEY waypoint on this route, excluding trajectories with significant deviations to ensure the quality of experimental data and the objectivity of model predictions. After processing, we selected 200 trajectories that met the criteria. The Figure 6.5 shows a schematic of the trajectory of inbound flights on this route.

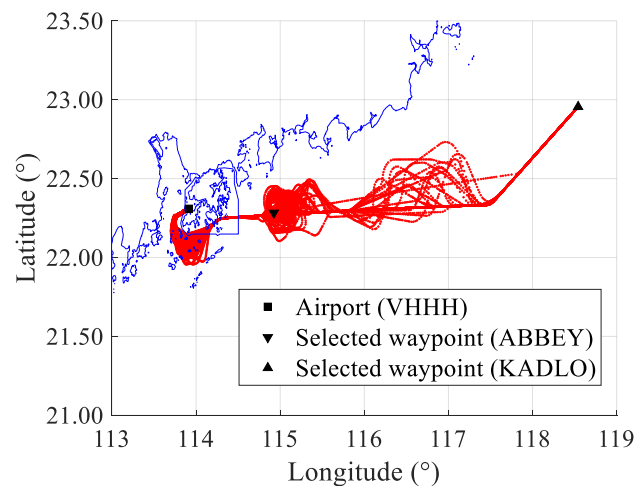


Figure 6.5 Inbound flight trajectories from KADLO to VHHH in the Hong Kong region.

6.5.2 Experimental environment and model hyperparameter settings

The experiment is performed on a PC with Microsoft Windows 10 (CPU: 12th-Gen Intel(R) Core i9-12900KF at 3.20 GHz, RAM: 64.0 GB, GPU: NVIDIA GTX 3080). The CTGAN-based models were built using Keras and TensorFlow. The parameters of the improved CTGAN architecture are based on our previous research shown in Table 6.1.

In this study, we conducted a comparative analysis using MATLAB to run prediction algorithms based on LSTM models. The ratio of the training set to the test set was set to 0.8 and 0.2, respectively. We selected four commonly used neural network architectures for comparison: LSTM, CNN-LSTM, CNN-LSTM-Attention, and CNN-BiLSTM. The specific parameter settings are shown in Table 6.2. With these model parameters, we could compare the performance of different neural network models in predicting aircraft trajectories under severe weather conditions and analyze their effects on the experimental dataset.

Table 6.2 The parameter setting for different LSTM-based models.

Parameter setting	LSTM	CNN-LSTM	CNN-LSTM-Attention	CNN-BiLSTM
Number of hidden layers	100	200	200	100
Training epochs	1000	500	500	200
Learning rate	0.005	0.005	0.005	0.001
Learning rate drop period	800	400	400	160
Learning rate drop factor	0.9	0.9	0.9	0.1
Number of convolution layers	-	2	2	2
Convolution kernel size	-	[3,1]	[3,1]	[3,1]
Number of feature maps in first conv layer	-	32	32	32
Number of feature maps in second conv layer	-	64	64	64
Activation function	-	ReLU	ReLU	ReLU
Minibatch size	-	200	100	200

6.5.3 Model testing and analysis

6.5.3.1 Data augmentation verification based on the improved CTGAN model

Since this study introduces a weather module as a condition that differs from previous research, it is necessary to verify which encoder is optimal for the improved CTGAN model. We used three different encoders, One-hot, Target, and Leave-one-out, and calculated their respective MMD and MED. The results are shown in Table 6.3. Smaller MMD and MED values indicate higher accuracy of the generated trajectories. When both values equal 0, the generated data is identical to the actual data. As shown in Table 6.3, the MMD values for all three encoding methods are the same, indicating that the differences in data distribution between the generated and actual data are consistent across different encoders. However, the MED value for the Leave-one-out encoder is the smallest, suggesting that it produces the most accurate trajectory distances. Therefore, the Leave-one-out encoder is the most effective.

Table 6.3 Evaluation results of the improved CTGAN model under different encoding methods.

Evaluation metrics	One-hot encoding	Target encoding	Leave-one-out encoding
MMD	0.0382	0.0382	0.0382
MED	0.0087	0.0088	0.0086

To intuitively observe the similarity between model-generated and real data distribution, we calculate the probability density distribution using the kernel function as the PDF for all parameters. This difference in probability density distribution visually reflects the distance between the generated and accurate data distribution.

Figure 6.6 shows the PDFs of the generated data for the variables of time, latitude, longitude, geometric height, ground speed, and track angle. The blue line representing the actual data distribution fits well with the red line representing the

generated data distribution. This result indicates that the distributions of the actual data and the generated data are highly similar and consistent with the calculated evaluation metrics. Therefore, the generated data can be used for data augmentation. Figure 6.7 shows the value of the loss function for the generator and discriminator with every 50 epochs for training. It shows that the model has converged.

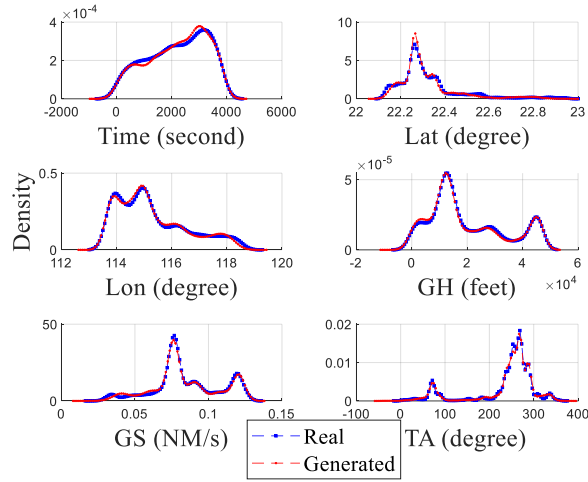


Figure 6.6 Distribution comparison between the generated and real data.

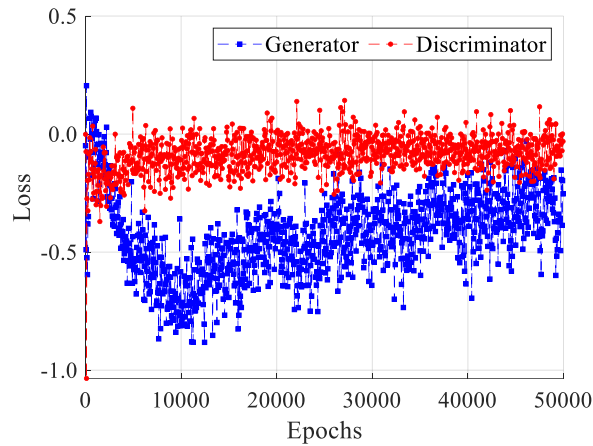


Figure 6.7 Loss function value of the generator and discriminator for training.

6.5.3.2 Model comparison under different data percentages

(1) Comparison of short-term prediction errors for LSTM-based models under severe weather conditions

LSTM-based models tend to be limited by the chaos phenomenon when predicting long time series data [150]. As the prediction period extends, the output

probability density of the expected trajectory tends to become uniform due to the flight uncertainty. Therefore, most LSTM models struggle to provide accurate medium to long-term predictions (prediction periods exceeding 10 minutes). Our comparative experiments only compare the short-term prediction accuracy under severe weather conditions.

We set the sliding window of the LSTM-based model to 40 and the prediction step to 5 minutes. Using our proposed improved CTGAN-LSTMs hybrid model, we enhance the proportion of trajectory data under severe weather conditions according to a specific ratio. We then conduct comparative experiments and calculate the average RMSE of trajectory parameters. The results are shown in Table 6.4 to Table 6.7. "Lat" represents latitude, "Lon" represents longitude, "GH" represents geometric height, "GS" represents ground speed, and "TA" represents track angle.

Table 6.4 RMSE of LSTM model with different data percentages of severe weather conditions.

Data percentage	Lat (degree)	Lon (degree)	GH (feet)	GS (NM/s)	TA (degree)
4.7%	0.0464	0.1947	1665.52	0.00552	63.73
9.1%	0.0412	0.1524	1430.24	0.00547	57.62
13.0%	0.0380	0.1247	1187.58	0.00536	52.47
16.7%	0.0371	0.1184	1202.98	0.00533	53.76
20.0%	0.0351	0.1401	1237.33	0.00541	49.50

Table 6.5 RMSE of CNN-LSTM model with different data percentages of severe weather conditions.

Data percentage	Lat (degree)	Lon (degree)	GH (feet)	GS (NM/s)	TA (degree)
4.7%	0.05568	0.1847	1808.35	0.00994	83.18
9.1%	0.04863	0.1498	1772.69	0.00822	79.03
13.0%	0.04219	0.1282	1704.75	0.00693	74.46
16.7%	0.04130	0.1197	1306.17	0.00669	70.19

20.0%	0.03407	0.1202	1312.64	0.00537	52.39
-------	---------	--------	---------	---------	-------

Table 6.6 RMSE of CNN-LSTM-Attention model with different data percentages of severe weather conditions.

Data percentage	Lat (degree)	Lon (degree)	GH (feet)	GS (NM/s)	TA (degree)
4.7%	0.05526	0.1849	2095.77	0.00957	85.65
9.1%	0.05081	0.1759	1963.36	0.00848	80.39
13.0%	0.04621	0.1423	1644.99	0.00764	75.75
16.7%	0.04001	0.1243	1458.42	0.00665	71.29
20.0%	0.04070	0.1154	1445.02	0.00699	70.84

Table 6.7 RMSE of CNN-BiLSTM model with different data percentages of severe weather conditions.

Data percentage	Lat (degree)	Lon (degree)	GH (feet)	GS (NM/s)	TA (degree)
4.7%	0.05547	0.1807	2165.73	0.00953	84.75
9.1%	0.05135	0.1622	1912.20	0.00874	80.92
13.0%	0.04595	0.1502	1728.44	0.00776	76.42
16.7%	0.04064	0.1268	1454.66	0.00680	72.02
20.0%	0.04036	0.1246	1419.62	0.00678	70.34

To more intuitively compare the impact of data augmentation on the prediction accuracy of aircraft trajectories under severe weather conditions, we plotted the data from Table 6.4 to Table 6.7 into line graphs based on latitude, longitude, geometric height, ground speed, and track angle, as shown in Figure 6.8. These charts allow us to compare the average RMSE of different models under various data percentage conditions, illustrating the effect of data augmentation on short-term prediction tasks under severe weather conditions.

Overall, data augmentation significantly improved the prediction accuracy of all models under severe weather conditions. As the proportion of augmented data

increased, the average RMSE of all models decreased. For CNN-LSTM, CNN-LSTM-Attention, and CNN-BiLSTM models, the CNN-LSTM model performed the best across most parameters, especially in predicting longitude, geometric height, and ground speed. Although introducing attention mechanisms and bidirectional LSTM improved performance for specific parameters, the overall performance of the CNN-LSTM model remained superior.

