

Generation of Synthetic Aircraft Landing Trajectories Using Generative Adversarial Networks

Master of Science Thesis

Sebastiaan C.P. Wijnands



Generation of Synthetic Aircraft Landing Trajectories Using Generative Adversarial Networks

Master of Science Thesis

by

Sebastiaan C.P. Wijnands

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Thursday October 3, 2024 at 09:30 AM.

Student number:	4668561	
Thesis committee:	Dr. M.D. Pavel	TU Delft, Chair
	Dr. O.A. Sharpanskykh	TU Delft, Supervisor
	Dr.ir. D.M. Pool	TU Delft, Examiner
	Ir. K.Y.S.B. Aly	TU Delft, Additional

An electronic version of this thesis is available at <http://repository.tudelft.nl/>.

Acknowledgements

This thesis represents the culmination of my Master of Science journey in Aerospace Engineering at the Delft University of Technology. It has been a challenging yet rewarding experience that marks the end of my academic endeavors and the beginning of new professional opportunities. Reflecting on my time here, I am appreciative for the knowledge gained, the skills developed, and the personal growth experienced.

I would like to extend my sincere gratitude to my supervisor, Dr. Sharpanskykh. His unwavering support, insightful critiques, and constant encouragement have been invaluable. Dr. Sharpanskykh guided me through every stage of the research process, helping me to overcome challenges and achieve a level of work I am truly proud of. His dedication and expertise have not only enhanced the quality of this thesis but also made the entire journey enjoyable. I am also thankful to Karim Aly; our productive discussions and his valuable input significantly contributed to the development of this thesis. Additionally, I want to acknowledge the members of the SynthAIR project. The shared experiences provided a sense of community that was helpful during the demanding phases of my research. The collaborative environment and mutual encouragement greatly enhanced my graduation experience.

I want to express my appreciation to my family, friends, and especially my girlfriend Paola. Her patience, understanding, and backing have been my anchor throughout this journey. She has endured my frustrations and celebrated my successes, making this journey all the more meaningful. Her belief in me and her constant encouragement have been a source of great strength and motivation. I look forward to the future, knowing that I have such a wonderful person by my side.

This thesis is a testament to the collective support and encouragement I have received from all those around me. To everyone who has contributed to this journey, I am truly grateful.

Sebastiaan C.P. Wijnands
The Hague, September 2024

Contents

List of Figures	vii
List of Tables	ix
List of Abbreviations	xi
Introduction	xiii
I Scientific Paper	1
II Literature Study	21
1 Introduction	23
2 Synthetic Data for Air Traffic Management Automation	25
2.1 Synthetic Data	25
2.1.1 Synthetic Data Definition and Rationale	25
2.1.2 Overview of Synthetic Data Types	26
2.1.3 Key Synthetic Data Applications	26
2.2 SynthAir Project - SESAR Joint Undertaking.	27
2.2.1 Synthetic Data Generation for Multivariate Time Series for ATM Automation	27
2.2.2 Universal Time Series Model for Prediction & Generation for ATM Automation	27
2.2.3 Identified Use Cases	28
2.2.4 4-Dimensional Aircraft Flight Trajectories	29
2.2.5 Available Open Data Sources.	29
2.3 Synthetic Data Generation - Generative Modeling	31
2.3.1 Generative vs. Discriminative Modeling	31
2.3.2 Density Function Estimation.	31
2.3.3 Variational Autoencoders	32
2.3.4 Generative Adversarial Networks.	33
2.3.5 Diffusion Models.	33
2.4 Temporal Dynamics in Synthetic Multivariate Time Series Data	34
2.4.1 Time Series & Recurrent Neural Networks	35
2.4.2 Long Short-Term Memory Model	36
2.4.3 Self-Attention & Transformers	37
2.5 Synthetic Time Series Quality Evaluation	38
2.5.1 The Generative Learning Trilemma	38
2.5.2 PCA and t-SNE for Data Diversity	38
2.5.3 Discriminative Scoring for Data Fidelity	39
2.5.4 Predictive Scoring for Data Usefulness.	39
2.5.5 Physical Flyability Using BlueSky	40
2.5.6 Distance Metrics	41
3 Existing Aircraft Flight Trajectory Generation Methods	45
3.1 Model-Driven Methods Based on Flight Dynamic Equations	45
3.1.1 Model-Driven Rationale & Limitations.	45
3.1.2 BADA: Base of Aircraft Data	46
3.1.3 OpenAP: Open Aircraft Performance.	47
3.2 Flight Plan Database Statistical Extrapolation.	47
3.3 Markov Chain Processes	48
3.3.1 Stochastic Model of an Aircraft Trajectory	48
3.3.2 Conflict Probability Estimation	49

3.4	Multivariate Density - Gaussian Mixture Models	50
3.4.1	Trajectory Dimensionality Reduction	50
3.4.2	Multivariate Statistical Distribution Fitting.	51
3.5	Convolutional Generative Adversarial Network	52
3.5.1	GAN for Trajectory Generation.	52
3.5.2	Abnormal Approach Detection.	53
3.6	Temporal Convolutional Variational Autoencoder	54
3.6.1	Background & Data	54
3.6.2	Temporal Convolution for Temporal Dependencies	54
3.6.3	VampPrior - Variational Mixture of Posteriors	55
3.6.4	Latent Space Analysis & Trajectory Generation.	55
4	Synthetic Multivariate Time Series Generation with Deep Generative Models	57
4.1	Variational Autoencoders	57
4.1.1	Recurrent Variational Autoencoder	57
4.1.2	Attention-Based Variational Recurrent Autoencoder	58
4.1.3	Vector Quantization and Bidirectional Priors	58
4.1.4	Integration of Domain-Specific Temporal Constructs	59
4.2	Generative Adversarial Networks	59
4.2.1	Recurrent Generative Adversarial Networks	60
4.2.2	Supervised Training and Embedding Network	60
4.2.3	Transformer-Based Generative Adversarial Networks	61
4.3	Diffusion Models	62
4.3.1	Unconditional Univariate and Short Time Series Generation	63
4.3.2	Constrained Time Series Generation.	63
4.3.3	Temporal Decomposition using Transformers & Fourier-Based Loss.	64
4.4	Summary - Application to (Conditional) Trajectory Generation.	65
5	Research Proposal	67
5.1	Objective and Research Questions	67
5.2	Relevance of the Project.	68
5.3	Planning of the Project	69
III	Supporting work	71
A	TimeGAN & TTS-CGAN - Architectures	73
A.1	TimeGAN	73
A.2	TTS-CGAN	74
B	Real & Synthetic Datasets	77
B.1	Real Datasets	77
B.2	Synthetic Datasets	80
C	TimeGAN Verification - Hyperparameter Variation & Model Behavior	83
C.1	Hyperparameter Variations	83
C.2	Notable Model Behavior	84
D	Software User Guide & Repository Structure	87
D.1	Code Repository Structure	87
D.2	Software User Guide	88
D.2.1	Module I: Data Loading, Preprocessing, Labeling & Clustering.	88
D.2.2	Module II: TimeGAN Model Training, Post-Processing & Plotting	88
D.2.3	Module III: Evaluation Framework - Diversity, Fidelity & Usefulness.	88
	Bibliography	91

List of Figures

2.1	Taxonomy of generative modeling approaches, based on [40]	31
2.2	Schematic representation of a variational autoencoder, based on [51]	32
2.3	Schematic representation of a generative adversarial network, based on [51]	33
2.4	Schematic representation of a diffusion model, based on [51]	34
2.5	Schematic representation of a recurrent neural network, from [142]	35
2.6	Schematic representation of a long short-term memory model, from [142]	36
2.7	RNN vs self-attention architectures, from [142]	37
2.8	Schematic representation of a transformer architecture, from [142]	37
2.9	Generative learning trilemma, from [134]	38
2.10	t-SNE and PCA results from TimeGAN (row 1) and RCGAN (row 2), from [138]	39
2.11	Curvature comparison of real and synthetic trajectories, from [77]	40
2.12	The BlueSky graphical user interface, from [56]	40
2.13	Cumulative distribution of DTW and SSPD distances between real and synthetic trajectories, from [100]	41
2.14	Boxplots of SSPD distances (between synthetic and real trajectories) for varying trajectory patterns, from [100]	41
3.1	Organization of the BADA model, from [97]	46
3.2	Geopotential altitude over time	46
3.3	Calibrated airspeed over time	46
3.4	Example trajectory generated by BADA, from [97]	46
3.5	Organization of the OpenAP model, from [119]	47
3.6	Example climb trajectory generated by OpenAP in terms of altitude, true airspeed, and vertical rate over time, from [119]	47
3.7	Geographic distribution of flights across Germany, peak instantaneous aircraft count (PIAC) of the simulation (left) and radar-correlated flight plan data (right), from [47]	48
3.8	Simulated aircraft trajectories based on a flight plan (left) and the separation distance between them over time, from [59]	49
3.9	5000 samples of separation distance processes (left) and the empirical pdf of the minimum distance over time, from [59]	49
3.10	Raw (left) and pre-processed (right) go-around trajectories, from [77]	50
3.11	Sampled version of a trajectory (red) based on the normal lines of a reference trajectory (red), the right depicts the marginal distributions at the sample points (in the lower dimensional representation space, from [77]	51
3.12	Synthetic trajectories generated by the Gaussian mixture model and the Vine copula (with and without dimensionality reduction), from [77]	52
3.13	Real (blue) and generated (green) trajectories in terms of longitude/latitude (left) and altitude (right) profiles, from [64]	52
3.14	Discriminator scoring of trajectories where green indicate typical patterns, red atypical, and orange intermediate, from [64]	53
3.15	Normalized reconstruction error distribution based on the encoder-generator network, where green indicates small errors and red high errors, from [64]	53
3.16	Causal convolution (left) and multiple causal convolution layers with dilatation (right), from [78]	54
3.17	The full VampPrior TCVAE architecture, from [78]	55
3.18	Visualization of the Gaussian clusters in the latent space after the PCA projection, from [78]	56
3.19	Synthetic trajectory generation using pseudo-input components representing <i>dense</i> regions in the latent space, from [78]	56
3.20	Synthetic trajectory generation using pseudo-input components representing <i>sparse</i> regions in the latent space, from [78]	56

4.1	Architecture of the Interpretable TimeVAE model, from [78]	59
4.2	Block diagram of TimeGAN model components, from [138]	61
4.3	Time series processing in TTS-CGAN, from [85]	61
4.4	Types of constraints imposed of time series, from [20]	63
4.5	Types of constraints imposed of time series, from [20]	64
A.1	Model architecture block diagram, from [138]	74
A.2	Data propagation, from [138]	74
A.3	TTS-CGAN model architecture, from [85]	75
A.4	Processing of multivariate time-series in TTS-CGAN, from [85]	75
B.1	Unconditional trajectories - 180 degree track angle	78
B.2	Unconditional trajectories - 270 degree track angle	78
B.3	Trajectories per runway - 180 degree track angle	78
B.4	Trajectories per runway - 270 degree track angle	78
B.5	Geographic profiles of various operational conditions	79
B.6	Altitude profiles of various operational conditions	79
B.7	Clustered trajectories (N = 2)	79
B.8	Clustered trajectories (N = 3)	79
B.9	Clustered trajectories (N = 4)	79
B.10	Clustered trajectories (N = 5)	79
B.11	Geographic profile heat map (5 clusters)	81
B.12	Altitude profile heat map (5 clusters)	81
B.13	Geographic profile heat map (3 clusters)	81
B.14	Altitude profile heat map (3 clusters)	81
B.15	Geographic profile heat map (runways 28 & 34)	81
B.16	Altitude profile heat map (runways 28 & 34)	81
B.17	Geographic profile heat map (go-arounds)	82
B.18	Altitude profile heat map (go-arounds)	82
B.19	Example of uninterpretable results of TTS-CGAN	82
C.1	Two layers	83
C.2	Three layers	83
C.3	Four layers	83
C.4	GRU Modules	84
C.5	LSTM Modules	84
C.6	24 hidden dimensions	84
C.7	36 hidden dimensions	84
C.8	48 hidden dimensions	84
C.9	500 training iterations	84
C.10	1.000 training iterations	84
C.11	2.500 training iterations	84
C.12	Mode collapse of the TimeGAN model for batch sizes 16, 32, and 64	85
C.13	Trade-off between the trajectory distribution coverage and noise, for 5.000 (left) and 1.000 (right) training iterations	85
D.1	Code Repository Structure	87

List of Tables

2.1	Common synthetic data types.	26
2.2	Available open data sources.	30

List of Abbreviations

ADEP	Aerodrome of Departure
ADES	Aerodrome of Destination
ADS-B	Automatic Dependent Surveillance-Broadcast
AI	Artificial Intelligence
AIRAC	Aeronautical Information Regulation and Control
ANSP	Air Navigation Service Provider
ATM	Air Traffic Management
AUA	Air Traffic Control Unit Airspace
CNN	Convolutional Neural Network
DDPM	Denoising Diffusion Probabilistic Model
DL	Deep Learning
DTW	Dynamic Time Warping
FAA	Federal Aviation Administration
FIR	Flight Information Region
GAN	Generative Adversarial Network
GDPR	General Data Protection Regulation
GRU	Gated Recurrent Unit
ICAO	International Civil Aviation Organization
IFR	Instrument Flight Rules
LSTM	Long Short-Term Memory
ML	Machine Learning
NM	Nautical Mile
PCA	Principal Component Analysis
RNN	Recurrent Neural Network
SDG	Synthetic Data Generation
SESAR	Single European Sky ATM Research
SSPD	Symmetric Segment-Path Distance
SVM	Support Vector Machine
t-SNE	T-Distributed Stochastic Neighbor Embedding
TMA	Terminal Maneuvering Area
TSTR	Train-Synthetic-Test-Real
VAE	Variational Autoencoder

Introduction

To ensure the future effectiveness of the air traffic management (ATM) system and its ability to meet growing demand and complexity, enhanced automation will be crucial. This will be heavily bolstered by emerging artificial intelligence (AI) and machine learning (ML) applications [113]. These technologies are key to realizing the Single European Sky ATM Research (SESAR) Joint Undertaking's objectives [114]. Automation in ATM will advance in stages, starting from basic information acquisition and decision support to full air traffic control and U-space services [35, 113]. AI/ML solutions are vital for meeting the increasingly complex demands of passengers, airlines, regulatory bodies, and air traffic controllers, ensuring seamless coordination for safe and efficient air travel. However, achieving widespread connectivity and trust in these developed systems is a significant challenge, compounded by the erratic stability of models and the risk of catastrophic failures due to inadequate training [122]. The increasing need for comprehensive datasets to train ML models for tasks like air traffic flow prediction, route optimization, anomaly detection, and safety improvement often exceeds the available data, making data acquisition both costly and difficult. This aptly named *data access problem* represents a major barrier to effective ML algorithm training [90].

Addressing these challenges, the European Commissions SynthAIR project, part of the SESAR Joint Undertaking, exemplifies the technological advancements driving the transformation of the aviation industry. SynthAIR focuses on generating realistic and diverse synthetic data while ensuring physical plausibility and adaptability to varying conditions. A significant aim of the project is to develop a Universal Time Series Generator (UTG) capable of creating realistic synthetic aircraft trajectories for ATM systems [115]. The Terminal Maneuvering Area (TMA) is a crucial segment of airspace characterized by high traffic density and diverse trajectory types, necessitating granular data to model these scenarios accurately. The main research objective of this work was to investigate the applicability of novel generative modeling techniques in generating synthetic 4-dimensional aircraft landing trajectories capable of capturing traffic patterns in this airspace. Generating synthetic flight trajectories (one of the identified use cases in SynthAIR) that closely mimic real-world data involves considering complex scenarios such as go-arounds and holding patterns, influenced by variables like weather, air traffic, and runway availability. Specifically, generative adversarial networks (GANs) were leveraged to generate synthetic multivariate time series representing aircraft trajectories, primarily using the TimeGAN model [138]. The novelty lies in the extrapolation of the success these techniques have had in other domains to ATM. As part of the broader SynthAIR objective, this thesis aims alleviate issues from lacking data sets to gain more insight into flight behavior within the TMA sector, improve decision-making algorithms for air traffic controllers, as well as refine flight planning and scheduling tools. Ultimately, the project aspires to enhance the safety and efficiency of aviation operations, supporting the broader goals of modernization and the transition to increased automation within the aviation industry.

This report contains all components of the thesis, structured into three parts. Part I features the core scientific paper, which encapsulates the primary objectives and findings of the study against the backdrop of existing literature. It elaborates on the methodology applied and provides a thorough examination of the effectiveness. This is achieved through the implementation of generative adversarial networks to the problem of synthetic aircraft trajectory generation, demonstrating the practical application and impact of the research findings in this context. A crucial precursor to this is a detailed literature review, which is presented in Part II. This section synthesizes existing knowledge and identifies gaps that the current research aims to address, laying a solid foundation for the subsequent scientific inquiry. The literature review not only contextualizes the research but also underscores the novelty and necessity of the proposed study. Part III concerns all supplementary work, not included in the scientific paper. These appendices provide more detailed insight into model architectures and training procedures, real and synthetic data sets, notable model behavior, and a code repository description and accompanying user guide.

I

Scientific Paper

Generation of Synthetic Aircraft Landing Trajectories Using Generative Adversarial Networks

An Exploratory Study on the Applicability of an Unconditional Recurrent-Based Network (TimeGAN) and Conditional Extensions to Synthetic 4-Dimensional Aircraft Landing Trajectory Generation

Sebastiaan Camille Pepijn Wijnands
Air Transport and Operations, Faculty of Aerospace Engineering
Delft University of Technology (TU Delft)
Delft, The Netherlands
S.C.P.Wijnands@student.tudelft.nl

Abstract—The increasing demand and complexity of air traffic management (ATM) systems necessitate significant advancements in automation to ensure safety and efficiency. Artificial intelligence (AI) and machine learning (ML) are emerging as promising solutions to manage this growing complexity, offering enhanced decision-making and predictive capabilities. However, the effectiveness of ML models in ATM heavily relies on the availability of extensive, high-quality data. In many cases, such data is scarce or incomplete, which presents a major challenge for training robust models. Synthetic data generation (SDG) represents a viable solution to address this data scarcity, enabling the creation of realistic, diverse datasets that can effectively fill data gaps and unlock the ML value proposition. The Terminal Maneuvering Area (TMA) is a crucial segment of airspace where aircraft perform complex maneuvers including rare go-arounds, and holding patterns. The TMA is characterized by high traffic density and diverse trajectory types, necessitating precise, granular data to model these dynamic scenarios accurately. The main research objective of this thesis project was to investigate the applicability of Generative Adversarial Networks (GANs) in generating synthetic 4-dimensional aircraft landing trajectories capable of capturing traffic patterns in this airspace. An unconditional recurrent-based network (TimeGAN) was applied and complemented with an investigation into the merit of conditioning to reap the rewards of transfer learning between classes. The resulting synthetic trajectories were evaluated in terms of data diversity, fidelity and usefulness. The main challenge identified during this research was the imbalance in data classes, which affected the models' ability to accurately capture data patterns, particularly in less frequent scenarios. Generating synthetic data based on separate groupings showed promise in addressing these imbalances, although this approach was sensitive to the number of clusters chosen. Smoothing filters frequently proved necessary to ensure realistic landings were devoid of unwanted noise. Overall, this work proves the capability of TimeGAN in generating diverse, realistic trajectories that are difficult to differentiate from real historical data.

Keywords—Air traffic management, Deep generative models, Generative Adversarial Networks, Multivariate time series generation, Synthetic data quality evaluation

I. INTRODUCTION

As air traffic management (ATM) systems face escalating demands and increasing complexity, advancing automation leveraging artificial intelligence (AI) and machine learning (ML) has become crucial for maintaining safety and oper-

ational efficiency [1]. However, the effectiveness of these technologies is limited by the availability of comprehensive datasets needed for training ML models, constraining the ability of capturing complex patterns. Acquiring sufficient training data is often hampered by high costs and logistical challenges; aptly named the *data access problem* [2]. Moreover, datasets must meet high standards of quality, adhere to data privacy laws, and promote fairness to avoid perpetuating algorithmic biases [3], [4]. The integration of artificial intelligence (AI) and machine learning (ML) stands as a cornerstone for achieving the ambitious goals set forth by the Single European Sky ATM Research (SESAR) Joint Undertaking [5]. With the growing complexity of ATM systems, balancing the varied requirements of passengers, airlines, regulatory bodies, and air traffic controllers becomes increasingly demanding and requires seamless coordination to ensure safe and efficient air travel. To achieve widespread connectivity and system deployment it is paramount to establish trust in automated solutions [6]. Synthetic data generation (SDG) addresses the limitation of lacking comprehensive data by creating realistic datasets that effectively bridge the data gap and allow for model training and validation. By generating accurate synthetic data, one could optimize automation, improve prediction accuracy, and enhance resource allocation within ATM systems.

The Terminal Maneuvering Area (TMA) is of particular interest to model because it experiences high traffic density, complex interactions between arriving and departing aircraft, and varied trajectory patterns, making it crucial for optimizing airspace efficiency [7]. Synthetic 4-dimensional aircraft trajectories, which include latitude, longitude, altitude, and time, help examine airspace constraints and delay propagation. They provide granular data for training ML models in tasks like flight delay prediction and full trajectory extension in synthetic traffic generators. This is one of the use cases considered as part of the SynthAir project [8]. Within the TMA, synthetic trajectories allow for detailed analysis of aircraft behavior during critical phases of flight, such as takeoff and landing. This data supports the design and evaluation of airspace procedures, air traffic control strategies, and collision avoidance systems, ensuring safety and efficiency in congested

airspace [9]. Well-trained models for generating synthetic data for landing phases have a high potential for generalizability to other airports globally and to different flight phases, including en-route operations. This is because landing data offers inherent diversity, encompassing scenarios like go-arounds, holding patterns, weather impacts, capacity issues, and variability in air traffic control (ATC) directives [7].

The majority of existing literature addressing the problem of trajectory generation has been model-driven based on flight dynamic equations [7]. More recently, deep generative modeling techniques for multivariate time series generation have shown significant potential in other domains such as finance [10], healthcare [11], and automotive industries [12]. Generative models, including variational autoencoders (VAEs) [13], generative adversarial networks (GANs) [14], [15], and diffusion models [16], form the core of this literature. The objective of this research is to investigate the applicability of GANs to the ATM setting. In this context, the inherent temporal relationships in time series data, beyond just feature distributions, increase the complexity of the generative process [17]. GANs are promising due to their abilities in generating realistic, high-dimensional synthetic data by learning from existing data distributions, but have virtually never been applied to a broad base on landing data at an airport encompassing multiple runway approach patterns, aircraft type, and operational conditions. A GAN architecture comprises a generator and discriminator and is trained through adversarial learning to generate realistic synthetic data. The generator transforms random noise into synthetic data samples, aiming to deceive the discriminator, which classifies samples as real or fake. The iterative training process involves the generator producing increasingly realistic samples while the discriminator improves at distinguishing between real and generated data [14], [15]. TimeGAN differs from a standard GAN by using a recurrent neural network (RNN) in its architecture to capture temporal dependencies in sequential data and is trained with both supervised and adversarial losses to ensure the generated sequences are realistic and time-consistent [17]. This can be extended with conditioning mechanisms to enhance control of the generation process and enforce diversity of generated samples. Thus, the main research question addressed in this paper: how effective are GANs in generating realistic synthetic 4-dimensional aircraft landing trajectories in an airport's Terminal Maneuvering Area, including go-arounds and holding patterns? Subsequently, the synthetic samples are evaluated to quantify the quality, diversity, fidelity, and usefulness.

The paper commences with collating the literature and state of the art comprising the theoretical background and related work, in Section II. This is framed in three dimensions, namely aircraft trajectory generation, synthetic multivariate time series generation model architectures, and the quality evaluation of synthetic samples. Subsequently, the five-step methodology of applying TimeGAN to the problem of landing trajectory generation is explained in Section III. The approach starts with data preprocessing, labeling, and clustering, before training and ultimately evaluating generated samples based

on several experiments. The latter two stages are inherently iterative in search for optimality. Section IV presents the results of the experiments in terms of the established evaluation framework (data diversity, fidelity, and usefulness), for various (clustered/labeled) training data subsets and (hyper)parameter configurations. Ultimately, the results from all experimentation are integrated and discussed in light of the originally proposed value proposition, in Section V. Concluding remarks alongside future implications of this work are contained in Section VI.

II. RELATED WORK

The theoretical background for this study is based on an assessment of the state-of-the-art in three fundamental research domains, namely prior efforts with respect to the aircraft trajectory generation problem, an overview of deep generative modeling architectures capable of capturing temporal dynamics inherent to multivariate time series data, and the upcoming research in systematically evaluating synthetic samples.

A. Aircraft Trajectory Generation

Methods for generating synthetic aircraft trajectories can be broadly classified into model-driven and data-driven approaches. Model-driven methods use mathematical models of aircraft dynamics and environmental constraints to generate trajectories, where Base of Aircraft Data (BADA) [18] and OpenAP [19] are the most well-known. These deterministic methods often involve optimization processes to achieve objectives like minimizing fuel consumption or flight time [20]. However, their deterministic nature limits their ability to incorporate uncertainty and randomness [7]. Flight plan database statistical extrapolation uses historical data to generate synthetic traffic sets, creating trajectories that replicate real-world traffic patterns. By analyzing and leveraging existing data, this approach can produce realistic synthetic trajectories without requiring detailed physical modeling of aircraft dynamics, though its efficacy depends considerably on the availability and quality of historical flight data [21]–[23]. Markov chain models introduce stochastic processes into trajectory generation, enabling the modeling of uncertainty and randomness. This has been used for estimating conflict probabilities in uncontrolled airspace, as it represents the trajectory as a sequence of states with probabilistic transitions, capturing the inherent variability [24].

Gaussian Mixture Models (GMMs) identify representative trajectory patterns by clustering trajectories [25], [26]. They generate trajectories reflecting the statistical properties of observed flight paths, capturing the diversity and variability of real flight trajectories. By fitting a mixture of Gaussian distributions to the trajectory data, this method effectively represents different common flight paths and deviations from them, making it suitable for scenarios where the data exhibits multi-modal distributions [27]. Most recently, deep generative modeling techniques have been applied to the generation of synthetic aircraft trajectories. A GAN was applied to generate aircraft trajectories for detecting atypical approaches at Paris

Only airport. The model produced realistic synthetic trajectories for the last 25 NM for a single runway and aircraft type, though it faced challenges with mode collapse and required smoothing to remove unrealistic noise [28]. The Temporal Convolutional Variational Autoencoder (TCVAE) presented in [7] represents the acme of the state-of-the-art, serving as the benchmark for this study. The authors developed a model capable of generating 4-dimensional landing paths at Zurich Airport, aimed at estimating aircraft conflict probabilities through Monte Carlo simulations. This model integrates a temporal convolutional network (TCN) to capture sequential dependencies and uses a VampPrior, a variational mixture of posteriors, to better handle complex latent distributions. While high-density latent space regions produced realistic trajectories, low-density areas generated less realistic paths.

B. Deep Generative Modelling - Multivariate Time Series

Three notable families of generative models have seen remarkable success in generating synthetic multivariate time series data: variational autoencoders (VAEs), generative adversarial networks (GANs) and diffusion models. These models form the core of what is proposed in literature, and serve as a springboard to explore adaptability to 4-dimensional landing trajectory generation. The following section discusses the variations in these techniques, alongside trend-breaking discontinuities in this rapidly evolving field.

The Variational Recurrent Autoencoder (VRAE) [29], introduced in 2014, pioneered the use of Recurrent Neural Networks (RNNs) to encode and decode sequential data into a latent space. Despite initial success, limitations in capturing long-term dependencies led to inconsistent findings on alternative benchmark datasets [30]. TrajVAE improved upon this by utilizing Long Short-Term Memory (LSTM) specifically for trajectory generation, integrating spatial embedding and a dedicated feature extraction module [31]. In [32], the Attention-Based Variational Recurrent Autoencoder (AVRAE) integrates attention mechanisms to enhance the model's ability to learn from multivariate input sequences. AVRAE uses cross-attention and self-attention layers to focus the decoder on relevant parts of the input sequence, improving accuracy. The Vector Quantized Variational Autoencoder (VQ-VAE) employs discrete latent spaces and learned priors to reconstruct input data using a codebook of discrete vectors [33]. This approach combined the bidirectional transformer model (MaskGIT) to enhance the model's capability to capture complex patterns in data. The Interpretable TimeVAE incorporates specific temporal structures such as trends and seasonality into its generative process, aiming to improve interpretability and explainability in modeling temporal dynamics [34].

Similar to VAEs, early applications of GANs to sequential data involved integrating RNNs directly into the architecture. The Continuous RNN-GAN (C-RNN-GAN) used LSTM networks in both generator and discriminator roles for generation, incorporating bidirectional discriminators to capture sequential context effectively [35]. Variants like Recurrent GAN (RGAN) and Recurrent Conditional GAN (RCGAN)

further refined these approaches, with RCGAN introducing conditional inputs to the generator to improve sequence fidelity [30]. An alternative approach is the Conditional Sig-Wasserstein GAN (SigCWGAN) [36], which introduces the Signature Wasserstein-1 (Sig-W1) metric to handle long time series data, enhancing temporal dependency modeling without the computational overhead. TimeGAN features an embedding function to transform static and temporal features into a latent space, facilitating efficient learning and reconstruction [17]. TransGAN [37] introduced Transformers for GANs, initially for image generation and later adapted for multivariate time series as in Transformer-Based Time Series GAN (TTS-GAN) [38]. TTS-GAN uses one-dimensional random vectors mapped to an M-dimensional space, employing transformer blocks with self-attention and feedforward layers to process sequences effectively. The discriminator, akin to a Vision Transformer, classifies real versus synthetic signals based on sequence characteristics. TTS-CGAN [39], an extension of TTS-GAN, enhances control over generation by incorporating conditional inputs and classification heads in the discriminator. This conditional approach improves the model's ability to generate diverse and realistic time series data across classes.

Recent advancements in diffusion models have expanded their application beyond time series imputation and forecasting to include synthesis [40]. Initial approaches primarily integrated recurrent elements to capture temporal dynamics, focusing on univariate data and often constrained by sequence length limitations. Early diffusion-based methods for time series synthesis, like DiffWave [41] and ChiroDiff [42], were tailored for specific data types such as audio and chirographic data. TSGM [40] pioneered diffusion-based generative models specifically for time series. It utilizes an encoder-decoder architecture with a conditional score-matching network, employing RNNs to embed time series into a latent space and learning conditional log likelihoods to model temporal dependencies. In addressing the challenge of maintaining temporal properties during diffusion-based generation, constraints have been crucial. Techniques such as DiffTime integrate trend constraints into the diffusion process, conditioning generation to follow predefined trends while incorporating penalty terms to enforce constraints effectively [43]. This approach improves fidelity to real data distributions while ensuring generated samples meet specified criteria. Diffusion-TS couples seasonal-trend decomposition with denoising diffusion models using transformer architectures [44]. This non-autoregressive model leverages encoders and decoders to capture temporal representations, guided by trend and seasonal components inspired by time series analysis.

C. Synthetic Data Quality Evaluation

GANs are well-established in their ability to generate high-quality samples rapidly, but simultaneously suffer lacking comprehensive mode coverage [14]. The instability during the training process and high sensitivity to hyperparameter configurations presents a challenge. Complementary to capturing the diversity in the *true* data, synthetic and real

samples should be indistinguishable from one another, and ultimately be of value for downstream prediction tasks, the latter two hereafter being referred to as data fidelity and usefulness, respectively. To achieve data diversity, synthetic data must match the distribution of real data. It has been proposed to assess this using Principal Component Analysis (PCA) [45] and t-Distributed Stochastic Neighbor Embedding (t-SNE) [46], to visually confirm mode coverage [17]. PCA is a *linear* technique that projects data onto lower dimensions while preserving key variations, making it useful for identifying variances. In contrast, t-SNE is a *nonlinear* technique that focuses on maintaining local structures and clusters by optimizing data embedding to minimize mismatches between high-dimensional and low-dimensional similarities.

Discriminative scores can evaluate synthetic data by training an RNN classifier, such as LSTM, to distinguish between synthetic and original sequences [17]. If the synthetic data resembles the real data closely, the classifier struggles to differentiate them, forming a binary classification task. The discriminative score, which can be formatted into a confusion matrix, quantifies this ability. Performance is further assessed using train/test errors and ROC curves [47]. Predictive scores can evaluate how well synthetic data captures the original data's predictive characteristics [17]. An RNN model, like LSTM, is trained on synthetic data to predict future vectors and then evaluated on the original/real dataset. Metrics like MAE quantify prediction accuracy [137]. Both TRTS (train-real-test-synthetic) and TSTR (train-synthetic-test-real) paradigms are used: TRTS benchmarks performance, while TSTR assesses if synthetic data enhances downstream prediction accuracy [17]. Evaluating synthetic aircraft trajectories for realism can further be assessed using curvature metrics for flight path smoothness [27]. For comprehensive evaluation, [20] suggests using BlueSky [48], an open-source air traffic simulator, to compute distances between synthetic trajectories and their replays in the simulator, using navigational and performance data from AIS and BADA. This approach aligns trajectories with waypoints and optionally simplifies them using algorithms like Douglas-Peucker, but faces simulator limitations for self-intersecting flight paths.

III. METHODOLOGY

This study employs a systematic approach to investigating the applicability of TimeGAN [17] to the aircraft trajectory generation problem, structured in five key stages: data collection and preprocessing (A), trajectory labeling and clustering (B), the TimeGAN model implementation (C), synthetic sample evaluation framework establishment (D), and the training and optimization of the model using a set of *experiments* (E). The latter two stages are repeated in an iterative process, the format and rationale of which are explained in this section. Three fixed/categorical and two variational/sensitivity experiments are conducted pertaining to trajectory grouping and complexity. The following subsections are dedicated to detailing each of the methodological steps. All results and analysis,

with the exception of intermediate results, are reserved for Section IV.

A. Data - Landing Trajectories

Considering the focus on generating synthetic flight movements within the TMA of an airport, a reliable set of historical flight trajectories forms the first requirement. This study makes use of the available dataset of landing trajectories at Zurich Airport (CH), parsed from Automatic Dependent Surveillance-Broadcast (ADS-B), data originally sourced from the OpenSky Network [49], [50]. Zurich is the busiest airport in Switzerland and operates multiple landing and take-off configurations using its three runways (depicted in Figure 1). The circa 18,000 recorded landing trajectories occurred in the months of October and November in 2019, each initiating when the aircraft is within a 40 nautical mile distance from the airport (see Figure 2). The vast majority ($\sim 90\%$) of landings within this two month period were on runway 14, with the remainder usually allocated to runway 28 ($\sim 7\%$) and 34 ($\sim 2\%$).

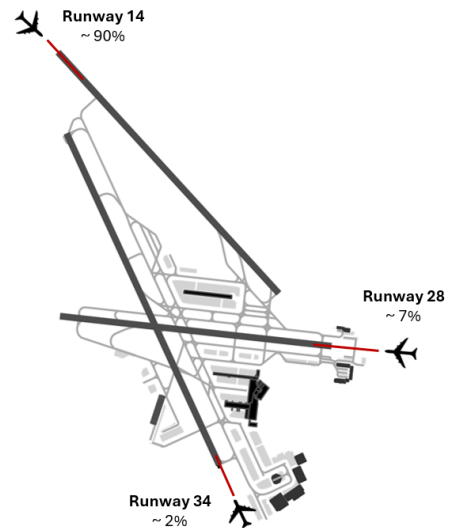


Figure 1. Zurich Airport (LSZH) runway configuration

Compared to take-offs, landing paths are much more complex and varied due to arrivals appearing from any radial and the crucial role of air traffic controllers in sequencing aircraft through varying climate and capacity conditions. This is reflected in occurrences of go-arounds and holding patterns. A go-around (missed approach) is a standard procedure for pilots when landing conditions are unsafe, involving full power, retracting flaps and gear if necessary, and climbing to a safe altitude to re-enter airspace, crucial for preventing runway accidents [51]. Holding patterns are predefined flight paths used by air traffic control to delay aircraft at congested airports or during airspace congestion. Aircraft fly in a racetrack pattern above a fix, awaiting approach clearance, helping to manage traffic flow and ensure safe separation [52].

Given the variation in approach procedures at Zurich Airport, the 18,000 trajectories are safely deemed representative of arrivals at other major airports subject to equal operational

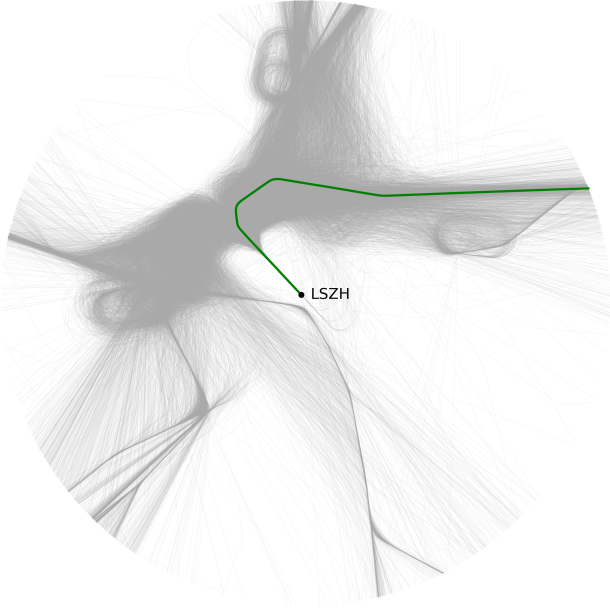


Figure 2. Landing trajectories in October and November (2019) on runway 14 at Zurich Airport, retrieved from the OpenSky Network

constraints due to local terrain, noise abatement, and emission regulation [7]. The value proposition of a synthetic trajectory generator is therefore safeguarded by this hypothesized generalizability to other geographic locations and flight phases (take-offs and en-route). To ensure consistency in trajectory representation, each trajectory is resampled at a fixed rate and condensed to a state vector of longitude, latitude, and altitude at each timestamp. The resampling and simplification of the input dataset was varied during the experimentation/training of the GANs to gain insight into performance sensitivity.

B. Trajectory Labeling & K-Means Clustering using DTW

The labeling of landing trajectories is essential for discerning between different archetypical landing behaviors at Zurich Airport. The two benefits thereof are the ability to conduct separate analysis on various landing sequences to boost explainability of the results, and it opens the door to incorporating conditioning mechanisms to the generative modeling process. The latter becomes especially relevant for addressing lacking mode coverage of GANs by attempting to exploit transfer-learning between labeled classes. Trajectories are divided into three categories: trajectories involving a go-around, involving a holding pattern, and *normal* procedures. Go-arounds and holding patterns occur significantly less frequently, and on multiple runways (visible in Figure 3). Go-arounds are detected based on multiple Instrument Landing System (ILS) alignments of the aircraft, separated by a climbing phase. In some cases, the aircraft lands on a different runway on the second attempt. Holding patterns identification is based on the total change in track angle of the aircraft’s flight path, and setting a minimum cumulative track angle threshold for a trajectory to be labeled as such. This requires the unwrapping of the track angle to handle the

angular discontinuity that occurs when an aircraft crosses the 360 degrees mark, by measuring the angle from -180 degrees to 180 degrees. Additionally, each trajectory is further labeled based on the specific runway used for landing.

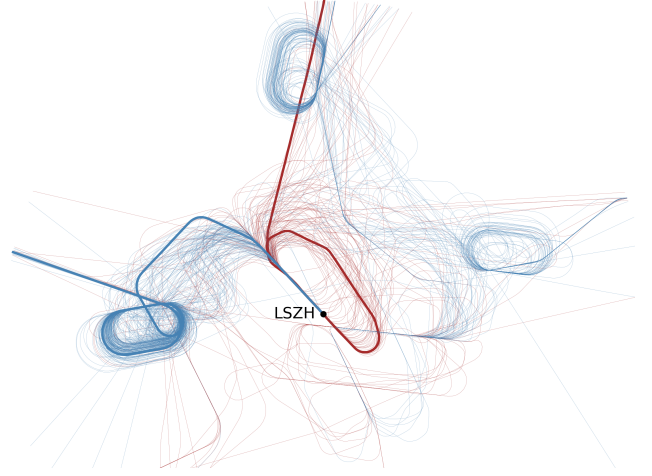


Figure 3. Landing trajectories involving go-arounds (red) or holding patterns (blue) on any runway at Zurich Airport, retrieved from OpenSky Network

Complementary to this rigid and explicit form of labeling, a more flexible approach was taken. Trajectories are also clustered by employing the k-means clustering algorithm with Dynamic Time Warping (DTW) as the dedicated distance metric [53]. Traditional distance metrics (e.g. Euclidean) often fall short when dealing with time series due to their inability to account for temporal distortions. Dynamic Time Warping (DTW) is a well-established algorithm used to measure the distance between two time series, even if they vary in speed [54]. It works by aligning the sequences such that distances between corresponding points are minimized, allowing for shifts and distortions in time. DTW finds an optimal match between points in the two series by stretching or compressing them along the time axis, which is especially useful for comparing sequences with similar overall patterns but differing temporal dynamics. The DTW distance is defined according to Equation 1 (from [53]).

$$DTW(x, y) = \min_{\pi} \sqrt{\sum_{(i,j) \in \pi} d(x_i, y_j)^2} \quad (1)$$

A viable path $\pi = [\pi_0, \dots, \pi_K]$ represents an ordered list of indices: $\pi_k = (i_k, j_k)$ with $0 \leq i_k < n$ and $0 \leq j_k < m$. It is subject to three constraints [from [53], namely:

- $\pi_0 = (0, 0)$ and $\pi_K = (n - 1, m - 1)$
- for all $k > 0$, $\pi_k = (i_k, j_k)$ is related to $\pi_{k-1} = (i_{k-1}, j_{k-1})$ as follows:
 - $i_{k-1} \leq i_k \leq i_{k-1} + 1$
 - $j_{k-1} \leq j_k \leq j_{k-1} + 1$

This similarity metric boils down to an optimization problem that seeks an optimal temporal alignment of the multi-variate time series, such that the subsequent Euclidean dis-

tance between them is at a minimum [53]. This approach ensures that similar-shaped trajectories are grouped together, even if there are temporal misalignments. The clustering was performed using the longitude and latitude coordinates alone. Omission of altitude proved effective as this lowered the dimensionality of the space. The altitude profile is also rather consistent between all trajectories, rendering this feature more informative. The elbow method heuristic allows determination of an optimal number of clusters by plotting the DTW inertia for different cluster numbers. In this study, the number of clusters at initialization was varied from between two and five. Figure 4 depicts the clustering of nominal trajectories on runway 14 for the case of two and four clusters centers, as examples of this technique.

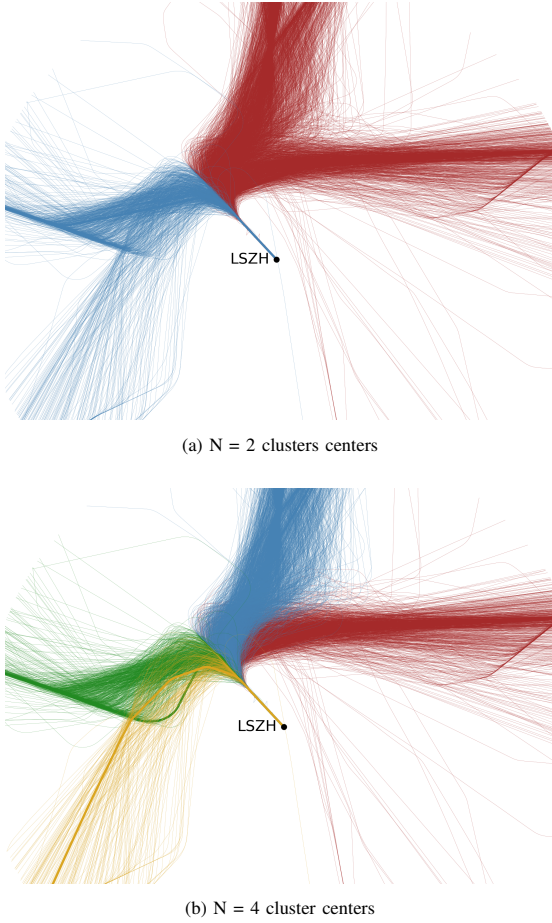


Figure 4. K-means clustering with DTW as a distance metric, for nominal landings on runway 14

C. TimeGAN Model Architecture

TimeGAN [17] synthesizes realistic time-series data using a combination of autoregressive and adversarial methods. The architecture consists of four main components (depicted in Figure 5): the embedding network, the recovery network, the sequence generator, and the sequence discriminator. The embedding network converts high-dimensional time-series data into a compact latent space using RNN layers such as LSTMs

or GRUs. The recovery network reconstructs the original time-series data from the latent space representation, ensuring accurate data preservation. The sequence generator produces synthetic latent sequences, which are then mapped back to the original data space by the recovery network. The sequence discriminator differentiates real sequences from synthetic ones, providing feedback to improve the generator's performance.

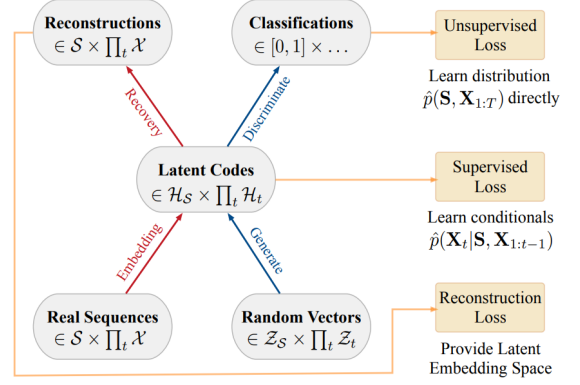


Figure 5. TimeGAN model architecture, retrieved from [17]. S , X , and Z are the static, temporal, and random vector spaces, respectively

TimeGAN uses a combination of supervised and unsupervised learning to capture both data distribution and temporal dynamics. The reconstruction loss ($\mathcal{L}_R$) ensures that the embedding and recovery networks accurately reconstruct the original data from the latent space. This loss is defined as the sum of the differences between the original ($s, x_{1:T}$) and reconstructed ($\tilde{s}, \tilde{x}_{1:T}$) data points, formulated in Equation 2 (from [17]). Where s and t refer to values of vectors in the static and temporal spaces.

$$\mathcal{L}_R = \mathbb{E}_{s, x_{1:T} \sim p} \left[\|s - \tilde{s}\|_2 + \sum_t \|x_t - \tilde{x}_t\|_2 \right] \quad (2)$$

The generator is trained in two modes: open-loop and closed-loop. In the autoregressive mode, it uses its previous synthetic outputs ($\hat{h}_S$ and $\hat{h}_{1:t-1}$) to generate the next synthetic vector ($\hat{h}_t$). This unsupervised loss aims to trick the discriminator, and is formulated in Equation 3 (from [17]). y_S and y_t are the discriminator's outputs for real data, and $\hat{y}_S$ and $\hat{y}_t$ for synthetic data.

$$\begin{aligned} \mathcal{L}_U = \mathbb{E}_{s, x_{1:T} \sim p} & \left[\log y_S + \sum_t \log y_t \right] \\ & + \mathbb{E}_{s, x_{1:T} \sim \hat{p}} \left[\log(1 - \hat{y}_S) + \sum_t \log(1 - \hat{y}_t) \right] \end{aligned} \quad (3)$$

In tandem, a supervised loss ($\mathcal{L}_S$) ensures that the generator models step-by-step relationships in time-series by using real data embeddings ($h_{1:t-1}$), computed by the embedding network to generate the next latent vector, in Equation 4 (from [17]).

$$\mathcal{L}_S = \mathbb{E}_{s, x_{1:T} \sim p} \left[\sum_t \|h_t - g_X(h_S, h_{t-1}, z_t)\|^2 \right] \quad (4)$$

D. Evaluation Framework

The evaluation framework used to determine the quality of synthetic samples rests on four pillars: data diversity, fidelity, usefulness, and a statistical distance metric between real and synthetic sample distributions.

To assess *diversity* of synthetic time series data, a dimensionality reduction approach was implemented to visualize and compare the distribution of original and synthetic samples. A subset of samples can randomly be selected from both datasets to ensure computational efficiency. The data can also optionally be preprocessed by averaging along a specified dimension: *sequence averaging* computed the mean for each feature across sequence length, while *sample averaging* averaged across the feature dimension for each time step. Principal Component Analysis (PCA) and t-Distributed Stochastic Neighbor Embedding (t-SNE) were then applied to reduce the data to two dimensions. PCA provided a 2D representation of the data, facilitating a comparison of distributions, whereas t-SNE emphasized local structures and neighborhood relationships. The resulting 2D projections were visualized using scatter plots, with original and synthetic samples distinguished by color. By examining these visualizations, one can determine whether the synthetic data effectively captures the original diversity, thereby indicating the quality and effectiveness of the synthetic data generation process.

To assess the *fidelity* of synthetic time series data, a discriminator model was employed to evaluate the ability of an LSTM network to distinguish between original and synthetic data. The process involved several key steps. Initially, the data was split into training and testing sets. The original and synthetic datasets are partitioned using a specified test size, ensuring that both sets were equally represented in the training and testing phases. A discriminator model was then built and trained. This model, comprising of three LSTM layers followed by a dense output layer, was compiled with the Adam optimizer and binary cross-entropy loss function. It was trained to classify samples as either original or synthetic, using concatenated datasets and corresponding binary labels. Following training, the discriminator was evaluated on separate test sets of original and synthetic data.

The model's performance was quantified by its accuracy through the generation of a confusion matrix. The three key metrics to provide understanding of the model performance are True Positive Rate (TPR), True Negative Rate (TNR), and Accuracy (all averaged for multiple runs). In this context, *positive* indicates *synthetic* and *negative* indicates *real* samples. True Positive Rate (TPR), also known as Recall or Sensitivity, measures the proportion of actual positive instances that are correctly identified by the model. True Negative Rate (TNR), also known as Specificity, assesses the proportion of actual negative instances that are accurately identified by the model.

Accuracy provides a general measure of overall correctness by calculating the proportion of all instances that are correctly predicted, both positive and negative. Data quality is implied by lower scores of each of these metrics, signaling that a trained model finds it difficult to differentiate between real and synthetic samples.

The *usefulness* of synthetic time series data was evaluated by measuring the predictive performance of an LSTM model trained on synthetic versus real data. The evaluation was conducted using a time series regression approach and was structured into several key steps. Initially, the real time series data were split into training and testing sets. Both of those datasets were partitioned into an input sequence used to predict future values, with the last time step serving as the label/output. The input sequence can be trimmed to omit values at the end of the sequence to create an n-steps ahead prediction problem. This is analogous to predicting an aircraft state vector further into the future. Similarly, synthetic data was prepared by extracting training sequences and corresponding labels based on the same prediction horizon. A three-layer LSTM model was then designed for time series regression. This model was trained separately on both the real and synthetic datasets to predict future values. During training, the model was optimized using the Adam optimizer and mean absolute error (MAE) as the loss function. After training, the performance of each model was evaluated on the real test data to measure how well the synthetic data-trained model generalized to unseen real data (for multiple runs). The MAE of predictions from models trained on both synthetic and real data was compared, providing insight into the predictive power of the synthetic data. The results included the training metrics for models trained on synthetic and real data, as well as the MAE on real test data for both models. This evaluation process facilitated a comparison of the predictive performance and overall usefulness of the synthetic data relative to the real data.

The *energy distance* metric was used to evaluate the similarity between synthetic and real time series data. The energy distance measures the divergence between the distributions of two sets of random vectors [55]. It approaches a positive constant if the distributions are similar and increases towards infinity if they differ significantly. Each time series was flattened and normalized to the range between -1 and 1, using MinMaxScaler. Energy distances were computed over 100 random subsets of 500 samples from both the true and synthetic datasets. The mean energy distance was then used to assess which generation method produces synthetic trajectories whose distribution is closest to the observed data [7]. A lower mean energy distance indicates that the synthetic data more closely approximates the distribution of the real data.

E. Experiments

The exploratory nature of this study is reflected in the design of five experiments, that will be conducted for the TimeGAN model. Three of these are aimed at investigating data quality of synthetic samples when TimeGAN is trained on

either clustered groups (using k-means with DTW), separate runway assignments, and flight operation (normal, go-around, holding pattern). These experiments aim to investigate whether or not TimeGAN can successfully generate realistic landing trajectories for data classes of decreasingly small datasets; the optimal results in each case are presented. The remaining two experiments are variational (resembling sensitivity analyses), where underlying variables in the data preprocessing step are varied while monitoring the effect on data quality. The cumulative track angle change (complexity proxy) and number of cluster centers are varied one-by-one to verify the robustness of the TimeGAN model in this context, and boost explanatory power for the variation in results. Experimental results are presented in Section IV, with hyperparameter tuning and noteworthy model behavior provided in the Appendices. The experiments are numbered and outlined here.

- *Experiment 1: Cluster-Based Generation* - This experiment evaluated TimeGAN’s ability to generate realistic synthetic landing trajectories when trained on clustered groups of flight data. Trajectories were grouped into five clusters using k-means with Dynamic Time Warping (DTW), with each cluster representing different flight patterns (visually confirmed main approach patterns at Zurich). The model was trained separately on each cluster to learn specific trajectory characteristics. Cluster-specific synthetic datasets could subsequently be combined to form the full dataset.
- *Experiment 2: Minority Runway-Specific Generation* - This experiment investigated the capacity of TimeGAN to replicate landing trajectories associated with different runway assignments. It focused on simulating landings on two distinct runways - 28 and 34 - each with unique approach patterns. Whereas runway 14 has a high frequency of use, data for runways 28 and 34 is limited, presenting a heavy data class imbalance challenge for the model generalization. The experiment examined whether TimeGAN can generate runway specific data despite uneven data representation.
- *Experiment 3: Rare Operational Scenario Adaptability* - This experiment evaluated TimeGAN’s ability to generate synthetic data for less frequent and challenging flight operations, namely go-arounds and holding patterns. Unlike normal landings, which are more common and serve as a baseline, go-arounds and holding patterns contain more abrupt and sustained maneuvers and constitute a small minority in historical data. The experiment assessed whether TimeGAN can effectively handle these rare scenarios and maintain data quality and realism in situations that deviate from standard landing operations.
- *Experiment 4: Increased Complexity Challenge* - This experiment examined how TimeGAN performs when generating synthetic trajectories with different levels of complexity, as measured by cumulative track angle change. Complexity thresholds of 180 and 270 degrees were used to represent varying degrees of flight path

variability. Higher thresholds correspond to more intricate trajectories involving greater changes in direction. The experiment evaluated the model’s ability to accurately reproduce these complex patterns, providing insights into its sensitivity and robustness with more varied datasets.

- *Experiment 5: Robustness to Cluster Count Reduction* - This experiment analyzed the impact of reducing the number of clusters on TimeGAN’s ability to generate high-quality synthetic data. By decreasing the cluster count from five to two, the model was tested on its capacity to handle broader and more diverse trajectory distributions. Fewer clusters translate to each group covering a wider range of flight patterns, increasing the challenge for the model to maintain fidelity and consistency. The experiment aimed to assess the model’s adaptability and robustness in generating realistic data under conditions of reduced specificity.

IV. RESULTS

This section provides an analysis and evaluation of the synthetic data generation capabilities of TimeGAN through the five experiments described in Section III.E. By analyzing performance across various scenarios, the strengths and limitations of the model are identified. Insights into data quality and model adaptability under different conditions are presented.

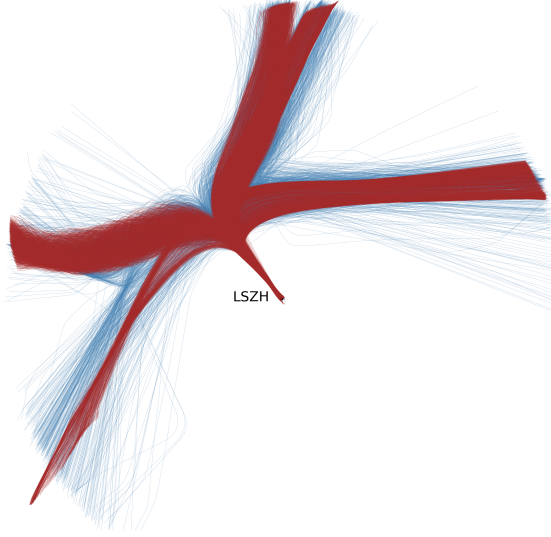
A. Experiment 1: Cluster-Based Generation

For the set of landing trajectories on runway 14, sampled at 100 data points, filtered for a maximum cumulative track angle change of 180 degrees (representing simpler trajectories), TimeGAN was trained on each cluster separately. The data imbalance for trajectories per cluster is evident considering the range of 364 to 1212 trajectories per cluster. The complexity of aircraft trajectories closer to the 40 NM cut-off from an airport is significantly higher. Within this critical distance, aircraft must adhere to Standard Terminal Arrival Routes (STARs), which are intricate with multiple way points, altitude restrictions, and speed constraints to ensure safe and orderly arrivals. Air Traffic Control (ATC) provides continuous instructions for descending, speed adjustments, and precise vectoring to align aircraft with the final approach course, adding to the complexity. Controllers must also manage the sequencing and spacing of multiple converging aircraft to prevent wake turbulence and maintain safety. Adverse weather conditions and noise abatement procedures further complicate the approach phase, requiring deviations from standard procedures and additional vectoring, all of which contribute to noisy and intricate data. To address this noise and improve data quality, a smoothing filter, such as a simple moving average, may be applied during post-processing.

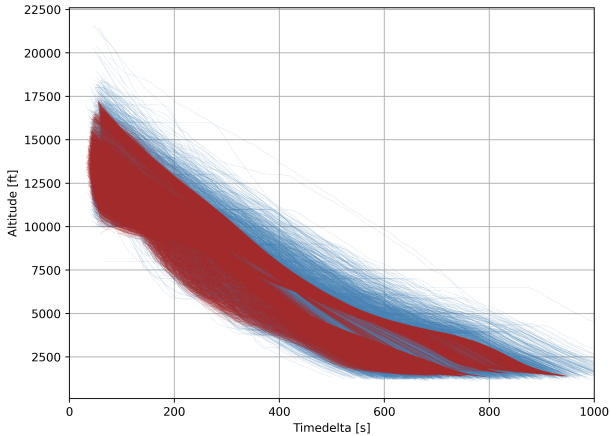
For a given sequence of state vectors describing the trajectory $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n$, where $\mathbf{x}_t = [x_{t,1}, x_{t,2}, x_{t,3}, x_{t,4}]$ with each vector having four features (longitude, latitude, altitude, and time), the moving average of window size w for the k -th feature at position t is defined as:

$$y_{t,k} = \frac{1}{w} \sum_{i=0}^{w-1} x_{t-i,k}, \quad k = 1, 2, 3$$

where $y_{t,k}$ is the output of the moving average filter for the k -th feature at position t , and $x_{t-i,k}$ represents the k -th feature of the input data points within the window size w . Following the post-processing, trajectories appear more realistic, but some remain unnecessarily noisy during the initial phase. The results for the separate classes are combined and plotted in Figure 6, after the application of filter with a rolling window of seven state vectors.



(a) Coordinate profiles for synthetic (red) and real (blue) trajectories; moving average filter (window = 7)



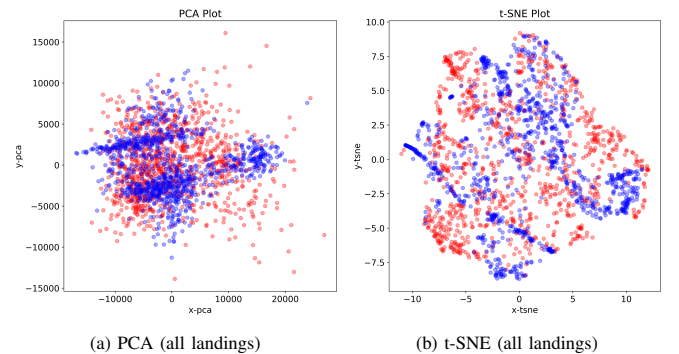
(b) Altitude profiles for synthetic (red) and real (blue) trajectories; moving average filter (window = 7)

Figure 6. TimeGAN synthetic samples (per cluster)

The trajectory distribution is wider and sparser at the start of the trajectories (when the aircraft is in the air, 40 NM from the runway), reflected in the difficulty of TimeGAN to properly capture this. The latter half of each trajectory is often considerably smoother, whereas the start sometimes contains jagged paths resembling noise due to overfitting of the model.

With ground speed being a decreasing function over time, data points at the start of each trajectory are more separated. This translates into a differential complexity along a trajectory that the GAN must capture. The distribution of longitude, latitude, and altitude become more narrow with as function of time. GANs are notoriously unstable during training due to the adversarial dynamics between the generator and discriminator, which can lead to significant variations in performance and realism across different clusters of landing trajectories. When training TimeGAN with the same parameters across various clusters, the instability can become pronounced because each cluster may have distinct characteristics and data distributions. This discrepancy can cause the GAN to perform well on some clusters while struggling with others, as the generator might be better at capturing the features of more common or less complex clusters but fail to generalize to more varied or less frequent ones. In Figure 6, it is evident that TimeGAN struggles the most with flights approaching from the west as there is a higher concentration of noise. Flights from the southwest are less frequent, and TimeGAN falls victim to mode collapse. The remaining three clusters are well captured by the model, and the synthetic samples are of high quality.

Pertaining to the full sets, both PCA and t-SNE analysis demonstrate reasonable coverage of the trajectory distribution, largely steering clear of mode collapse (visualized in Figure 7). In both cases however, clear concentrations are observed, as opposed to the more uniform spread in the original data. The overlap in PCA suggests that the real and synthetic flight paths share similar broad patterns, likely in terms of overall variance in features like altitude or speed. However, the differences revealed by t-SNE indicate that there are subtle, non-linear variations in the flight paths, such as small trajectory deviations or unique maneuvers, that distinguish the real paths from the synthetic ones. t-SNE captures these local differences that PCA, being a linear method, cannot detect. During hyperparameter tuning, the risk of mode collapse proved to be proportional to the number of iterations. Simultaneously, an inadequate number of iterations leads to notable mode collapse at the start of a trajectory, where the model resorts *averaging* of the real data, with a minuscule standard deviation in the starting point (see Appendix C).



(a) PCA (all landings)

(b) t-SNE (all landings)

Figure 7. PCA & t-SNE analysis for the combined set of trajectories; synthetic (blue) and real (red) trajectories

The results for the assessment of data fidelity and usefulness are provided in Table I. Pertinent to data usefulness, the Train-Real-Test-Real (TRTR) and Train-Synthetic-Test-Real (TSTR) 5-step ahead prediction performance figures are included. The quoted value is a percentage change in performance (in MAE) of the TSTR compared to the TRTR evaluation paradigm. Lastly, e-distance provides a quantitative metric for the distance between the synthetic and real trajectory distributions (complementary to the qualitative PCA and t-SNE analyses). The quoted value is the average of 100 retrials, each consisting of sets of 1500 samples. The mediocre accuracy indicates the trained discriminator is fooled often. Real and synthetic samples are difficult to differentiate, and the higher TPR indicates that this largely stems from synthetic samples misclassified as real (as opposed to the inverse). The low e-distance value suggests a high degree of similarity between the real and synthetic data distributions. With regard to usefulness, the trained LSTM seems to learn similar patterns for both datasets. If the real and synthetic datasets have nearly identical time series patterns, trends, and variations, the model’s performance will be very similar on both datasets. Simultaneously, this diminishes the *usefulness* of the augmentation with synthetic samples for predictive power.

TABLE I. PERFORMANCE METRICS (ALL LANDINGS)

Metric	Value
Accuracy	0.608
True Positive Rate (TPR)	0.947
True Negative Rate (TNR)	0.269
TSTR vs. TRTR	-0.473%
E-distance	1.482

The combination of all synthetic samples masks model limitations for certain clusters, alluded to earlier. For this reason, the quality assessment has been performed on an individual cluster basis. The varying performance of TimeGAN for different clusters is clearly visible in PCA and t-SNE plots for separate clusters. Figure 8 depicts the results for two clusters with a large discrepancy in diversity and similar coverage of the synthetic samples. This coverage can be matched with visual inspection of the trajectories in Figure 6. Whereas PCA aligns, the t-SNE plot suggests considerable latent differences between the real and synthetic samples in both clusters. Table II contains the remaining performance indicators for each of the five clusters, showcasing similar variations. In some cases, figures are less representative due to fewer samples within a cluster.

TABLE II. PERFORMANCE METRICS (PER CLUSTER)

Metric	N	NE	E	SW	W
Accuracy	0.737	0.651	0.704	0.869	0.728
TPR	0.707	0.697	0.722	0.893	0.745
TNR	0.767	0.603	0.685	0.844	0.709
TSTR vs. TRTR	-0.224%	-0.127%	-0.152%	-2.401%	-0.122%
E-distance	2.880	4.621	5.500	4.856	3.210

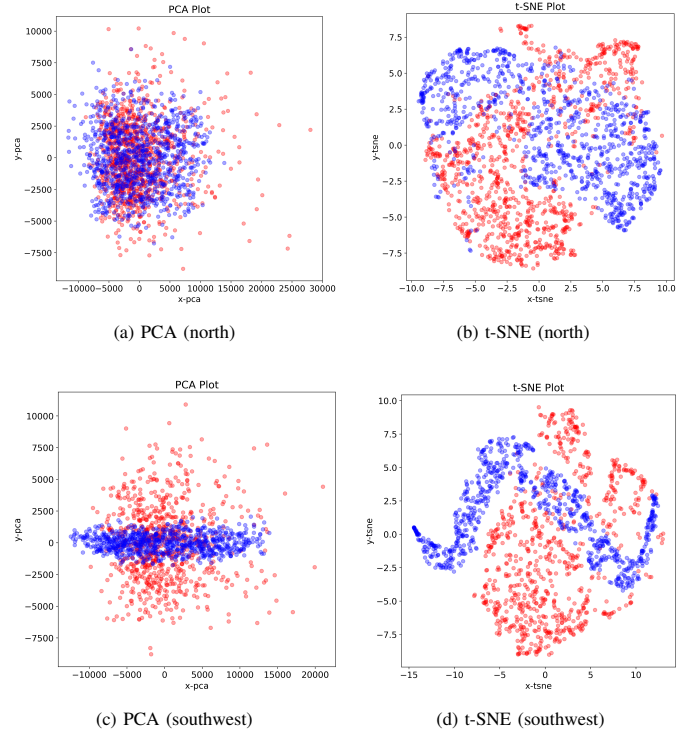


Figure 8. PCA & t-SNE analysis for separate sets of trajectories; synthetic (blue) and real (red) trajectories

In many cases, the LSTM model performing binary classification cannot accurately distinguish between real and synthetic samples. In the case of flights approaching from the southwest, the real and synthetic samples are seen to be most accurately distinguished. Interestingly, the results for the TSTR and TSTR evaluation metrics prove an indifference in prediction ability (except for flights originating from the southwest). The e-distance values all approach a constant positive value, which by the design of this test point to similarity in the distributions. This ensures sampling from the learned model generates consistently realistic results.

B. Experiment 2: Minority Runway-Specific Generation

The results for runway 28 and runway 34 were surprisingly positive (depicted in Figure 9), demonstrating the effectiveness of TimeGAN in generating realistic synthetic landing trajectories despite the limited data available. For runways 28 and 34 the model achieved notable success in replicating landing patterns. This performance can partially be attributed to the choices made during pre and post-processing stages. By keeping the complexity of trajectories low, with a threshold for maximum cumulative track angle set to 180 degrees, and using a low sampling rate of 75, the data remained manageable and less prone to high variability. Additionally, a robust moving average filter with a rolling window of 13 was applied, effectively smoothing out noise and enhancing the clarity of the synthetic trajectories.

Despite the lower data count for runway 34 leading to some noise during the final turn towards runway alignment, the

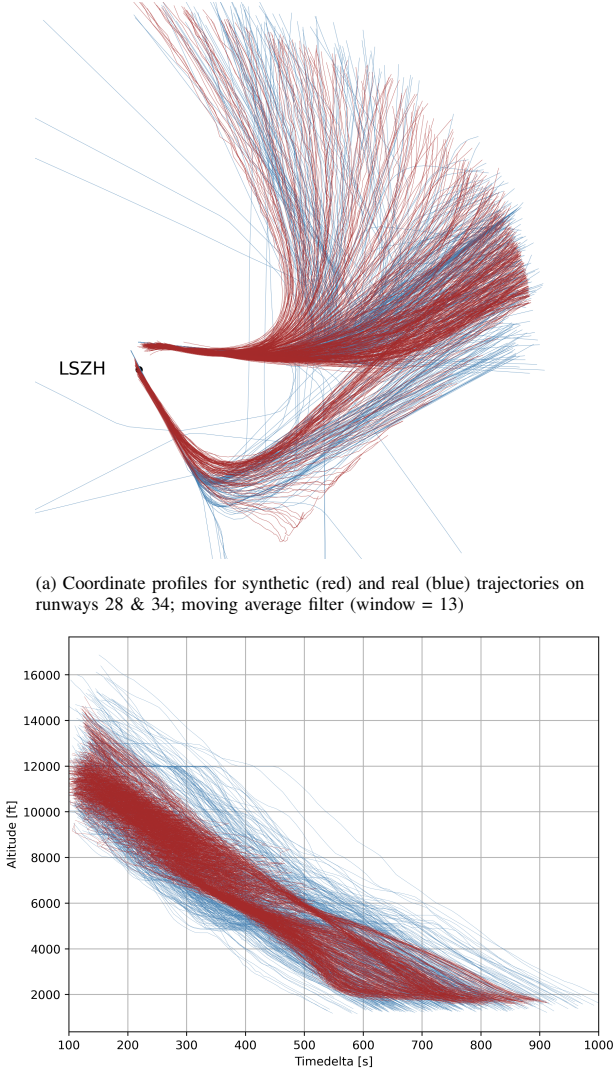


Figure 9. TimeGAN synthetic samples (per runway)

combination of these preprocessing techniques helped mitigate data sparsity issues and allowed TimeGAN to generate trajectories that were closely aligned with real landing patterns. These results highlight TimeGAN’s capability to produce high-quality synthetic data even in situations with limited original samples, underscoring its potential for applications requiring reliable synthetic trajectory generation. The altitude profiles are seen to suffer more from noise, even with the application of a moving average filter. This noise is concentrated at the start of the trajectories and hints to a differential over-underfitting phenomenon along the temporal dimension, as well as between geographic and altitude features. The data diversity of both runways is visualized in Figure 10. Runway 28 is seen to be more covered according to both PCA and t-SNE analysis. This can be largely attributed to the fact that runway 28 contained nearly four times as many historical samples to train from. Additionally, despite the complexity cutoff angle, flights

approached from more varied angles. This higher variation on runway 34 combined with its inherent rarity for landing usage put strain on the stability of the training process.