When comparing CNN-LSTM and LSTM models, despite the CNN-LSTM model showing more significant accuracy improvements with increased data proportions—especially in longitude, geometric height, and ground speed predictions—the LSTM model performed well and stably under minor sample conditions for some indicators. Therefore, while the CNN-LSTM model may perform better on comprehensive metrics after data augmentation reaches a certain percentage, using the LSTM model can yield better prediction results when handling minority-class sample severe weather aircraft TP tasks.

From Figure 6.8, we can observe that for the LSTM model, the average RMSE of each model decreased with increasing data proportions. The prediction accuracy improvement, around 13%, was the most significant in data augmentation. Specifically, when the data proportion increased from 4.7% to 13%, the RMSE for latitude (Lat) decreased by about 18.10%, longitude (Lon) by approximately 35.93%, geometric height (GH) by around 28.68%, ground speed (GS) by the smallest margin of about 2.90%, and track angle (TA) by about 17.68%. On average, the RMSE decreased by about 20.46%, according to the LSTM model's results. This indicates that increasing the data volume positively impacts model performance, enabling more accurate predictions of target variables.

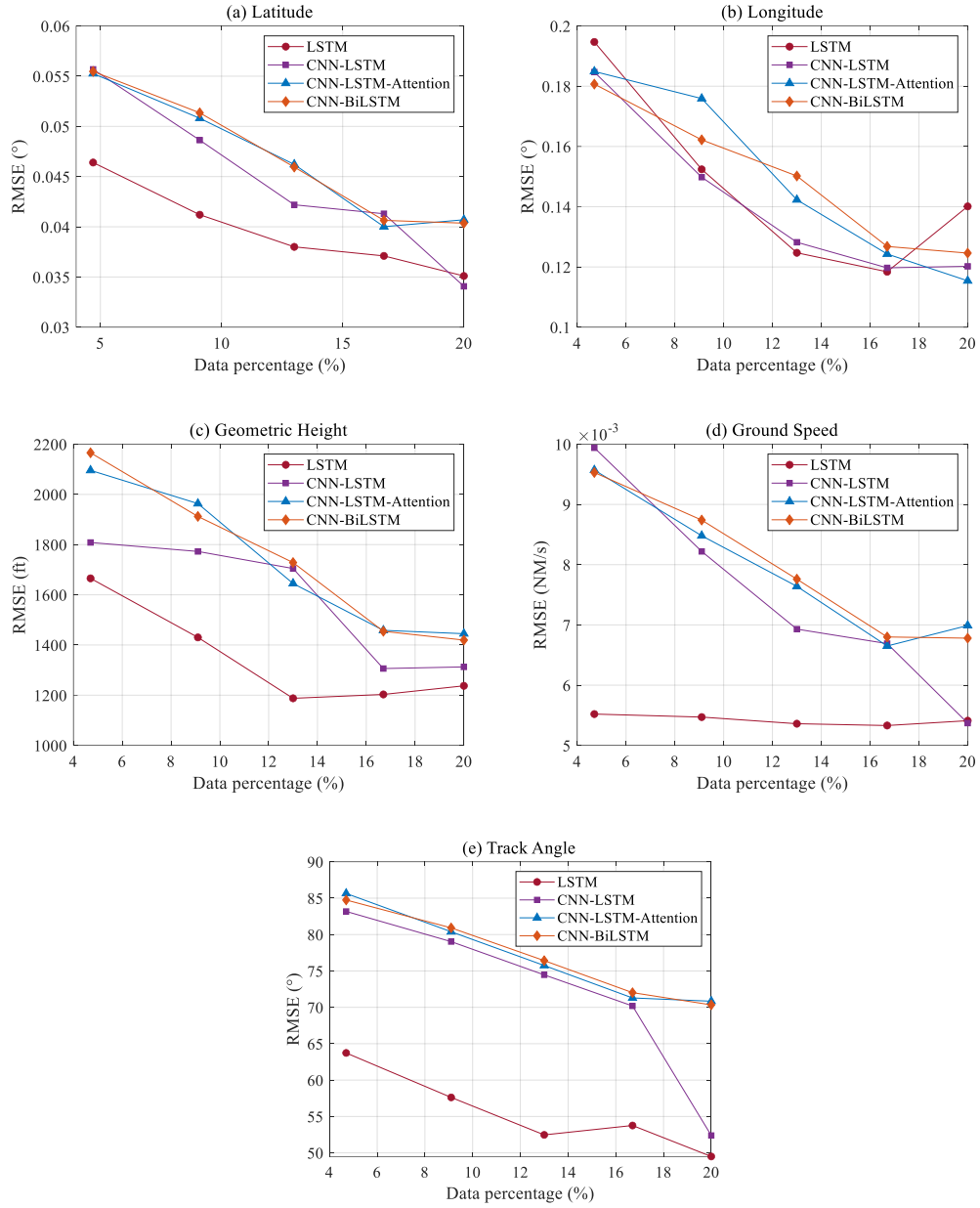


Figure 6.8 Comparison of the prediction RMSE of the LSTM, CNN-LSTM, CNN-LSTM-Attention, and CNN-BiLSTM models for different data percentages.

(2) Visual analysis of typical trajectory prediction

To intuitively compare the effect of our proposed improved CTGAN-LSTM framework on TP tasks under severe weather conditions, we selected flight CPA475 as an example to analyze the TP parameter differences of the LSTM model. Specifically, we analyzed and visually compared the changes in data proportions of 4.7% and 13%. The results are shown in Figure 6.9. For the LSTM model, when improved CTGAN enhanced the minority-class sample trajectory data under severe weather conditions, the prediction accuracy for all parameters

saw significant improvement. This result is consistent with the findings shown in Table 6.4.

We also calculated the mean horizontal and vertical errors to show the difference further. It should be noted that this comparative experiment uses a prediction time span of 5 minutes, which causes an apparent vertical error. However, this error will be significantly reduced if we shorten the prediction time span.

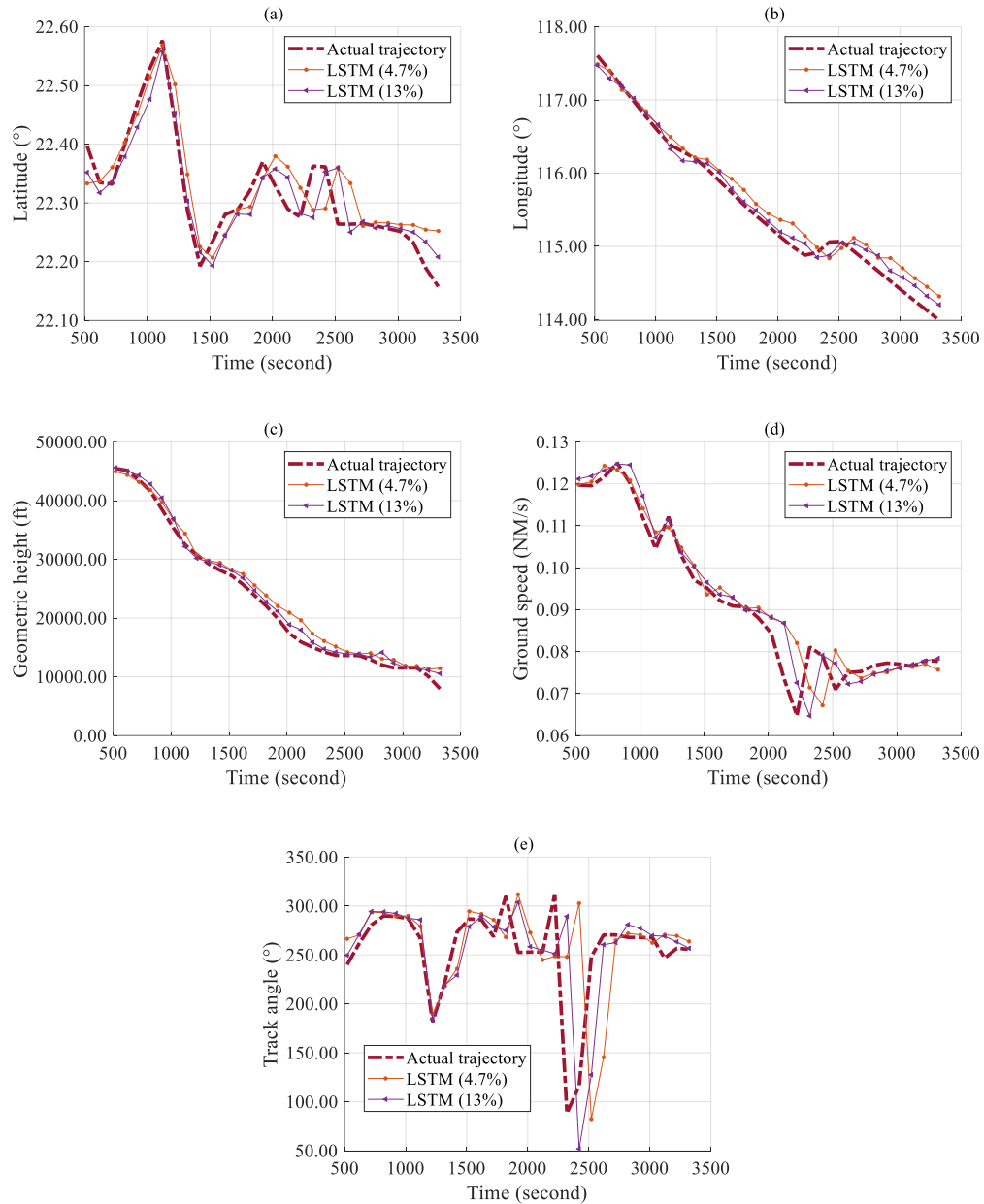


Figure 6.9 Comparison of the prediction results of the LSTM model for flight CPA475.

Table 6.8 Comparison of mean horizontal and vertical errors of the LSTM model for flight CPA475. The prediction time span is 5 minutes.

Data	Mean horizontal errors	Mean vertical errors
percentage	(nmi)	(m)
4.7%	9.8422	413.14
13.0%	6.4487	269.73

6.5.3.3 Comparison with SMOTE data augmentation

The quality of synthetic data is a crucial factor influencing the effectiveness of data augmentation. Therefore, it is necessary to compare the quality of data generated by the improved CTGAN with that generated by the commonly used SMOTE technique. Considering that the LSTM model yields better prediction results when handling minority-class sample TP tasks under severe weather conditions, the comparative experiment was conducted using the LSTM model with the same settings as in Table 6.2. Here, the number of nearest neighbors for SMOTE was set to 5.

The MMD for the enhanced data generated is 0.0442, and the MED is 0.00075. Compared to the improved CTGAN model (MMD is 0.0382, MED is 0.0087), the data generated by SMOTE has a larger MMD but a more petite MED. This indicates that from the data distribution perspective, the data generated by the improved CTGAN is closer to the actual data distribution. However, regarding trajectory point distance, the data generated by SMOTE has a more minor error.

Table 6.9 shows the impact of different data percentages on the average RMSE of five variables predicted by the LSTM model after using the SMOTE data augmentation technique. Lat represents latitude, Lon represents longitude, GH represents geometric height, GS represents ground speed, and TA represents track angle.

Table 6.9 RMSE of LSTM model with different data percentages using SMOTE data augmentation.

Data	Lat	Lon	GH	GS	TA
percentage	(degree)	(degree)	(feet)	(NM/s)	(degree)

4.7%	0.0464	0.1947	1665.52	0.00552	63.73
9.1%	0.0515	0.1842	1434.17	0.00591	72.76
13.0%	0.0434	0.1507	1314.68	0.00552	62.87
16.7%	0.0418	0.1706	1326.72	0.00493	60.03
20.0%	0.0434	0.1255	1489.81	0.00553	61.65

For latitude, when the data percentage increases from 4.7% to 9.1%, the RMSE increases from 0.0464 to 0.0515. As the data percentage rises to 13.0% and 16.7%, the RMSE decreases to 0.0434 and 0.0418, respectively. However, when the data percentage increases to 20%, the RMSE rises to 0.0434 again. The RMSE shows a trend of first increasing, then decreasing and slightly increasing, indicating that increasing the data volume and using SMOTE can improve the model's prediction performance for latitude to a certain extent, but there are fluctuations.

For longitude, when the data percentage increases from 4.7% to 13.0%, the RMSE decreases rapidly from 0.1947 to 0.1507. As the data percentage increases to 16.7%, the RMSE rises to 0.1706. However, as the data percentage increases to 20%, the RMSE decreases again to 0.1255. The RMSE shows a general downward trend as the data percentage increases, but there are significant fluctuations at specific points. These fluctuations may be due to SMOTE's tendency to lose authenticity when handling long time-series data, which in turn affects the performance of the LSTM model.

For geometric height, when the data percentage increases from 4.7% to 13%, the RMSE decreases rapidly from 1665.52 to 1314.68. As the data percentage rises to 16.7%, the RMSE slightly increases to 1326.72, and when the data percentage increases to 20%, the RMSE significantly rises to 1489.81. This indicates that in the early stages of growing data volume, the model's prediction accuracy improves significantly, but after reaching a particular data volume, the prediction accuracy decreases. This may be caused by SMOTE's tendency to generate data that converges locally.

For ground speed, the RMSE trend is similar to that of latitude, showing a trend of first increasing, then decreasing, and slightly increasing. Specifically, at 4.7%, 13.0%, and 20% data percentages, the RMSE is very close, being 0.00552,

0.00552, and 0.00553, respectively. When the data percentage is 9.1%, the RMSE increases to 0.00591. At 16.7%, the RMSE decreases to the lowest value of 0.00493. It can be seen that SMOTE augmented data did not significantly improve the LSTM model's prediction performance for GS, especially when the data percentage is 9.1%, the RMSE increases. This indicates that increasing the data volume and using SMOTE have limited effects on the prediction of ground speed and may even cause performance fluctuations and deterioration in some cases.

For track angle (TA), the RMSE trend is the same as that of ground speed. It can be seen that SMOTE augmented data did not significantly improve the LSTM model's prediction performance for track angle.

To visually compare the difference between improved CTGAN and SMOTE for data augmentation, we plotted the RMSE data from Table 6.4 and Table 6.9 for latitude, longitude, geometric height, ground speed, and track angle as line graphs, as shown in Figure 6.10. It can be observed that, in most cases, the improved CTGAN performs significantly better than SMOTE.

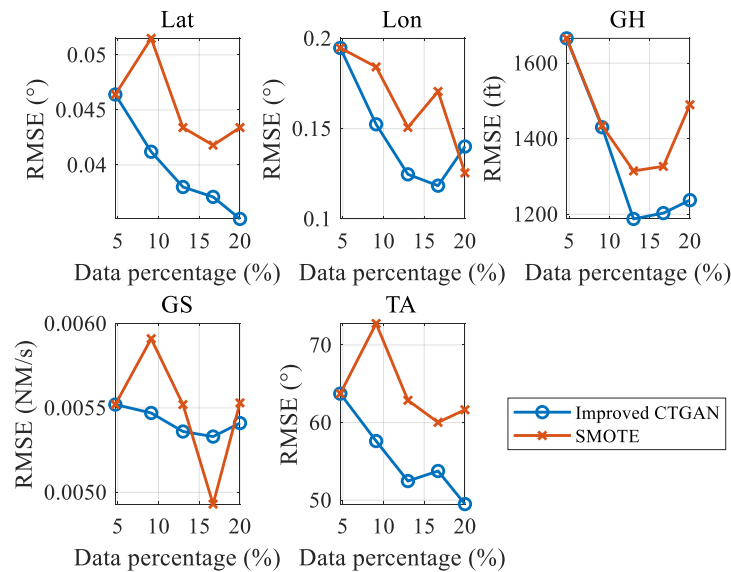


Figure 6.10 RMSE values of different parameters for various data percentages using improved CTGAN and SMOTE data augmentation techniques.

In summary, for minority-class sample trajectory data under severe weather conditions, using the SMOTE data augmentation technique and increasing the data percentage can improve the prediction accuracy of LSTM models for some

variables. However, there is inevitable fluctuation in the RMSE changes for certain variables, and increasing the data volume leads to performance degradation at specific data percentages. SMOTE-enhanced data did not significantly improve model performance and even showed performance degradation at some data percentages, particularly for ground speed and track angle. Compared to the improved CTGAN, although SMOTE has higher precision in generating trajectories, it is not a suitable data augmentation model for minority-class sample trajectory data under severe weather conditions. The root cause may be that SMOTE performs data augmentation based on the uniform distribution assumption rather than the actual distribution. At the same time, the improved CTGAN generates synthetic data using the distribution learned from samples.

6.5.4 Experimental conclusion

(1) Trajectory data under severe weather conditions occupies a small proportion in aircraft trajectory sample data, leading to data imbalance. Current deep neural networks are usually trained using multiple rounds of small-batch random sampling of sample data, which cannot effectively solve the data imbalance problem, and aircraft TP under severe weather conditions is challenging. The proposed aircraft TP framework, which includes a minority-class sample data enhancement network and a prediction network, achieves significant improvements in short-term aircraft TP accuracy under severe weather conditions, demonstrating the effectiveness of the improved CTGAN-LSTMs model. Experimental results show that when the proportion of minority-class sample data increases from 4.7% to 13%, the average RMSE of the LSTM model for 5-minute short-term prediction decreases by 20.46%. Based on this architecture, other data augmentation and prediction networks can be selected according to specific application characteristics to replace the improved CTGAN and LSTM-based models, forming new hybrid models. The idea and method proposed in this paper to solve the problem of sample data imbalance through data enhancement can provide a reference for solving other data imbalance problems in aircraft TP tasks. Given the significant impact of severe weather on ATMs, this study is helpful to enhance ATM capabilities.

(2) We compared four commonly used LSTM-based prediction models (LSTM, CNN-LSTM, CNN-LSTM-Attention, and CNN-BiLSTM) to evaluate the

performance of the proposed the improved CTGAN-LSTM hybrid model. Experimental results show that the improved CTGAN-LSTMs hybrid model can significantly improve the accuracy of TP under severe weather conditions. The LSTM model performs best for minority-class sample data, with trajectory prediction accuracy substantially better than other comparative models. Before the proportion of minority-class sample data reaches 13%, the accuracy indicators increase rapidly with the proportion, and after gaining 16.7%, the prediction accuracy stabilizes.

(3) We compared the improved CTGAN with the commonly used data augmentation model SMOTE to assess the impact of different data augmentation methods on aircraft TP tasks for minority-class sample data under severe weather conditions. Experimental results show that the improved CTGAN is more suitable as a data augmentation method for TP of minority-class sample data compared to SMOTE. When using the LSTM model for 5-minute prediction with a data percentage of 13%, the average RMSE decreases by 11.76%. Although SMOTE has higher synthetic trajectory accuracy than the improved CTGAN, it is not a good model for data augmentation of minority-class sample trajectories under severe weather conditions. This also reveals an exciting issue: synthetic data distribution characteristics seem more critical than data accuracy for trajectory data augmentation.

In summary, data augmentation of minority-class samples can effectively improve aircraft TP accuracy under severe weather conditions without the need to adjust or enhance the proven prediction model. Different data augmentation models for minority-class samples can significantly affect the training and prediction of TP models based on generated trajectories (or generated trajectories mixed with original trajectories). The ideas and methods proposed in this article to solve the problem of sample data imbalance through data enhancement provide a valuable reference for solving other data imbalance problems in aircraft TP. The proposed improved CTGAN-LSTMs hybrid model is only a verification method for this method, and the corresponding hybrid model can be used according to specific application scenarios.

6.6 Chapter summary

This chapter addresses the challenges of applying weather data in aircraft TP and outlines the following key contributions:

1). Minority-class sample data imbalance in severe weather conditions: It identifies that the main challenge in TP during severe weather is the limited availability of relevant data. Severe weather significantly impacts aircraft flight, making it a critical area of study. However, weather-related trajectory data forms a small portion of available samples, leading to an imbalance that complicates accurate TPs.

2). Representation of weather data in deep learning: The chapter analyzes the methods for representing and applying weather data in deep learning models for TP. This includes how weather elements are encoded and the concept of a "weather cube," which encapsulates weather information and is widely used in TP models.

3). Addressing minority-class sample data issues: It explores the characteristics and challenges of working with minority-class sample data in TP. Specifically, the chapter highlights that data augmentation for such cases must consider spatial distribution properties to ensure enhanced data maintains realistic characteristics relevant to flight paths.

4). Data augmentation using improved CTGAN for severe weather prediction: A novel approach using a hybrid improved CTGAN-LSTM model is proposed to handle the imbalance of minority-class sample data. The improved CTGAN is employed to augment underrepresented weather-related trajectory samples, while the LSTM network is used to predict and assess the impact of this enhanced data. The model improves TP accuracy and serves as a framework for handling similar data imbalance issues in aircraft TP.

5). Comprehensive experimental validation: Extensive comparative experiments are conducted to validate the effectiveness of the improved CTGAN-LSTM model. These include comparisons with other LSTM-based models (e.g., LSTM, CNN-LSTM, CNN-LSTM-attention, CNN-BiLSTM) and the improved CTGAN with the SMOTE data augmentation technique. The results demonstrate that the

CTGAN-LSTM hybrid model improves TP accuracy in severe weather conditions. The experiments further emphasize that the choice of data augmentation technique can significantly influence the performance of TP models.

This work thoroughly analyzes how to address severe weather TP challenges by augmenting minority-class sample data and offers a practical solution using an improved CTGAN-LSTM hybrid model.

Chapter 7.