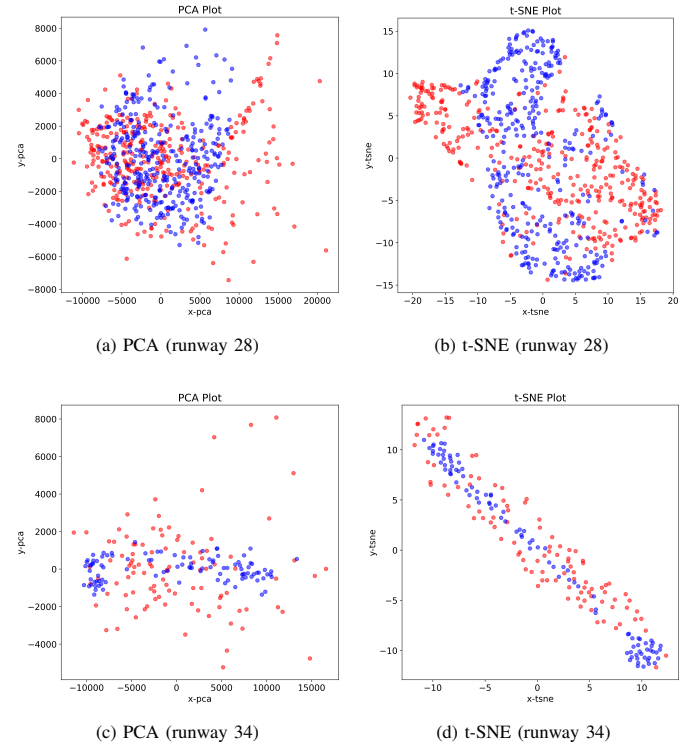


Figure 10. PCA & t-SNE analysis for separate sets of trajectories; synthetic (blue) and real (red) trajectories

Overall, the data fidelity and usefulness analysis demonstrates that TimeGAN performed relatively consistently across both runways, with similar accuracy metrics and slight differences in e-distance. This finding is significant because it indicates that the model’s performance does not degrade substantially with fewer samples. The TSTR vs. TRTR values provided limited additional insight. The lower e-distance for runway 28 suggests that the synthetic trajectories for this runway were slightly closer to the real data, which is further supported by the diversity assessment results. These findings highlight TimeGAN’s ability to generate realistic synthetic trajectories even with varying data sizes and emphasize the effectiveness of preprocessing steps in mitigating the impact of data limitations.

TABLE III. PERFORMANCE METRICS (PER RUNWAY)

Metric	Runway 28	Runway 34
Accuracy	0.694	0.704
TPR	0.528	0.722
TNR	0.861	0.685
TSTR vs. TRTR	-0.129%	-0.144%
E-distance	3.198	3.795

C. Experiment 3: Rare Operational Scenario Adaptability

This experiment highlights significant challenges in incorporating complex flight maneuvers like holding patterns and go-arounds into TimeGAN training. Holding patterns, as in Zurich's airspace, involve intricate circular routes with variable entry points depending on daily conditions. Modeling these patterns in isolation might be feasible but useless on its own. The problem intensifies when integrating them with the full trajectory. Aircraft can enter the same holding pattern from various directions, creating a high-dimensional space that the current dataset struggles to cover adequately. A similar conclusion is made with regard to the mere 50 go-around samples available in the dataset. This leads to difficulties in capturing the full range of maneuver behaviors, with TimeGAN prone to overfitting noise from the remaining trajectory and failing to generate realistic representations. TimeGAN must navigate a vast and complex space of potential maneuvers with sparse data points, which can lead to incomplete learning and poor generalization. Instead, following a similar procedure as laid out in [27], go-around trajectories are pre-processed to start when the aircraft initiates the go-around, and end when it has landed. With fewer than 50 samples it remained infeasible to generate realistic samples (with post-processing) and results in unflyable trajectories, as depicted in Figure 11.

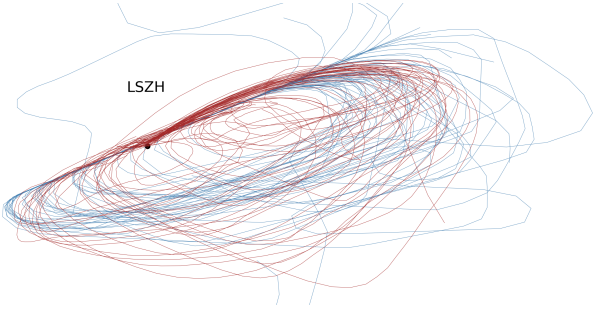


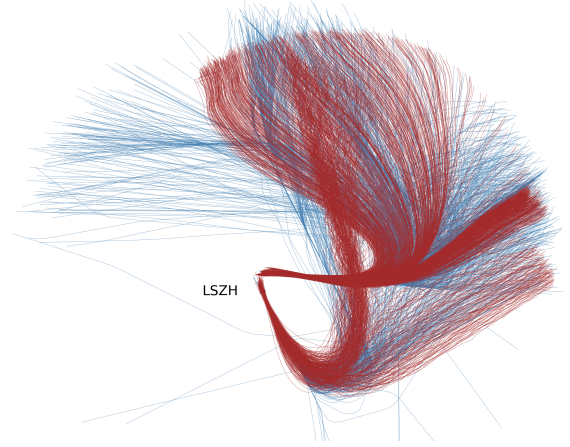
Figure 11. Coordinate profiles for synthetic (red) and real (blue) go-arounds

To address these issues, transfer learning offers a valuable strategy. By leveraging models pre-trained on standard landing trajectories, TimeGAN can apply learned representations to better handle the complexities of go-arounds and holding patterns. This approach allows the model to build on the more abundant data of complete landing trajectories, adapting this knowledge to the temporary segments of go-arounds and holding patterns. Transfer learning could thereby bridge the gap created by sparse data. A discussion on this value proposition alongside the potential merit in other experiments is provided in Section V.

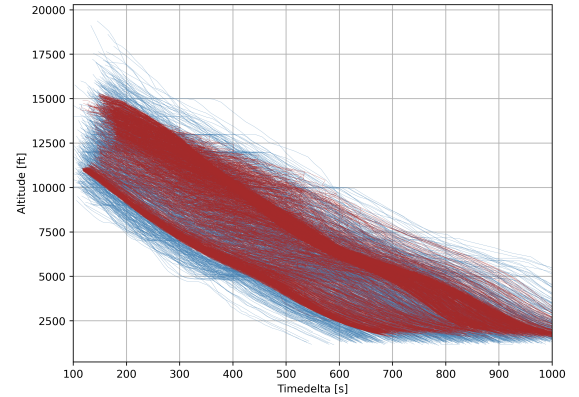
D. Experiment 4: Increased Complexity Challenge

This experiment evaluated how trajectory complexity influences TimeGAN's synthetic data generation capabilities, focusing on the effects of cumulative track angle change thresholds of 180 and 270 degrees. As the complexity increases, the requirement for more extensive post-processing, such as smoothing with a moving average, becomes evident.

Despite these adjustments, TimeGAN encounters difficulties in capturing the full diversity of high-complexity trajectories. While the generated data remains flyable based on geographical and altitude profiles, the model tends to concentrate on more common patterns, resulting in a narrower range of trajectories (evident in Figure 12). For example, flights approaching from the west are complexity unaccounted for in the synthetic dataset. This indicates that the model struggles to generalize across the full spectrum of trajectory types as complexity rises.



(a) Coordinate profiles for synthetic (red) and real (blue) trajectories on runways 28 & 34; moving average filter (window = 13)



(b) Altitude profiles for synthetic (red) and real (blue) trajectories on runways 28 & 34; moving average filter (window = 13)

Figure 12. TimeGAN synthetic samples (per runway), inclusion of higher complexity landings

To address this issue and improve the representation of diverse trajectories, it may be necessary to refine the clustering approach for each specific runway. By further grouping or clustering data within a runway based on more granular characteristics, TimeGAN could better capture and differentiate between the various patterns of high-complexity trajectories. This approach could help the model to generate a more comprehensive range of realistic flight paths by reducing the focus on prevalent patterns.

E. Experiment 5: Robustness to Cluster Count Reduction

The sensitivity analysis of cluster count reduction reveals several nuanced behaviors in TimeGAN’s ability to generate trajectories. With three clusters, the model exhibits volatile behavior, particularly for flights originating from the west (Figure 13). This volatility suggests that the model struggles to maintain consistent learning when the data is partitioned into a limited number of clusters. When the number of clusters is further reduced to two, the model appears to attempt to learn the separate groupings within each cluster, but this consolidation results in a lack of coverage of historical real patterns (Figure 14). TimeGAN’s focus on internal cluster variation may cause it to miss broader, more representative patterns, thus compromising the overall fidelity of the generated outputs. In general, increasing the number of clusters provides a more detailed representation of trajectory types, allowing the GAN to capture specific variations within each cluster. However, this can lead to challenges like data sparsity, where smaller clusters have fewer data points. This sparsity can cause overfitting, limiting the ability to generalize and generate diverse trajectories. Conversely, reducing the number of clusters consolidates data into larger groups, simplifying the learning process. Larger clusters provide more robust datasets, reduce overfitting risks, and improve training stability, helping the TimeGAN model capture broader trends and develop a more generalized understanding of the data. When pushed to the extreme in the form of a single cluster (meaning no clustering algorithm is employed), TimeGAN falls victim to mode collapse at the start of the trajectory.

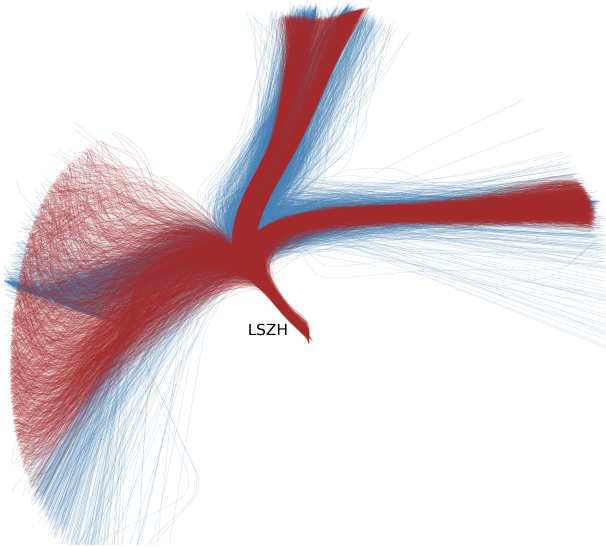


Figure 13. Coordinate profiles for synthetic (red) and real (blue) trajectories for three clusters; moving average filter (window = 7)

An intriguing aspect of these findings is the role of ‘noise’ in synthetic samples—variations that deviate from the historical true data. In the context of generative modeling for flight trajectories, some degree of noise can be beneficial. Instead of merely replicating observed patterns, introducing controlled variations could allow the model to generate new flight paths

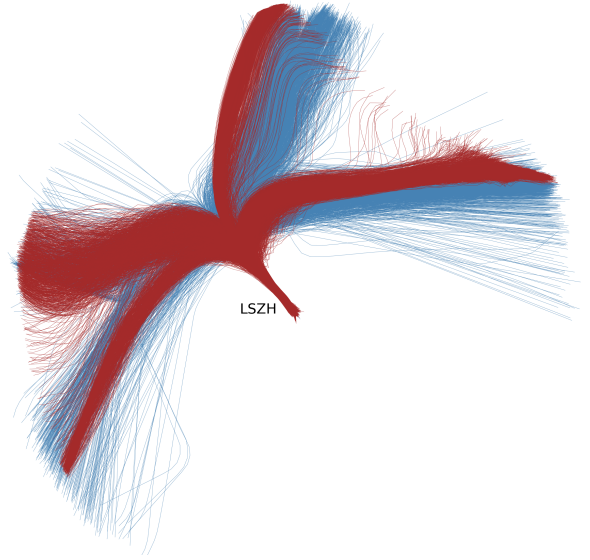


Figure 14. Coordinate profiles for synthetic (red) and real (blue) trajectories for two clusters; moving average filter (window = 9)

that have not yet been observed but are plausible within the operational constraints. This ability is particularly valuable in the context of evolving airspace management strategies, where there may be less strict adherence to STARS in the future. A model capable of synthesizing trajectories that anticipate such changes could provide significant value, enabling ATM to explore and prepare for new operational scenarios. Thus noise, when appropriately managed, could enhance the model’s utility by offering a richer set of possible trajectories that extend beyond the confines of historical data. This of course puts into question the applicability of data diversity, fidelity, and usefulness as this rewards the generation of datasets that match the real sets in terms of distribution. Moreover, the relationship between the characteristics of the input dataset and the optimal number of clusters or groupings remains inadequately defined. Determining a more precise relationship is essential to tailor the model architecture effectively to the specific complexities of the dataset. An empirical guideline could be established given similar testing on multiple other airports.

V. DISCUSSION

TimeGAN demonstrated the potential to generate high quality synthetic aircraft landing trajectories, but encountered challenges, especially when dealing with data imbalances. Specifically, there was significant trouble in adequately capturing the underlying trajectory distribution in sparser areas, notably comparatively rare operational conditions. Training and generation based on individual clusters proved to be a promising approach. However, this method was highly sensitive to the number of cluster centers, meaning that careful consideration is needed when clustering the data to avoid poor generation quality in less populated clusters. Generation of landings on specific runways could be effectively trained for separately, but operational conditions (go-arounds and holding

patterns) proved infeasible. Sparser regions of trajectory data distributions could be learned by optimizing the sampling rate parameter in the preprocessing step. A higher sampling rate provides more frequent data points, increasing the dimensionality of the data space and allowing the GAN to better capture intricate patterns and variations in landing trajectories. It also requires more computational resources and longer training times. Conversely, a lower sampling rate reduces the dimensionality and computational burden, simplifying the data and making it easier for the GAN to capture temporal dependencies, particularly in shorter sequences. However, this simplification may result in the loss of important details, producing less accurate and varied outputs. Therefore, finding the optimal sampling rate is crucial: a higher rate enhances detail and complexity, while a lower rate reduces complexity but risks oversimplification. The optimal rate depends on the number of samples per set, with fewer observations necessitating a lower sampling rate to avoid the curse of dimensionality.

Previous work centered around go-arounds used a two-step process of dimensionality reduction and multivariate statistical distribution modeling [27]. By filtering and normalizing a dataset of 407 trajectories and applying a method to map two-dimensional aircraft paths to a single dimension, the authors achieved more realistic synthetic data with Gaussian mixture models (GMM) and statistical copulas. However, the study's limitations included its specificity to a single airport and runway, the exclusion of irregular flight movements, and the overall lacking potential for generalizability. Compared to the 400 go-arounds in their dataset, the 45 available in this case proved insufficient. In [28], the authors applied a convolutional GAN to generate landing trajectories and detect anomalies at Paris Orly Airport, focusing on Airbus A320 aircraft. Similar to TimeGAN in some cases, this approach required smoothing to remove high-frequency noise. The authors relied on trajectories that started at 25 nautical miles from the airport, inevitably decreasing complexity and variation compared to the 40 nautical mile cut-off used in this study. No clustering or labeling of trajectories was performed, limiting the controllability of the generation process. The most recent work involved an integrated temporal convolutional network (TCN) to capture sequential dependencies and a Variational Mixture of Posteriors (VampPrior) for the prior distribution, in a VAE architecture [7]. The authors used the same data but excluded go-arounds from the set. Sparser regions of the learned latent space resulted in unrealistic trajectories. The e -distance values obtained for the TCVAE are lower than for TimeGAN, which could indicate greater similarity. Compared to all previous work, this study expanded the evaluation framework of synthetic samples. The inclusion of assessments on data diversity, fidelity, and usefulness were particularly valuable in revealing varying performance among classes.

This study's assumptions, while necessary for model development, introduce limitations that impact the generalizability and broader applicability of its findings. The focus on a specific airport (Zurich, LSZH) and trajectory type (landing) assumes that this data is representative of other airports,

which simplifies the modeling process but restricts the model's applicability to diverse environments. This assumption may not hold true across airports with different layouts, traffic patterns, or environmental conditions, potentially limiting the effectiveness of the generated synthetic data in broader air traffic management scenarios. Testing of the established methodological framework established in this work can unlock value for multiple stakeholders. Airports can use it to optimize traffic flow and enhance operational efficiency through improved simulations of peak demand scenarios. Airlines can leverage synthetic data for route optimization, cost reduction, and safety assessments, while air navigation service providers can test and refine air traffic control strategies in a risk-free environment, improving airspace management and mitigating potential disruptions. Researchers can use it to develop and test new air traffic management algorithms, study flight efficiency, or simulate rare safety-critical scenarios without needing access to proprietary or sensitive data.

The concept of a Universal Time Series Generator (UTG) provides the ultimate form of generalization. By training on data from a few specific airports, the UTG is able to generate synthetic flight path data for entirely different airports. This capability allows for the modeling and optimization of operations at unfamiliar and even hypothetical airports without needing real-world data from those specific locations. Incorporating controlled noise in generative models for flight trajectories could enable the synthesis of plausible, yet unobserved paths that anticipate future changes in airspace management. This approach challenges traditional notions of data fidelity and distribution-matching, suggesting a broader utility for models that extend beyond historical data. However, the unclear relationship between dataset characteristics and optimal clustering calls for further empirical research to tailor models effectively for evolving operational scenarios. This can call into question the evaluation metrics, as they implicitly reward models for generating datasets that closely resemble historical data, potentially overlooking the value of generating novel, plausible trajectories that deviate from past patterns but align with future operational needs. Further research is needed to validate these models across diverse datasets and explore alternative or complementary approaches that can better capture the complexities of different ATM scenarios.

Conditioning Extension & Transfer Learning

Conditioning in generative models involves guiding the data generation process based on specific labels or constraints to produce outputs that adhere to certain criteria. This has been explored as an extension to work presented thus far, using the TTS-CGAN model [39]; a detailed architecture and training description can be found in Appendix A. In attempting to generate synthetic data for aircraft landing trajectories using TTS-CGAN, several technical challenges emerged that hindered the model's ability to produce realistic outputs. While conditioning on specific labels and utilizing advanced model architectures containing self-attention layers certainly seem promising (aiming to benefit from transfer-

learning across classes), the model ultimately struggled to handle the complexity and variability of the data, failing to generate meaningful results.

Transfer learning aims to leverage knowledge learned from one cluster of trajectories to improve the model's performance on another cluster. By sharing latent representations across classes, the model can transfer features like altitude transitions, speed profiles, or general flight dynamics, which are common across different trajectory types. This approach enables the model to generalize better on classes with limited data by utilizing the shared structure inherent in aircraft trajectories, reducing the need for extensive training on every individual class. However, the complexity of trajectories and their class-specific constraints (e.g., different operational rules or environmental factors) can hinder the effectiveness of transfer learning, as mismatches between classes may introduce unrealistic features into the generated sequences.

One significant reason the TTS-CGAN model may have failed to produce the desired results in synthetic data generation is due to the mismatch between the model's complexity and the available data. TTS-CGAN consists of a combination of a conditional GAN and self-attention mechanisms to generate time-series data that adhere to specific conditional labels, such as different clusters, runways, or operational conditions (normal procedures, go-arounds, and holding patterns). The self-attention layers are particularly adept at capturing long-range dependencies within the data, theoretically allowing the model to generate more contextually consistent sequences. However, in the context of generating synthetic aircraft landing trajectories, the complexity of the self-attention mechanism can become a drawback. The self-attention layers require substantial amounts of diverse and high-quality data to learn meaningful patterns effectively. Given the high dimensionality and variability of the data—characterized by diverse trajectories under varying operational conditions—the model's self-attention mechanism may have become overwhelmed, resulting in outputs that are unexplainable and uninterpretable.

Furthermore, the attempts to simplify the task by constraining the datasets to similar trajectories, lowering the sampling rate, and performing intra-cluster labeling highlight the difficulties in managing the data's high variability and complexity. By reducing the sampling rate and limiting the data to more homogeneous subsets, the intention was to make the learning process more manageable for the model, thereby improving the generation quality. However despite these efforts, TTS-CGAN still failed to capture the essential temporal and spatial correlations necessary for realistic trajectory generation. This failure could be attributed to the model's conditional mechanism, which may not be robust enough to handle the variations between groups effectively. [39] tested multiple label embedding strategies to select the best approach for conditioning the model. They found that concatenating label embeddings to both the generator and discriminator produced noisy results, while adding label embeddings to the input performed poorly. Their proposed method, which concatenates label embeddings only in the generator and adds a classifica-

tion head to the discriminator, resulted in the highest quality outputs. When intra-cluster labeling is applied, the model is expected but unable to distinguish between differences in trajectories that occur under similar conditions. This limitation is compounded by the GAN's inherent training instability, where the generator and discriminator networks fail to converge to a stable solution, particularly when the data distribution is complex and multi modal. The latent space dimensionality and generator and discriminator networks were tested for various hyperparameter configurations to seek a stable training process. In all cases, the output data was not meaningful enough. For a dataset of 18,000 trajectories, the geographical coordinates (longitude and latitude) vary too great of an extent between clusters, runways, and operational conditions. The labeling/conditioning mechanism of TTS-CGAN was not able to handle this effectively. As a reminder, TimeGAN is trained on each group/cluster individually, significantly reducing the breadth of the input data distribution.

TTS-CGAN struggles to model the non-linear dependencies and physical constraints of aircraft trajectories and account for the stochastic factors and domain-specific constraints like weather, pilot behavior, and regulatory standards. As a result, the model could generate synthetic data that are unexplainable and physically implausible, overfitting to noise rather than capturing true flight dynamics. This suggests that an alternative modeling approach, perhaps integrating more domain-specific knowledge (e.g. flight trends) or constraints directly into the model architecture, could be more effective for generating realistic synthetic data according to a conditioning mechanism in this application. Further testing with other conditioning variables is also advisable as less-informative variables mislead the generator. The integration of a prior and time series decomposition analysis might be of value here. Overall, due to the lacking interpretability of the results at this stage, providing sound reasoning for the lacking synthetic data quality remains largely speculative.

VI. CONCLUSIONS & FUTURE WORK

This research explored the effectiveness of GANs in generating realistic synthetic aircraft trajectories during the landing phase, including complex scenarios such as go-arounds and holding patterns. It also examined suitable evaluation metrics to quantify the quality, diversity, fidelity, and usefulness of the synthetic samples. Increased complexity in landing trajectories, represented by greater variability and higher dimensionality, posed significant challenges for the GAN models. The risk of mode collapse and overfitting was notable, emphasizing the need for a balanced approach in model training and evaluation. The results demonstrate that TimeGAN can successfully generate realistic synthetic aircraft trajectories, particularly when data is clustered and trained separately for each class. This clustering was essential in managing the complexity during training and ensuring that the generated trajectories were diverse and reflective of varied real-world operations. Key strategies that proved effective included setting a maximum cumulative track angle threshold to manage

trajectory complexity and reducing the sampling rate to make the training process more computationally feasible (lowering the sample space dimensionality). Post-processing techniques, by applying smoothing filters, further refined the synthetic data. Even in the event of severe minority classes, as was the case for runway use, TimeGAN proved effective at capturing the patterns in data, but required significant smoothing of the output time series. With respect to go-arounds and holding patterns, the available real datasets proved too limited for the model to capture the diversity yet simultaneously maintain realism. The investigation into conditioning a GAN to allow for simultaneous training across all data classes (allowing for transfer learning) proved ineffective. Results of the novel TTS-CGAN model remained uninterpretable, likely due to exceedingly large discrepancies in flight paths across clusters. The evaluation framework, incorporating metrics for data diversity, fidelity, and usefulness, validated these methods, demonstrating that the clustered approach, combined with complexity management techniques, produced high-fidelity synthetic trajectories with potential for various ATM applications. Nonetheless, variation in performance of TimeGAN between groups of data underscores the sensitivity of the training procedure to the underlying real distribution. At this stage, this relationship has been assessed empirically, but has not been rendered to a precise relationship. This would enable optimal tailoring of the model architecture and training to the specific complexities of each input dataset, improving both the fidelity and interpretability of the generated data.

Future research on synthetic trajectory generation can benefit from exploring several innovative avenues. Naturally, one promising direction is the testing of diffusion model architectures, which could offer new perspectives and improvements over current methodologies. Enhancing scalability and computational efficiency will be crucial, particularly for managing large-scale datasets. Furthermore, testing of models across various airports and operational environments, as well as conditioning on factors such as weather and specific airport data, can provide additional insight into their adaptability. Addressing domain-specific constraints and developing novel evaluation metrics are essential for refining the utility and accuracy of synthetic trajectories. Incorporating flight simulators to assess flyability and physical realism, including complex trajectories, will further enhance the evaluation framework. Regarding the implementation and training to improve the efficiency of data generation, integrating global sensitivity analysis and Bayesian optimization holds potential. Global sensitivity analysis can pinpoint critical parameters that impact model performance, enabling targeted optimization and enhancing the robustness of synthetic trajectories. Bayesian optimization, in turn, offers a systematic approach to hyperparameter tuning by skewing the allocation computational resources to promising areas of the parameter space, as opposed to traditional methods. This combined approach can streamline model development and adoption in applications. However, a major challenge remains: the limited availability of diverse real-world data, which constrains model training. This paradox highlights the

critical need for rich real data to produce high-quality synthetic data, emphasizing the intricate balance between innovation and reality in the field of trajectory generation.

REFERENCES

- [1] SESAR Joint Undertaking, "Automation in air traffic management: Long-term vision and initial research roadmap," Publications Office of the European Union, Tech. Rep., 2020. [Online]. Available: <https://www.sesarju.eu/automation>
- [2] Y. Roh, G. Heo, and S. E. Whang, "A survey on data collection for machine learning: A big data - ai integration perspective," *IEEE Transactions on Knowledge and Data Engineering*, vol. 33, no. 4, pp. 1328–1347, April 2021. [Online]. Available: <https://ieeexplore.ieee.org/document/8862913>
- [3] Eurocontrol and Partners, "Fly ai report - demystifying and accelerating ai in aviation/atm," European Aviation High Level Group on AI, Tech. Rep., 2020. [Online]. Available: <https://www.eurocontrol.int/publication/fly-ai-report>
- [4] European University Institute of Florence, "The impact of the general data protection regulation (gdpr) on artificial intelligence," European Parliamentary Research Service, Tech. Rep., 2020. [Online]. Available: <https://www.eurocontrol.int/publication/fly-ai-report>
- [5] SESAR Joint Undertaking, "European atm master plan - digitalising europe's aviation infrastructure," Publications Office of the European Union, Tech. Rep., 2020. [Online]. Available: <https://www.sesarju.eu/masterplan2020>
- [6] D. Timotic and F. Netjasov, "Automation in air traffic control: Trust, teamwork, resilience, safety," *Transportation Research Procedia*, vol. 65, pp. 13–23, November 2022. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S2352146522006706>
- [7] T. Krauth, A. Lafage, J. Morio, X. Olive, and M. Waltert, "Deep generative modelling of aircraft trajectories in terminal maneuvering areas," *Machine Learning with Applications*, vol. 11, p. 100446, March 2023. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S2666827022001219>
- [8] Delft University of Technology, Sintef, Deep Blue, and Eurocontrol, "D.2.1 state of the art," Publications Office of the European Union, Tech. Rep., January 2024.
- [9] SESAR Joint Undertaking, "Sesar innovation pipeline - air traffic management research and innovation," Publications Office of the European Union, Tech. Rep., 2023. [Online]. Available: <https://www.sesarju.eu/innovation-pipeline>
- [10] S. A. Assefa, D. Dervovic, M. Mahfouz, R. E. Tillman, P. Reddy, and M. Veloso, "Generating synthetic data in finance: opportunities, challenges and pitfalls," in *Proceedings of the First ACM International Conference on AI in Finance*, ser. ICAIF '20. Association for Computing Machinery, October 2021. [Online]. Available: <https://dl.acm.org/doi/10.1145/3383455.3422554>
- [11] A. Gonzales, G. Guruswamy, and S. R. Smith, "Synthetic data in health care: A narrative review," *PLOS Digital Health*, vol. 2, no. 1, pp. 1–16, 01 2023. [Online]. Available: <https://journals.plos.org/digitalhealth/article?id=10.1371/journal.pdig.0000082>
- [12] N. Gadipudi, I. Elamvazuthi, M. Sanmugam, L. I. Izhar, T. Prasetyo, R. Jegadeeshwaran, and S. S. A. Ali, "Synthetic to real gap estimation of autonomous driving datasets using feature embedding," in *5th International Symposium in Robotics and Manufacturing Automation (ROMA)*, 2022, pp. 1–5. [Online]. Available: <https://ieeexplore.ieee.org/document/9915679>
- [13] D. P. Kingma and M. Welling, "An introduction to variational autoencoders," *Found. Trends Mach. Learn.*, vol. 12, no. 4, p. 307–392, November 2019. [Online]. Available: <https://dl.acm.org/doi/10.1561/22000000056>
- [14] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems*, vol. 27. Curran Associates, Inc., June 2014. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf
- [15] —, "Generative adversarial networks," *Commun. ACM*, vol. 63, no. 11, p. 139–144, October 2020. [Online]. Available: <https://dl.acm.org/doi/10.1145/3422622>

- [16] J. Ho, A. Jain, and P. Abbeel, "Denosing diffusion probabilistic models," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 6840–6851. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2020/file/4c5bcfec8584af0d967f1ab10179ca4b-Paper.pdf
- [17] J. Yoon, D. Jarrett, and M. van der Schaar, "Time-series generative adversarial networks," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2019/file/c9efe5f26cd17ba6216bbe2a7d26d490-Paper.pdf
- [18] 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, May 2010. [Online]. Available: <https://onlinelibrary.wiley.com/doi/abs/10.1002/acs.1176>
- [19] J. Sun, J. M. Hoekstra, and J. Ellerbroek, "Openap: An open-source aircraft performance model for air transportation studies and simulations," *Aerospace*, vol. 7, no. 8, July 2020. [Online]. Available: <https://www.mdpi.com/2226-4310/7/8/104>
- [20] X. Olive, J. Sun, M. Murça, and T. Krauth, "A framework to evaluate aircraft trajectory generation methods," in *14th USA/Europe Air Traffic Management Seminar, ATM2021*, September 2021. [Online]. Available: <https://research.tudelft.nl/en/publications/a-framework-to-evaluate-aircraft-trajectory-generation-methods>
- [21] T. Gräupl, "Validation of simulated synthetic air traffic against recorded air traffic in germany," in *Integrated Communications Navigation and Surveillance (ICNS)*, January 2017, pp. 2063–2068. [Online]. Available: <https://elib.dlr.de/111344/>
- [22] F. Hoffmann, U. Epple, M. Schnell, and U.-C. Fiebig, "Feasibility of Idacsl cell planning in european airspace," in *2012 IEEE/AIAA 31st Digital Avionics Systems Conference (DASC)*, October 2012, pp. 1–17. [Online]. Available: <https://ieeexplore.ieee.org/document/6383053>
- [23] C.-H. Roktansky, M. Ehammer, and T. Graupl, "Newsky - building a simulation environment for an integrated aeronautical network architecture," in *2007 IEEE/AIAA 26th Digital Avionics Systems Conference*, 2007, pp. 4.B.4–1–4.B.4–11. [Online]. Available: <https://ieeexplore.ieee.org/document/4391905>
- [24] D. Jacquemart and J. Morio, "Conflict probability estimation between aircraft with dynamic importance splitting," *Safety Science*, vol. 51, no. 1, pp. 94–100, 2013. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0925753512001300>
- [25] M. Poppe and J. Buxbaum, "Clustering climb profiles for vertical trajectory analysis," in *Proceedings of the 10th SESAR Innovation Days*, December 2020. [Online]. Available: https://www.sesarju.eu/sites/default/files/documents/sid/2020/papers/SIDs_2020_paper_66red.pdf
- [26] M. C. Rocha Murça and M. de Oliveira, "A data-driven probabilistic trajectory model for predicting and simulating terminal airspace operations," in *2020 AIAA/IEEE 39th Digital Avionics Systems Conference (DASC)*, October 2020, pp. 1–7. [Online]. Available: <https://ieeexplore.ieee.org/document/9256644>
- [27] T. Krauth, J. Morio, X. Olive, B. Figuet, and R. Monstein, "Synthetic aircraft trajectories generated with multivariate density models," *Engineering Proceedings*, vol. 13, no. 1, 2021. [Online]. Available: <https://www.mdpi.com/2673-4591/13/1/7>
- [28] G. Jarry, N. Couellan, and D. Delahaye, "On the use of generative adversarial networks for aircraft trajectory generation and atypical approach detection," in *Air Traffic Management and Systems IV*, 2021, pp. 227–243. [Online]. Available: https://link.springer.com/chapter/10.1007/978-981-33-4669-7_13
- [29] O. Fabius and J. R. van Amersfoort, "Variational recurrent auto-encoders," *arXiv preprint arXiv:1412.6581 [cs]*, December 2014. [Online]. Available: <https://arxiv.org/abs/1412.6581>
- [30] C. Esteban, S. L. Hyland, and G. Rätsch, "Real-valued (medical) time series generation with recurrent conditional gans," *arXiv preprint arXiv:1706.02633 [cs]*, June 2017. [Online]. Available: <https://arxiv.org/abs/1706.02633>
- [31] X. Chen, J. Xu, R. Zhou, W. Chen, J. Fang, and C. Liu, "Trajvae: A variational autoencoder model for trajectory generation," *Neuro-computing*, vol. 428, pp. 332–339, March 2021. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0925231220312017>
- [32] S. Jeon and J. T. Seo, "A synthetic time-series generation using a variational recurrent autoencoder with an attention mechanism in an industrial control system," *Sensors*, vol. 24, no. 1, 2024. [Online]. Available: <https://www.mdpi.com/1424-8220/24/1/128>
- [33] D. Lee, S. Malacarne, and E. Aune, "Vector quantized time series generation with a bidirectional prior model," in *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, F. Ruiz, J. Dy, and J.-W. van de Meent, Eds., vol. 206. PMLR, 25–27 Apr 2023, pp. 7665–7693. [Online]. Available: <https://proceedings.mlr.press/v206/lee23d.html>
- [34] A. Desai, C. Freeman, Z. Wang, and I. Beaver, "Timevae: A variational auto-encoder for multivariate time series generation," *arXiv preprint arXiv:2111.08095 [cs]*, November 2021. [Online]. Available: <https://arxiv.org/abs/2111.08095>
- [35] O. Mogren, "C-rnn-gan: Continuous recurrent neural networks with adversarial training," *arXiv preprint arXiv:1611.09904 [cs]*, November 2016. [Online]. Available: <https://arxiv.org/abs/1611.09904>
- [36] S. Liao, H. Ni, L. Szpruch, M. Wiese, M. Sabate-Vidales, and B. Xiao, "Conditional sig-wasserstein gans for time series generation," *arXiv preprint arXiv:2006.05421 [cs]*, June 2020. [Online]. Available: <https://arxiv.org/abs/2006.05421>
- [37] Y. Jiang, S. Chang, and Z. Wang, "Transgan: Two pure transformers can make one strong gan, and that can scale up," in *Advances in Neural Information Processing Systems*, vol. 34, 2021, pp. 14745–14758. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2021/file/7c220a2091c26a7f5e9f1cfb099511e3-Paper.pdf
- [38] X. Li, V. Metsis, H. Wang, and A. H. H. Ngu, "Tts-gan: A transformer-based time-series generative adversarial network," in *Artificial Intelligence in Medicine: 20th International Conference on Artificial Intelligence in Medicine, AIME 2022, Halifax, NS, Canada, June 14–17, 2022, Proceedings*, June 2022, p. 133–143. [Online]. Available: https://dl.acm.org/doi/10.1007/978-3-031-09342-5_13
- [39] X. Li, A. H. H. Ngu, and V. Metsis, "Tts-cgan: A transformer time-series conditional gan for biosignal data augmentation," *arXiv preprint arXiv:2206.13676 [cs]*, June 2022. [Online]. Available: <https://arxiv.org/abs/2206.13676>
- [40] H. Lim, M. Kim, S. Park, and N. Park, "Regular time-series generation using sgm," *arXiv preprint arXiv:2301.08518 [cs]*, January 2023. [Online]. Available: <https://arxiv.org/abs/2301.08518>
- [41] Z. Kong, W. Ping, J. Huang, K. Zhao, and B. Catanzaro, "Diffwave: A versatile diffusion model for audio synthesis," *arXiv preprint arXiv:2009.09761 [cs]*, September 2020. [Online]. Available: <https://arxiv.org/abs/2009.09761>
- [42] A. Das, Y. Yang, T. Hospedales, T. Xiang, and Y.-Z. Song, "Chirodiff: Modelling chirographic data with diffusion models," *arXiv preprint arXiv:2304.03785 [cs]*, April 2023. [Online]. Available: <https://arxiv.org/abs/2304.03785>
- [43] A. Coletta, S. Gopalakrishnan, D. Borrajo, and S. Vyetenko, "On the constrained time-series generation problem," *arXiv preprint arXiv:2307.01717 [cs]*, July 2023. [Online]. Available: <https://arxiv.org/abs/2307.01717>
- [44] X. Yuan and Y. Qiao, "Diffusion-ts: Interpretable diffusion for general time series generation," *arXiv preprint arXiv:2403.01742 [cs]*, March 2024. [Online]. Available: <https://arxiv.org/abs/2403.01742>
- [45] F. Bryant and P. Yarnold, "Principal-component analysis and exploratory and confirmatory factor analysis," January 1995. [Online]. Available: <https://psycnet.apa.org/record/1995-97110-004>
- [46] L. van der Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, no. 86, pp. 2579–2605, August 2008. [Online]. Available: <http://jmlr.org/papers/v9/vandermaten08a.html>
- [47] S. Jansen, *Machine Learning for Algorithmic Trading: Predictive Models to Extract Signals from Market and Alternative Data for Systematic Trading Strategies with Python*, ser. Expert Insight. Packt Publishing, 2020. [Online]. Available: <https://books.google.nl/books?id=cki6zQEACAAJ>
- [48] J. M. Hoekstra and J. Ellerbroek, "Bluesky atc simulator project: an open data and open source approach," in *Proceedings of the 7th international conference on research in air transportation*, vol. 131, 2016, p. 132. [Online]. Available: <https://repository.tudelft.nl/record/uuid:d1131a90-f0ea-4489-a217-ad29987689a1>
- [49] M. Schäfer, M. Strohmeier, V. Lenders, I. Martinovic, and M. Wilhelm, "Bringing up opensky: a large-scale ads-b sensor network for research," in *Proceedings of the 13th International Symposium on Information Processing in Sensor Networks*, ser. IPSN '14. IEEE Press, April 2014, p. 83–94.

- [50] X. Olive, “traffic, a toolbox for processing and analysing air traffic data,” *Journal of Open Source Software*, vol. 4, p. 1518, 2019.
- [51] A. Khattak, J. Zhang, P.-W. Chan, F. Chen, and C. M. Matara, “Ai-supported estimation of safety critical wind shear-induced aircraft go-around events utilizing pilot reports,” *Heliyon*, vol. 10, no. 7, p. e28569, January 2024. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S2405844024046000>
- [52] K. Artiouchine, P. Baptiste, and C. Dürr, “Runway sequencing with holding patterns,” *European Journal of Operational Research*, vol. 189, no. 3, pp. 1254–1266, September 2008. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0377221707005978>
- [53] R. Tavenard, J. Faouzi, G. Vandewiele, F. Divo, G. Androz, C. Holtz, M. Payne, R. Yurchak, M. Rußwurm, K. Kolar, and E. Woods, “Tslearn, a machine learning toolkit for time series data,” *Journal of Machine Learning Research*, vol. 21, no. 118, pp. 1–6, 2020. [Online]. Available: <http://jmlr.org/papers/v21/20-091.html>
- [54] D. J. Berndt and J. Clifford, “Using dynamic time warping to find patterns in time series,” in *KDD Workshop*. AAAI Press, April 1994, pp. 359–370. [Online]. Available: <http://dblp.uni-trier.de/db/conf/kdd/kdd94.html#BerndtC94>
- [55] G. Szekely and M. Rizzo, “Testing for equal distributions in high dimension,” *InterStat*, vol. 5, November 2004. [Online]. Available: https://www.researchgate.net/publication/228918499_Testing_for_equal_distributions_in_high_dimension

II

Literature Study

Introduction

Maintaining the effectiveness of the current air traffic management (ATM) system in the face of rising demand will require greater automation. Artificial intelligence (AI) and machine learning (ML) will play a pivotal role in supporting this advancement [113]. These technologies are fundamental to achieving the ambitious goals of the Single European Sky ATM Research (SESAR) Joint Undertaking [114]. A gradual increase in the degree of automation from information acquisition and decision support, to air traffic control and U-space services can be realized with increasingly complex and trained AI/ML solutions [35, 113]. The complexities of the ATM system involves balancing the diverse needs and expectations of passengers, airlines, regulatory bodies, and air traffic controllers, all of whom rely on seamless coordination for safe and efficient air travel. The breakthrough in technological challenges to achieve widespread connectivity and deployment of the developed systems is paramount, and a prerequisite for trust in them [122]. The erratic stability characteristics of models and the potential catastrophic failure in situations stemming from inadequate training and experience [35, 113] underscore this.

The surge in demand for comprehensive datasets to fuel the training and validation of ML models for various ATM tasks, such as predicting air traffic flow, optimizing routes, detecting anomalies, and improving safety protocols, is often in excess of what can be supplied [35]. Simultaneously, obtaining the required volumes of training data is often non-trivial, impeded by latent costs and obstacles [106]. The aptly named "data access problem" plagues the effective training of ML algorithms to attain acceptable levels of performance [90]. Scarcity of data clearly imposes limits on the complexity of patterns that can be captured by a model, but mere abundance of data is not the holy grail. Datasets that are available should be held to a high standard in terms of quality (correct and minimal noise), adhere to data privacy laws such as the General Data Protection Regulation (GDPR) [36], and promote fairness on legal and ethical domains, rather than perpetuate algorithm biases [37, 90]. Synthetic data generation (SDG) aims to provide a solution to the aforementioned challenges, with the promise of an accelerated and effective unlocking of strategic ML capabilities.

Under the umbrella of the European Union funded SESAR Joint Undertaking initiative, the SynthAIR project aims to leverage AI-based universal models for the purpose of synthetic data generation in the domain of ATM automation and simulation [115]. Unavailability of accurate data constitutes a major barrier to reaping the full benefits of ML models to optimize ATM [35]. Optimization in this context ranges from accelerating the degree of automation, to improving the accuracy of downstream prediction tasks and resource allocation [115]. One of the core areas therein revolves around aircraft trajectories (time series), with the purpose of reducing uncertainty and increasing capacity of the airspace [35]. Despite the fact that application of deep generative models to aircraft trajectory generation is still in its infancy, time series generation has gained traction and risen to prominence in diverse alternative industries and research domains [90]. The objective is to leverage novel deep generative modelling techniques developed for multivariate time series applications in domains including finance [5], healthcare [43], and automotive [42], in an effort to take significant strides compared to the relatively limited existing implementations within the ATM domain.

The SynthAIR project revolves around generative modelling techniques, rather than statistical modelling and agent-based modelling (both other viable synthetic data generation paradigms). Three notable families of generative models (that have seen remarkable success) are variational autoencoders (VAEs) [75], generative adversarial networks (GANs) [44, 46], and diffusion models [55]. These form the core of this literature study and serve as a springboard to explore options for generating 4-dimensional aircraft trajectories. Nearly

all models in these families fail to simultaneously meet the three requirements pivotal to success: high quality of the generated samples, mode coverage (sample diversity), and fast sampling; coined as the generative learning trilemma [135]. Despite this being an active field of current research, these techniques have proven to be successful across industries with regard to generating images, textual, and tabular data [69]. The inherent temporal relationships of time series data, supplementary to feature distributions, add complexity to the generative process [138]. Generation of time series data has found success in other domains after the integration with recurrent network elements and novel architectures such as the transformer based on the self-attention mechanism [84]. Regardless of the nature of the generation process, ultimately, synthetic samples need to be evaluated against specified objective-related criteria; this remains an active area of research. Inspired by other domains, generated artificial flight trajectories are required to be well-diversified, indistinguishable from real flights, enhance predictive power in downstream tasks, and be flyable from a physics and aircraft performance standpoint [138].

The research scope of the thesis is framed by the literature study presented in this report, where the state-of-the-art in synthetic aircraft trajectory generation is reviewed, in tandem with model architectures applied to multivariate time series in other domains. [Chapter 2](#) commences with a definition of synthetic data, alongside a rationale and exemplary applications. This background is used to taper towards use cases with regard to ATM systems, and 4-dimensional aircraft trajectories in particular. The generative modeling approach is described and the implications of the temporal dynamics intrinsic to time series is discussed. Evaluation metrics are discussed in terms of diversity, fidelity, usefulness, and flyability. [Chapter 3](#) reviews the existing models that have been proposed to generate synthetic flight trajectories, including model driven methods relying on flight dynamics, statistical extrapolation, stochastic processes, and most recently, deep generative models. There is an abundance of variance in model architectures within the broader families of variational autoencoders, generative adversarial networks, and diffusion models. Providing an exhaustive account of all variations would constitute an insurmountable task with little merit. Instead, a selection of models that each represent a discontinuity in framework is elaborated upon in [Chapter 4](#), and are framed within the broader taxonomy backdrop of techniques. The literature study represents the bedrock for the final research proposal presented in [Chapter 5](#). The content of that chapter is threefold. It contains a concise statement of the objective and the research questions, a rationale for the relevance of the research, and a detailed planning to ensure objectives yield measurable results.

2

Synthetic Data for Air Traffic Management Automation

Synthetic data can play a pivotal role in driving automation within ATM systems. For this reason, this chapter delves into the fundamental significance of this artificially generated data. [Section 2.1](#) commences with a definition of synthetic data, alongside an exploration of its diverse types and applications. The envisioned applications of the SynthAir project pertinent to ATM are discussed in [Section 2.2](#). The generative modeling paradigm is juxtaposed with discriminative modeling and the concept of density function estimation is explained in [Section 2.3](#). The temporal dimension intrinsic to time series is explicitly discussed in [Section 2.4](#), together with corresponding model architectures. Lastly, proposed evaluation methodologies for synthetically generated time series samples is presented in [Section 2.5](#).

2.1. Synthetic Data

The notion of synthetic data holds promising value in widespread industries, including ATM systems. This section provides the necessary background to understand the nature of this data by commencing with a definition and rationale, in [Section 2.1.1](#). An overview of various synthetic data types is given in [Section 2.1.2](#) and exemplary applications across different industries in [Section 2.1.3](#) highlight recent success when coupled with downstream machine learning techniques.

2.1.1. Synthetic Data Definition and Rationale

Synthetic data can be defined as artificial information that has been generated according to computer algorithms [91]. This data is meant to be statistically representative of real-world (historic) data, but does not correspond to any existing objects or events; it is purely artificial [90]. Nonetheless, it can be an alternative to reliance on unreliable and volatile *real* data. The collection and processing of data is expensive, often comprising a majority share of the machine learning workflow. The data access problem condenses these data-related difficulties into three pillars: data quality, data scarcity, and data privacy and fairness [72, 90].

When available data is not of the highest quality, machine learning models trained on it are vulnerable to misinterpretations and develop an erroneous understanding of the underlying problem [90]. This can translate into false predictions and diminished model usefulness [102, 116]. Scarcity refers to data unavailability [6]. Data is often branded as the *fuel* for machine learning applications. While there is truth to this statement, it constitutes only one of the three pillars of the data access problem that plagues effective machine learning. Lastly, data privacy and fairness concerns are on a meteoric rise. When data contains private and sensitive information it cannot be readily released and hinders its use for machine learning purposes [37]. Anonymized synthetic datasets offer a way of circumventing these difficulties. Overall, synthetic data in various forms provides an alternative method of unlocking the potential of emerging models by circumventing the data access problem.

2.1.2. Overview of Synthetic Data Types

Unavoidably, different forms of synthetic data are demanded to serve the diverse data requirements of various industries [90]. Table 2.1 provides an overview of the different classes of synthetic data types, a description, and the value proposition. Clearly, synthetic data has emerged in many logical forms and is readily finding use cases. SynthAIR's objective places emphasis on the generation of tabular and time series data (the temporal extension of the former). Due to the focus of this thesis on 4-dimensional aircraft trajectories, time series is the naturally applicable data type.

Table 2.1: Common synthetic data types.

Synthetic Data Type	Description	Example Value Propositions
Tabular	Organized data in rows and columns (table/spreadsheet)	Imbalanced class correction, anonymization of health records, and non-destructive testing [111], [54, 126], [53]
Image	Visual representation of objects or scenes	Computer vision for medical imaging, construction mapping, autonomous driving [98], [137], [12]
Text	Written or typed language content for communication	Natural language processing (NLP) including machine translation, sentiment analysis, and virtual assistants [26, 70]
Audio	Sound recordings or signals transmitted over time	Speech and emotion recognition, music and melody generation (based on lyrics, video, genre) [79], [117]
Time Series	Sequential data points recorded over time intervals	Enhance prediction of trading strategies, electricity demand, weather pattern generators [13], [136], [141]
Spatial	Geographical or geometric data with spatial references	Global extreme sea level events assessment, urban mobility planning, health risk transmission [83], [143], [21]
Video	Sequential frames creating moving visual content	Virtual instructor videos for learning, development of interactive media, security surveillance [81], [16], [103]

2.1.3. Key Synthetic Data Applications

The merits of synthetic data have been discovered in a multitude of domains. The financial, healthcare, automotive, and manufacturing industries are examples of these successes and foreshadow the potential within ATM. For each of these sectors, two specific use cases are outlined next.

Finance

Within the financial sector, synthetic data is utilized for ML model training on balanced datasets, testing software, and conducting scenario analyses while preserving sensitive and confidential information [5]. For example, a synthetic data generator has been proposed specifically for anti-money laundering. An agent-based model is trained to mimic historical transactions (with maximum realism) to avoid banks from missing transactions of their customers with other institutions, decrease the high rates of mislabeled laundering transactions (false negatives), and help identify patterns in behavior of malicious practices [2]. Synthetic data augmentation has also led to improvement in financial trading using deep reinforcement learning, as measured by profitability and generalizability [89]. Datasets become more representative of all possible market circumstances and therefore lead to more informed decisions.

Healthcare

Synthetic data in healthcare is accelerating research and development while safeguarding patient privacy [43]. With respect to demographic ageing, micro-simulations are used to create synthetic datasets to investigate the impact of policy options concerning medical visit rates, prescriptions, and referrals [25]. These studies can become unwieldy without synthetic simulation. Recurrent autoencoders and GANs have also been employed to generate synthetic longitudinal electronic health records for model training [120]. This promotes the usage of clinical data in medical education and research without compromising sensitive information.

Automotive

Autonomous driving tasks including visual odometry and object detection require vast amounts of data, where obtaining annotations is expensive [42]. Many scenarios that are of interest for testing are dangerous and impractical to repeat many times over. The work in [12] highlights the use of synthetic data as a method of creating virtual testing and validation conditions for the automotive sector. Furthermore, synthetic tabular transaction data representing passenger carsharing trips has been generated using GANs and VAEs [1]. The goal of improving mobility in this way resonates with generating tabular data to minimize delay propagation through an airline network.

Manufacturing

In manufacturing, synthetic data can be employed to optimize production processes without risking real-world operational disruptions. Deep learning vision systems are being utilized to generate samples of minority class defects, to ultimately improve machine-learning based quality control in the manufacturing process [101]. Limited training data also presents difficulties in non-destructive ultrasonic testing, where SDG methods have been proposed to translate experimental and simulated testing data into synthetic defective and non-defective images [94].

2.2. SynthAir Project - SESAR Joint Undertaking

The SESAR Joint Undertaking SynthAir project's mission is to explore novel deep generative modeling techniques for synthetic data generation in the ATM context. At the core of the project is a partnership between Sintef (NO), Deep Blue (IT), Eurocontrol (EU), and the Delft University of Technology (NL), with a projected duration of 30 months. Section 2.2.1 and Section 2.2.2 present the two-step objective of the project. In Section 2.2.3, identified use cases (albeit non-exhaustive) based on stakeholder discussions are presented. The focus of this thesis is on the generation of 4-dimensional aircraft trajectories, which are elaborated upon in Section 2.2.4. Lastly, the contents of available open data bases are highlighted in Section 2.2.5.

2.2.1. Synthetic Data Generation for Multivariate Time Series for ATM Automation

The main objective of the SynthAir project is to explore the applicability of AI-based generation of synthetic data in ATM systems [115]. SDG using generative AI applied to air transport is very limited thus far, where primarily statistical and simulation-based techniques have been relied on. Generation of synthetic data enables the exploration of diverse and potentially rare scenarios, contributing to the robustness and resilience of ATM automation systems against unexpected events and anomalies [35]. Both safety and efficiency in terms of resource allocation should ultimately improve as measured by downstream tasks. The work is done in conjunction with various stakeholders including airports, airlines, and air network service providers (ANSPs). This collaboration also provides scope for the uses cases introduced in the SynthAir Description of Action [28]: prediction of turnaround time (UC1), flight delay prediction (UC2), passenger flow prediction (UC3), synthetic traffic generation (UC4), and flight diversion prediction (UC5). These are further delineated in Section 2.2.3.

2.2.2. Universal Time Series Model for Prediction & Generation for ATM Automation

The premise of a universal time series generator (UTG) is central to the SynthAir project [115]. In essence, a model is learned on various datasets to subsequently generate synthetic new datasets, representing new situations. An example is the training on a set of existing airports, to afterwards use the model in generating data concerning a different, new airport [115]. The natural extension is to establish a universal time series forecaster (UTF) that does not rely on accompanying training data for each new environment [109, 133]. The

pre-training on a large collection of datasets would allow for coverage of a wide variety of downstream tasks. Attaining this means overcoming significant challenges such as cross-frequency learning, effective handling of arbitrary variables of multivariate time series, and varying underlying distributions [133], by a minimal number of assumptions on the generating mechanism [109]. However, it is with this type of modeling that the majority of value lies for the stakeholders.

2.2.3. Identified Use Cases

In collaboration with airports, airlines, ANSPs, and researchers, five potential uses cases have been identified in the SynthAIR Description of Action [28]. Based on a set of selection criteria covering data, analysis and modelling, integration, stakeholders, and validation, select use cases (UC) have been identified as most promising and subject to continuation in the project. Synthetic data generation using generative deep learning remains in its infancy within ATM, therefore the use cases are largely described in terms of downstream tasks. A concise account of each of the use cases is provided here, based on [28].

UC1: Turnaround Time Prediction

Turnaround time is the duration from when an aircraft is parked at a gate until it departs. It is a common source of primary delay accumulation and is influenced by a myriad of factors including the aircraft type, airline, weather conditions, and airport infrastructure [28]. Previous efforts in modeling turnaround time have often adopted low levels of granularity, using data from individual airports [4, 92]. It is recognized by stakeholders that provision of early indicators of impending delay in the turnaround process is needed. This will increase accuracy of time estimates for the turnaround and smaller deviations from these predictions are likely to lead to reduced delays [112]. Granular data for modeling remains confidential, and combined with the complexity and existing simulation studies, provided reason not to select this use case for SynthAIR.

UC2: Flight Delay Prediction

Delay represents a key performance indicator for air transport systems and can propagate through a network due to aircraft rotation, crew changes, and passenger transfers [14]. Flight delay prediction models are crucial tools for airports to mitigate congestion and improve operational efficiency, without requiring investments in expanding capacity [28]. Most previous efforts in flight delay prediction relied on real historical data for training; generative AI has not been deployed for this purpose as of yet [14, 132]. Delay propagation models are either microscopic (treat aircraft separately and detailed airport layout), mesoscopic (reliance on operational data such as runway configuration and short-term demand), or macroscopic (networks effects by limiting to airport level modeling) [60]. This use case is of high interest for all stakeholders and has been selected for continuation in the project.

UC3: Passenger Flow Prediction

Accurate prediction of passenger flows at an airport can lead to improved operational efficiency, resource allocation, and passenger service quality [28]. Detailed information on passenger movements across the main airport terminal activities (arrival, check-in, security, passport control, boarding/deboarding, and shopping) is required to generate synthetic data with generative AI approaches. Modeling passenger flows has previously been approached using simulation-based [62] or data-driven [63] models. The prediction step specifically, has been performed using time series, causal, AI-based, or hybrid models [15]. This use cases has not been chosen for further continuation because obtaining training data for the synthetic data models poses a significant challenge. Furthermore, passenger flow management is mostly employed in an ad-hoc fashion which complicates generalizability [28].

UC4: Synthetic Traffic Generation

Synthetic traffic generation is an extension of the flight delay use case. Various traffic scenarios can be generated as a result of specific disruptions in the network. The traffic generation can be extended to full 4-dimensional aircraft trajectories rather than at the level of flight schedules [28]. These trajectories can be used to model delay propagation through a network or generate synthetic trajectories of less common situations such as go-arounds. Previous efforts have often focused on extrapolation from flight plan databases [48] and were limited to granularity of airspace entry points. In [78], a generative AI method is proposed where a variational autoencoder is used to generate 4-dimensional aircraft trajectories in the Terminal Maneuvering Area (TMA) of Zurich airport. By extension of UC2, this use case is of interest to the SynthAIR project.

UC5: Flight Diversion Prediction

Flight diversions constitute a small fraction of all flights, occurring when an aircraft cannot land at its intended arrival airport but must *divert* to an alternative. These disruptions have severe economic and operational consequences, and the accurate prediction of impending diversions may alleviate this to some extent [28]. The underrepresentation of flight diversions in historical training sets have lead to difficulties in proposed prediction models, including tree-based [22], clustering [23], and Support Vector Machines (SVM) [30], where adverse weather conditions play a dominant role in the predictions. This use case has been selected for the SynthAir project, but is expected to be very challenging given the limited available data and the complex (unpredictable) nature of flight diversions.

2.2.4. 4-Dimensional Aircraft Flight Trajectories

The designated use cases of flight delay prediction and the full trajectory extension in the synthetic traffic generator implicitly depend on the generation of 4-dimensional flight trajectories [28]. The generation of these flight trajectories forms the focus of this thesis project. Synthetic aircraft trajectories serve as invaluable assets in various domains of aviation research and development, providing insights into airspace management, flight optimization, and safety assessments [35, 124]. The trajectory generation can be conducted at multiple *levels*. Trajectories simulated within the TMA afford granular examination of aircraft behaviors during critical phases of flight, aiding in the design and evaluation of airspace procedures, air traffic control strategies, and collision avoidance systems [78]. On a full-network level, a more holistic approach allows for investigation of airspace constraints, operational protocols, and flight delay propagation [35, 124]. Data on historic flight trajectories allows models to be learned that can facilitate research into capacity expansions, route diversions, and airspace redesigns. In short, it is deemed that efforts in generating synthetic flight trajectories are certainly worthwhile.

Moreover, integrating priors and conditioning mechanisms into synthetic trajectory generation holds the potential for improving the generalization of models to novel circumstances [93]. Generating similar trajectories to historical data holds some value, but this can be greatly increased when the models are flexible enough to assimilate other variables into the generation process. By encoding historical trends, probabilistic constraints, and expert knowledge, generative techniques can capture nuanced behaviors and anticipate future airspace challenges with more limited volumes of data. Leveraging priors and conditioning mechanisms also facilitates the generation of diversified synthetic trajectories, spanning a spectrum of operational conditions and contingencies; a limitation of many generative modeling techniques [135]. Chapter 3 and Chapter 4 are dedicated to a literature study of previously proposed aircraft trajectory models and multivariate time series generation in general, respectively. The goal is not to be exhaustive, but to highlight the merits of specific models and ultimately select techniques that have found success in other domains, and translate that to the aircraft flight trajectory generation problem.

Go-Arounds & Holding Patterns

Two specific types of trajectories that occur less frequently are go-arounds and holding patterns (both defined by ICAO [58] and FAA [38]). Deep generative modeling of these circumstances can ensure well-diversified datasets for ATM automation training. Conditioning mechanism can be used to allow synthetic data for various weather, runway, or approach pattern scenarios. A go-around (missed approach) is a standard aviation procedure initiated by pilots during the landing phase when conditions are deemed unsafe for a successful touchdown. The onset can be poor visibility, runway incursions, or inadequate aircraft positioning. The procedure involves applying full power, retracting flaps and landing gear if necessary, and executing a climb to a predefined altitude, thereby re-entering the airspace. Go-arounds are a crucial safety measure to prevent runway accidents and ensure the safe operation of aircraft during adverse conditions [73]. Holding patterns on the other hand, are predetermined flight paths established by air traffic control to delay aircraft before landing at congested airports or during unexpected airspace congestion. Aircraft are instructed to fly within a defined racetrack-shaped pattern, typically above a designated navigation fix, while awaiting clearance for further approach or landing instructions. Holding patterns serve to regulate air traffic flow, allowing controllers to maintain safe separation distances between aircraft and efficiently manage traffic volume [3].

2.2.5. Available Open Data Sources

In order to generate synthetic data, generative models need to be trained on existing, historical data. Three open databases are available for the purpose of this thesis, namely the Eurocontrol R&D archive [34], the

OpenSky Network [110], and the Bureau of Transportation Statistics (BTS) of the US Department of Transportation [125]. There are specific Python frameworks available that facilitate the parsing and handling of data sourced from these databases, the *traffic* library is an example [99]. The reader is referred to Table 2.2 for details on the specific type of data contained in each of the three databases.

The Eurocontrol R&D archive provides researchers (strictly prohibited for commercial purposes) with air traffic management quality data for commercial flights over Europe. Data is released for four months of each year (comprising of approximately two to three million flights a year), with a two-year delay. The data consists of last-filed flight plans and actual flown routes, alongside details on the airspace and route network in place at the time of the flight. Eurocontrol receives the latest submitted flight plans for all instrument flight rules (IFR) flights and updates this based on live incoming data from ANSPs from flight data systems, radar, and data link communications [34]. The OpenSky network provides air traffic surveillance data in the form of Automatic Dependent Surveillance-Broadcast (ADS-B) and Mode S messages. An estimated 30% of commercial traffic is captured in OpenSky's historical database. Complementary to essential aircraft state vector information (position and velocity), the status messages provide insight into operational/emergency information which can be used to reconstruct aircraft movements in specific situations [110]. Contrary to Eurocontrol and OpenSky, the BTS database provides coverage of air traffic over the US. The database captures air carrier statistics, airline on-time performance, itinerary details of passengers, and intermodal passenger connectivity [125]. For the purposes of this thesis, this data source is expected to be of less value.

Table 2.2: Available open data sources.

Data Source	Dataset & Description
Eurocontrol R&D Archive [34]	• <i>Key flight information</i>: ADEP/ADES, filed arrival time, actual arrival time, aircraft type, operator, registration, requested flight level, and actual distance flown • <i>Flight trajectories (filed and actual)</i>: sequence number, time over, flight level, latitude, and longitude (also in terms of sequence of flight information regions (FIRs) and airspace blocks (AUAs)) • <i>Airspace structure and route network</i>: Routes in terms of latitude and longitude, Aeronautical Information Regulation and Control (AIRAC) cycle information
OpenSky Net- work [110]	• <i>Automatic Dependent Surveillance-Broadcast (ADS-B)</i>: aircraft identification and aircraft state vector (position and velocity) • <i>Mode S messages</i>: various status messages on priority, capability, operational modes, and emergencies
Bureau of Trans- portation (BTS) [125]	• <i>Air carrier statistics</i>: monthly data on passengers and cargo transported (including aircraft type, service class, capacity, and flight hours) • <i>Airline on-time performance</i>: monthly data on scheduled and actual flight times for flights • <i>Airline itineraries</i>: 10% sample of sold airline tickets (itineraries) of carriers • <i>Intermodal passenger connectivity</i>: connection to public transportation modes (rail, air, bus, transit, ferry) at nodes in the network.

2.3. Synthetic Data Generation - Generative Modeling

The paradigm of generative modeling is a mechanism with which synthetic samples can be generated. [Section 2.3.1](#) provides an introduction to this idea by contrasting generative modeling with the perhaps wider-known discriminative modeling. The taxonomy of generative modeling techniques is laid out using the approach to density estimation in [Section 2.3.2](#). Variational autoencoders, generative adversarial networks, and diffusion models are three of the most popular generative modeling approaches (comprising the bulk of existing literature) and are introduced in [Section 2.3.3](#), [Section 2.3.4](#), and [Section 2.3.5](#), respectively.

2.3.1. Generative vs. Discriminative Modeling

Generative modeling, and synthetic data generation by extension, is the branch of machine learning concerned with modeling the underlying distribution of a dataset to generate new samples [40]. It differs from discriminative modeling, which focuses on learning the boundary between different classes in the data. Instead, generative models aim to capture the full probability distribution of the input data. Generative models aim to learn a joint probability distribution $P(x, y)$, or $P(x)$ if there are no labels present. By contrast, discriminative models focus on learning the conditional probability distribution $P(y|x)$ [40, 45]. It is the modeling of the entire joint distribution that allows for the generation of new samples that resemble the training data.

Generative models can also be trained to model the conditional probability distribution $P(x|y)$, in case specific samples should be generated. Additionally, generative models can also be used as classifiers by utilizing Bayes' rule on the class-conditional densities for each class, and the prior class probabilities [96]. The true merit however, lies in the sampling from the learned probability density to generate synthetic content.

2.3.2. Density Function Estimation

Following the taxonomy laid out in [40], generative models can be categorized based on the approach to modeling the underlying probability distribution of the input. At the highest level, the probability density function is either modeled implicitly or explicitly. Implicit density models produce a stochastic process that generates data, rather than determine a probability density function. The generative adversarial network [44, 46] is a prime example of this, and will be further elaborated in [Section 2.3.4](#). In some forms, explicit models set constraints on the model architecture (for example ordering of features) to ensure that calculation of the density function does not become intractable. Alternatively, the model approximates the density function (for example by working in a lower dimensional latent space) to warrant a tractable calculation. Variational autoencoders [75] and diffusion models [55] are examples thereof, explained in [Section 2.3.3](#) and [Section 2.3.5](#). Overall, generative modeling encompasses diverse families of models, each with distinct methods for generating new data. Hybrid approaches may also emerge, combining elements from different families to leverage their respective strengths. For the purpose of this thesis, these models require specific adjustment and augmentation to effectively support the inherent temporal dynamics of (multivariate) time series.

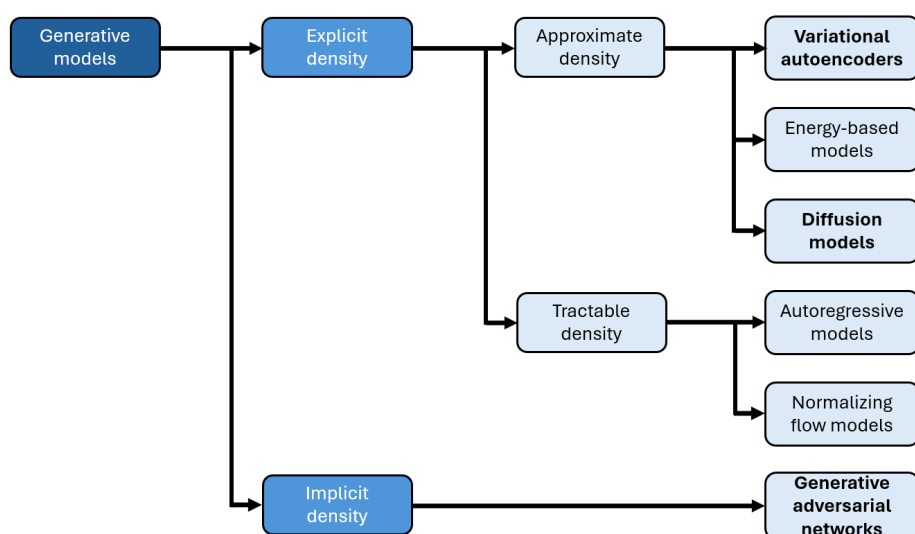


Figure 2.1: Taxonomy of generative modeling approaches, based on [40]

2.3.3. Variational Autoencoders

The variational autoencoder (VAE) was first introduced by Diederik P. Kingma and Max Welling in a seminal paper presented at the International Conference on Learning Representations (ICLR) in 2014 [74]. A VAE comprises of an encoder and decoder network designed for unsupervised learning of a probability distribution through a continuous latent vector [40], schematically depicted in Figure 2.2. Each element in the latent vector corresponds to a distinct dimension in the latent space, capturing different aspects of the higher dimensional input data. The encoder network, $q_\phi(\mathbf{z}|\mathbf{x})$, is responsible for mapping the high dimensional input data into a lower dimensional latent space. This latent space is usually represented by a multivariate Gaussian distribution. The latent variables vector representing the original data is subsequently the input for the decoder network, $p_\theta(\mathbf{x}|\mathbf{z})$, which converts it back to the higher dimensional space [75]. From the learned latent space, new samples can be drawn and decoded back to the original input space, thus allowing new data to be sampled with similar characteristics.