Summary and outlook

7.1 Main research work of this thesis

This research comprehensively explored and validated the application of a novel deep learning framework, centered on the CTGAN, for aircraft trajectory prediction. The proposed technical framework, encompassing system architecture, model construction, and specialized evaluation metrics, has proven effective across various challenging scenarios. The core findings, substantiated by key quantitative results, are summarized as follows:

1). A highly stable and data-efficient core model for 4D TP: The developed CTGAN model demonstrates exceptional performance in generating high-fidelity 4D trajectories. On a key test route, it achieved remarkable similarity to actual trajectories, with a MED of only **0.0085 nautical miles**, alongside low distributional errors (JS divergence: **0.0539**; MMD: **0.0150**). The model exhibits high stability and low data demand, maintaining performance with as few as **5 training flights** and retaining high fidelity even with **80% data thinning**.

2). Significant accuracy gains in medium- and long-term prediction through a hybrid approach: The C-CTGAN model significantly outperforms both traditional and contemporary benchmarks. Compared to the best-performing LSTM variant (CNN-LSTM), it reduced the MAE in predicting critical parameters—**latitude, geometric altitude, and ground speed**—by **69.89%, 74.07%, and 84.21%**, respectively. This demonstrates that the integration of clustering effectively enhances the model's capability for complex, long-horizon predictions.

3). Effective mitigation of data imbalance for severe weather scenarios: The proposed data augmentation framework successfully addresses the challenge of imbalanced data in severe weather conditions. By leveraging an improved CTGAN to augment minority-class samples, the short-term prediction accuracy

(5-minute RMSE) for a standard LSTM model was boosted by **20.46%**. Furthermore, the CTGAN-based augmentation outperformed the commonly used SMOTE method, yielding an **11.76% lower RMSE**, highlighting the importance of realistic data distribution over mere synthetic accuracy.

In summary, this research moves beyond a proof-of-concept and delivers a quantitatively validated, practical framework. The CTGAN-based models have been shown to offer superior prediction accuracy, robust stability, and data efficiency, making them a highly promising foundation for developing next-generation trajectory prediction capabilities within the Trajectory-Based Operations framework. The findings effectively bridge the gap between advanced deep learning techniques and the rigorous demands of air traffic management.

7.2 Main innovations of the thesis

1). Proposed a deep learning-based TP technical framework, providing a reference framework for system construction and application

Given the lack of systematic research and insufficient practical application of deep learning-based TP technology, a TP system was proposed. This system considers fundamental information system components, the TBO-based general TP model architecture, and the integrated application needs of ML-based TP models. It comprises essential business processing units like data preprocessing, data updates, prediction output, TP models, trajectory integration and fusion, and auxiliary units for system operation such as scheduling control, data management, and input data reading. A particular focus was placed on the integrated application framework for trajectory prediction models and the descriptive information required for model sharing. Regarding model training and validation characteristics in deep learning-based TP systems, a system construction workflow and evaluation metric framework were proposed, with particular attention to prediction accuracy as the core metric and its associated indicators and calculation methods. Based on a study of the characteristics of deep learning TP models, typical structural patterns and methods for determining model parameters were introduced.

2). Introduced CTGAN into the TP field, offering a new option for medium- and long-term aircraft TP

CTGAN, a recent advancement in GAN technology, has shown practical value in applications such as data synthesis, augmentation, and cybersecurity. Based on an analysis of CTGAN's advantages, which align well with the characteristics of aircraft trajectory data, comparative experiments on prediction accuracy, stability, and adaptability of sample data were conducted. The results confirmed CTGAN's potential for aircraft TP. Subsequently, a new hybrid model combining K-means clustering and the CTGAN architecture was designed, optimizing the CTGAN network to create a new TP model. This model demonstrated a significant improvement in prediction accuracy compared to typical models, especially when the prediction time span exceeded 10 minutes, providing a new hybrid model option for medium- and long-term aircraft TP.

3). Applied data augmentation methods to address imbalanced TP data, providing a new solution for minority-class sample data

Weather conditions, especially severe ones like convective weather and severe turbulence, significantly impact flight safety and are major causes of trajectory deviations. Such weather-related flight data typically constitutes a small proportion of the overall sample data, classifying it as a minority-class sample, which leads to data imbalance. Data-driven methods often treat minority-class samples as anomalies or fail to capture their characteristics due to insufficient sample size, resulting in reduced prediction accuracy. Given the characteristics of aircraft trajectory data and CTGAN's exceptional performance in data augmentation, a hybrid model combining CTGAN and LSTM and an optimized CTGAN network was designed. TP tests were conducted through data augmentation using CTGAN to enhance minority-class samples, significantly improving prediction accuracy. Compared to the classic data augmentation method SMOTE, CTGAN proved more suitable for aircraft TP. This research demonstrates that data augmentation is a practical approach for solving the TP data imbalance problem, offering a new method for predicting minority-class sample data.

7.3 Research directions and prospects

Building upon the findings of this research, future work will evolve along two

parallel tracks: the technical refinement of the prediction models themselves, and the strategic expansion of their application to support the future of autonomous and advanced ATM.

1). Technical refinement of prediction models

Knowledge transfer and lifelong learning for evolving airspaces: The dependency on route-specific models, as observed in our experiments where accuracy decreased without distinguishing trajectory modes, highlights an efficiency bottleneck. Future research will explore transfer learning and meta-learning techniques. The goal is to enable a core model, like the CTGAN framework we developed, to efficiently and reliably adapt its knowledge to new routes, aircraft types (especially uncrewed aerial vehicles (UAV) and supersonic transports), and operational contexts without exhaustive retraining.

Integrated multi-factor modeling: While historical trajectories implicitly contain information on various factors, future models must evolve to explicitly and dynamically incorporate a wider set of environmental and operational constraints. This includes air traffic congestion, airspace structure, and airline decisions, moving beyond weather-centric models to achieve a holistic understanding of trajectory determinants.

Verifiable reliability and explainable AI for operational trust: For the adoption of data-driven models like ours in safety-critical ATM, building trust is paramount. Research is needed to develop rigorous reliability verification frameworks and explainable AI techniques. Quantifying prediction uncertainty and providing intuitive explanations for model outputs are essential prerequisites for human operators to accept and act upon AI-generated predictions in daily operations.

2). Expansion towards autonomous systems and advanced applications

AI-Powered strategic Conflict Resolution: The high-fidelity, probabilistic trajectories generated by models like our improved CTGAN can serve as the foundation for next-generation, AI-driven conflict detection and resolution systems. Future work will explore using these predictions not just for situational awareness, but as direct input to automated tools that can propose strategic deconfliction maneuvers to controllers or directly to autonomous aircraft.

Application in urban air mobility (UAM) and advanced air mobility (AAM): The low-altitude, high-density, and network-structured operations envisioned for UAM present a novel and demanding prediction challenge. The ability of generative models to learn complex distributions from data, a key strength of the CTGAN framework validated in this study, makes them ideally suited for predicting the trajectories of air taxis and autonomous cargo drones in congested urban corridors.

Integration with large-scale digital twins for ATM: The proposed TP framework can serve as a core component of a large-scale, real-time digital twin of the airspace. By simulating and predicting the behavior of all aircraft under various scenarios, this digital twin would become an invaluable tool for strategic planning, stress-testing new procedures, and training next-generation AI-based air traffic management systems.

Cross-domain synergies: The methodologies developed for data augmentation and handling imbalanced datasets in this research have potential applications beyond traditional aviation. The core ideas could be adapted to challenges in autonomous vehicle trajectory prediction and maritime traffic management, opening avenues for cross-domain research collaboration and knowledge transfer.

By pursuing these directions, the research community can transition trajectory prediction from a supportive tool to a core, enabling technology for the autonomous, highly efficient, and safe air transportation system of the future.

References

- [1] International Civil Aviation Organization, “2016-2030 Global air navigation plan.” ICAO, 2019.
- [2] E. Schulz, “Global Networks, Global Citizens. Global Market Forecast 2018–2037,” Airbus, Blagnac Cedex, France, 2018.
- [3] International Civil Aviation Organization, “2013-2028 Global Air Navigation Plan - Doc 9750-AN/963. 4th edition.” ICAO, 2013.
- [4] K. Legrand, “Correction and optimization of 4D aircraft trajectories by sharing wind and temperature information,” Ph.D. dissertation, INSA de Toulouse, Toulouse, France, 2019.
- [5] International Civil Aviation Organization, “Why Global TBO Concept.” ICAO. Accessed: Jul. 11, 2024. [Online]. Available: <https://www.icao.int/airnavigation/tbo/Pages/Why-Global-TBO-Concept.aspx>
- [6] J. Garcia-Chico, R. Vivona, and K. Cate, “Characterizing intent maneuvers from operational data: Step towards trajectory prediction uncertainty estimation,” presented at the AIAA Guidance, Navigation and Control Conference and Exhibit, 2008, p. 6520.
- [7] A. Degas *et al.*, “A survey on artificial intelligence (ai) and explainable ai in air traffic management: Current trends and development with future research trajectory,” *Applied Sciences*, vol. 12, no. 3, p. 1295, 2022.
- [8] Organisation de l’aviation civile internationale, *Global air traffic management operational concept*. ICAO, 2005.
- [9] E. FAA, “Common trajectory prediction structure and terminology in support of SESAR & NextGen,” Technical Report. USA: FAA/EUROCONTROL AP 16, 2010.
- [10] FAA, “A report from the portable electronic devices (PED) aviation rule making committee (ARC) to the Federal Aviation Administration,” Federal Aviation Administration, 2013.
- [11] G. N. Lui, T. Klein, and R. P. Liem, “Data-driven approach for aircraft arrival flow investigation at terminal maneuvering area,” presented at the AIAA Aviation 2020 Forum, 2020, p. 2869.
- [12] W. Zeng, Z. Quan, Z. Zhao, C. Xie, and X. Lu, “A deep learning approach for aircraft trajectory prediction in terminal airspace,” *IEEE Access*, vol. 8, pp. 151250–151266, 2020.
- [13] Z. Wang, M. Liang, and D. Delahaye, “Short-term 4d trajectory prediction using machine learning methods,” presented at the SID 2017, 7th SESAR Innovation Days, 2017.
- [14] Y. Pang, X. Zhao, H. Yan, and Y. Liu, “Data-driven trajectory prediction with weather uncertainties: A Bayesian deep learning approach,” *Transportation Research Part C: Emerging Technologies*, vol. 130, p. 103326, 2021.
- [15] S. Mondoloni and N. Rozen, “Aircraft trajectory prediction and synchronization for air traffic management applications,” *Progress in* 246

aerospace sciences, vol. 119, p. 100640, 2020.