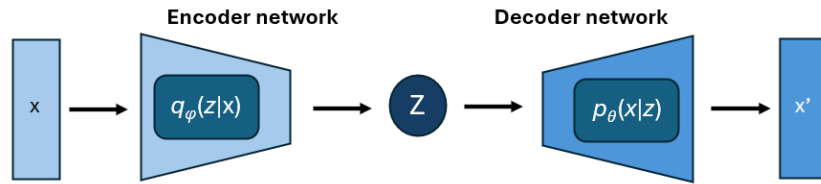


Figure 2.2: Schematic representation of a variational autoencoder, based on [51]

The encoder network is not able to exactly represent the true posterior of the decoder, as this is often intractable; instead it provides an approximation. The log-likelihood of observing the input according to the decoder model can be formulated according to Equation (2.1) [from [75]].

$$\log p_\theta(\mathbf{x}) = \underbrace{\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[\log \left[\frac{p_\theta(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} \right] \right]}_{=\mathcal{L}_{\theta, \phi}(\mathbf{x}) \text{ (ELBO)}} + \underbrace{\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[\log \left[\frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{z}|\mathbf{x})} \right] \right]}_{=D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z}|\mathbf{x}))} \quad (2.1)$$

The Kullback-Leibler (KL) divergence term (second term in Equation (2.1)) penalizes the divergence of the approximate posterior from the prior, ensuring that the latent space representation remains close to the pre-defined prior distribution (which can arbitrarily chosen, but should be easy to sample from). This term is non-negative, Equation (2.2) [from [75]].

$$D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z}|\mathbf{x})) \geq 0 \quad (2.2)$$

The first term in Equation (2.1) is the The Evidence Lower Bound (ELBO), and can be rewritten as in Equation (2.3) [from [75]].

$$\mathcal{L}_{\theta, \phi}(\mathbf{x}) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} [\log p_\theta(\mathbf{x}, \mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x})] \quad (2.3)$$

The ELBO is used as the optimization objective, providing a lower bound for the log-likelihood of the input data. The ELBO consists of a reconstruction term and a Kullback-Leibler (KL) divergence term, formulated in Equation (2.4) from [75]. The joint optimization process of maximizing the reconstruction term and minimizing the KL divergence term enables the VAE to learn a meaningful lower-dimensional latent representation of the input data and generate representative new samples by leveraging the trained decoder.

$$\begin{aligned} \mathcal{L}_{\theta, \phi}(\mathbf{x}) &= \log p_\theta(\mathbf{x}) - D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z}|\mathbf{x})) \\ &\leq \log p_\theta(\mathbf{x}) \end{aligned} \quad (2.4)$$

The VAE minimizes a combination of the reconstruction loss, measuring the similarity of the reconstructed data to the original input, and the divergence loss, encouraging the learned latent distribution to approximate a predefined prior distribution (often a standard (multivariate) Gaussian).

2.3.4. Generative Adversarial Networks

A generative adversarial network (GAN) learns to generate realistic synthetic data through adversarial training of a generator and a discriminator [40], schematically depicted in Figure 2.3. The generator network, $G(z)$, samples from a random noise distribution and is trained to transform this into realistic synthetic data samples. The discriminator network, $D(x)$, is trained to perform the binary classification task of distinguishing the generator's *fake* samples from the *real* ones. The output indicates the assigned probability that the data is real as opposed to fake. The components are trained iteratively, where in the process the generator learns to produce samples that are increasingly indistinguishable from real data, while the discriminator learns to better distinguish between real and generated samples [44].

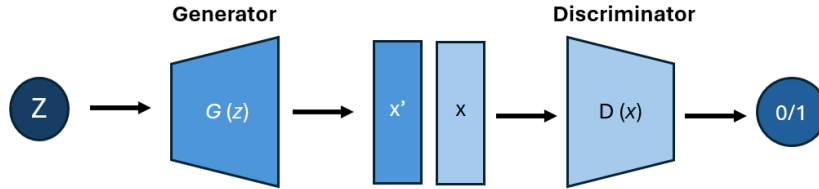


Figure 2.3: Schematic representation of a generative adversarial network, based on [51]

At a given training step, the generator is trained (keeping the discriminator fixed) to generate data samples (often starting from a random noise vector) that are as close to the input training data as possible [44, 90]. This can be formulated as minimizing the loss shown in Equation (2.5) [from [90]]. The loss is formulated as the ability to fool the discriminator network.

$$\min \mathcal{L}(G) = \mathbb{E}_{z \sim p(z)} [\log(1 - D(G(z)))] \quad (2.5)$$

The discriminator aims to correctly classify both real and generated samples, where the loss can be understood as cross-entropy [44, 90]. Remembering that the output of the discriminator is the probability that the data is real, the objective function can be formulated according to Equation (2.6) [from [90]].

$$\max \mathcal{L}(D) = \mathbb{E}_{x \sim \text{training set}} [\log D(x)] + \mathbb{E}_{z \sim p(z)} [\log(1 - D(G(z)))] \quad (2.6)$$

Ultimately, the discriminator and generator engage a two-player minimax game [44] with the complete adversarial objective function formulated according to Equation (2.7) [from [90]]. The trained generator network can subsequently be leveraged to generate new realistic synthetic data samples.

$$\min_G \max_D \mathcal{L}(D, G) = \mathbb{E}_{x \sim \text{training set}} [\log D(x)] + \mathbb{E}_{z \sim p(z)} [\log(1 - D(G(z)))] \quad (2.7)$$

Since the introduction of the GAN architecture, many variants have been proposed to counteract the instability of the training process and the common problem of mode collapse [90]. There is a latent need for balancing the constituent networks, the generator and discriminator, alongside challenges related to uninformative loss and hyperparameters tuning. If the discriminator becomes too powerful, it can quickly distinguish between real and fake samples, rendering the generator unable to produce realistic outputs. Conversely, an overpowering generator may generate samples that fool the discriminator too easily, leading to a lack of meaningful feedback for both networks and hindering convergence. Furthermore, GANs often face difficulties with uninformative loss, where traditional loss functions fail to provide meaningful gradients for effective learning, and are misaligned with the true quality of the samples [40]. Variants of GANs aimed at targeting these shortcomings will be discussed in more detail in Chapter 4.

2.3.5. Diffusion Models

Diffusion models (focusing on the Denoising Diffusion Probabilistic Model (DDPMs) formulation) are comprised of a forward and reverse process on data. The forward diffusion process, $q(x_t | x_{t-1})$, incrementally adds Gaussian noise to data transforming the original input into pure noise. Conversely, the reverse diffusion

process, $p_\theta(x_{t-1} | x_t)$, incrementally reverts this process to construct data samples starting from the noisy distribution [40]. It is the sequence of invertible transformations that is learned, and this spreads the noise throughout the data space and keeps the latent structure of the true data distribution [55]. A schematic depiction of this process is provided in Figure 2.4.

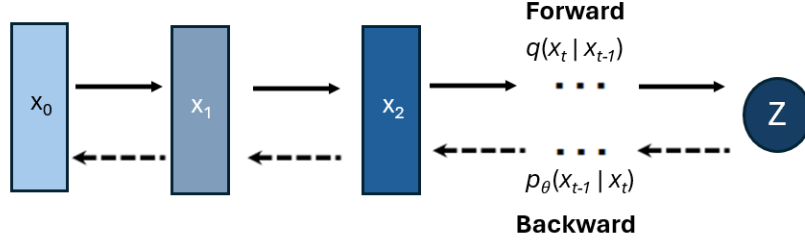


Figure 2.4: Schematic representation of a diffusion model, based on [51]

The forward diffusion process transforms data from the input data to random noise in a sequence of steps. Each step corresponds to the incorporation of unparameterized Gaussian noise (with a varying variance) in Equation (2.8) [from [55]], and these latent variables are of the same dimensionality as the input.

$$q(\mathbf{x}_{1:T} | \mathbf{x}_0) := \prod_{t=1}^T q(\mathbf{x}_t | \mathbf{x}_{t-1}), \quad q(\mathbf{x}_t | \mathbf{x}_{t-1}) := \mathcal{N}(\mathbf{x}_t; \sqrt{1 - \beta_t} \mathbf{x}_{t-1}, \beta_t \mathbf{I}) \quad (2.8)$$

The backward diffusion process reverses this and represents a Markov chain with the *learned* Gaussian steps (unlike the unparameterized forward process) [55]. The backward process corresponding to Equation (2.8) is formulated in Equation (2.9) [from [55]]. The starting point is pure Gaussian noise $p(\mathbf{x}_T) = \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$.

$$p_\theta(\mathbf{x}_{0:T}) := p(\mathbf{x}_T) \prod_{t=1}^T p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t), \quad p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t) := \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \Sigma_\theta(\mathbf{x}_t, t)) \quad (2.9)$$

Training of the diffusion model is performed by optimizing the variational lower bound (VLB) of the negative log-likelihood of the observed data (while simultaneously approximating the posterior) [55], formulated in Equation (2.10) [from [55]].

$$\mathbb{E}[-\log p_\theta(\mathbf{x}_0)] \leq \mathbb{E}_q \left[-\log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T} | \mathbf{x}_0)} \right] = \mathbb{E}_q \left[-\log p(\mathbf{x}_T) - \sum_{t=1}^T \log \frac{p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t)}{q(\mathbf{x}_t | \mathbf{x}_{t-1})} \right] =: L \quad (2.10)$$

The forward process can be sampled at an arbitrary time step using the closed form equation in Equation (2.11) [from [55]], where $\alpha_t := 1 - \beta_t$ and $\bar{\alpha}_t := \prod_{s=1}^t \alpha_s$. This has the beneficial effect of speeding up training as the latent space can more rapidly be explored in fewer iterations [40, 55].

$$q(\mathbf{x}_t | \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t} \mathbf{x}_0, (1 - \bar{\alpha}_t) \mathbf{I}) \quad (2.11)$$

With proved success in images, these diffusion based models are finding applications in time series forecasting, imputation, and generation. The score-based generative model and stochastic differential equations based formulations of diffusion models are alternatives [88].

2.4. Temporal Dynamics in Synthetic Multivariate Time Series Data

Given the scope of generating realistic aircraft trajectories, the temporal component of the underlying data is given special attention. The sequential dependencies and class of neural networks, Recurrent Neural Networks (RNNs), designed to handle this are presented in Section 2.4.1. The Long Short-Term Memory (LSTM) model, described in Section 2.4.2, is a variant of RNNs that addresses the limitation of traditional RNN architectures in capturing long-term dependencies in the data: *the vanishing gradient problem*. Recently, transformers (using the self-attention mechanism) have gained traction as a new method of capturing temporal dependencies, explained in Section 2.4.3. Both of these models are often incorporated in generative architectures aimed at time-series data, in order to capture the sequential dependencies.

2.4.1. Time Series & Recurrent Neural Networks

Time Series

Time series data are a sequence of data points indexed in time, adhering to chronological order. The spacing between data points can be regular (uniform) or irregular [50]. Data points over time enable analysts to discern temporal dependencies and make predictions about future values. Time series have far-reaching applications including in communications, finance, meteorology, and astronomy. In its simplest form, each time step in the sequence contains one variable, representing a *univariate* series. When multiple variables are associated with each time step, the series becomes *multivariate* [50].

Aircraft trajectories represent multivariate time series due to the simultaneous observation of multiple flight parameters including latitude, longitude, altitude, airspeed, and heading over time [124]. A set of variables is required to specify the aircraft's state at a given time step. Furthermore, these flight parameters evolve with a temporal dependency on prior observations.

Recurrent Neural Networks

Recurrent neural networks (RNNs) represent a distinct type of networks designed for handling sequential data effectively. The description and main outline of an exemplary RNN with hidden states provided in this section are based on [142]. RNNs take advantage of parameter sharing to generalize well to sequences of variable length, essentially performing the same operation at each time step. At their core is the recurrent connection, which is a cycle of the nodes in the network. Figure 2.5 is a schematic showing the logic of an RNN with hidden states.

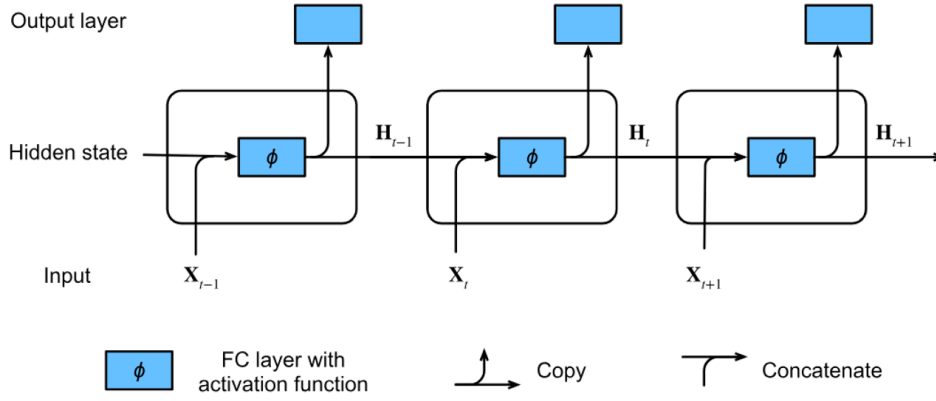


Figure 2.5: Schematic representation of a recurrent neural network, from [142]

To avoid the exponential increase in model parameters with sequence length, a latent variable model is relied on using hidden state that store all information up to that time step, Equation (2.12) [from [142]]. The hidden state is computed using the corresponding input at that time step and the previous hidden state, $h_t = f(x_t, h_{t-1})$.

$$P(x_t | x_{t-1}, \dots, x_1) \approx P(x_t | h_{t-1}) \quad (2.12)$$

At a given time step, the previous hidden state is weighted with fixed learned parameters ($\mathbf{W}_{hh}$) and this information is concatenated with the weighted input at the current time step ($\mathbf{W}_{xh}$) and a bias term, according to Equation (2.13) [from [142]]. The network is *recurrent* because the computations are equivalent at each layer of the network.

$$\mathbf{H}_t = \phi(\mathbf{X}_t \mathbf{W}_{xh} + \mathbf{H}_{t-1} \mathbf{W}_{hh} + \mathbf{b}_h) \quad (2.13)$$

The output follows the familiar form of a Multilayer Perceptron (MLP), through multiplication of the current hidden state with a weight matrix and adding bias term, formulated in Equation (2.14) [from [142]].

$$\mathbf{O}_t = \mathbf{H}_t \mathbf{W}_{hq} + \mathbf{b}_q \quad (2.14)$$

2.4.2. Long Short-Term Memory Model

An intrinsic problem to the design of RNNs is the fragility of the gradient when propagated over long sequences, either vanishing or exploding (the former being more common) [40]. The difficulties of capturing critical long-term dependencies in data required to tailored approaches, where the Long Short-Term Memory (LSTM) model is arguably the most effective and widely adopted. The description and main outline of the LSTM model provided in this section are based on [142]. The LSTM incorporates specialized memory cells and gating mechanisms that enable capturing and retaining information over extended sequences (schematically depicted in Figure 2.6). The memory cell maintains an internal state and specific mechanisms control the *updating* and *resetting* of the hidden state.

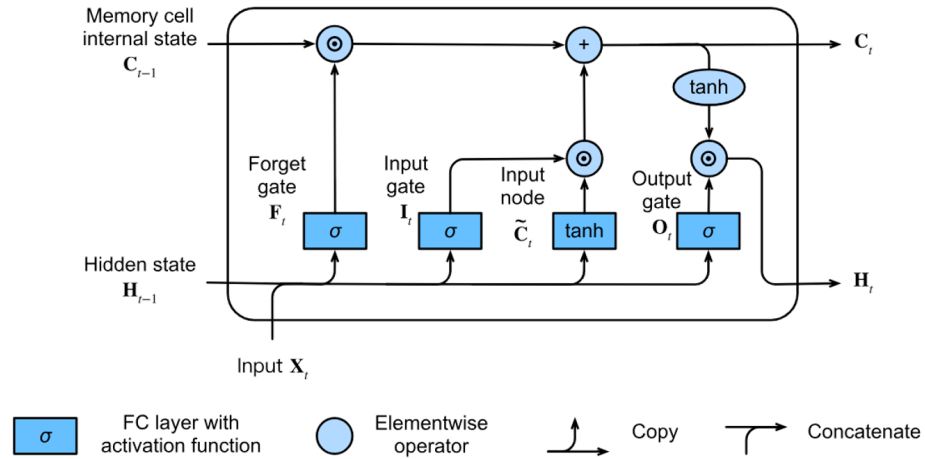


Figure 2.6: Schematic representation of a long short-term memory model, from [142]

The input at the current time step and the previous step's hidden state are fed into the LSTM gating mechanisms, Equation (2.15) [from [142]]. The input gate, I_t , controls the input of new data into the memory cell, determining whether information is stored in the cell state. The forget gate, F_t , controls which information should be discarded from the memory cell, preventing the overwhelming of the network due to irrelevant information. The output gate, O_t , regulates the information flow from the memory cell to the network's output. All three gates employ a sigmoid activation function to ensure the output is between zero and one.

$$\begin{aligned} I_t &= \sigma(\mathbf{X}_t \mathbf{W}_{xi} + \mathbf{H}_{t-1} \mathbf{W}_{hi} + \mathbf{b}_i) \\ F_t &= \sigma(\mathbf{X}_t \mathbf{W}_{xf} + \mathbf{H}_{t-1} \mathbf{W}_{hf} + \mathbf{b}_f) \\ O_t &= \sigma(\mathbf{X}_t \mathbf{W}_{xo} + \mathbf{H}_{t-1} \mathbf{W}_{ho} + \mathbf{b}_o) \end{aligned} \quad (2.15)$$

To design the memory cell, an input node (candidate cell, $\tilde{C}_t$) is also introduced using a tanh activation function (Equation (2.16) [from [142]]), resulting in a value between negative one and one. The memory cell's internal state, C_t , can be computed by weighing the current candidate internal state and the previous internal state with the input gate and forget gate, respectively (Equation (2.16) [from [142]]).

$$\begin{aligned} \tilde{C}_t &= \tanh(\mathbf{X}_t \mathbf{W}_{xc} + \mathbf{H}_{t-1} \mathbf{W}_{hc} + \mathbf{b}_c) \\ C_t &= F_t \odot C_{t-1} + I_t \odot \tilde{C}_t \end{aligned} \quad (2.16)$$

The hidden state, H_t , is computed with the Hadamard product between the memory cell's internal state (after an activation function) and the output gate, formulated in Equation (2.17) [from [142]]. An output gate value of zero means that the current memory does not have an impact on the other layers at this time step.

$$\mathbf{H}_t = O_t \odot \tanh(C_t) \quad (2.17)$$

It is worth noting that the gated recurrent unit (GRU) is another popular model that provides relief of the vanishing gradient problem [142], and that other architectures exist for handling sequential data (e.g. temporal convolution neural networks (CNNs) [78]). Some of these are further discussed in Chapter 4.

2.4.3. Self-Attention & Transformers

Unlike traditional fixed receptive fields in convolutional or recurrent architectures, self-attention leads elements in the input sequence to attend to any other element, effectively weighing their relevance. The goal of self-attention is to compute a weighted representation of each element in the sequence based on its relationship with every other element (schematically depicted in Figure 2.7). Unlike RNNs that inherently rely on sequential operations, the self-attention mechanism takes advantage of parallel computation [142].

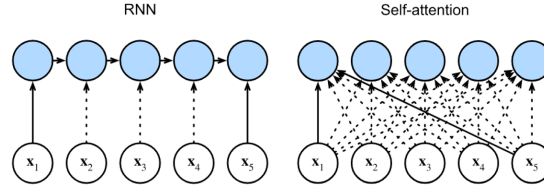


Figure 2.7: RNN vs self-attention architectures, from [142]

The attention value for each point in the sequence is computed by evaluating the similarity between the current point and all other points in the sequence. This involves transforming the input sequence into three sets of vectors: query $\mathbf{Q}$, key $\mathbf{K}$, and value $\mathbf{V}$, vectors. Variation in attention functions exist, but the scaled dot product version formulated in Equation (2.18) [from [142]] is very common. The function computes an output through a weighted sum of the values, whereby the assigned weight is a function of the query and key [129]. Different representations of the input can be attended to by using multi-head attention [129, 142].

$$\begin{aligned}
 \mathbf{Q} &= \mathbf{x}\mathbf{W}_Q \\
 \mathbf{K} &= \mathbf{x}\mathbf{W}_K \\
 \mathbf{V} &= \mathbf{x}\mathbf{W}_V
 \end{aligned}
 \tag{2.18}$$

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d}}\right)\mathbf{V}$$

The transformer architecture consists of an encoder-decoder structure, each composed of multiple layers of self-attention and feedforward neural networks (schematically depicted in Figure 2.8). In the encoder, self-attention allows the model to weigh the relevance of each input value, capturing both local and global dependencies efficiently. The decoder utilizes masked self-attention to ensure causal generation during training and employs cross-attention to attend over the encoder's output during decoding. Residual connections and layer normalization stabilize training and enhance information flow across layers [129, 142].

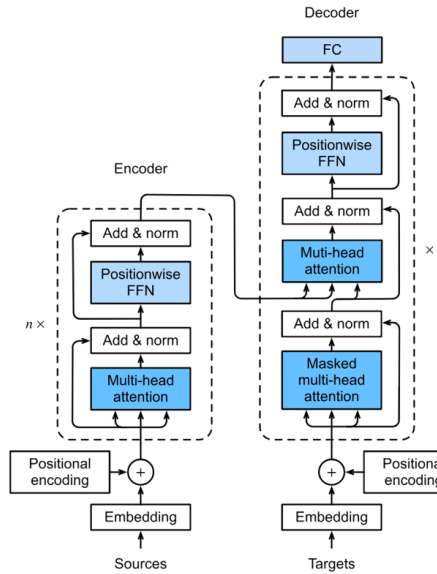


Figure 2.8: Schematic representation of a transformer architecture, from [142]

2.5. Synthetic Time Series Quality Evaluation

The evaluation of synthetic data quality is nontrivial and remains an active area of research. The generative learning trilemma, explained in [Section 2.5.1](#), describes the difficulty in simultaneously generating high quality samples, a well diversified set of them, and ensuring fast sampling. Previously proposed models have often followed the framework delineated in [\[138\]](#), used to analyze the TimeGAN model. This relies on a three-part assessment, namely of data diversity, data fidelity, and data usefulness. [Section 2.5.2](#) presents two techniques that can be used to visually assess the diversity of generated samples by reducing the high-dimensional data. [Section 2.5.3](#), highlights how a discriminative model (classifier) can be used to assess realism. The usefulness of the synthetic data can be quantified by a train-synthesized-test-real (TSTR) evaluation paradigm, presented in [Section 2.5.4](#). Pertinent to aircraft trajectories, the synthetic trajectories must also be flyable by an aircraft from a laws of physics and aircraft performance point of view. The open-source air traffic simulator *BlueSky* [\[56\]](#) can be leveraged to replay versions of synthetic trajectories and quantify physical realism by measuring the distance between the synthetic and replayed versions, as described in [Section 2.5.5](#). A study on the distance metrics needed for this is presented in [Section 2.5.6](#).

2.5.1. The Generative Learning Trilemma

The *generative learning trilemma* has been coined by [\[134\]](#), referring to the ongoing struggle of deep generative models to simultaneously meet the most important requirements in this field: high sample quality, mode coverage, and a fast way to sample new data. [Figure 2.9](#) depicts the tradeoff of common generative models with respect to the trilemma. The discussion of this concept provides a natural next step in developing a framework to assess the quality of the generated samples.

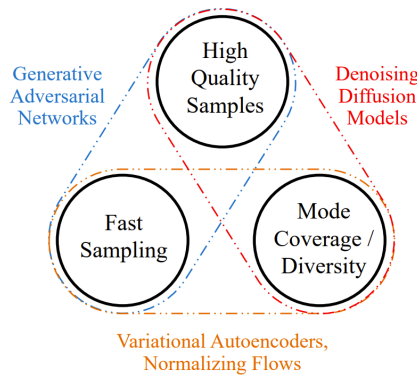


Figure 2.9: Generative learning trilemma, from [\[134\]](#)

The trilemma arises from the tension between the desire for high-quality samples, comprehensive mode coverage, and fast and computationally efficient sampling. The three families of generative models central to this thesis address these requirements differently. GANs [\[44\]](#) excel at generating high-quality samples rapidly but may lack comprehensive mode coverage. VAEs [\[75\]](#) are better at covering all data modes but may suffer from lower sample quality. Diffusion models [\[55\]](#) have emerged as powerful generative models that demonstrate good results in sample quality and mode coverage, but sampling from them often brings a significant computational cost [\[134\]](#). The authors of the paper ultimately propose a denoising diffusion GAN, where conditional GANs are used to model the denoising distributions.

2.5.2. PCA and t-SNE for Data Diversity

With regard to data diversity, it is the goal that the distribution of synthetic data generated by a model is comparable to the distribution of real (historical) data. Visual assessment of the diversity can be realized with the use of Principal Component Analysis (PCA) [\[10\]](#) and t-Distributed Stochastic Neighbor Embedding (t-SNE) [\[128\]](#). Both techniques allow for the visualization of data in a lower-dimensional space, referred to as a dimensionality reduction. PCA, a linear technique, identifies the principal axes of maximum variance in data, allowing for the projection of data points onto a lower-dimensional subspace while preserving the most significant sources of variation. In contrast, t-SNE, a nonlinear technique, focuses on preserving local

structure and clustering patterns by optimizing the embedding of data points in a low-dimensional space to minimize the mismatch between pairwise similarities in the high-dimensional and low-dimensional spaces. Figure 2.10 shows (a selection of) the results of the PCA and t-SNE analysis in the paper presenting TimeGAN [138]. Both techniques reveal clear similarities in the distributions suggesting that the synthetically generated data is representative of true data characteristics.

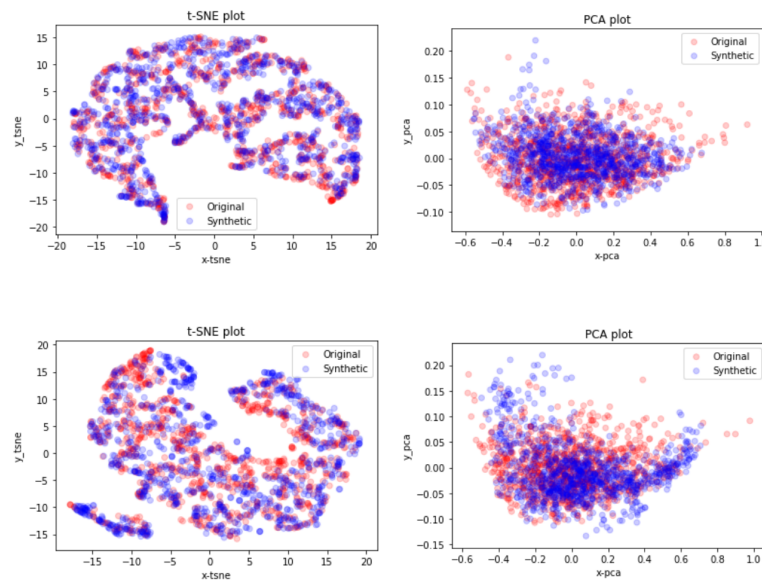


Figure 2.10: t-SNE and PCA results from TimeGAN (row 1) and RCGAN (row 2), from [138]

2.5.3. Discriminative Scoring for Data Fidelity

Another effective method for evaluating synthetic data is through the use of discriminative scores. This technique involves training a classification model, often an RNN such as the LSTM model, to distinguish between sequences generated by a synthetic data generator and those originating from the original/historical real dataset. The fundamental premise is that if the synthetic data accurately captures the characteristics of the real data, it should pose as a challenge for the classifier to differentiate between the two. To implement this approach, the initial step entails labeling sequences from the original dataset as *real* and those generated by the synthetic data process as *fake*. Subsequently, a binary classification task is formulated to train a classifier. The model is trained to discern patterns that differentiate between real and synthetic sequences. The discriminative score then quantifies the classifier's proficiency in this task, providing a metric for gauging the degree of similarity between the two types of sequences [138].

A low discriminative score indicates that the synthetic data closely mirrors the original data, making it difficult for the classifier to distinguish between the two. Conversely, a high discriminative score suggests notable discrepancies between the synthetic and real data, facilitating easier differentiation by the classifier. The train and test classification errors and the Receiver Operator Curve (ROC) area under the curve can be plotted as a function of epochs to inspect performance as well [61, 138].

2.5.4. Predictive Scoring for Data Usefulness

The predictive score serves as a measure of how well a synthetic dataset captures the predictive characteristics of the original data, particularly in terms of conditional distributions over time. This technique involves training a sequence prediction model, again typically using an RNN like LSTM, on the synthetic dataset to predict next-step temporal vectors over each input sequence. The model is then evaluated on the original dataset to assess its predictive performance. There are various evaluation metrics (e.g. MAE) that can be used to quantify the average discrepancy between the predicted next-step temporal vectors and the actual observed values [138].

To implement this technique, use is made of both a train-real-test-real (TRTS) and train-synthetic-test-real (TSTR) paradigm. The former acts as benchmark for performance, whereas the latter reveals whether the synthetic data improves downstream tasks (by increasing performance) [61, 138].

2.5.5. Physical Flyability Using BlueSky

Naturally, the context of the synthetic data should be taken into account in the evaluation of it. In this setting, it is of upmost importance that the generated aircraft trajectories are flyable from a physics point of view, where this physical realism is perhaps the most difficult to assess. The work in [77] provides a crude, initial attempt to quantify this realism with the curvature of variation. Aircraft trajectories largely consist of straight line segments connected by turns, where a flight path should not be unnecessarily *wiggly*. The authors measure the number of times a flight trajectory exceeds a certain curvature threshold, where Figure 2.11 provides an example of the results. Clearly this metric does not suffice as an encompassing measure of flyability, but it can serve as an initial filter.

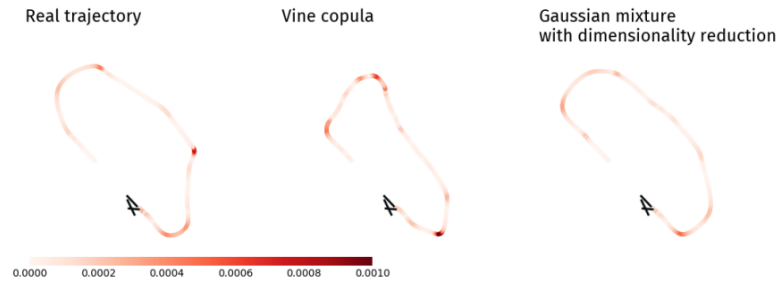


Figure 2.11: Curvature comparison of real and synthetic trajectories, from [77]

A more complete framework to evaluate generated aircraft trajectories is proposed in [100]. Synthetically generated trajectories are replayed in BlueSky, an open-source air traffic simulator developed at TU Delft [56]. BlueSky was proposed as an initiative to create common ground on traffic scenarios, metrics, and simulations tools for researchers in the ATM domain. It relies on an open-source, open-data approach where the model is comparable in fidelity to commercial-grade alternatives. The required data for simulations consists of navigation data (including geographical information, way points, airports, and sector boundaries) and aircraft performance data (including drag polars, engine performance, autopilot settings, and procedure speeds) [56]. The former is sourced from the publicly available Aeronautical Information Service (AIS) and the latter based on the Base of Aircraft Data (BADA), conceptual aircraft design methods, and ML performance models based on vast quantities of ADS-B data. BlueSky comes with a graphical user interface (depicted in Figure 2.12 and is built to be user-friendly.

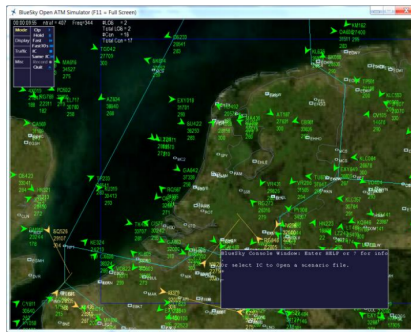


Figure 2.12: The BlueSky graphical user interface, from [56]

The physical realism evaluation method based on relaying generated trajectories using the BlueSky simulator proposed in [100], uses two different modeling approaches. The first method requires detection of alignments on navigational way points and directing the simulator with instructions to follow them. Secondly, the Douglas-Peucker algorithm is used to recursively simplify trajectories by focusing on the most prominent points and using those as instructions for the simulator. In both cases, the underlying assumption is that BlueSky is able to replay all trajectories exactly, which is known not to be the case. For example, BlueSky is not equipped to handle runway alignment or trajectories that intersect with itself (characteristic of holding patterns) [78, 100]. To circumvent this, complementary to replaying synthetic trajectories, a reference dataset of historic trajectories is replayed in BlueSky and used as a benchmark for realism.

Use is made of a dedicated Python library (*traj-dist*, [49]) to compute nine distance metrics between two paths (synthetic and replayed), among them the Dynamic Time Warping (DTW) [8] and the Symmetric Segment-Path Distance (SSPD) [9]. The cumulative densities of a set of these two distances are visualized in Figure 2.13, to account for variability in the degree of accuracy in a set of trajectories.

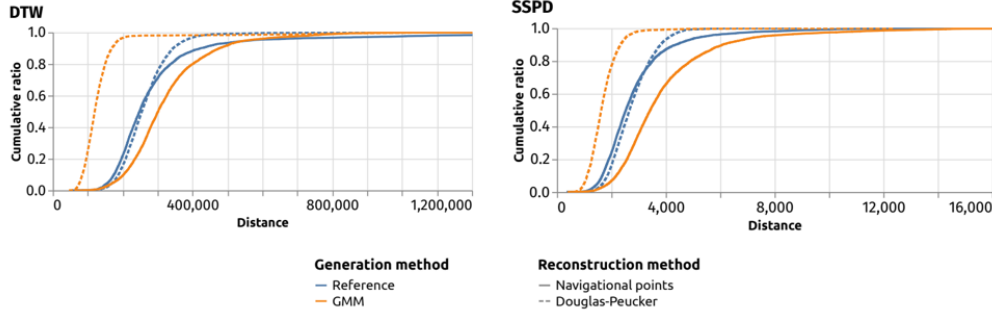


Figure 2.13: Cumulative distribution of DTW and SSPD distances between real and synthetic trajectories, from [100]

This analysis can be taken one step further by plotting distance distributions for varying trajectory patterns Figure 2.14, revealing whether certain trajectories are more difficult to simulate realistically. Lastly, it must be noted that operational relevance of generated trajectories can also be determined by experts such as air traffic controllers and pilots who are familiar with the pertinent geographic area [100]. The next section provides more details on the various warping based, shape-based, and statistical distance metrics.

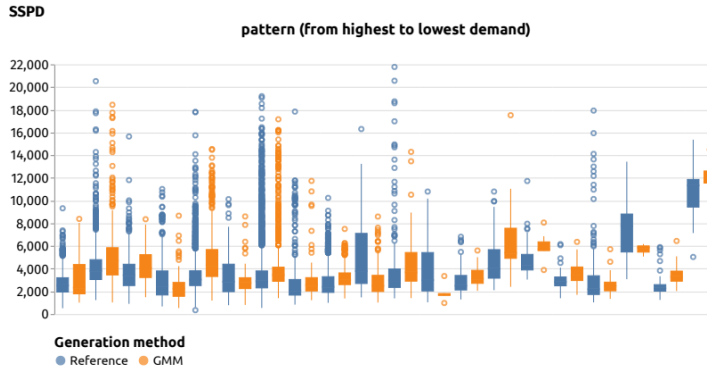


Figure 2.14: Boxplots of SSPD distances (between synthetic and real trajectories) for varying trajectory patterns, from [100]

2.5.6. Distance Metrics

Warping Based Distances

When real and synthetic trajectories are compared for realism quantification, distance metrics are used. With trajectories, tailored distance metrics are required to accommodate trajectories of different lengths and time indexing. The distance metrics can be categorized as being shape-based or warping based [9]. *Classical* distances including Euclidean and Manhattan compare objects' indices one by one, meaning that variations in time between trajectories render these methods inappropriate. Warping distances overcome this shortcoming by seeking an optimal alignment between trajectories given a cost function. DTW [8], LCSS [130], EDR [18], and ERP [17] are well-known warping based distances that vary only in the cost function that is optimized. To calculate the warping distance, two trajectories (T^i, T^j) are arranged on a $n^i \times n^j$ grid, where a grid cell $g_{k,l}$ points the pair (p_k^i, p_l^j) . The warping path traverses the grid, whereby locations in a trajectory can be repeated, deleted, or replaced, as long as the order is maintained. The distance is computed as the optimum (minimum or maximum) cost between paired trajectory locations formulated in Equation (2.19) [from [9]], where $\delta(w_k) = \delta(g_{k_i, k_j}) = \delta(p_{k_i}^i, p_{k_j}^j)$ is the cost function of elements of the warping path W [9].

$$D(T^i, T^j) = \min_W \left[\sum_{k=1}^{|W|} \delta(w_k) \right] \text{ or } = \max_W \left[\sum_{k=1}^{|W|} \delta(w_k) \right] \quad (2.19)$$

Warping based distances allow for comparison of trajectories that differ in length, however, they require a clear definition of a reference. In this context, the reference for the synthetic trajectory is the replayed version in the BlueSky simulator. As an alternative for correcting the time indexing, the temporal nature of the data can also be removed all together.

Shape-Based Distances

Aptly named, shape-based distances instead focus on geometric features (the shape) of the data, rather than aligning them in time [9]. Distances quantify the similarity between trajectories based on their geometric properties, such as curvature, direction changes, or spatial distribution. Two age-old distances regularly used in literature are the Hausdorff distance [52] and the Fréchet distance [41]. The Hausdorff distance quantifies the maximum discrepancy between two trajectories by measuring the maximum distance of a point in one trajectory to the nearest point in the other trajectory. It captures the worst-case scenario of how far apart points on one trajectory can be from points on the other trajectory, formulated in Equation (2.20) [from [9]].

$$D_{\text{Hausdorff}}(X, Y) = \max \left\{ \sup_{x \in X} \inf_{y \in Y} \|xy\|_2, \sup_{y \in Y} \inf_{x \in X} \|xy\|_2 \right\} \quad (2.20)$$

The Fréchet distance [41] measures the similarity between two curves or trajectories by considering the closest approach of a parametric point moving along one trajectory to a point moving along the other trajectory. It captures the similarity in terms of the closest distance that must be traveled by both trajectories simultaneously, formulated in Equation (2.21) [from [9]].

$$D_{\text{Fréchet}}(A, B) = \inf_{\alpha, \beta \in X} \max_{t \in [0,1]} \{ \|A(\alpha(t)), B(\beta(t))\|_2 \} \quad (2.21)$$

More recently, the Symmetric Segment-Path Distance (SSPD) [9] was introduced. It relies on minimum distances between points (p) on one trajectory and all segments (s) that comprise the other trajectory. The segment path from one trajectory to the other is the mean of these distances (D_{pt}) of all points on the former trajectory, formulated in Equation (2.22) [from [9]].

$$D_{\text{SSPD}}(T^1, T^2) = \frac{1}{n_1} \sum_{i_1=1}^{n_1} D_{pt}(p_{i_1}^1, T^2) \quad (2.22)$$

$$\text{where, } D_{pt}(p_{i_1}^1, T^2) = \min_{i_2 \in [0, \dots, n_2-1]} D_{ps}(p_{i_1}^1, s_{i_2}^2)$$

Due to the fact that this distance is not symmetric, the mean of both *directions* is computed, justifying the name of the distance. The final distance is formulated in Equation (2.23) [from [9]]. SSPD, alongside all other previously mentioned distances, can be readily computed using packages software in the *traj-dist* Python library [49].

$$D_{\text{SSPD}}(T^1, T^2) = \frac{D_{\text{SSPD}}(T^1, T^2) + D_{\text{SSPD}}(T^2, T^1)}{2} \quad (2.23)$$

Statistical Distances

The distances outlined in the previous section compute pairwise comparisons between trajectories. Complementary to this approach, statistical evaluation can be performed by comparing the distribution of synthetic trajectories with the distribution of real trajectories. The Kolmogorov-Smirnov test, Kullback-Leibler divergence, Kolmogorov distance, Total Variation distance, and Wasserstein distance are well-known metrics used for this type of statistical evaluation [100]. The Kolmogorov-Smirnov test is a statistical method used to determine whether two samples originate from the same continuous distribution. It computes the maximum difference between the cumulative distribution functions, and returns a p-value indicating the likelihood of the observed difference under the null hypothesis of identical distributions. In contrast, the Kullback-Leibler divergence quantifies the difference between probability distributions by measuring the average information gain when one distribution is used to approximate another. The Kolmogorov distance and Total Variation distance both assess the discrepancy between distributions, with the former focusing on the maximum difference between their probability densities, while the latter measures the total variation of their mass functions. Finally, the Wasserstein distance, popular GANs, calculates the minimum cost of transforming one

distribution into another, often used for comparing distributions with different shapes and supports. Each of these metrics provides valuable insights into the dissimilarity between probability distributions, facilitating various applications in statistics, machine learning, and beyond.

Lastly, Mahalanobis distance and e-distance have been employed as distance metrics when evaluating multi-variate density models for synthetic aircraft trajectories, based on samples of generated synthetic trajectories $A = \{\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_{N_1}\}$ and real trajectories $B = \{\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_{N_2}\}$ [77]. The Mahalanobis distance of the generated trajectories to the mean of the real trajectories can be computed according to Equation (2.24) [from [77]].

$$D_M = \frac{1}{N_1} \sum_{i=1}^{N_1} \sqrt{(\mathbf{a}_i - \mu)^T \Sigma^{-1} (\mathbf{a}_i - \mu)} \quad (2.24)$$

Additionally, the Mean Mahalanobis distance of a synthetic trajectory to each real trajectory can be computed according to Equation (2.25) [from [77]].

$$D_G = \frac{1}{N_1 N_2} \sum_{i=1}^{N_1} \sum_{j=1}^{N_2} \sqrt{(\mathbf{a}_i - \mathbf{b}_j)^T \Sigma^{-1} (\mathbf{a}_i - \mathbf{b}_j)} \quad (2.25)$$

E-distance is used to determine if two sets of trajectories of varying length follow the same distribution [121]. It is based on the expected value of the Euclidean distances between points on both trajectories, and is computed according to Equation (2.26) [from [77]]. The value of this metric will asymptote to a (positive) constant value if trajectories from both sets follow the same distribution, else it increases to infinity.

$$e(A, B) = \frac{N_1 N_2}{N_1 + N_2} \left(\frac{2}{N_1 N_2} \sum_{i=1}^{N_1} \sum_{j=1}^{N_2} \|\mathbf{a}_i - \mathbf{b}_j\| - \frac{1}{N_1^2} \sum_{i=1}^{N_1} \sum_{j=1}^{N_1} \|\mathbf{a}_i - \mathbf{a}_j\| - \frac{1}{N_2^2} \sum_{i=1}^{N_2} \sum_{j=1}^{N_2} \|\mathbf{b}_i - \mathbf{b}_j\| \right) \quad (2.26)$$

3

Existing Aircraft Flight Trajectory Generation Methods

Synthetic aircraft trajectory generation encompasses a diverse range of methods and techniques presented in literature. This chapter provides an overview of this variation, highlighting the underlying assumptions, merits, and results. [Section 3.1](#) arguably covers the majority of the literature, namely model-driven methods based on flight dynamic equations. Conversely, data-driven methods rely purely on existing trajectory data for generation. The common approach of flight plan database extrapolation is presented in [Section 3.2](#), allowing for entire synthetic traffic sets. In [Section 3.3](#), a method based on a Markov chain model is presented, and a Gaussian Mixture model for identifying representative trajectory patterns is explained in [Section 3.4](#). Recently, deep generative modeling techniques have been employed, specifically a GAN in [Section 3.5](#) and a temporal convolutional VAE in [Section 3.6](#).

3.1. Model-Driven Methods Based on Flight Dynamic Equations

Model-driven methods refer to (mostly) deterministic techniques to generate aircraft trajectories under constraints. [Section 3.1.1](#) provides an introduction to this modeling paradigm through comparison with data-driven techniques and discussing its key shortcomings. Two well-recognized aircraft performance models are Base of Aircraft Data (BADA) [\[97\]](#) and Open Aircraft Performance (OpenAP) [\[119\]](#), presented in [Section 3.1.2](#) and [Section 3.1.3](#), respectively. These will be described in more detail to get insights into their value proposition as synthetic aircraft trajectory generators.

3.1.1. Model-Driven Rationale & Limitations

At the highest level, a distinction between model-driven and data-driven approaches is made. Model-driven aircraft trajectory generation relies on mathematical models to calculate trajectories based on aircraft dynamics and environmental constraints, optimizing objectives like fuel consumption and flight time [\[100\]](#). Data-driven approaches utilize historical flight data, frequently employing machine learning techniques to predict trajectories without relying on predefined models, offering adaptability but requiring significant data for training. Model-driven generation is considered the classical approach and comprises most existing methods in literature [\[78\]](#). Established kinematics and aerodynamic models of aircraft performance sit at their core and trajectories are ultimately generated through an optimization of a specified criterion.

The optimization process can take different forms, an example is the use of dynamic programming to generate optimal trajectories (in terms of fuel consumption and emissions) given designated arrival slots [\[131\]](#). A wide variety of constraints can also be incorporated into the optimization, for the purpose of avoiding restricted areas [\[27\]](#) and adverse weather scenarios [\[71\]](#), for example. Physical and statistical modeling of aircraft performance characteristics are an alternative. BADA [\[97\]](#) and OpenAP [\[119\]](#) are industry-wide recognized examples of such models and are capable of generating flyable trajectories. The deterministic nature of these models represents a limitation. There is a requirement for information on aircraft performance, environmental factors, and various other flight input parameters that inherently withhold the model from

including uncertainty. This translates into an inability to generate larger sets of randomly diversified trajectories. Previous attempts to address this largely rely on explicitly including slight variations in the flight dynamics models, however, this included randomness is not necessarily representative of what is observed [78, 100]. Data-driven methods make no such simplifying assumptions when incorporating uncertainty.

3.1.2. BADA: Base of Aircraft Data

BADA [97], developed by Eurocontrol, stands as a cornerstone in modern aviation for its comprehensive repository of aircraft performance data. This standardized database catalogues parameters governing aircraft behavior, including aerodynamics, propulsion, weight characteristics, and more nuanced factors such as flap and gear configurations. Drawing from extensive empirical data collected from flight tests, manufacturer specifications, and operational data, BADA is able to simulate and predict aircraft performance across various flight phases and operating conditions, used for ATM R&D and strategic planning of ground operations units [97]. From the database, a set of performance parameters of a specific aircraft referred to as the *Aircraft Characteristics* is used by both the *Aircraft Performance Model (APM)* and the *Airline Procedure Model (ARPM)*, where Figure 3.1 outlines the structure of the model. The APM treats an aircraft as a point and uses a kinematic approach to modeling forces on the varying aircraft weight. The aerodynamic, propulsive, and gravitational forces are computed in the action model. The motion model uses kinematic relations to calculate the performance of the aircraft along a trajectory. Different operational possibilities are handled by the operations model, and the limitations model ensures the aircraft flies within the safe limits of the flight envelope. The ARPM contains information on nominal aircraft procedures, which vary for airspace.

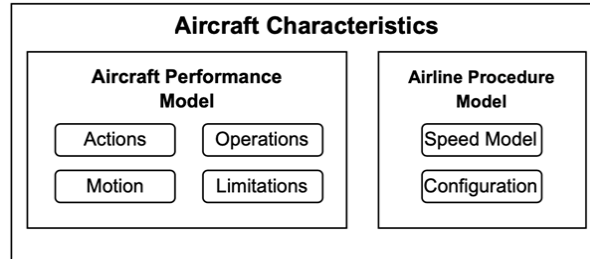


Figure 3.1: Organization of the BADA model, from [97]

The aircraft motion model is formulated in a system of differential equations (which may vary for different operating conditions). The initial/boundary conditions of these equations can vary according to flight segment, and the concatenation of the solutions thereof results in a computed trajectory. Figure 3.4 provides an example of a computed trajectory in terms of geopotential pressure altitude and calibrated airspeed. The full trajectory is an ensemble of a piece-wise defined set of segments.

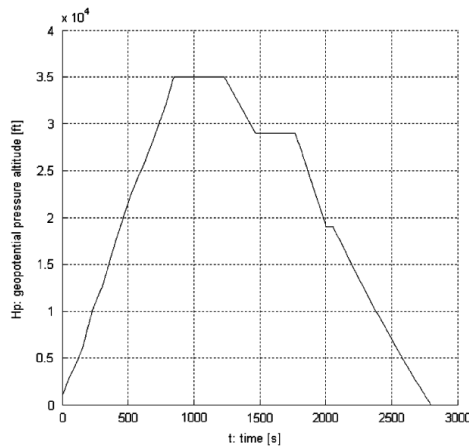


Figure 3.2: Geopotential altitude over time

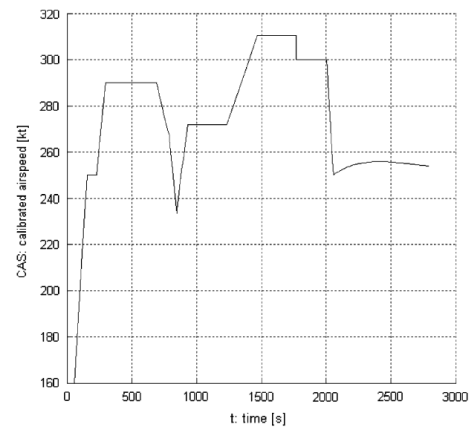


Figure 3.3: Calibrated airspeed over time

Figure 3.4: Example trajectory generated by BADA, from [97]

3.1.3. OpenAP: Open Aircraft Performance

OpenAP [119] serves as the open-source alternative to BADA. The model structure consists of four components (aircraft properties, kinematics, dynamics, and utilities), visualized in Figure 3.5. Aircraft properties holds information on characteristics of the aircraft, alongside details on the engine configurations and performance of common engine types. Kinematics concerns the motion of the aircraft along seven flight phases, whereby the results of a data-driven approach is used. The dynamic performance module extends this by incorporating metrics on mass and force on the aircraft, again reducing the aircraft to a point. The utilities module contains information on flight phases, standard calculations/conversions, and holds navigation data.

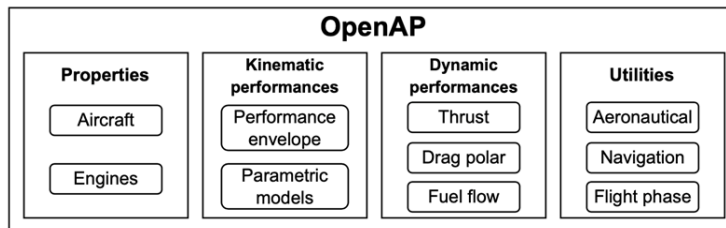


Figure 3.5: Organization of the OpenAP model, from [119]

The OpenAP model comes with a dedicated trajectory generator capable of generating high quantities of flights. Initial conditions and time step latency can be explicitly specified or be drawn from a probability distribution. In the case of full trajectories, individual flight segments are concatenated using cruise altitude and Mach number guides. Alternatively, noise can be introduced using the inherent noisy characteristics of ADS-B data. Figure 3.6 provides an example climbing trajectories, with random variation (gray) around the characteristic (red) climb profile.

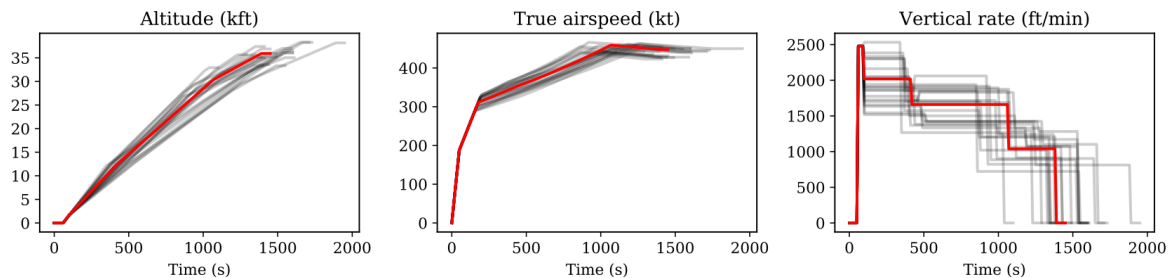


Figure 3.6: Example climb trajectory generated by OpenAP in terms of altitude, true airspeed, and vertical rate over time, from [119]

3.2. Flight Plan Database Statistical Extrapolation

A frequently implemented method to generating full synthetic air traffic situations (as opposed to individual trajectories) is using simple models for aircraft populations and trajectories [47, 57, 107]. The following description of such a procedure is based on [47], and can be considered characteristic. Central to this methodology is a reliable, verified flight plan database. An example is the database compiled by the International Air Transport Association (IATA), storing data on passenger and cargo IFR flights around the world. Two reference days are selected for their representation of typical daily flight volumes, including overnight flights. This data provides a starting point from which can be extrapolated into the future, where this particular study picked two days in May of 2007 as the reference period.

The number of flights between any set of European airports, for each hour (T_t) during the reference day, serves as the basis for computing the hourly aircraft generation rate ($\gamma(d, a)$) for each departure (d) and arrival (a) pair appearing in the set of all airports in the data base. The extrapolation of these rates to future dates is performed by applying growth factors defined by Eurocontrol to the generation rates γ . On a per-country basis, growth factors are published for four scenarios resembling gradations of growth rates. For years not covered by Eurocontrol forecasts, growth rates are interpolated linearly. This specific study restricted the simulation to German airspace, where the extrapolated data base was used to simulate flights asynchronously.

Every trajectory was represented with a function (Equation (3.1) [from [47]]) where the output contains the state vector (s_t) at the specified input time.

$$f(t) = s_t \text{ for } \alpha \leq t \leq \omega \quad (3.1)$$

The state vector contains the position, altitude, and domain of a flight, varying from start (α) to the end (ω). The domain refers to a categorization of the flight region based on the current altitude, including the airport domain (APT), terminal maneuvering area (TMA), and the en-route (ENR) domain. When amalgamating all generated trajectories across each origin-destination pair ($\gamma(d, a)$), one obtains the set of state vectors for all flights, formulated in Equation (3.2) [from [47]].

$$\mathfrak{F}(t) = \bigcup_{s,d \in \mathcal{U}} \left\{ \begin{array}{l} \{f_{i,s,d}(t) \mid t \in \{T_0, \dots, T_{23}\} \\ 1 \leq i \leq \gamma_T(s, d)\} \end{array} \right. \quad (3.2)$$

For trajectory generation, key flight parameters are randomly sampled from a uniform distribution varying by 10% around the average in order to induce some variation. This includes the climb and descent angle, airspeed during different flight phases, and cruising flight level. The simulation approach taken is largely dependent on earlier works, including [57, 107]. The authors validate the simulations against radar data provided by the local ANSP. Figure 3.7 depicts the flight trajectories in German airspace, for the simulation and the radar-correlated flight plans.



Figure 3.7: Geographic distribution of flights across Germany, peak instantaneous aircraft count (PIAC) of the simulation (left) and radar-correlated flight plan data (right), from [47]

With a resolution of approximately $1 \text{ km} \times 1 \text{ km}$, the error ratio (defined as the ratio of peak instantaneous aircraft count (PIAC) of the proposed model and radar data) shows that trajectories are concentrated stronger than is actually observed at major airports (e.g. Frankfurt and Munich).

3.3. Markov Chain Processes

For the purpose of researching conflict probability in uncontrolled airspace, [59] reviews and applies a methodology to formulate aircraft trajectories as outcomes of a stochastic process: a Markov chain model. This section provides a description of the proposed stochastic model in Section 3.3.1 and an account of the downstream conflict probability estimation in Section 3.3.2. The following explanations are based on [59].

3.3.1. Stochastic Model of an Aircraft Trajectory

Rather than introducing randomness in the flight dynamics describing aircraft motion (as described in Section 3.1), uncertainty in aircraft trajectories can be modeled by incorporating a stochastic element into the trajectory itself. The positional error of an aircraft increases over time, but can be modeled with a Gaussian density function. With this in mind, a trajectory can be specified according to the stochastic differential equation formulated in Equation (3.3) [from [59]].

$$dX_t = v \cdot dt + \sigma_t dW_t \quad (3.3)$$

In this equation, the aircraft position is denoted by X_t . v is speed vector, σ_t a correlation matrix, and W_t represents pure Brownian motion (a stochastic process). The paper focuses on 2-dimensional aircraft trajectories, where the positional vector is comprised of the along-track and cross-track coordinates, (X_t^a, X_t^c) . In

the case of two aircraft, Equation (3.4) [from [59]] formulates two stochastic processes to define the trajectories, where r_a , p , and q are set fitting to empirically observed behavior. R_θ is a rotational matrix and θ is the angle with the abscissa axis. The induced stochasticity is representative of the deviation from the flight plan due to wind, tracking, navigation, and control variations [78]. This forms the basis for analyzing conflict probabilities, which is the subject of the next section.

$$\begin{cases} dX_t^1 = R_{\theta_1} \begin{pmatrix} v_1 \\ 0 \end{pmatrix} dt + R_{\theta_1} \begin{pmatrix} (r_a t + v_0)^2 & 0 \\ 0 & (p t^2 + q t + v_0)^2 \end{pmatrix} R_{-\theta_1} dW_t^1 \\ dX_t^2 = R_{\theta_2} \begin{pmatrix} v_2 \\ 0 \end{pmatrix} dt + R_{\theta_2} \begin{pmatrix} (r_a t + v_0)^2 & 0 \\ 0 & (p t^2 + q t + v_0)^2 \end{pmatrix} R_{-\theta_2} dW_t^2 \end{cases} \quad (3.4)$$

3.3.2. Conflict Probability Estimation

Ultimately, this paper focuses on accurately estimating conflict probability between two aircraft in an uncontrolled airspace. The distance metric between the trajectories is formulated in Equation (3.5) [from [59]].

$$D_t = \|X_t^1 - X_t^2\|, \quad t \in [0; T] \quad (3.5)$$

For two aircraft trajectories, such as those formulated in Equation (3.4), the trajectories and separation distance can be computed over time (Figure 3.8 [from [59]]). These are based on the flight plans of the aircraft.

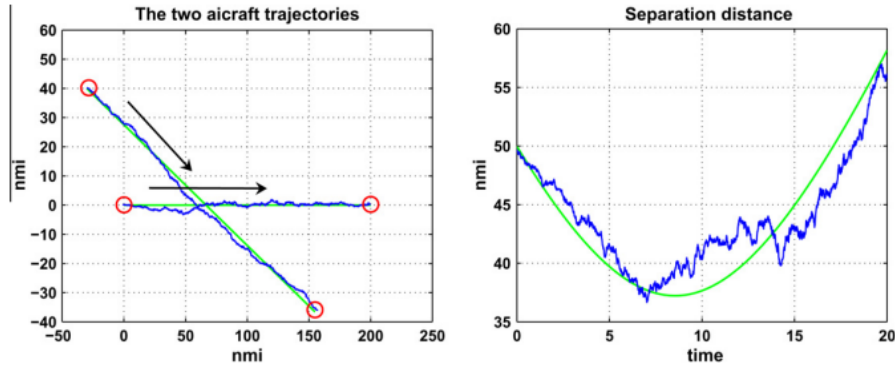


Figure 3.8: Simulated aircraft trajectories based on a flight plan (left) and the separation distance between them over time, from [59]

The underlying stochastic processes allow for the fast simulation of thousands of trajectories, leading to variations in the distance process. Given that a conflict is defined as a loss of separation (where the separation threshold is specified), its probability can be computed using the empirical density function (pdf) of the minimum distance of the aircraft traversing the trajectory depicted in Figure 3.9 [from [59]].

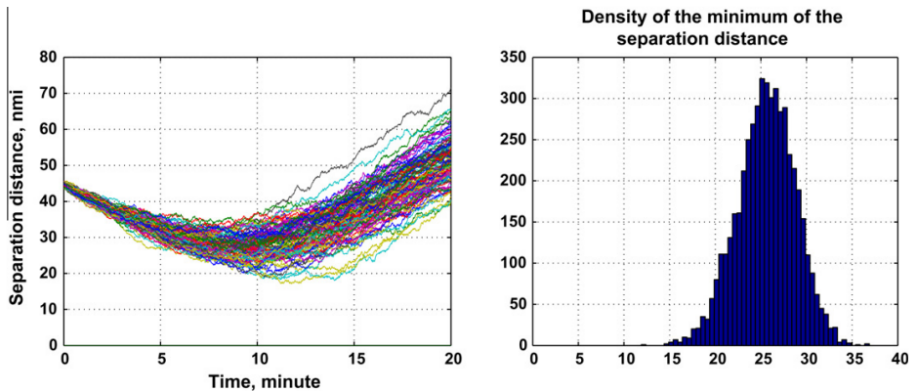


Figure 3.9: 5000 samples of separation distance processes (left) and the empirical pdf of the minimum distance over time, from [59]

The clear limitations remains that generated trajectories are restricted to the 2-dimensional case, consisting of straight segments and a constant altitude and speed vector.

3.4. Multivariate Density - Gaussian Mixture Models

A more structured approach to generating realistic trajectories consists of identifying patterns in existing data and estimating multivariate statistical distributions to sample from. In essence, new synthetic trajectories represent slight deviations from the reference pattern(s) [77]. This type of modeling has been applied to the analysis of climb profiles by using clustering techniques to identify characteristic profiles [104]. Alternatively, landing patterns have been clustered and modeled using Gaussian mixture models [105]. Most recently, [77] builds on this foundation to generate realistic go-around trajectories at Zurich Airport. This presented model (the focus of this section) is the product of a two step process: a dimensionality reduction (Section 3.4.1) to simplify the analysis of trajectories by selecting a subset of the most relevant points, and the modelling of the statistical distribution (Section 3.4.2) of flight flow using multivariate joint distributions.

3.4.1. Trajectory Dimensionality Reduction

In comparison to *standard* landings, go-arounds are a rare event. The authors compiled a dataset of 646 go-arounds at Zurich Airport, within a three year time frame. This base was further reduced by limiting to a single runway and removing irregular flight movements including mistaken turns and holding patterns. A normalization was also applied to ensure that each trajectory series starts when the aircraft initiates the go-around (at the end of the runway) and ends 1.5 NM after starting the final segment of the approach. Following this pre-processing, the authors are left with a dataset of 407 trajectories, where Figure 3.10 [from [77]] depicts a visualization of both the raw and pre-processed trajectories.

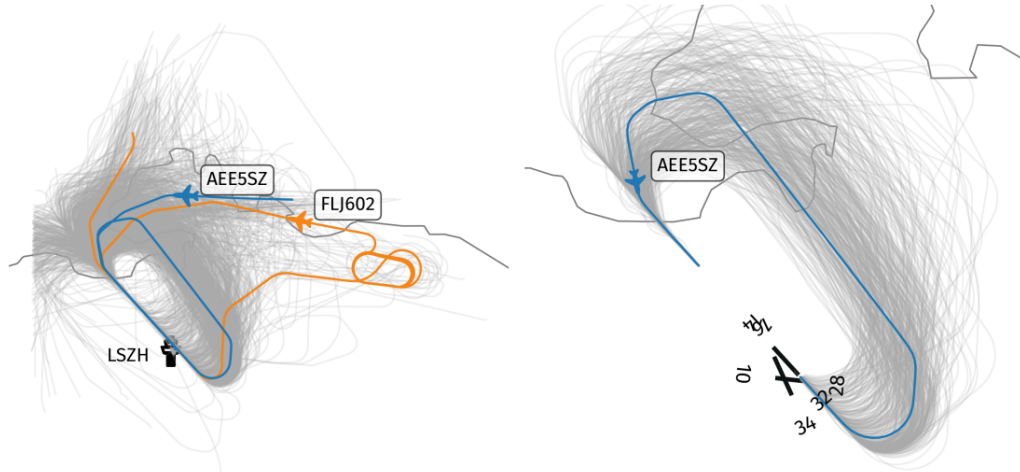


Figure 3.10: Raw (left) and pre-processed (right) go-around trajectories, from [77]

In service of the density estimation, a dimensionality reduction was applied to the trajectories. Previous efforts have relied on applied PCA [32] and FPCA [66] for projecting altitude and speed along trajectories to a lower dimension. In this spirit, the authors map the 2-dimensional aircraft paths to a single dimension, while ensuring that the i -th observation in each trajectory aligns with all others. The methodology relies on the selection of a representative reference trajectory, with the characteristic shape of a go-around for the runway at hand. This benchmark trajectory is sampled into a specified number of features. At each of these features, the orthogonal lines to the trajectory direction are calculated. All other go-around trajectories on the dataset can then be resampled based on the intersections with the established normal lines.

This method ensures that each trajectory is represented consistently (in a lower dimension), and that trajectories are aligned along the computed orthogonal lines of the chosen reference. Formally, the features of each 2-dimensional trajectory $[x_1, y_1, x_2, y_2, \dots, x_n, y_n] \in \mathbb{R}^{2 \times n}$ are now in $[z_1, z_2, \dots, z_n] \in \mathbb{R}^n$, where each feature corresponds to an intersection with a normal line. Figure 3.11 [from [77]] demonstrates the sampling of a trajectory (in red) based on the normal lines according to the reference trajectory (in blue). The joint probability distribution $[z_1, z_2, \dots, z_n]$ forms the basis of the next step, which is approached using multivariate Gaussian and Gaussian mixtures, and leveraging theory on statistical copulas. The marginal distributions along the sampling points of a trajectory are visualized in the right of the same figure.

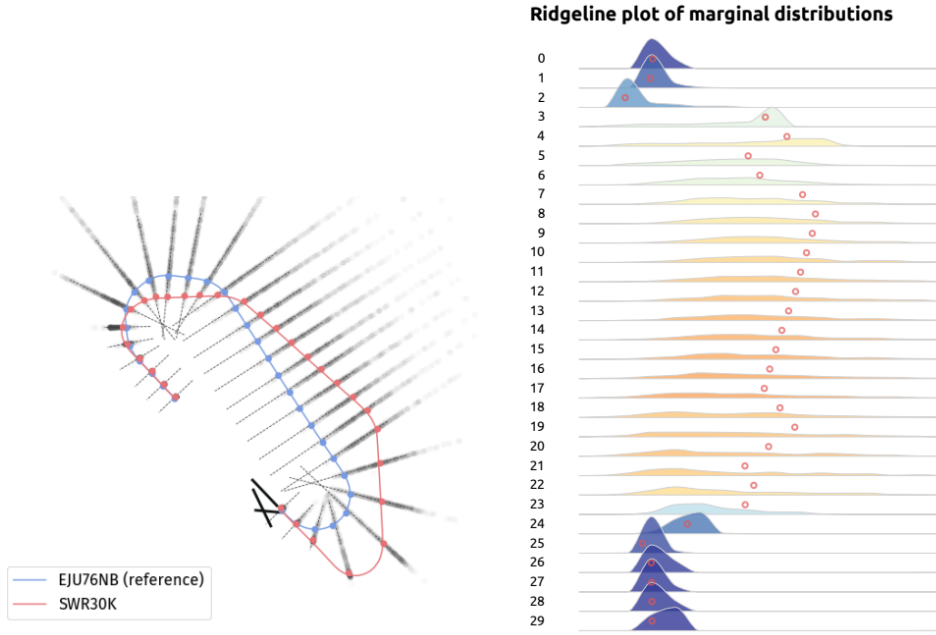


Figure 3.11: Sampled version of a trajectory (red) based on the normal lines of a reference trajectory (red), the right depicts the marginal distributions at the sample points (in the lower dimensional representation space, from [77])

3.4.2. Multivariate Statistical Distribution Fitting

The remaining problem is to estimate a d -variate distribution of the N independent realizations from it. The proposed dimensionality reduction means that $d = n$, where n is the number of sampled points (normal lines) along the reference trajectory. A multivariate normal distribution (MVN) is deemed inappropriate due to the assumption of a single representative normal distribution and foreseen difficulties of the covariance matrix in capturing all dependencies between the marginals. Instead, a Gaussian mixture model (GMM) is proposed, where the corresponding density definition is formulated in Equation (3.6) [from [77]]. The mean and variance of each multivariate normal density f_i is estimated through maximization of likelihood using the expectation-maximization (EM) algorithm. k is chosen for optimal accuracy (where an anti-over-fitting procedure has been applied).

$$\forall \mathbf{X} \in \mathbb{R}^d, g(\mathbf{X}) = \sum_{i=1}^k w_i f_i(\mathbf{X}) \quad (3.6)$$

Alternatively, the authors employ statistical copulas to simplify the estimation of the density function. As background, Sklar's Theorem [118] is invoked to demonstrate that the multivariate cumulative distribution function representing a trajectory can be disentangled in terms of marginal distributions that are *connected* using a copula function, formulated in Equation (3.7) [from [77]]. The two-dimensional extension (vine copula) is presented as an alternative formulation of the density estimation with many inherent dependencies, and is tested as a suitable model for generating realistic trajectories.

$$F(x_1, x_2, \dots, x_d) = C(F_1(x_1), F_2(x_2), \dots, F_d(x_d)) \quad (3.7)$$

Metrics used by the authors have been presented in Section 2.5.6, namely the curvature of variation, Mahalanobis distance, and e-distance. First and foremost, the dimensionality reduction proved to be beneficial, reducing the distance between the real and generated trajectories for all model variations (MVN, GMM, and Vine Copula). Furthermore, the synthetic data as a result of this dimensionality reduction was more realistic, as the turns observed without it proved too sharp to be flyable, visible in Figure 3.12 [from [77]]. Overall, the GMM with dimensionality reduction generates the most realistic trajectories and ensures dense coverage of the variations in pattern. The optimization process thereof relies on maximum likelihood.