- [16] W. Zeng, X. Chu, Z. Xu, Y. Liu, and Z. Quan, "Aircraft 4D trajectory prediction in civil aviation: A review," *Aerospace*, vol. 9, no. 2, p. 91, 2022.
- [17] W. Schuster, M. Porretta, and W. Ochieng, "High-accuracy four-dimensional trajectory prediction for civil aircraft," *The Aeronautical Journal*, vol. 116, no. 1175, pp. 45–66, 2012.
- [18] International Civil Aviation Organization, "Performance-based navigation (PBN) manual," ICAO, 2013.
- [19] J. Bronsvort, "Contributions to trajectory prediction theory and its application to arrival management for air traffic control," Ph.D. dissertation, Universidad Politécnica de Madrid, Madrid, Spain, 2014.
- [20] T. Zeh, J. Rosenow, and H. Fricke, "Bayesian Inference of Aircraft Operating Speeds for Stochastic Medium-Term Trajectory Prediction," presented at the 2023 IEEE/AIAA 42nd Digital Avionics Systems Conference (DASC), IEEE, 2023, pp. 1–10.
- [21] X. Wu, H. Yang, H. Chen, Q. Hu, and H. Hu, "Long-term 4D trajectory prediction using generative adversarial networks," *Transportation Research Part C: Emerging Technologies*, vol. 136, p. 103554, 2022.
- [22] R. Slattery and Y. Zhao, "Trajectory synthesis for air traffic automation," *Journal of guidance, control, and dynamics*, vol. 20, no. 2, pp. 232–238, 1997.
- [23] J. Zhang et al., "Data-driven flight time prediction for arrival aircraft within the terminal area," *IET Intelligent Transport Systems*, vol. 16, no. 2, pp. 263–275, 2022.
- [24] M. G. Hamed, D. Gianazza, M. Serrurier, and N. Durand, "Statistical prediction of aircraft trajectory: regression methods vs point-mass model," presented at the ATM 2013, 10th USA/Europe Air Traffic Management Research and Development Seminar, 2013, p. pp-xxxx.
- [25] E. C. de Souza, C. C. Fattori, and S. Silva, "Improved Trajectory Prediction and Simplification Analysis for ATM System Modernization," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 15627–15632, 2017.
- [26] R. Alligier, D. Gianazza, and N. Durand, "Learning the aircraft mass and thrust to improve the ground-based trajectory prediction of climbing flights," *Transportation Research Part C: Emerging Technologies*, vol. 36, pp. 45–60, 2013.
- [27] J. Zhang, J. Liu, R. Hu, and H. Zhu, "Online four dimensional trajectory prediction method based on aircraft intent updating," *Aerospace Science and Technology*, vol. 77, pp. 774–787, 2018.
- [28] X. Zhang and S. Mahadevan, "Bayesian neural networks for flight trajectory prediction and safety assessment," *Decision Support Systems*, vol. 131, p. 113246, 2020.
- [29] I. Lymperopoulos, J. Lygeros, and A. Lecchini, "Model based aircraft trajectory prediction during takeoff," presented at the Aiaa guidance, navigation, and control conference and exhibit, 2006, p. 6098.
- [30] W. Schuster, W. Ochieng, and M. Porretta, "High-performance trajectory prediction for civil aircraft," presented at the 29th Digital Avionics Systems Conference, IEEE, 2010, pp. 1–C.
- [31] M. Porretta, M.-D. Dupuy, W. Schuster, A. Majumdar, and W. Ochieng, "Performance evaluation of a novel 4D trajectory prediction model for civil aircraft," *The Journal of Navigation*, vol. 61, no. 3, pp. 393–420, 2008.
- [32] A. Nuic, D. Poles, and V. Mouillet, "BADA: An advanced aircraft

- performance model for present and future ATM systems,” *International journal of adaptive control and signal processing*, vol. 24, no. 10, pp. 850–866, 2010.
- [33] D. P. Thippavong, C. A. Schultz, A. G. Lee, and S. H. Chan, “Adaptive algorithm to improve trajectory prediction accuracy of climbing aircraft,” *Journal of Guidance, Control, and Dynamics*, vol. 36, no. 1, pp. 15–24, 2013.
 - [34] G. B. Chatterji, “Fuel burn estimation using real track data,” presented at the 11th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference, including the AIAA Balloon Systems Conference and 19th AIAA Lighter-Than, 2011, p. 6881.
 - [35] T. Baklacioglu and M. Cavcar, “Aero-propulsive modelling for climb and descent trajectory prediction of transport aircraft using genetic algorithms,” *The Aeronautical Journal*, vol. 118, no. 1199, pp. 65–79, 2014.
 - [36] J. Sun, J. Ellerbroek, and J. M. Hoekstra, “WRAP: An open-source kinematic aircraft performance model,” *Transportation Research Part C: Emerging Technologies*, vol. 98, pp. 118–138, 2019.
 - [37] R. Alligier and D. Gianazza, “Learning aircraft operational factors to improve aircraft climb prediction: A large scale multi-airport study,” *Transportation research part C: emerging technologies*, vol. 96, pp. 72–95, 2018.
 - [38] X. Tang, P. Chen, and Y. Zhang, “4D trajectory estimation based on nominal flight profile extraction and airway meteorological forecast revision,” *Aerospace Science and Technology*, vol. 45, pp. 387–397, 2015.
 - [39] R. Alligier, D. Gianazza, and N. Durand, “Energy rate prediction using an equivalent thrust setting profile,” presented at the ICRAT 2012, 5th International Conference on Research in Air Transportation, 2012, p. pp-xxxx.
 - [40] R. Alligier, D. Gianazza, and N. Durand, “Machine Learning Applied to Airspeed Prediction During Climb,” 11th USA/Europe Air Traffic Management Research and Development Seminar (ATM 2015), Lisbon, Portugal, June 2015.
 - [41] M. Marinescu, A. Olivares, E. Staffetti, and J. Sun, “On the estimation of vector wind profiles using aircraft-derived data and Gaussian process regression,” *Aerospace*, vol. 9, no. 7, p. 377, 2022.
 - [42] A. Hadjaz, G. Marceau, P. Savéant, and M. Schoenauer, “Online learning for ground trajectory prediction,” *arXiv preprint arXiv:1212.3998*, 2012.
 - [43] G. Slater, “Adaptive improvement of aircraft climb performance for air traffic control applications,” presented at the Proceedings of the IEEE International Symposium on Intelligent Control, IEEE, 2002, pp. 602–607.
 - [44] C. Schultz, D. Thippavong, and H. Erzberger, “Adaptive trajectory prediction algorithm for climbing flights,” presented at the AIAA Guidance, Navigation, and Control Conference, 2012, p. 4931.
 - [45] K. Tastambekov, S. Puechmorel, D. Delahaye, and C. Rabut, “Aircraft trajectory forecasting using local functional regression in Sobolev space,” *Transportation research part C: emerging technologies*, vol. 39, pp. 1–22, 2014.
 - [46] M. Hrastovec and F. Solina, “Machine learning model for aircraft performances,” presented at the 2014 IEEE/AIAA 33rd Digital Avionics Systems Conference (DASC), IEEE, 2014, pp. 8C4-1.
 - [47] M. Hrastovec and F. Solina, “Prediction of aircraft performances based on data collected by air traffic control centers,” *Transportation Research Part C:*

- Emerging Technologies*, vol. 73, pp. 167–182, 2016.
- [48] R. Dalmau, M. Pérez-Batlle, and X. Prats, “Real-time identification of guidance modes in aircraft descents using surveillance data,” presented at the 2018 IEEE/AIAA 37th Digital Avionics Systems Conference (DASC), IEEE, 2018, pp. 1–10.
 - [49] R. Dalmau, X. Prats, R. Verhoeven, F. Bussink, and B. Heesbeen, “Comparison of various guidance strategies to achieve time constraints in optimal descents,” *Journal of Guidance, Control, and Dynamics*, vol. 42, no. 7, pp. 1612–1621, 2019.
 - [50] Y. Lin, J. Zhang, and H. Liu, “An algorithm for trajectory prediction of flight plan based on relative motion between positions,” *Frontiers of Information Technology & Electronic Engineering*, vol. 19, no. 7, pp. 905–916, 2018.
 - [51] P. P. Choi and M. Hebert, “Learning and predicting moving object trajectory: a piecewise trajectory segment approach,” *Robotics Institute*, vol. 337, pp. 1–17, 2006.
 - [52] S. Ayhan and H. Samet, “Aircraft trajectory prediction made easy with predictive analytics,” presented at the Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 21–30.
 - [53] Y. Lin, B. Yang, J. Zhang, and H. Liu, “Approach for 4-d trajectory management based on HMM and trajectory similarity,” *Journal of Marine Science and Technology*, vol. 27, no. 3, p. 7, 2019.
 - [54] R. Rezaie and X. R. Li, “Trajectory modeling and prediction with waypoint information using a conditionally Markov sequence,” presented at the 2018 56th Annual Allerton Conference on Communication, Control, and Computing (Allerton), IEEE, 2018, pp. 486–493.
 - [55] G. Chatterji, “Short-term trajectory prediction methods,” presented at the Guidance, Navigation, and Control Conference and Exhibit, 1999, p. 4233.
 - [56] I. Lymperopoulos and J. Lygeros, “Sequential Monte Carlo methods for multi-aircraft trajectory prediction in air traffic management,” *International Journal of Adaptive Control and Signal Processing*, vol. 24, no. 10, pp. 830–849, 2010.
 - [57] X. R. Li and V. P. Jilkov, “Survey of maneuvering target tracking. Part V. Multiple-model methods,” *IEEE Transactions on aerospace and electronic systems*, vol. 41, no. 4, pp. 1255–1321, 2005.
 - [58] A. Franco, D. Rivas, and A. Valenzuela, “Probabilistic aircraft trajectory prediction in cruise flight considering ensemble wind forecasts,” *Aerospace science and technology*, vol. 82, pp. 350–362, 2018.
 - [59] Y. Song, P. Cheng, and C. Mu, “An improved trajectory prediction algorithm based on trajectory data mining for air traffic management,” presented at the 2012 IEEE International Conference on Information and Automation, IEEE, 2012, pp. 981–986.
 - [60] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan, *Estimation with applications to tracking and navigation: theory algorithms and software*. John Wiley & Sons, 2004.
 - [61] W. Liu and I. Hwang, “Probabilistic trajectory prediction and conflict detection for air traffic control,” *Journal of Guidance, Control, and Dynamics*, vol. 34, no. 6, pp. 1779–1789, 2011.
 - [62] C. E. Seah and I. Hwang, “A hybrid estimation algorithm for terminal-area

- aircraft tracking,” presented at the AIAA guidance, navigation and control conference and exhibit, 2007, p. 6691.
- [63] C. E. Seah and I. Hwang, “Terminal-area aircraft tracking using hybrid estimation,” *Journal of Guidance, Control, and Dynamics*, vol. 32, no. 3, pp. 836–849, 2009.
 - [64] I. Hwang and C. E. Seah, “Intent-based probabilistic conflict detection for the next generation air transportation system,” *Proceedings of the IEEE*, vol. 96, no. 12, pp. 2040–2059, 2008.
 - [65] Y. Chen, Z. Cheng, and S.-L. Wen, “Modified IMM algorithm for unmatched dynamic models,” *Systems Engineering and Electronics*, vol. 33, no. 12, pp. 2593–2597, 2011.
 - [66] ZHANG junfeng, Wu xiaoguang, and Wang fei, “Aircraft trajectory prediction based on modified interacting multiple model algorithm,” *Journal of Donghua University (English Edition)*, vol. 32, no. 2, pp. 180–184, 2015.
 - [67] J. L. Yepes, I. Hwang, and M. Rotea, “New algorithms for aircraft intent inference and trajectory prediction,” *Journal of guidance, control, and dynamics*, vol. 30, no. 2, pp. 370–382, 2007.
 - [68] L. Xi, Z. Jun, Z. Yanbo, and L. Wei, “Simulation study of algorithms for aircraft trajectory prediction based on ADS-B technology,” presented at the 2008 Asia Simulation Conference-7th International Conference on System Simulation and Scientific Computing, IEEE, 2008, pp. 322–327.
 - [69] Y. Liu and X. R. Li, “Intent based trajectory prediction by multiple model prediction and smoothing,” presented at the AIAA Guidance, Navigation, and Control Conference, 2015, p. 1324.
 - [70] X. Guan, R. Lv, L. Sun, and Y. Liu, “A study of 4D trajectory prediction based on machine deep learning,” presented at the 2016 12th World Congress on Intelligent Control and Automation (WCICA), IEEE, 2016, pp. 24–27.
 - [71] Z. Shi, M. Xu, Q. Pan, B. Yan, and H. Zhang, “LSTM-based flight trajectory prediction,” presented at the 2018 International joint conference on neural networks (IJCNN), IEEE, 2018, pp. 1–8.
 - [72] K. Yang, M. Bi, Y. Liu, and Y. Zhang, “LSTM-based deep learning model for civil aircraft position and attitude prediction approach,” presented at the 2019 Chinese control conference (CCC), IEEE, 2019, pp. 8689–8694.
 - [73] Z. Yang, X. Kang, Y. Gong, and J. Wang, “Aircraft trajectory prediction and aviation safety in ADS-B failure conditions based on neural network,” *Scientific reports*, vol. 13, no. 1, p. 19677, 2023.
 - [74] Z. Ma, M. Yao, T. Hong, and B. Li, “Aircraft surface trajectory prediction method based on LSTM with attenuated memory window,” presented at the Journal of Physics: Conference Series, IOP Publishing, 2019, p. 012003.
 - [75] Z. Zhao, W. Zeng, Z. Quan, M. Chen, and Z. Yang, “Aircraft trajectory prediction using deep long short-term memory networks,” in *CICTP 2019*, 2019, pp. 124–135.
 - [76] Z. Xu, W. Zeng, X. Chu, and P. Cao, “Multi-aircraft trajectory collaborative prediction based on social long short-term memory network,” *Aerospace*, vol. 8, no. 4, p. 115, 2021.
 - [77] P. Han, W. Wang, Q. Shi, and J. Yang, “Real-time short-term trajectory prediction based on GRU neural network,” presented at the 2019 IEEE/AIAA 38th Digital Avionics Systems Conference (DASC), IEEE, 2019, pp. 1–8.
 - [78] J. Huang and W. Ding, “Aircraft trajectory prediction based on bayesian optimised temporal convolutional network–bidirectional gated recurrent unit

- hybrid neural network,” *International Journal of Aerospace Engineering*, vol. 2022, 2022.
- [79] I. Goodfellow *et al.*, “Generative adversarial nets,” *Advances in neural information processing systems*, vol. 27, 2014.
 - [80] Y. Pang and Y. Liu, “Conditional generative adversarial networks (CGAN) for aircraft trajectory prediction considering weather effects,” presented at the AIAA Scitech 2020 Forum, 2020, p. 1853.
 - [81] Q. Hu, G. Huang, H. Shi, Y. Lin, and D. Guo, “A Short-term Aircraft Trajectory Prediction Framework Using Conditional Generative Adversarial Network,” presented at the 2022 IEEE 4th International Conference on Civil Aviation Safety and Information Technology (ICCASIT), IEEE, 2022, pp. 433–439.
 - [82] G. Jarry, N. Couellan, and D. Delahaye, “On the use of generative adversarial networks for aircraft trajectory generation and atypical approach detection,” presented at the Air Traffic Management and Systems IV: Selected Papers of the 6th ENRI International Workshop on ATM/CNS (EIWAC2019) 6, Springer, 2021, pp. 227–243.
 - [83] A. Vaswani, “Attention is all you need,” *Advances in Neural Information Processing Systems*, 2017.
 - [84] S. Luo, M. Zhao, Z. Zhao, L. Li, S. Zhang, and X. Zhang, “FT-TF: A 4D Long-Term Flight Trajectory Prediction Method Based on Transformer,” presented at the 2023 42nd Chinese Control Conference (CCC), IEEE, 2023, pp. 4616–4621.
 - [85] Q. Tong, J. Hu, Y. Chen, D. Guo, and X. Liu, “Long-term trajectory prediction model based on Transformer,” *IEEE Access*, 2023.
 - [86] M. Tan, G. Yu, and H. Wang, “Research on 4D Trajectory Prediction Algorithm Based on Hybrid Neural Network Model,” presented at the 2023 IEEE 6th International Conference on Automation, Electronics and Electrical Engineering (AUTEEE), IEEE, 2023, pp. 112–119.
 - [87] H. Wang, G. Yu, and M. Tan, “Research on Aircraft Trajectory Prediction Algorithm Based on Hybrid Neural Network Model,” presented at the 2023 IEEE 6th International Conference on Automation, Electronics and Electrical Engineering (AUTEEE), IEEE, 2023, pp. 120–126.
 - [88] Y. Wu, H. Yu, J. Du, B. Liu, and W. Yu, “An aircraft trajectory prediction method based on trajectory clustering and a spatiotemporal feature network,” *Electronics*, vol. 11, no. 21, p. 3453, 2022.
 - [89] H. Shafienya and A. C. Regan, “4D flight trajectory prediction using a hybrid deep learning prediction method based on ads-b technology: A case study of hartsfield–jackson Atlanta international airport (ATL),” *Transportation Research Part C: Emerging Technologies*, vol. 144, p. 103878, 2022.
 - [90] L. Ma and S. Tian, “A hybrid CNN-LSTM model for aircraft 4D trajectory prediction,” *IEEE access*, vol. 8, pp. 134668–134680, 2020.
 - [91] Y. Liu and M. Hansen, “Predicting aircraft trajectories: A deep generative convolutional recurrent neural networks approach,” *arXiv preprint arXiv:1812.11670*, 2018.
 - [92] N. Schimpf, Z. Wang, S. Li, E. J. Knoblock, H. Li, and R. D. Apaza, “A Generalized Approach to Aircraft Trajectory Prediction via Supervised Deep Learning,” *IEEE Access*, 2023.
 - [93] Y. Pang, H. Yao, J. Hu, and Y. Liu, “A recurrent neural network approach

- for aircraft trajectory prediction with weather features from sherlock,” presented at the AIAA Aviation 2019 Forum, 2019, p. 3413.
- [94] J. Patrikar, B. Moon, J. Oh, and S. Scherer, “Predicting like a pilot: Dataset and method to predict socially-aware aircraft trajectories in non-towered terminal airspace,” presented at the 2022 international conference on robotics and automation (icra), IEEE, 2022, pp. 2525–2531.
 - [95] S. T. Kanneganti, P. B. Chilson, and R. Huck, “Visualization and prediction of aircraft trajectory using ADS-B,” presented at the NAECON 2018-IEEE National Aerospace and Electronics Conference, IEEE, 2018, pp. 529–532.
 - [96] A. De Leege, M. van Paassen, and M. Mulder, “A machine learning approach to trajectory prediction,” presented at the AIAA Guidance, Navigation, and Control (GNC) Conference, 2013, p. 4782.
 - [97] S. Hong and K. Lee, “Trajectory prediction for vectored area navigation arrivals,” *Journal of Aerospace Information Systems*, vol. 12, no. 7, pp. 490–502, 2015.
 - [98] Y. Liu, M. Hansen, D. J. Lovell, and M. O. Ball, “Predicting aircraft trajectory choice—a nominal route approach,” presented at the Proc. of the International Conference for Research in Air Transportation, 2018.
 - [99] T. H. Le, N. P. Tran, D.-T. Pham, M. Schultz, and S. Alam, “Short-term trajectory prediction using generative machine learning methods,” in proceedings of the 9th International Conference on Research in Air Transportation (ICRAT 2020), 2020.
 - [100] S. I. R. Rizvi and J. H. Markani, “A meta-learning approach for aircraft trajectory prediction,” *Communications and Network*, vol. 15, no. 2, pp. 43–64, 2023.
 - [101] S. Qiao, D. Shen, X. Wang, N. Han, and W. Zhu, “A self-adaptive parameter selection trajectory prediction approach via hidden Markov models,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 1, pp. 284–296, 2014.
 - [102] S. T. Barratt, M. J. Kochenderfer, and S. P. Boyd, “Learning probabilistic trajectory models of aircraft in terminal airspace from position data,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 9, pp. 3536–3545, 2018.
 - [103] X. Gui, J. Zhang, X. Tang, J. Bao, and B. Wang, “A data-driven trajectory optimization framework for terminal maneuvering area operations,” *Aerospace Science and Technology*, vol. 131, p. 108010, 2022.
 - [104] P. Besse, B. Guillouet, J.-M. Loubes, and R. François, “Review and perspective for distance based trajectory clustering,” *arXiv preprint arXiv:1508.04904*, 2015.
 - [105] P. Jia, H. Chen, L. Zhang, and D. Han, “Attention-lstm based prediction model for aircraft 4-d trajectory,” *Scientific reports*, vol. 12, no. 1, p. 15533, 2022.
 - [106] Y. Pang, N. Xu, and Y. Liu, “Aircraft trajectory prediction using LSTM neural network with embedded convolutional layer,” presented at the Proceedings of the Annual Conference of the PHM Society, PHM Society Scottsdale, AZ, USA, 2019, pp. 1–8.
 - [107] C. E. V. Gallego, V. F. G. Comendador, F. J. S. Nieto, G. O. Imaz, and R. M. A. Valdés, “Analysis of air traffic control operational impact on aircraft vertical profiles supported by machine learning,” *Transportation research*