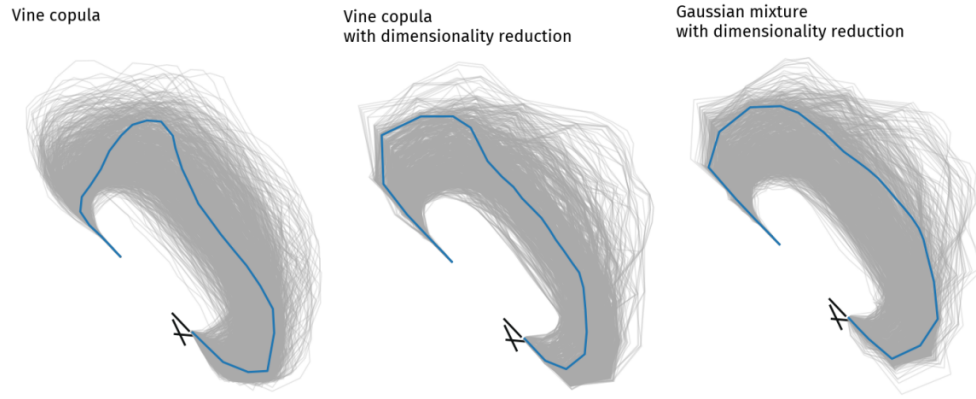


Figure 3.12: Synthetic trajectories generated by the Gaussian mixture model and the Vine copula (with and without dimensionality reduction), from [77]

3.5. Convolutional Generative Adversarial Network

In 2019, the first application of generative deep learning techniques to aircraft trajectory generation was performed. [64] documents the use of a GAN to generate trajectories for the purpose of detecting atypical aircraft approaches at Paris Orly airport during the final landing phase. Section 3.5.1 describes the used architecture and fidelity of the generated landing trajectories. Section 3.5.2 focuses on the downstream task of using the trained model for anomaly detection.

3.5.1. GAN for Trajectory Generation

The training data consists of 4401 landing trajectories on runway 26 at Paris Orly airport, all conducted by Airbus A320 aircraft. For each trajectory, data points on longitude, latitude, altitude, and ground speed are available for the last 25 nautical miles (NM) before touchdown on the runway. This restriction of a single aircraft type on a single runway considerably reduces the expected generalizability of the model. Although each trajectory was sampled every four seconds, the authors resampled the data to obtain 256 uniformly spread points along each trajectory to match the required $4 \times 256 \times 1$ input dimensionality of the network architecture. The discriminant is comprised of two convolutional layers (with max pooling, batch normalization, and dropout) before the final fully connected layer maps to the output. The generator network consists of similar batch normalization and dropout, but uses up-sampling convolutional transpose layers to map back to a 4-dimensional trajectory.

Using the minimax training process described in Section 2.3.4, the generator is able to generate synthetic trajectories that were not previously observed. Despite adhering to the generic shape of landing trajectories, the generated data required smoothing using a cubic spline interpolation to remove the high-frequency noise (unrealistic for aircraft). Despite this smoothing filter, the known limitation of *mode collapse* specific to GANs is evident as not all approach patterns are captured by the generator, depicted in Figure 3.13 [from [64]]. Two examples of this struggle are lack of capturing level flight before descent along the glide path and imprecise adherence to the extended runway centerline (following the localizer path) for the last few miles.

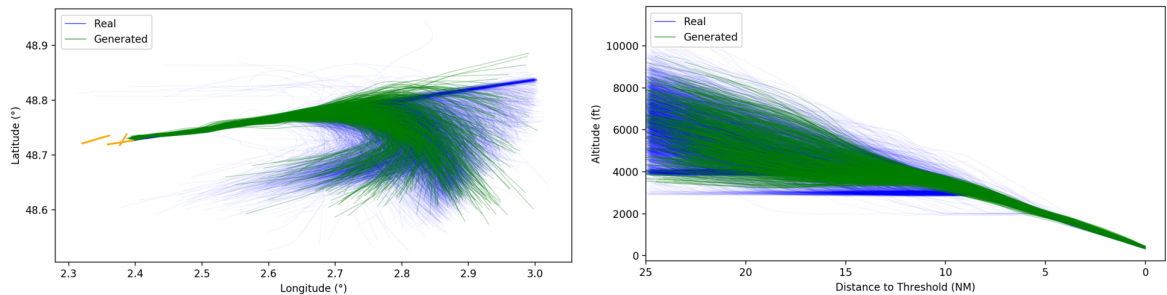


Figure 3.13: Real (blue) and generated (green) trajectories in terms of longitude/latitude (left) and altitude (right) profiles, from [64]

3.5.2. Abnormal Approach Detection

The landing anomaly detection problem is approached using two techniques, first using the trained discriminator and secondly using a separate encoder architecture. The discriminator is trained to differentiate between real and synthetic samples, implying it's ability to pinpoint trajectories belonging to the original dataset and those resemble atypical behaviors. Essentially, the anomaly detection is performed by scoring trajectories based on similarity to the original dataset. Figure 3.14 [from [64]] predicts the scoring of trajectories according to this methodology. Evidently, the discriminator is able to identify anomalies according to longitude/latitude and altitude profiles. The former deems approaches from the west and east more atypical, and holding patterns very unusual. In terms of altitude, the level-flight prior to the glideslope is seen as typical, whereas later slope inception is seen as an anomaly. Differentiation in terms of ground speed profiles proved inconclusive.

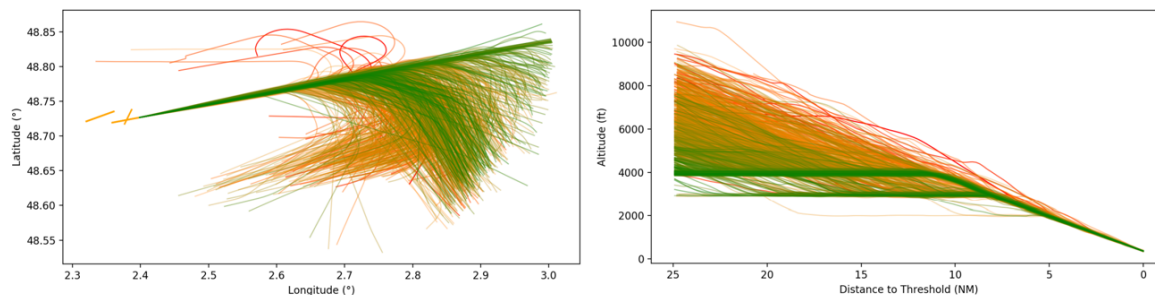


Figure 3.14: Discriminator scoring of trajectories where green indicate typical patterns, red atypical, and orange intermediate, from [64]

Alternatively, an encoder network is proposed to detect anomalies (tuned after the training of the original GAN). The encoder embeds a high dimensional trajectory to a lower dimensional latent space, as outlined in Section 2.3.3. The generator subsequently maps from the latent space back to the high dimensional 4-dimensional trajectory space. Between this rebuild of the original trajectory and the original itself, a reconstruction error is computed (chosen by the authors as the L2 norm). A high reconstruction error serves as a proxy for an atypical approach.

The authors relied on the use specific total energy trajectories to compare with previous efforts in anomaly detection using functional PCA [65], where the specific total energy is calculated as the sum of potential and specific kinetic energy per unit of mass (where the latter is assumed constant for the last 25 NM of a flight in this setting). Effective energy management is assumed as a proxy for safe approaches and deviation thereof being atypical. Figure 3.15 [from [64]] plots the distribution of reconstruction errors based on this proxy, where orange/red (on the right) indicate atypical profiles.

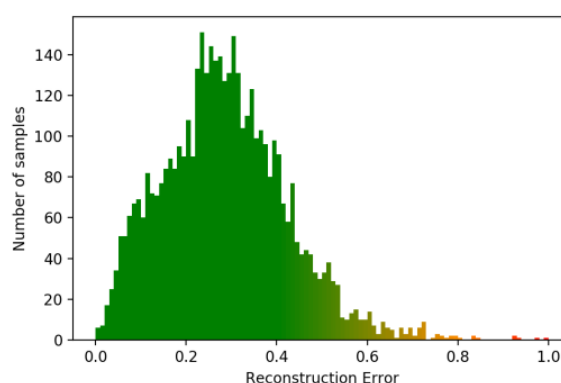


Figure 3.15: Normalized reconstruction error distribution based on the encoder-generator network, where green indicates small errors and red high errors, from [64]

This paper marks a premier for the use of GANs (and deep generative modelling in general for that matter) to aircraft trajectory generation and the specific atypical approach detection downstream task. However, the simplistic nature of the trajectories and the highly filtered dataset severely limit the use in other contexts, coupled with a latent space that holds negligible information on approach patterns [78].

3.6. Temporal Convolutional Variational Autoencoder

The most recent advancement in aircraft trajectory generation was the adaptation of the VAE (outlined in [Section 2.3.3](#)) to generate 4-dimensional paths in the TMA of Zurich Airport, named the Temporal Convolutional Variational Autoencoder (TCVAE) [78]. The foreseen downstream task is for aircraft conflict probability estimation using Monte Carlo simulation, similar to the stochastic process approach outlined in [Section 3.3](#). The specific adaptations are the integration of a temporal convolutional network (TCN) to capture sequential dependencies, and using a the *VampPrior* prior distribution consisting of a Variational Mixture of Posteriors. Following a short background in [Section 3.6.1](#), this section describes the proposed model ([78]) in terms of its differences with the *vanilla* VAE structure (temporal convolution in [Section 3.6.2](#) and VampPrior in [Section 3.6.3](#)) and summarizes the analysis of the learned latent space and results in [Section 3.6.4](#).

3.6.1. Background & Data

The authors confine themselves to data on 14,000 landing aircraft on Zurich Airport runway 14 over a two month time span, citing that landings are more complex and diverse (and include holding patterns that are not observable in other flight phases). Furthermore, it is claimed that the breadth of trajectories collected at Zurich Airport are representative of those found at major airports around the world, and that results of the model can safely be extrapolated to other locations and *simpler* flight phases (landing and en-route). Trajectories are clipped to ensure the starting point is 40 NM from the airport and the ending point 1.5 NM after the final approach point of the specific runway. Each trajectory is linearly interpolated to 200 points to ensure consistency, and that the exploding gradient problem does not take over during training (due to huge variations in points). Interestingly, mirroring the work in [77] ([Section 3.4](#)), go-arounds were excluded from the data due to separate complexity. Lastly, longitude and latitude were replaced by track (which together with ground speed can derive the former) to avoid the need to learn complex relations in coordinates.

3.6.2. Temporal Convolution for Temporal Dependencies

To capture the temporal dependencies, use is made of a temporal convolutional network (TCN) rather than a recurrent network, for the simplicity and allowance for parallel computation. TCNs employ causal convolutional layers with adapted zero-padding, ensuring that output at time t only depends on inputs from time t and earlier. Dilatation in TCNs allows for long effective history coverage when multiple of such layers are used, depicted in [Figure 3.16](#) [from [78]]. Previous efforts have also combined residual elements into the architecture to control problems with the gradient [7].

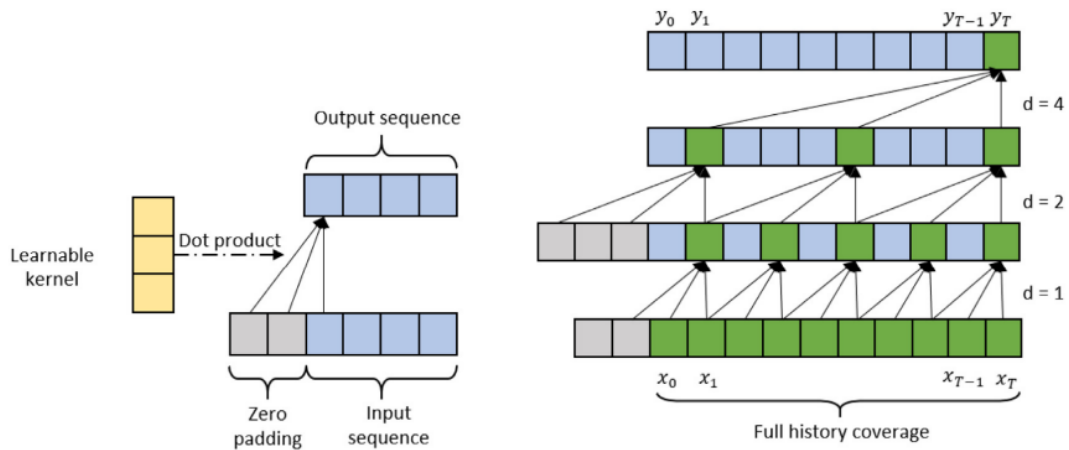


Figure 3.16: Causal convolution (left) and multiple causal convolution layers with dilatation (right), from [78]

Usage of a TCN in a VAE enhances the representation of trajectories in the latent space to reduce information loss as a result of the dimensionality reduction, leading to better reconstruction. The selection of an appropriate prior distribution becomes more critical because the posterior distribution is much more complex due to the included temporal correlations. To address this, the authors use a Variational Mixture of Posteriors.

3.6.3. VampPrior - Variational Mixture of Posteriors

Traditionally, VAEs use a standard multivariate Gaussian as the prior distribution. This, for the purpose of ensuring that the latent space can be readily sampled from and that it is continuous in space (small distances between points in the latent space should decode to similar output). In general this allows VAEs to estimate high dimensional distributions with greater effect than other statistical methods. However, the addition of the TCN element leads to a more complex latent distribution, and this can impede the quality of generation when sampling. To address this, multi-modal priors can help maintain a diverse latent space while adhering to the prior. [123] first proposed a Variational Mixture of Posteriors (VampPrior), using a Gaussian mixture prior derived from posterior distributions conditioned on learnable pseudo-inputs (x_{pseudo}). When K is the number of pseudo-inputs for the encoder, the posterior in the latent space is specified by $q_\phi(z | x_i^{pseudo})$, where each constitutes a component for the prior. The full prior distribution is formulated in Equation (3.8) [from [78]].

$$p(z) = \frac{1}{K} \sum_{i=1}^K q_\phi(z | x_i^{pseudo}) = \frac{1}{K} \sum_{i=1}^K f(z | \mu_{VP,i}, \sigma_{VP,i}^2) \quad (3.8)$$

Each Gaussian component of the mixture has a mean $\mu_{VP,i}$ and a standard deviation $\sigma_{VP,i}^2$ parameter, rather than a single multivariate normal. Instead, the approach optimizes the prior during VAE training through a pseudo-inputs generator that ensures the encoder minimizes the ELBO loss term by producing the best component parameters leading to the most effective coverage and generation capabilities in the latent space.

The VampPrior acts as a conditioning mechanism for the generation process, as the pseudo-inputs represent trajectories aligned with the average shape of trajectories in that proxy of the latent space. The final entire architecture proposed by the authors is shown in Figure 3.17 [from [78]]. Hyperparameters have been tuned for Zurich Airport specifically. Training time can also be reduced when considering that simpler trajectories require a lower-dimensional latent space. Generated data is realized through sampling in the latent space, after a specific region (and corresponding pseudo-inputs) has been selected.

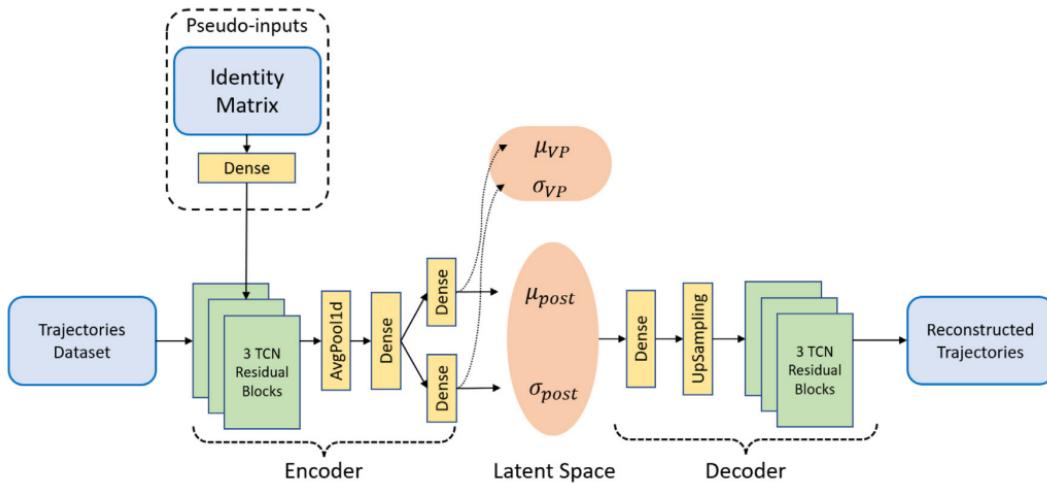


Figure 3.17: The full VampPrior TCVAE architecture, from [78]

3.6.4. Latent Space Analysis & Trajectory Generation

To visually analyze the learned latent space, PCA analysis is used to plot the Gaussian clustering, the results of which are depicted in Figure 3.18 [from [78]]. The authors compare the TCVAE architecture with a FCVAE and the difference is significant. The TCVAE is able to accurately reconstruct trajectories of all profiles. FCVAE proved to allocate trajectories in the latent space primarily based on the direction of arrival, whereas the TCVAE was able to differentiate lower-level variations, even within a single direction of arrival. The green and purple colored directions (in Figure 3.18) both represent approaches from the east, but the former represents complex trajectories and the latter direct routes. The simple fact that the TCVAE contains more clusters in the latent space is indicative of the explanatory power.

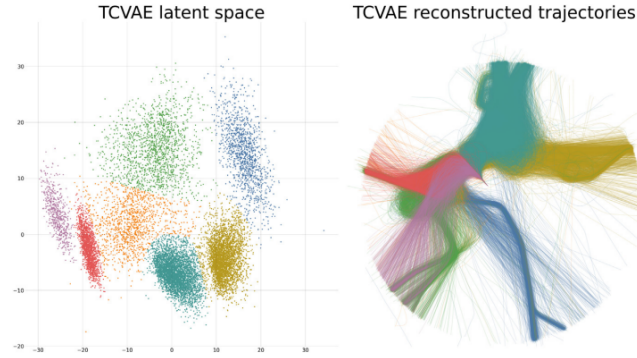


Figure 3.18: Visualization of the Gaussian clusters in the latent space after the PCA projection, from [78]

The VampPrior TCVAE allows for the clustering of the latent space in groups of trajectories around characteristic shapes. The latent space exhibits variations in densities, where high density is indicative of more common trajectories (lower variance) and low density areas of more complex and diverse approach patterns. The latter is at risk of rare lacking realism in the trajectory in the sense that the trajectories are very unlikely from an operational point of view. This may still be of interest when considering research into novel airspace structure designs. Examples of the former are depicted in Figure 3.19 [from [78]]. These denser areas of the latent space represent the vast majority of trajectories, and the decoded points lead to realistic/flyable trajectories. The authors employ the flyability assessment described in Section 2.5.5 to assess trajectories.

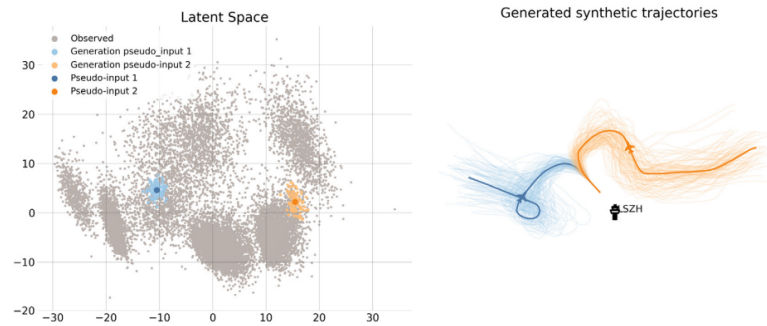


Figure 3.19: Synthetic trajectory generation using pseudo-input components representing *dense* regions in the latent space, from [78]

To demonstrate the high variance of the VampPrior in sparser regions of the latent space, the authors consider the generation of synthetic trajectories in this regions, visualized in Figure 3.20 [from [78]]. Evidently, the trajectories feature many loops and turns varying in magnitude, and self-intersect much more than is observed in the original dataset. Despite being mathematically sound in the optimization of the loss, the trajectories are not flyable and operationally relevant. Nonetheless, the pseudo-inputs allow for control of the generation of trajectories to avoid this if desired.

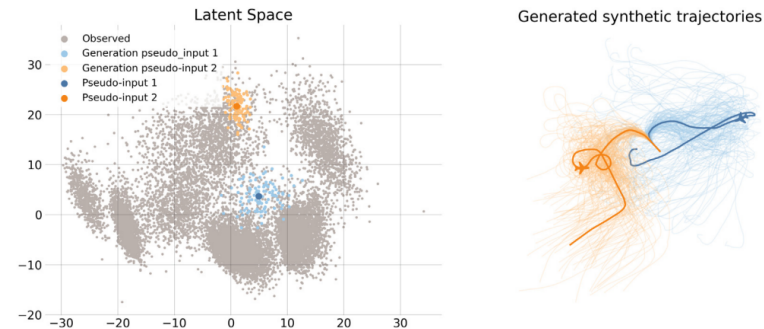


Figure 3.20: Synthetic trajectory generation using pseudo-input components representing *sparse* regions in the latent space, from [78]

4

Synthetic Multivariate Time Series Generation with Deep Generative Models

The playing field of generative AI-based architectures and techniques grows by the day. An exhaustive search over all models amounts to an near insurmountable task with questionable merit. Instead, each of the following sections is dedicated to one of the base architectures discussed in [Section 2.3](#). Therein, a selection of models covers the variation in techniques and trend-breaking discontinuities in the past years. Prior to any further search, exploratory incentives dictate that either GANs or diffusion models are preferred for further continuation in the thesis project, given that the closet competitor (TCVAE [\[78\]](#)) involves a VAE. In similar spirit, the TCN should preferably be offset with a recurrent or self-attention component. The chapter ends with a summary of model architectural landscape, and provides a preliminary selection based on five criteria:

1. **High-Fidelity Data Generation:** The model must produce synthetic data that closely mirrors real-world trajectories, ensuring realism and accuracy.
2. **Temporal Consistency:** The model should accurately capture the dynamic nature of flight paths over time, maintaining the temporal relationships inherent in the data.
3. **Adaptability:** The model should be adaptable, allowing for the incorporation of specific flight conditions and operational parameters to generate diverse scenarios.
4. **Scalability:** The model should be able to handle varying data volumes and complexities, ensuring they can be applied to different datasets and operational contexts.
5. **Robustness:** The model should effectively manage anomalies and produce reliable outputs under diverse scenarios, ensuring reliability and stability in their performance.

4.1. Variational Autoencoders

The most recent advancement in aircraft trajectory generation presented in [\[78\]](#) uses a VAE as the underlying generative model. The authors' choice of using a TCN and a Variational Mixture of Priors (VampPrior) proved effective, but alternatives exist. This section outlines the landscape of using VAEs for the purpose of generating time series. [Section 4.1.1](#) begins with the natural step of combining recurrent elements with the VAE architecture. The recent success of attention mechanisms has also been proposed in tandem with a VAE, discussed in [Section 4.1.2](#). Alternatively, vector quantization has been used to address time series generation combined with a bidirectional prior model, explained in [Section 4.1.3](#). Lastly, [Section 4.1.4](#) proves the merit of integrating domain specific temporal constructs in the architecture.

4.1.1. Recurrent Variational Autoencoder

The implementation of RNNs as the encoder and decoder of a VAE was notably introduced in 2014, with the Variational Recurrent Autoencoder (VRAE) [\[39\]](#). The model mapped sequential music data to a latent space

representation after resampling each sequence into 50 time steps. The model was kept very simple and relied on a vanilla RNN to capture the temporal dependencies. The encoder network consists of a single recurrent (hidden) layer, where the last state of the RNN determines the distribution over the latent space. Following the reparameterization trick, the representation is mapped back to the higher dimension using a decoding RNN with a single weight matrix. The two-dimensional latent space revealed limited clustering and the actual 20-dimensional learned space allowed for music generation lasting less than a minute. In later years, testing of this model on alternative benchmark datasets proved ineffective and led to inconsistent findings [33].

The clear limitation of this approach is the limit placed on capturing long-term dependencies in the data, inherent of standard RNNs. TrajVAE addresses this by utilizing an LSTM and applying it specifically to trajectory generation for self-driving cars [19]. The encoder consists of a spatial embedding module, LSTM module, and feature extraction module. The spatial dimension captured by the embedding is pre-trained separately. The feature extraction module consists of MLPs that use the LSTM module output to map to the standard mean and standard deviation vector in the latent space. The decoder also employs an LSTM network.

More recently, causal mechanisms have been incorporated into the VAE framework [82]. Leveraging previous efforts focused on discovering causal relations between higher dimensional time series data, the authors improve interpretability and explainability in the medical domain. The proposed Causal Recurrent Variational Autoencoder (CR-VAE) uses RNNs, but learns a Granger causal graph and merges this knowledge when generating. A multi-head decoder generates the individual dimensions of the multivariate data separately, and is regularized by the causal relations.

4.1.2. Attention-Based Variational Recurrent Autoencoder

An alternative to capturing the temporal dependencies in multivariate data is the use of an attention mechanism, which has been combined with a variational autoencoder in [67]. The proposed Attention-Based Variational Recurrent Autoencoder (AVRAE) is used to generate industrial control systems data. The encoder and decoder are still RNN-based and learn entire input sequences to ultimately output sequences of various lengths. The RNN operation is adjusted to follow a stochastic process, by sampling the hidden state from a probability distribution. By introducing attention layers, AVRAE can direct the decoder's focus on key parts of the sequence, optimizing the ability to learn and generate better predictions. AVRAE uses two distinct attention layers: cross-attention and self-attention. Cross-attention finds which areas of the input sequence the decoder should pay particular attention to when generating the output, using hidden states from the encoder and decoder to compute attention weights. A look-ahead mask prevents the decoder from considering future inputs. This masked cross-attention layer computation is formulated in Equation (4.1) [from [67]], where H_{enc} and H_{dec} are the hidden states over time, and the right-hand side implements the look-ahead mask.

$$W_c = \text{softmax} \left(\frac{H_{dec} H_{enc}^T}{\sqrt{d}} + (1 - M) \times -\infty \right) \quad (4.1)$$

The output of the cross-attention layer is used as input for the next self-attention layer. Lastly, the output is reshaped to the identical shape as the encoder's input using an MLP. The model makes use of a particular algorithm for generating new data samples. The process begins with taking a real data point (seed input) from a dataset. The encoder processes the seed to create a context that includes the mean and standard deviation for the hidden state and the context vector. These are sampled from a Gaussian distribution, creating variation in the data. The decoder then uses these sampled hidden state and context to generate synthetic data that follows the pattern of the original input but with some randomness. This method can be repeated multiple times to generate as much synthetic data as needed [67].

4.1.3. Vector Quantization and Bidirectional Priors

The VQ-VAE, or Vector Quantized Variational Autoencoder, is another type of variational autoencoder, one that uses vector quantization and leverages a learned prior [127]. Contrary to the standard continuous space, the latent space is discrete. The encoder maps input data to a latent space consisting of a set of discrete codebook vectors. The quantization refers to the comparison of each input data point to the closest codebook vector: quantization of the data. The decoder then reconstructs the input data from the quantized vectors and uses a loss function combining reconstruction loss and codebook/commitment loss to guide learning and ensure efficient codebook usage. Secondly, the priors of the latent space are not static. The posterior collapse problem is avoided through the learning process and the granted flexibility to the prior.

Having found success in image generation, [80] leverage the VQ-VAE model and combine it with a bidirectional transformer model (MaskGIT) for learning the prior. The combination of the two priors allows the capturing of more nuanced patterns and provides more flexibility in modeling the latent space, resulting in improved generalization. Two separate models are leveraged in the split of high and low frequency domains to retain as much of the characteristics in the input data as possible. Lastly, the authors demonstrate that sampling can be conducted both unconditionally and class-conditionally by including a class token in the input, giving greater control of the generation process.

4.1.4. Integration of Domain-Specific Temporal Constructs

The integration of specific temporal structures when generating new data can be seen as a different route. TimeVAE [29] was proposed in 2021 and explicitly takes into account trends and seasonality to model temporal dynamics, arguing that this boosts interpretability and explainability. The usual VAE serves as the foundation for the generative power and it is optimized using the ELBO loss function. The *Base* Time-VAE relies on convolutional and dense layers to map the multivariate time series input into the standard normal latent space. Sequences of variable length are automatically padded with zero to ensure consistency. Using the reparameterization trick, samples from this space are decoded back through fully-connected and transposed convolutional layers to end in identical shape as the input. This base model does not yet include the specialized temporal constructs.

The *Interpretable* Time-VAE injects level, trend, and seasonal components in the model, depicted in Figure 4.1. The encoder is identical to that in the base model, but the decoder relies on parallel usage of the temporal structures to compute the outcome. The original base decoder can serve as a residual block, but just like any of the trend and seasonality components it can be excluded.

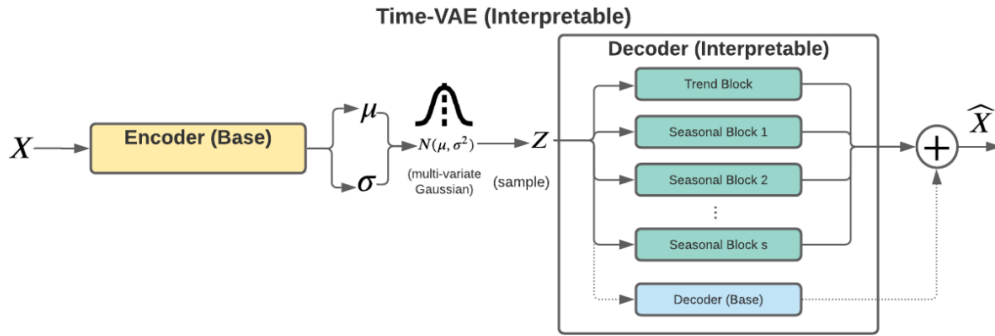


Figure 4.1: Architecture of the Interpretable TimeVAE model, from [78]

The trend block uses a polynomial function, where the degree is a hyperparameter. The latent space vector from the encoder is used to compute an expansion coefficient that is separate for both samples and feature dimensions, ensuring interpretability. The trend is reconstructed by multiplying the expansion coefficient with a time vector matrix. A polynomial function with zero degree means there is no trend. Modeling of the seasons is achieved using a seasonality block, where it defines different seasonality patterns based on two parameters: the number of seasons and the duration of each season. The latent space vector allows for the computation of matrix of basis expansion coefficients, allowing the indexing of specific seasons at individual time steps. Ultimately, these seasonal patterns are summed element-wise to arrive at a single location in the seasonal cycle.

4.2. Generative Adversarial Networks

GANs have become a cornerstone in the generative AI domain, and success in image generation has started to transfer to time series data. Section 4.2.1 discusses the integration of recurrent elements into the vanilla GAN architecture, for both continuous and discrete data, as well as conditioning mechanisms. Section 4.2.2 is a departure from this, where the presented model argues that adversarial training alone cannot learn temporal dynamics. Lastly, Section 4.2.3 presents the application of transformer architectures to GANs, again for both unconditional and conditional models.

4.2.1. Recurrent Generative Adversarial Networks

Analogous to VAEs, the first step in applying GANs to sequential data was the direct integration of RNNs in the GAN architecture. The Continuous RNN-GAN (C-RNN-GAN) [95] relied on the use of LSTM networks for both the generator and discriminator of the GAN when generating music, where a 4-dimensional signal consists of tone length, frequency, intensity, and time since the previous tone. The integrated RNNs were two stacked LSTM layers, and the discriminator was bidirectional to include sequential context from either direction. The generation process is recurrent because the input to each cell in the generator is concatenated with the output of the previous cell. The input itself is the standard sequence of uniform random vectors of multivariate dimensionality. Adversarial loss training combined with backpropagation through time (BPTT) was used to reduce the loss.

Shortly after C-RNN-GAN was proposed, variants emerged including the Recurrent GAN (RGAN) [33] and the Recurrent Conditional GAN (RCGAN) [33]. Similarly to C-RNN-GAN, LSTM RNNs are utilized in both the generator and discriminator to capture temporal dependencies. The RCGAN removes the dependence on previous outputs, but introduces a conditioning mechanism on additional input. The discriminator aims to minimize the average cross-entropy CE on a per time step basis, between the prediction and the true label of the sequence, formulated in Equation (4.2) [from [33]].

$$D_{\text{loss}}(X_n, \mathbf{y}_n) = -CE(\text{RNN}_D(X_n), \mathbf{y}_n) \quad (4.2)$$

The generator has the objective of minimizing the average *negative* cross-entropy between the true labels and the predictions on generated sequences, made by the discriminator (Equation (4.3) [from [33]]). Noise vectors are sampled independently in this case.

$$G_{\text{loss}}(Z_n) = D_{\text{loss}}(\text{RNN}_G(Z_n), \mathbf{1}) = -CE(\text{RNN}_D(\text{RNN}_G(Z_n)), \mathbf{1}) \quad (4.3)$$

The authors experimented with non-independent sampling of the noise vector z but this proved to lack clear benefits for performance. The auxiliary inputs representing the conditional information c_n for a specific sample are concatenated to the corresponding noise vector at each time step, seen in Equation (4.11) [from [33]]. This is done to avoid discounting of this information through the forget gate of the LSTM.

$$\mathbf{z}_{nt} \rightarrow [\mathbf{z}_{nt}; \mathbf{c}_n] \quad \mathbf{x}_{nt} \rightarrow [\mathbf{x}_{nt}; \mathbf{c}_n] \quad (4.4)$$

Alternatively, Conditional Sig-Wasserstein GAN (SigCWGAN) [86] addresses the challenge of handling long time series data streams in generative modeling by introducing the Signature Wasserstein-1 (Sig-W1) metric as a discriminator, which captures temporal dependency and avoids costly computations. In the discrete domain, SeqGAN [139] presents a sequential data generation framework to handle the challenges of generating discrete data, outperforming previous generative modeling methods. The generator consists of RNNs with LSTM cells, while the discriminative model is a CNN. SeqGAN uses a policy gradient and Monte Carlo search to train the generator. Ultimately, discrete and continuous data can be converted through interpolation and sampling techniques.

4.2.2. Supervised Training and Embedding Network

The TimeGAN model [138] is a framework for generating synthetic time series data with both static and temporal features. The architecture is specifically tailored to time series data. It consists of four main network components: an embedding function, recovery function, sequence generator, and sequence discriminator (shown in Figure 4.2). The embedding function transforms static and temporal features into latent spaces, providing lower-dimensional representations of the data for efficient learning. The recovery function performs the reverse operation, converting the latent space back to the original features to ensure the integrity of the learned representations. The sequence generator produces synthetic data in the latent space by taking static and temporal random vectors as input and outputting synthetic latent codes. Meanwhile, the sequence discriminator classifies data as either real or synthetic based on the static and temporal codes, and is implemented as a bidirectional recurrent network with a feedforward output layer.