- part C: emerging technologies*, vol. 95, pp. 883–903, 2018.
- [108] L. Zhang, H. Chen, P. Jia, Z. Tian, and X. Du, “WGAN-GP and LSTM based Prediction Model for Aircraft 4-D Trajectory,” presented at the 2022 International Wireless Communications and Mobile Computing (IWCMC), IEEE, 2022, pp. 937–942.
 - [109] Y. Pang and Y. Liu, “Probabilistic aircraft trajectory prediction considering weather uncertainties using dropout as Bayesian approximate variational inference,” presented at the AIAA Scitech 2020 Forum, 2020, p. 1413.
 - [110] S. Feng, G. Wang, P. Zhao, X. Chao, and K. Cai, “Short-term Aircraft Trajectory Prediction Considering Weather Effect,” presented at the 2023 IEEE/AIAA 42nd Digital Avionics Systems Conference (DASC), IEEE, 2023, pp. 1–6.
 - [111] S. Hao, Y. Zhang, S. Cheng, R. Liu, and Z. Xing, “Probabilistic multi-aircraft conflict detection approach for trajectory-based operation,” *Transportation Research Part C: Emerging Technologies*, vol. 95, pp. 698–712, 2018.
 - [112] Y. Chen, M. Hu, and L. Yang, “Autonomous planning of optimal four-dimensional trajectory for real-time en-route airspace operation with solution space visualisation,” *Transportation Research Part C: Emerging Technologies*, vol. 140, p. 103701, 2022.
 - [113] W. Dai, J. Zhang, D. Delahaye, and X. Sun, “A heuristic algorithm for aircraft 4d trajectory optimization based on bezier curve,” presented at the ATM 2019, 13th USA/Europe Air Traffic Management Research and Development Seminar, 2019.
 - [114] R. Sáez García *et al.*, “Alternative 4D Trajectories for the avoidance of weather-and contrail-sensitive volumes,” presented at the 12th SESAR Innovation Days: Inspiring long-term research in the field of air traffic management: Budapest, Hungary: December 5-8, 2022, Single European Sky ATM Research (SESAR), 2022, pp. 1–8.
 - [115] Z. Chang, M. Hu, and Y. Zhang, “Multi-objective aircraft robust trajectory optimization considering various predictability metrics under uncertain wind,” *IEEE Access*, vol. 11, pp. 104333–104346, 2023.
 - [116] M. Soler, A. Olivares, E. Staffetti, and D. Zapata, “Framework for aircraft trajectory planning toward an efficient air traffic management,” *Journal of Aircraft*, vol. 49, no. 1, pp. 341–348, 2012.
 - [117] J. Zhou, H. Zhang, W. Lyu, J. Wan, J. Zhang, and W. Song, “Hybrid 4-dimensional Trajectory prediction model, based on the reconstruction of prediction time span for aircraft en Route,” *Sustainability*, vol. 14, no. 7, p. 3862, 2022.
 - [118] A. Radford, “Unsupervised representation learning with deep convolutional generative adversarial networks,” *arXiv preprint arXiv:1511.06434*, 2015.
 - [119] J. Adler and S. Lunz, “Banach wasserstein gan,” *Advances in neural information processing systems*, vol. 31, 2018.
 - [120] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, “Improved training of wasserstein gans,” *Advances in neural information processing systems*, vol. 30, 2017.
 - [121] H. Petzka, A. Fischer, and D. Lukovnicov, “On the regularization of wasserstein gans,” *arXiv preprint arXiv:1709.08894*, 2017.

- [122] C. Chu, A. Zhmoginov, and M. Sandler, “Cyclegan, a master of steganography,” *arXiv preprint arXiv:1712.02950*, 2017.
- [123] X. Chen, Y. Duan, R. Houthoofd, J. Schulman, I. Sutskever, and P. Abbeel, “Infogan: Interpretable representation learning by information maximizing generative adversarial nets,” *Advances in neural information processing systems*, vol. 29, 2016.
- [124] A. Spurr, E. Aksan, and O. Hilliges, “Guiding infogan with semi-supervision,” presented at the Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 2017, pp. 119–134.
- [125] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena, “Self-attention generative adversarial networks,” presented at the International conference on machine learning, PMLR, 2019, pp. 7354–7363.
- [126] T. Karras, S. Laine, and T. Aila, “A style-based generator architecture for generative adversarial networks,” presented at the Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2019, pp. 4401–4410.
- [127] A. Brock, J. Donahue, and K. Simonyan, “Large scale GAN training for high fidelity natural image synthesis,” *arXiv preprint arXiv:1809.11096*, 2018.
- [128] L. Xu, M. Skoularidou, A. Cuesta-Infante, and K. Veeramachaneni, “Modeling tabular data using conditional gan,” *Advances in neural information processing systems*, vol. 32, 2019.
- [129] M. Arjovsky, S. Chintala, and L. Bottou, “Wasserstein generative adversarial networks,” presented at the International conference on machine learning, PMLR, 2017, pp. 214–223.
- [130] X. Mao, Q. Li, H. Xie, R. Y. Lau, Z. Wang, and S. Paul Smolley, “Least squares generative adversarial networks,” presented at the Proceedings of the IEEE international conference on computer vision, 2017, pp. 2794–2802.
- [131] M. Mirza, “Conditional generative adversarial nets,” *arXiv preprint arXiv:1411.1784*, 2014.
- [132] S. Yazdani *et al.*, “Generative AI in depth: A survey of recent advances, model variants, and real-world applications,” *Journal of Big Data*, vol. 12, no. 1, pp. 1–43, 2025.
- [133] J. L. Elman, “Finding structure in time,” *Cognitive science*, vol. 14, no. 2, pp. 179–211, 1990.
- [134] Z. Che, S. Purushotham, K. Cho, D. Sontag, and Y. Liu, “Recurrent neural networks for multivariate time series with missing values,” *Scientific reports*, vol. 8, no. 1, p. 6085, 2018.
- [135] T. Mikolov, M. Karafiát, L. Burget, J. Cernocký, and S. Khudanpur, “Recurrent neural network based language model,” presented at the Interspeech, Makuhari, 2010, pp. 1045–1048.
- [136] N. Kalchbrenner and P. Blunsom, “Recurrent continuous translation models,” presented at the Proceedings of the 2013 conference on empirical methods in natural language processing, 2013, pp. 1700–1709.
- [137] Q. You, H. Jin, Z. Wang, C. Fang, and J. Luo, “Image captioning with semantic attention,” presented at the Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 4651–4659.
- [138] A. Graves, “Generating sequences with recurrent neural networks,” *arXiv preprint arXiv:1308.0850*, 2013.
- [139] K. Kawakami, “Supervised sequence labelling with recurrent neural

- networks,” Ph.D. dissertation, Univ. Cambridge, Cambridge, U.K., 2008.
- [140] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, “Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks,” presented at the Proceedings of the 23rd international conference on Machine learning, 2006, pp. 369–376.
 - [141] A. Graves, A. Mohamed, and G. Hinton, “Speech recognition with deep recurrent neural networks,” presented at the 2013 IEEE international conference on acoustics, speech and signal processing, Ieee, 2013, pp. 6645–6649.
 - [142] M. Schuster and K. K. Paliwal, “Bidirectional recurrent neural networks,” *IEEE transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.
 - [143] J. Choi, T. Kim, and S. Lee, “Cell-aware stacked LSTMs for modeling sentences,” presented at the Asian conference on machine learning, PMLR, 2019, pp. 1172–1187.
 - [144] A. Miller, A. Fisch, J. Dodge, A.-H. Karimi, A. Bordes, and J. Weston, “Key-value memory networks for directly reading documents,” *arXiv preprint arXiv:1606.03126*, 2016.
 - [145] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997. doi: 10.1162/neco.1997.9.8.1735.
 - [146] K. Cho, “On the Properties of Neural Machine Translation: Encoder-decoder Approaches,” *arXiv preprint arXiv:1409.1259*, 2014.
 - [147] W.-N. Hsu, Y. Zhang, and J. Glass, “A prioritized grid long short-term memory RNN for speech recognition,” presented at the 2016 IEEE Spoken Language Technology Workshop (SLT), IEEE, 2016, pp. 467–473.
 - [148] J. Li, C. Liu, and Y. Gong, “Layer trajectory LSTM,” *arXiv preprint arXiv:1808.09522*, 2018.
 - [149] H.-S. Lee and S. Oh, “LSTM-based deep learning for time series forecasting: The case of corporate credit score prediction,” *The Journal of Information Systems*, vol. 29, no. 1, pp. 241–265, 2020.
 - [150] L. Lin, W. Li, H. Bi, and L. Qin, “Vehicle trajectory prediction using LSTMs with spatial-temporal attention mechanisms,” *IEEE Intelligent Transportation Systems Magazine*, vol. 14, no. 2, pp. 197–208, 2021.
 - [151] J. Devlin, “Bert: Pre-training of deep bidirectional transformers for language understanding,” *arXiv preprint arXiv:1810.04805*, 2018.
 - [152] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, “Improving language understanding by generative pre-training,” OpenAI, San Francisco, CA, USA, Tech. Rep., 2018.
 - [153] C. Raffel *et al.*, “Exploring the limits of transfer learning with a unified text-to-text transformer,” *Journal of machine learning research*, vol. 21, no. 140, pp. 1–67, 2020.
 - [154] Z. Dai, “Transformer-xl: Attentive language models beyond a fixed-length context,” *arXiv preprint arXiv:1901.02860*, 2019.
 - [155] Z. Yang, “XLNet: Generalized Autoregressive Pretraining for Language Understanding,” *arXiv preprint arXiv:1906.08237*, 2019.
 - [156] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
 - [157] J. Hu, L. Shen, and G. Sun, “Squeeze-and-excitation networks,” presented at the Proceedings of the IEEE conference on computer vision and pattern

- recognition, 2018, pp. 7132–7141.
- [158] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” presented at the Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.
 - [159] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” *Advances in neural information processing systems*, vol. 25, 2012.
 - [160] C. Szegedy *et al.*, “Going deeper with convolutions,” presented at the Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, pp. 1–9.
 - [161] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, “Densely connected convolutional networks,” presented at the Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 4700–4708.
 - [162] EUROCONTROL, “Surveillance Data Exchange - Part 1: All Purpose Structured Eurocontrol Surveillance Information Exchange (ASTERIX),” EUROCONTROL, Brussels, Belgium, Tech. Rep. SUR.ET1.ST05.2000-STD-01-01, Nov. 2007.
 - [163] B. S. Ali, W. Y. Ochieng, and R. Zainudin, “An analysis and model for Automatic Dependent Surveillance Broadcast (ADS-B) continuity,” *GPS Solutions*, vol. 21, pp. 1841–1854, 2017.
 - [164] I. Cao, “Guidance material on comparison of surveillance technologies (GMST),” *International Civil Aviation Organization*, 2007.
 - [165] X. Olive, J. Sun, M. C. R. Murça, and T. Krauth, “A framework to evaluate aircraft trajectory generation methods,” presented at the Proceedings of the 14th USA/Europe Air Traffic Management Research and Development Seminar, Saclay, France, 2021, pp. 17–19.
 - [166] S. Mondoloni, S. Swierstra, and M. Paglione, “Assessing trajectory prediction performance-metrics definition,” presented at the 24th Digital Avionics Systems Conference, IEEE, 2005, pp. 3–C.
 - [167] H. Ryan, M. Paglione, and S. Green, “Review of trajectory accuracy methodology and comparison of error measurement metrics,” presented at the AIAA Guidance, Navigation, and Control Conference and Exhibit, 2004, p. 4787.
 - [168] Z. Zhang, D. Guo, S. Zhou, J. Zhang, and Y. Lin, “Flight trajectory prediction enabled by time-frequency wavelet transform,” *Nature Communications*, vol. 14, no. 1, p. 5258, 2023.
 - [169] J. Garcia-Chico, R. Vivona, and K. Cate, “Characterizing intent maneuvers from operational data: Step towards trajectory prediction uncertainty estimation,” presented at the AIAA Guidance, Navigation and Control Conference and Exhibit, 2008, p. 6520.
 - [170] C. Vondrick, H. Pirsiavash, and A. Torralba, “Generating videos with scene dynamics,” *Advances in neural information processing systems*, vol. 29, 2016.
 - [171] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee, “Generative adversarial text to image synthesis,” presented at the International conference on machine learning, PMLR, 2016, pp. 1060–1069.
 - [172] A. Brock, J. Donahue, and K. Simonyan, “Large scale GAN training for high fidelity natural image synthesis,” *arXiv preprint arXiv:1809.11096*, 2018.
 - [173] W. Ding, W. Wang, and D. Zhao, “Multi-vehicle trajectories generation for vehicle-to-vehicle encounters,” presented at the 2019 IEEE International

- Conference on Robotics and Automation (ICRA), 2019.
- [174] C. Wang *et al.*, “Framework of nacelle inverse design method based on improved generative adversarial networks,” *Aerospace Science and Technology*, vol. 121, p. 107365, 2022.
 - [175] T. Afonja, D. Chen, and M. Fritz, “MargCTGAN: A ‘Marginally’ Better CTGAN for the Low Sample Regime,” presented at the DAGM German Conference on Pattern Recognition, Springer, 2023, pp. 524–537.
 - [176] A. Majeed and S. O. Hwang, “CTGAN-MOS: Conditional Generative Adversarial Network Based Minority-Class-Augmented Oversampling Scheme for Imbalanced Problems,” *IEEE Access*, 2023.
 - [177] C.-H. Chen, C.-K. Tsung, and S.-S. Yu, “Designing a hybrid equipment-failure diagnosis mechanism under mixed-type data with limited failure samples,” *Applied Sciences*, vol. 12, no. 18, p. 9286, 2022.
 - [178] C. An, J. Sun, Y. Wang, and Q. Wei, “A k-means improved ctgan oversampling method for data imbalance problem,” presented at the 2021 IEEE 21st International Conference on Software Quality, Reliability and Security (QRS), IEEE, 2021, pp. 883–887.
 - [179] M.-P. Cote, B. Hartman, O. Mercier, J. Meyers, J. Cummings, and E. Harmon, “Synthesizing property & casualty ratemaking datasets using generative adversarial networks,” *arXiv preprint arXiv:2008.06110*, 2020.
 - [180] A. A. Alqarni and E.-S. M. El-Alfy, “Improving intrusion detection for imbalanced network traffic using generative deep learning,” *International Journal of Advanced Computer Science and Applications*, vol. 13, no. 4, 2022.
 - [181] L. Xu and K. Veeramachaneni, “Synthesizing tabular data using generative adversarial networks,” *arXiv preprint arXiv:1811.11264*, 2018.
 - [182] G. Gugliotta, “An air-traffic upgrade to improve travel by plane,” *The New York Times*, vol. 17, 2009.
 - [183] R. A. Nugraha, H. F. Pardede, and A. Subekti, “Oversampling based on generative adversarial networks to overcome imbalance data in predicting fraud insurance claim: 10.48129/kjs. splml. 19119,” *Kuwait Journal of Science*, 2022.
 - [184] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” presented at the International conference on machine learning, PMLR, 2015, pp. 448–456.
 - [185] E. Jang, S. Gu, and B. Poole, “Categorical reparameterization with gumbel-softmax,” *arXiv preprint arXiv:1611.01144*, 2016.
 - [186] Z. Lin, A. Khetan, G. Fanti, and S. Oh, “Pacgan: The power of two samples in generative adversarial networks,” *Advances in neural information processing systems*, vol. 31, 2018.
 - [187] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: a simple way to prevent neural networks from overfitting,” *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.
 - [188] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, “Improved training of wasserstein gans,” *Advances in neural information processing systems*, vol. 30, 2017.
 - [189] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, “Improved techniques for training gans,” *Advances in neural information processing systems*, vol. 29, 2016.
 - [190] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter,

- “Gans trained by a two time-scale update rule converge to a local nash equilibrium,” *Advances in neural information processing systems*, vol. 30, 2017.
- [191] A. Borji, “Pros and cons of gan evaluation measures,” *Computer vision and image understanding*, vol. 179, pp. 41–65, 2019.
 - [192] M. Soloveitchik, T. Diskin, E. Morin, and A. Wiesel, “Conditional frechet inception distance,” *arXiv preprint arXiv:2103.11521*, 2021.
 - [193] X. Guan, X. Zhang, D. Han, Y. Zhu, J. Lv, and J. Su, “A strategic flight conflict avoidance approach based on a memetic algorithm,” *Chinese Journal of Aeronautics*, vol. 27, no. 1, pp. 93–101, 2014.
 - [194] X. Guan, X. Zhang, J. Wei, I. Hwang, Y. Zhu, and K. Cai, “A strategic conflict avoidance approach based on cooperative coevolutionary with the dynamic grouping strategy,” *International Journal of Systems Science*, vol. 47, no. 9, pp. 1995–2008, 2016.
 - [195] D. J. Brudnicki and D. B. Kirk, “Trajectory modeling for automated en route air traffic control (AERA),” presented at the Proceedings of 1995 American Control Conference-ACC’95, IEEE, 1995, pp. 3425–3429.
 - [196] D. Delahaye, S. Puechmorel, P. Tsiotras, and E. Féron, “Mathematical models for aircraft trajectory design: A survey,” presented at the Air Traffic Management and Systems: Selected Papers of the 3rd ENRI International Workshop on ATM/CNS (EIWAC2013), Springer, 2014, pp. 205–247.
 - [197] A. Harada, T. Ezaki, T. Wakayama, and K. Oka, “Air traffic efficiency analysis of airliner scheduled flights using collaborative actions for renovation of air traffic systems open data,” *Journal of Advanced Transportation*, vol. 2018, no. 1, p. 2734763, 2018.
 - [198] S. Ruiz, L. Guichard, N. Pilon, and K. Delcourte, “A New Air Traffic Flow Management User-Driven Prioritisation Process for Low Volume Operator in Constraint: Simulations and Results,” *Journal of advanced transportation*, vol. 2019, no. 1, p. 1208279, 2019.
 - [199] E. Andrés, D. González-Arribas, M. Soler, M. Kamgarpour, M. Sanjurjo-Rivo, and J. Simarro, “Iterative graph deformation for aircraft trajectory planning considering ensemble forecasting of thunderstorms,” *Transportation Research Part C: Emerging Technologies*, vol. 145, p. 103919, 2022.
 - [200] H.-S. Park and C.-H. Jun, “A simple and fast algorithm for K-medoids clustering,” *Expert systems with applications*, vol. 36, no. 2, pp. 3336–3341, 2009.
 - [201] S. Shah and M. Singh, “Comparison of a time efficient modified K-mean algorithm with K-mean and K-medoid algorithm,” presented at the 2012 international conference on communication systems and network technologies, IEEE, 2012, pp. 435–437.
 - [202] J. Park, J. Jeong, and Y. Park, “Ship trajectory prediction based on bi-LSTM using spectral-clustered AIS data,” *Journal of marine science and engineering*, vol. 9, no. 9, p. 1037, 2021.
 - [203] S. J. Corrado, T. G. Puranik, O. P. Fischer, and D. N. Mavris, “A clustering-based quantitative analysis of the interdependent relationship between spatial and energy anomalies in ADS-B trajectory data,” *Transportation Research Part C: Emerging Technologies*, vol. 131, p. 103331, 2021.
 - [204] P. Arora and S. Varshney, “Analysis of k-means and k-medoids algorithm for big data,” *Procedia Computer Science*, vol. 78, pp. 507–512, 2016.

- [205] S. Zhang, L. Wang, M. Zhu, S. Chen, H. Zhang, and Z. Zeng, “A bi-directional LSTM ship trajectory prediction method based on attention mechanism,” presented at the 2021 IEEE 5th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), IEEE, 2021, pp. 1987–1993.
- [206] S. Ioffe, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” *arXiv preprint arXiv:1502.03167*, 2015.
- [207] E. Jang, S. Gu, and B. Poole, “Categorical reparameterization with gumbel-softmax,” *arXiv preprint arXiv:1611.01144*, 2016.
- [208] Y. Liu and M. Hansen, “Predicting aircraft trajectories: A deep generative convolutional recurrent neural networks approach,” *arXiv preprint arXiv:1812.11670*, 2018.
- [209] Y. Pang and Y. Liu, “Conditional generative adversarial networks (CGAN) for aircraft trajectory prediction considering weather effects,” presented at the AIAA Scitech 2020 Forum, 2020, p. 1853.
- [210] V. S. Spelman and R. Porkodi, “A review on handling imbalanced data,” presented at the 2018 international conference on current trends towards converging technologies (ICCTCT), IEEE, 2018, pp. 1–11.
- [211] K. Cheng, C. Zhang, H. Yu, X. Yang, H. Zou, and S. Gao, “Grouped SMOTE with noise filtering mechanism for classifying imbalanced data,” *IEEE Access*, vol. 7, pp. 170668–170681, 2019.
- [212] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, “SMOTE: synthetic minority over-sampling technique,” *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.
- [213] M. Gao, X. Hong, S. Chen, C. J. Harris, and E. Khalaf, “PDFOS: PDF estimation based over-sampling for imbalanced two-class problems,” *Neurocomputing*, vol. 138, pp. 248–259, 2014.
- [214] H. Zhang and M. Li, “RWO-Sampling: A random walk over-sampling approach to imbalanced data classification,” *Information Fusion*, vol. 20, pp. 99–116, 2014.
- [215] D. Elreedy, A. F. Atiya, and F. Kamalov, “A theoretical distribution analysis of synthetic minority oversampling technique (SMOTE) for imbalanced learning,” *Machine Learning*, vol. 113, no. 7, pp. 4903–4923, 2024.
- [216] R. DeLaura, M. Robinson, M. Pawlak, and J. Evans, “Modeling convective weather avoidance in enroute airspace,” presented at the 13th Conference on Aviation, Range, and Aerospace Meteorology, AMS, New Orleans, LA, 2008.
- [217] H. Zhang and Z. Liu, “Four-Dimensional Aircraft Trajectory Prediction Based on Generative Deep Learning,” *Journal of Aerospace Information Systems*, pp. 1–13, 2024.