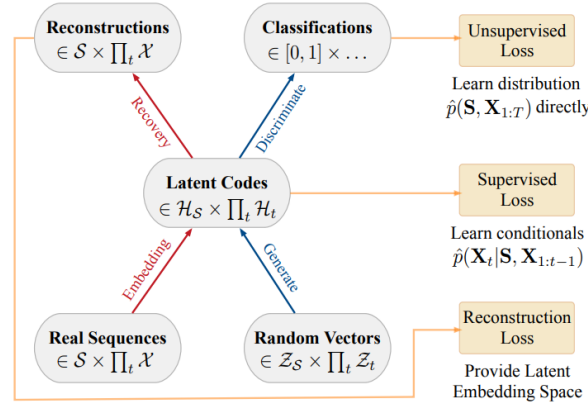


Figure 4.2: Block diagram of TimeGAN model components, from [138]

The training process of TimeGAN constitutes a departure from sole reliance on adversarial loss, instead it combines reconstruction loss, supervised loss, and unsupervised loss. The reconstruction loss measures how well the model can reconstruct the original data from the latent (simplified) representations while supervised loss guides the generator to produce realistic step wise transitions similar to the actual data. Unsupervised loss is the standard GAN loss that pits the generator and discriminator in an adversarial training loop, where the generator strives to produce data indistinguishable from real data and the discriminator aims to classify real and synthetic data accurately. TimeGAN's training approach effectively integrates encoding features, generating latent representations, and iterating over time to create realistic synthetic time series data, maintaining the dynamics and relationships inherent in the original data.

4.2.3. Transformer-Based Generative Adversarial Networks

In 2021, TransGAN [68] was proposed as the first GAN relying entirely on Transformers for image generation. Direct adoption of the dedicated Vision Transformer (ViT) [31] was made for the discriminator network. A year later, the first transformer-based GANs specifically designed for multivariate time series were proposed, notably the Transformer-Based Time Series GAN (TTS-GAN) [84]. The generator requires a one-dimensional vector of N uniformly distributed random numbers as input (between zero and one), where N is a hyperparameter signaling the latent dimension of the synthetic time series. Immediately thereafter, it is mapped to an M -dimensional space where the sequence length is identical to the real data. It then divides the sequence into patches, applies positional encoding, and processes it through transformer encoder blocks. These blocks feature a multi-head self-attention module and a feed-forward multilayer perceptron (MLP) with GELU activation, with normalization and dropout layers for regularization. After the transformer encoder blocks, a Conv2D layer adjusts the dimensions to match the real data, transforming random noise into a sequence resembling real signals.

The discriminator, similar to a ViT model, acts as a binary classifier, distinguishing between real and synthetic signals by treating the input sequence like an image, with the time steps as the image's width and the number of signal channels as its height. The processing of multivariate time series data takes inspiration from images. A given multivariate sequence is treated as an image of height, where the height is one, the width is the number of time steps, and the number of channels is the feature dimension of the series. This ensures that positional encoding can be added with the sole requirement of dividing the width dimension into patches, depicted in Figure 4.3.

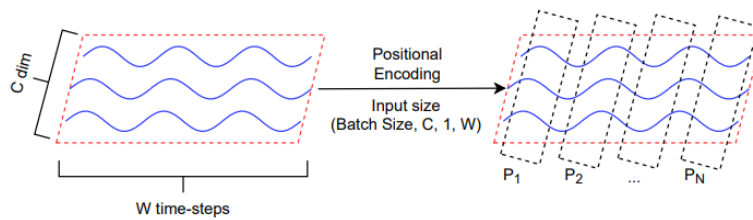


Figure 4.3: Time series processing in TTS-CGAN, from [85]

The authors of the TTS-GAN admitted to the clear downside of having to train the model on separate data classes in order to exercise control on the generation process. This hinders the benefit of transfer learning effects between classes, and undercuts the model's ability to learn low-level features applicable to all classes, while tweaking high-level features in the deepest layers of the network. Given the established frameworks of conditional GANs (CGANs), the same authors convert to a conditional version of the model: TTS-CGAN [85]. The conditional generator is trained to generate synthetic time series based on both a random vector z and a target class label c . The conditional discriminator now predicts not only whether a sequence is real or synthetic (D_{adv}), but also what class the signal belongs to (D_{cls}).

Multiple label embedding strategies were evaluated, where use was made of the MNIST dataset for interoperability reasons. Four strategies were evaluated, including concatenation and addition of label embedding to generator and discriminator inputs in linear and convolutional GANs. The most effective method involved concatenating label embedding only to the generator. For the discriminator, an additional classification head was added, resulting in the highest quality synthetic digit images.

During training of the model, the objective functions for the discriminator and generator networks consist of two components, an adversarial loss and a classification loss (Equation (4.5) [from [85]]). Similarly to TimeGAN, a supervised loss is deemed necessary to capture temporal relations missed through mere reliance on adversarial training.

$$\begin{aligned} L_D &= -L_{adv} + \lambda L_{cls}^r, \\ L_G &= L_{adv} + \lambda L_{cls}^f, \end{aligned} \quad (4.5)$$

The adversarial loss resembles the minimax game played between the generator and discriminator, but rather than the standard loss, the authors use the Wasserstein loss, formulated in Equation (4.6) [from [85]]. This loss contains a gradient penalty and increases the stability of the training procedure of a GAN, which is known to be highly unstable.

$$\begin{aligned} L_{adv} &= E_x [D_{adv}(x)] - E_{z,c} [D_{adv}(G(z, c))] \\ &\quad - \lambda_{gp} E_{\hat{x}} [\|\nabla_{\hat{x}} D_{adv}(\hat{x})\|_2 - 1]^2 \end{aligned} \quad (4.6)$$

For the categorical loss, the goal is to minimize the missclassification of the data with the target class. This is achieved by adding a classification head on the discriminator D and imposing categorical classification loss during the optimization process of both the generator and discriminator formulated in Equation (4.7) [from [85]]. This ensures the discriminator learns to classify real series to the true class, and helps the generator generate series that can be classified correctly.

$$\begin{aligned} L_{cls}^r &= E_{x,c'} [-\log D_{cls}(c' | x)], \\ L_{cls}^f &= E_{z,c} [-\log D_{cls}(c | G(z, c))] \end{aligned} \quad (4.7)$$

The results are promising when compared to competitor models C-RNN-GAN, RCGAN, TimeGAN, and SigCW-GAN, however, the authors note discrepancies in the quality of synthetic data for stationary vs non-stationary and single vs. multidimensional data. Ultimately, the quality of data seems to be an function of the amount of training data available.

4.3. Diffusion Models

Following the successful application of diffusion models to time series imputation and forecasting, recent works have begun leveraging diffusion for time series synthesis. Unsurprisingly, initial attempts relied on the integration of recurrent elements as the means to model the temporal dynamics. These models are described in Section 4.3.1 and are primarily focused on the univariate case or constrained in terms of sequence length. Reliance on the diffusion process has been argued to lead to losses in temporal properties, which has been addressed through constraining the generation as described in Section 4.3.2. Most recently, the constraining through seasonal-trend decomposition has been coupled with denoising diffusion models with a transformer architecture, described in Section 4.3.3.

4.3.1. Unconditional Univariate and Short Time Series Generation

First efforts in the application of diffusion methods to time series were usually largely focused on univariate data and tailored to specific data types, where DiffWave [76] (for audio synthesis) and ChiroDiff [24] (for chirographic data) are recent examples. Regular time series generation using SGM (TSGM [87]) is regarded as the first proposed generative model based on diffusion [88]. The model is comprised of an encoder, decoder, and conditional score-matching network, where the former two are RNNs and therefore defined recursively. The encoder and decoder are trained first, learning to embed the input time series into a latent space. This pre-trained model is subsequently used to train the conditional score network, which is ultimately used to sample synthetic data. To take the temporal dependencies into account, the score network learns the conditional log likelihood on previous time steps in a sequence. This is achieved by implementing the U-net architecture [108], and limiting the convolutional layers to a single dimension.

The training of the model involves two objective functions. The reconstruction loss (pre-training objective) is measured between the original series $\mathbf{x}_{1:T}$ and the reconstructed version $\hat{\mathbf{x}}_{1:T}$ (following the encoder and decoder processes), formulated in Equation (4.8) [from [87]].

$$\mathcal{L}_{ED} = \mathbb{E}_{\mathbf{x}_{1:T}^0} \left[\left\| \hat{\mathbf{x}}_{1:T}^0 - \mathbf{x}_{1:T}^0 \right\|_2^2 \right] \quad (4.8)$$

The hidden states learned during the pre-training is input for the score-matching network. The objective function governing learning of the gradient of the conditional log-likelihood function is formulated in Equation (4.9) [from [87]].

$$\begin{aligned} \mathcal{L}_{\text{Score}} &= \mathbb{E}_{\mathbf{h}_{1:T}, \mathbf{k}} \sum_{t=1}^T [\mathcal{L}(t, k)] \\ \mathcal{L}(t, k) &= \mathbb{E}_{\mathbf{h}_t^k} \left[\delta(k) \left\| s_{\theta}(\mathbf{h}_t^k, \mathbf{h}_{t-1}, k) - \nabla_{\mathbf{h}_t} \log q_{0k}(\mathbf{h}_t | \mathbf{h}_t^0) \right\|^2 \right]. \end{aligned} \quad (4.9)$$

Ultimately, new samples are generated through sampling hidden states and decoding them. The proposed TSGM method is able to generate high quality and diverse samples in comparison to GANs, but suffers from a much higher computational cost.

4.3.2. Constrained Time Series Generation

Aside from the unconstrained generation of time series, the authors of [20] formulate four types of constraints that can be enforced (visualized in Figure 4.5): trend, fixed value, global minimum, and multivariate. The goal now becomes to generate data of which the distribution approximates the true distribution, and simultaneously meets the constraints specified by the user. Hard constraints are enforced using inequality and equality constraints, where a generated series is rejected if the constraint is not met. Soft constraints are optimized for, thereby not leading to any sample rejection. Either constraint type can be enforced on a trajectory level or dataset level. Trends constraints are handled through a minimization of the L2 distance between the points of a synthetic sample and the trend-line series. Fixed points demand the series to take specified values at a given point. Soft and hard constraints can be enforced on both local and global sets.

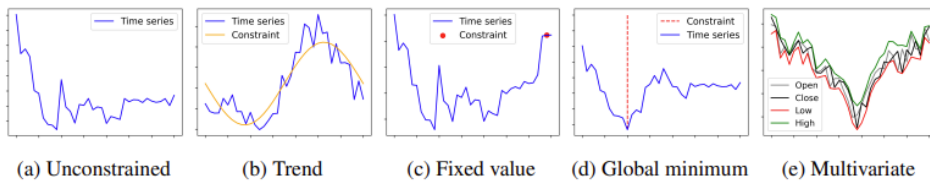


Figure 4.4: Types of constraints imposed of time series, from [20]

The constraints are incorporated into the generation process according to four different methods. Initially, the problem is formulated as a constrained optimization problem (COP), where the individual time stamps of a series represent decision variables in the optimization (solved using the computationally expensive Sequential Least Squares Programming (SLSQP)). The clear limitation of this method is the requirement for

explicit formulation of all data properties and constraints, which is addressed by using a conditional denoising diffusion model (DiffTime [20]), allowing for the incorporation of trend and fixed points constraints. A trend s is enforced by conditioning the diffusion process on it, formulated in Equation (4.10) [from [20]].

$$p_\theta(\mathbf{x}_{0:T} | \mathbf{s}) := p(\mathbf{x}_T) \prod_{t=1}^T p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{s}), \quad p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{s}) := \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t | \mathbf{s}), \Sigma_\theta(\mathbf{x}_t, t | \mathbf{s})) \quad (4.10)$$

The trend is extracted from the input data and is integrated into the diffusion process at each time step. The trend itself is determined using standard interpolation techniques. A trend is considered a soft constraint because it is merely a guide, and is not expected to be followed exactly. This is achieved by approximating it with a low order polynomial to hinder direct copying of the trend when generating samples. Fixed points (hard constraints) are enforced explicitly by assigning a value at each diffusion step. Mere conditioning of the generation process does not allow for the integration of all types of constraints. Loss-DiffTime extends the previously discussed formulation by adding a loss-term that penalizes violation of constraints proportionally. The optimization problem is re-parameterized to model the final generated sample at each diffusion step.

$$L(\theta) := \mathbb{E}_{t, \mathbf{x}_0, \epsilon} [\|\epsilon - \epsilon_\theta(\mathbf{x}_t, t | \mathbf{s})\|^2 + \rho f_c(\hat{\mathbf{x}}_0)] \quad (4.11)$$

Loss-DiffTime [20] delivered promising results but requires re-training for each new constraint. In an effort to reduce the carbon footprint and increase efficiency, Guided-DiffTime [20] is proposed as the first where re-training to accommodate new constraints is not a necessity. Instead, the generation process can be adjusted according to constraints at inference time. A denoising diffusion implicit model (DDIM) is used to reduce the required diffusion steps at inference. The gradients of differentiable constraints serve as the guide for a pre-trained diffusion model through conditioning at each sampling step. This process speeds up sample generation with a lower computational cost.

4.3.3. Temporal Decomposition using Transformers & Fourier-Based Loss

In [140], the proposed Diffusion-TS model is a non-autoregressive diffusion model relying on a transformer-base architecture. The transformer encoder-decoder network is responsible for capturing the temporal representations, which is guided by a trend and seasonal decomposition inspired by time series analysis. This combination of seasonal-trend decomposition techniques with denoising diffusion models allows Diffusion-TS to efficiently capture meaningful temporal properties from data. The time series model used for the trend and multiple seasonality representation is formulated in Equation (4.12) [from [140]], where x_j indicates the observed series, ζ_j a trend component, $s_{i,j}$ a seasonal component, and e_j the remaining noise.

$$x_j = \zeta_j + \sum_{i=1}^m s_{i,j} + e_j, \quad j = 0, 1, \dots, \tau - 1, \quad (4.12)$$

On a high level, the modeling process is depicted in Figure 4.5. The forward and reverse processes are at the core, where the reverse (denoising) process predicts the time series from noise, based on the interpretable decomposition that is gradually injected into the system.

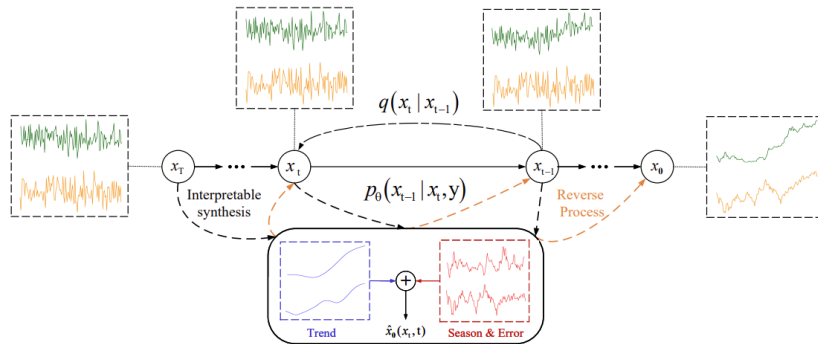


Figure 4.5: Types of constraints imposed of time series, from [20]

To capture the temporal correlations and patterns in the data, an encoder-decoder transformer architecture is used. The encoding process captures information from the entire noisy sequence before decoding, which is then used to inform the deep decomposition architecture of the decoder. This network contains multiple blocks, each consisting of a transformer, feed-forward network, and interpretable decomposition layers. Disentanglement is achieved by imposing specific constraints on the various components, capturing semantic knowledge and boosting interpretability. Trend representation models the underlying mean of the data with a polynomial regressor, aiming to reflect slow-varying behavior. Seasonality and error synthesis aim to capture periodic and non-periodic components from the noisy input using trigonometric representation and Fourier synthetic layers. The Fourier synthetic layers identify the most significant frequencies and amplitudes, reconstructing the seasonality in the time domain. Finally, the model combines the trend, seasonality, and residual noise to generate the original signal.

A Fourier-based loss function is used for reconstruction, focusing on samples rather than noise in each diffusion step, which improves accuracy of time series generation. The Fourier-based loss assesses the differences in the frequency domain using Fast Fourier Transformation (FFT). This approach helps the model accurately reconstruct time series signals. Lastly, conditional generation allows the model to produce new data tailored to specific conditions by using a classifier's gradients to guide the generation process (by iterating through gradient updates and controlling the strength of adjustments). Overall, by achieving disentangled representations through distinct constraints on different components, the model effectively captures specific semantic knowledge and isolates temporal information into different parts, providing explainable and reliable modeling of complex underlying dynamics.

4.4. Summary - Application to (Conditional) Trajectory Generation

This chapter provided a condensed overview of the state-of-the-art around deep generative models tailored to time series generation. The applicability of these models to the trajectory generation process is highly promising, provided they meet several essential requirements. The recent success of a VAE model to this problem calls for a similar study of GANs, where TimeGAN and TTS-CGAN are promising candidates. Both are designed to handle the complex, multi-dimensional data typical of aircraft trajectories, capturing both static and temporal features to produce realistic synthetic data. TimeGAN employs a combination of reconstruction loss, supervised loss, and adversarial loss to generate high-fidelity time series data, effectively maintaining the dynamics and relationships inherent in the original datasets. On the other hand, TTS-CGAN leverages transformer-based architectures, incorporating conditional generation and adversarial training to produce multivariate time series data.

Firstly, the models must generate high-fidelity synthetic data that closely mirrors real-world trajectories. Secondly, they should maintain temporal consistency, accurately capturing the dynamic nature of flight paths over time. Thirdly, the models must be adaptable, allowing for the incorporation of specific flight conditions and operational parameters. Fourthly, scalability is crucial, ensuring the models can handle varying data volumes and complexities. Lastly, the models should support robustness, effectively managing anomalies and producing reliable outputs under diverse scenarios. Both TimeGAN and TTS-CGAN exhibit these characteristics: TimeGAN's integration of various loss functions ensures quality and consistency, while TTS-CGAN's transformer-based approach enhances adaptability and robustness. These strengths suggest that TimeGAN and TTS-CGAN are not only capable of producing high-fidelity synthetic trajectories but are also scalable and adaptable to various operational contexts, marking them as highly valuable tools for advancing trajectory generation in air traffic management. Provided both are implemented, comparisons can be drawn between the RNN and attention-based mechanisms, and the added value proposition of conditional generation concerning specific runways, clusters, and trajectory types (normal, go-around and holding pattern).

5

Research Proposal

The conducted literature review on synthetic data, aircraft trajectory generation, and deep generative models tailored to multivariate time series, has identified a research gap for the thesis project. In [Section 5.1](#), the objective and research questions that are presented will provide guidance during the next phases of the thesis. [Section 5.2](#) provides a rationale on the relevance of the project, linking back to the foreseen downstream prediction tasks and involved stakeholders. Lastly, [Section 5.3](#) is dedicated to concretizing the planning of the research activities over the coming months.

5.1. Objective and Research Questions

The reviewed literature simultaneously reveals an unmet demand for high quality data in the domain of air traffic management (ATM), and a plethora of novel generative modeling architectures that have recently found success in multivariate time series generation in other domains. Variational autoencoders (VAEs), generative adversarial networks (GANs), and diffusion models are popular generative frameworks that can be combined with recurrent, convolutional, and self-attention components to learn the complex temporal dependencies. By incorporating synthetic trajectories, air traffic controllers and ATM systems can practice handling a variety of scenarios, including high traffic volumes, different weather conditions, and emergency situations. This aids in training, refining airspace management, and improving safety and efficiency. The promising use cases of synthetic trajectories leads to the following research objective:

The research aims to investigate the applicability of generative deep learning techniques, notably variational autoencoders, generative adversarial networks, and diffusion models, to generating synthetic aircraft trajectories for air traffic management automation and optimization.

Using synthetic trajectories for ATM automation can be challenging due to the complexity and diversity of landing trajectories and other flight operations. Landing trajectories are influenced by numerous factors, including prevailing weather conditions, runway availability, ATC procedures, and surrounding traffic. Simulating real-world adjustments made by pilots and ATC, such as go-arounds and holding patterns, adds complexity due to the unpredictability and need for precise coordination. Additionally, different aircraft types have varying performance characteristics and handling capabilities, requiring advanced modeling to replicate unique flight behaviors. To compare with the recent implementation of a temporal convolutional VAE [78], generative adversarial networks will be considered to provide an alternative. The evaluation framework consists of an assessment of the diversity, fidelity, and usefulness of the synthetic trajectories. The difficulty of the trajectory generation task will be step-wise increased, from *standard* landings, to go-arounds and holding patterns, to alternative hypothetical situational configurations. The research objective demands for the following research question:

How effective are GANs in generating realistic synthetic 4-dimensional aircraft landing trajectories in an airport's Terminal Maneuvering Area, including go-arounds and holding patterns?

The following sub-questions are aimed to guide the research efforts according to three dimensions:

1. How effective are GANs in generating realistic synthetic aircraft trajectories during the landing phase, including go-arounds and holding patterns?
 - (a) How do various GANs differ in terms of the types and frequency of errors observed in the generated synthetic trajectories compared to actual flight data?
 - (b) What challenges arise in training GANs to convergence, and how do these challenges impact the quality and realism of synthetic trajectories?
 - (c) How interpretable are GANs in terms of understanding how they generate specific trajectory paths and holding patterns?
 - (d) How well do GANs generalize across different types of aircraft, airports, and flight conditions, and what scalability challenges do they face?
2. What are the effects of different conditioning approaches on the generation of synthetic aircraft trajectories during the landing phase using GANs?
 - (a) How can conditioning GANs on runway assignment, normal/go-around/holding types, and cluster assignment improve the realism and adaptability of synthetic trajectories?
 - (b) How can conditioning approaches enhance the interpretability of GANs, allowing researchers to understand the influence of different conditions on trajectory generation?
 - (c) What impact does conditioning have on the computational efficiency of GANs, and how can these effects be optimized?
3. How can the quality of synthetic aircraft trajectories be systematically evaluated in terms of diversity, fidelity, and usefulness?
 - (a) How does model performance measure in terms of sample diversity after applying PCA and t-SNE dimensionality reduction?
 - (b) How does model performance measure in terms of fidelity using a LSTM binary classifier on real and synthetic data?
 - (c) How does model performance measure in terms of usefulness using a Train-Synthetic-Test-Real (TSTR) paradigm for downstream prediction task?

This study is exploitative in nature, aiming to investigate the potential of advanced generative models in the context of trajectory generation. By examining the capabilities of (conditional) GANs (such as TimeGAN [138] and TTS-CGAN [85]), this research seeks to uncover new insights and possibilities for enhancing the realism and effectiveness of synthetic trajectory data. The findings are intended to lay the groundwork for future developments and applications in this evolving field.

5.2. Relevance of the Project

The aviation industry is in the midst of a transformative phase, where amongst other developments, technological advancements fueled by data-driven approaches aim to modernize air traffic management without sacrificing safety. Within this context, the European Commission's SynthAir project (part of the SESAR Joint Undertaking) aims to leverage novel AI-based methods for synthetic data generation in the domain of Air Traffic Management (ATM) systems. An ambitious effort due to inherent scarcity of data and the challenge of effective model generalization, but with the promise of allowing more robust testing, validation, and deployment of ATM systems worldwide. Analysis and optimization of air traffic is dependent on the availability of large volumes of data distributed over all types of scenarios (including specific events such as go-arounds and holding patterns). This has led to a surge in demand for synthetic flight trajectories that are virtually indistinguishable from observed historical data. Complementary to resemblance of the true underlying distribution, artificially generated trajectories must be physically plausible to fly, and be adaptable to varying weather, traffic density, and geographic variables. The SynthAir project refers to the ability of training and developing a model on given datasets and subsequently using this to generate based on previously unseen datasets: the Universal Time Series Generator (UTG).

Focusing on landing trajectories, particularly go-arounds and holding patterns, is an area of significant importance in aviation research due to its complexity and impact on safety. Landing is one of the most challenging phases of a flight, with numerous variables such as weather conditions, air traffic, and runway availability influencing aircraft trajectories. By generating synthetic landing trajectories, researchers can create realistic datasets to analyze these complex scenarios and develop more effective solutions for handling them (which is in the interest of airlines, airports, and ANSPs). This includes improved performance in a myriad of downstream tasks including decision-making algorithms for pilots and air traffic controllers, as well as enhanced flight planning and scheduling tools. Recent advancement in generative deep learning techniques has risen to prominence in various other industries, but has yet to gain traction in the ATM domain. Inspired by initial findings using a VAE architecture, this thesis explores the implementation of GANs with recurrent and transformer elements to handle sequential dependencies, as viable alternatives seeking to reach superior performance. Overall, this research aims to enhance safety and efficiency in the landing phase of flights, contributing to more reliable and streamlined aviation operations.

5.3. Planning of the Project

Planning is a critical step in any successful thesis project. This planning section outlines the timeline, milestones, and goals for the completion of the thesis project, containerizing research activities into work packages. Each work package contains a description and the key tasks that will be performed, alongside specific timelines and deadlines. If necessary, this planning will be updated on a rolling-basis.

Work Package 1: Data Collection and Preprocessing

Description: Collect and preprocess flight trajectory data. Label data according to runway assignment, go-arounds and holding patterns. Cluster and label trajectories using K-Means with Dynamic Time Warping distance. Gather data primarily from ADS-B OpenSky.

Key Activities:

- *Data sourcing* - Collect data from OpenSky and/or Eurocontrol databases
- *Data cleaning* - Perform exploratory data analysis and remove noise, outliers, and errors from the data
- *Feature selection* - Extract key data for trajectories (longitude, latitude, airspeed, and altitude)
- *Data clustering & labeling* - Label trajectories according to runway assignment and normal/go-around/holding landing types; perform k-means clustering using dynamic time warping distance and labeling
- *Data sampling/interpolation* - Format data into the required format by sampling and/or interpolating

Work Package 2: Establishing the Evaluation Framework

Description: Establishing a comprehensive evaluation framework to assess the quality and effectiveness of the GANs in generating synthetic aircraft trajectories during the landing phase. The framework focuses on evaluating data diversity, fidelity, and usefulness.

Key Activities:

- *Diversity assessment* - Qualitative assessment of sample diversity using PCA and t-SNE dimensionality reductions
- *Fidelity assessment* - Quantitative assessment of binary time series classifier test error between synthetic and real data
- *Usefulness assessment* - Quantitative assessment of time series by using the TSTR evaluation paradigm
- *Energy distance assessment* - Statistical distance measurement between probability distributions of synthetic and real data

Work Package 3: Model Development and Training

Description: Implementation and training of the (conditional) GANs.

Key Activities:

- *Unconditional GAN training* - Implementation and training of an unconditional GAN.
- *Conditional GAN training* - Implementation and training of a conditional GAN.
- *Hyperparameter tuning* - Optimize learning rates, batch sizes, and all other hyperparameters pertinent to the specific model architectures

Work Package 4: Conditioning Approaches

Description: Explore the different conditioning scenarios to improve the quality of synthetic trajectories generated by TTS-CGAN. Condition the models on runway assignment, normal/go-around/holding type, and cluster assignment. Investigate the controllability advantage gained from the conditioning mechanism.

Key Activities:

- *Conditioning Feature identification* - Determine relevant conditioning features, such as weather patterns, runway availability, and cluster assignment
- *Impact assessment* - Evaluate how conditioning improves the quality (diversity) and accuracy of synthetic trajectories
- *Iterative refinement* - Continuously refine conditioning methods based on performance feedback

Work Package 5: Model Evaluation and Validation

Description: Evaluate and validate the performance of the GANs in generating realistic synthetic aircraft trajectories during the landing phase, particularly in go-arounds and holding patterns (testing for varying sampling rates, trajectory complexities, runway assignments, and cluster assignments).

Key Activities:

- *Performance metrics* - Assess both the synthetic data generated by the GANs according to the defined evaluation framework (diversity, fidelity, and usefulness) for step-wise increasingly difficult modeling tasks
- *Error analysis* - Analyze the discrepancies between synthetic and real trajectories to identify potential areas for improvement
- *Model validation* - Test the models' robustness and generalization across different landing scenarios and conditions

Work Package 6: Integration Analysis

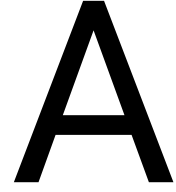
Description: Integrate results and insights from the previous work packages to produce a comprehensive analysis of the GANs' effectiveness in generating synthetic aircraft trajectories during the landing phase.

Key Activities:

- *Results synthesis* - Combine findings from model training, conditioning, and evaluation to identify overall trends and conclusions
- *Comparative analysis* - Compare and contrast the performance of the GANs across different (conditioning) scenarios
- *Final report preparation* - Prepare the final thesis literature study and research paper presenting the project's findings and conclusions
- *Reflection & future research directions* - Identify potential areas for further research based on outcomes

III

Supporting work



TimeGAN & TTS-CGAN - Architectures

This appendix provides detailed outlines of the two GAN models implemented as part of this thesis project: TimeGAN [138] and TTS-CGAN [85]. Both models represent advancements in the field, offering different approaches to capturing complex patterns found in time-series data. TimeGAN is older and has risen to become a standard benchmark for performance. TTS-CGAN has been proposed more recently and extends the previously proposed TTS-GAN model (of the same authors) with a conditioning mechanism. Provision of the detailed model architectures aims to provide transparency for the reader who wishes to obtain more details (without consulting the full papers). The descriptions that follow are based on the original papers.

A.1. TimeGAN

TimeGAN [138] was proposed in 2019 and has become a well-known benchmark in the field of time-series data synthesis. It has proved a valuable tool for various applications such as in financial modeling, medical data analysis, and more. The section provides a description of the model architecture and the training and loss functions.

Architecture

TimeGAN is designed to produce time-series by combining autoregressive and adversarial methods. Its architecture includes four interconnected components: the embedding network, the recovery network, the sequence generator, and the sequence discriminator. Each of these components has a specific role in capturing and generating complex temporal patterns in the data. Figure B.9 depicts the block diagram of the TimeGAN model, whereas Figure B.10 depicts the training scheme (solid lines represent forward data propagation and dashed represent backpropagation).

The embedding network's primary function is to transform the raw, high-dimensional time-series data into a compact, lower-dimensional latent space. The key is to simplify while retaining essential temporal features. The embedding network uses layers of recurrent neural networks (RNNs), such as LSTMs (Long Short-Term Memory networks) or GRUs (Gated Recurrent Units), which are adept at handling sequential data. The recovery network's role is to take the latent space representation from the embedding network and reconstruct the original time-series data. This step ensures that the latent representation preserves all the necessary information from the original data. The recovery network also employs neural network layers designed for sequential data. The objective here is to learn the mapping back from the latent to the actual data space accurately.

The sequence generator generates synthetic sequences in the latent space. Sequences are then mapped back to the original data space through the recovery network, resulting in synthetic time-series data. The generator produces latent sequences that, once processed by the recovery network, result in synthetic data indistinguishable from the real data. The sequence discriminator's function is to differentiate between real and synthetic time-series sequences. It provides feedback to the generator, encouraging generation of more realistic sequences over time. The discriminator too, is a neural network with layers suitable for sequential data, such as RNNs, LSTMs, or GRUs. It aims to maximize its accuracy in distinguishing real sequences from synthetic ones, thereby providing robust feedback to the generator.

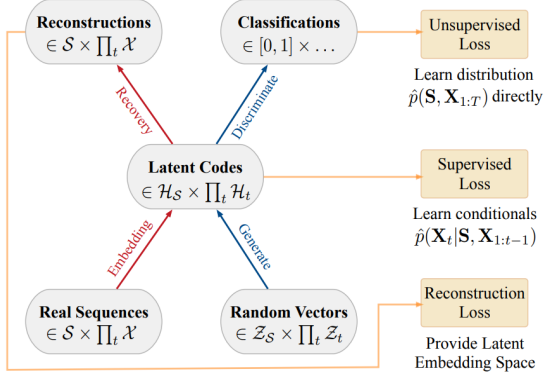


Figure A.1: Model architecture block diagram, from [138]

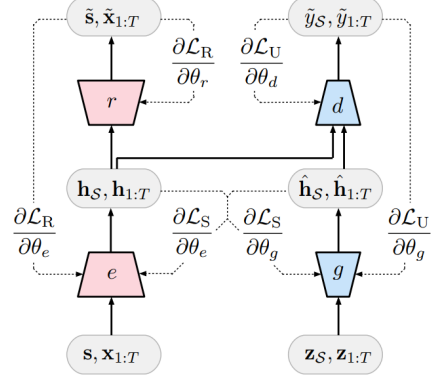


Figure A.2: Data propagation, from [138]

Training & Loss Functions

The training of TimeGAN involves a combination of supervised and unsupervised learning. This dual approach helps the model learn both the distribution of the data and the temporal dynamics. The reconstruction loss ensures that the embedding and recovery networks accurately capture and reconstruct the original time-series data from the latent space. The reconstruction loss ($\mathcal{L}_R$) is defined as the sum of the differences between the original $(\mathbf{s}, \mathbf{x}_{1:T})$ and reconstructed $(\tilde{\mathbf{s}}, \tilde{\mathbf{x}}_{1:T})$ data points, formulated in Equation (A.1).

$$\mathcal{L}_R = \mathbb{E}_{\mathbf{s}, \mathbf{x}_{1:T} \sim p} \left[\|\mathbf{s} - \tilde{\mathbf{s}}\|_2 + \sum_t \|\mathbf{x}_t - \tilde{\mathbf{x}}_t\|_2 \right] \quad (\text{A.1})$$

The generator is trained using two modes: open-loop and closed-loop, each contributing differently to the training process. The generator operates in an autoregressive manner, using its own previous synthetic outputs ($\hat{h}_S$ and $\hat{h}_{1:t-1}$) to synthesize the next vector ($\hat{h}_t$). This mode relies on the unsupervised loss ($\mathcal{L}_U$), which helps the generator learn to produce sequences that the discriminator finds difficult to distinguish from real sequences. The unsupervised loss function is formulated in Equation (A.2).

$$\mathcal{L}_U = \mathbb{E}_{\mathbf{s}, \mathbf{x}_{1:T} \sim p} \left[\log y_S + \sum_t \log y_t \right] + \mathbb{E}_{\mathbf{s}, \mathbf{x}_{1:T} \sim \hat{p}} \left[\log(1 - \hat{y}_S) + \sum_t \log(1 - \hat{y}_t) \right] \quad (\text{A.2})$$

Here, y_S and y_t are the outputs of the discriminator in the case of real data, while $\hat{y}_S$ and $\hat{y}_t$ pertain to synthetic data. In the closed-loop mode, the generator uses real data embeddings ($h_{1:t-1}$) computed by the embedding network. These can then be used to create the next latent vector. This approach employs a supervised loss ($\mathcal{L}_S$), which measures the discrepancy between the real conditional distributions and the generated ones, according to Equation (A.3).

$$\mathcal{L}_S = \mathbb{E}_{\mathbf{s}, \mathbf{x}_{1:T} \sim p} \left[\sum_t \|h_t - g_X(h_S, h_{t-1}, z_t)\|^2 \right] \quad (\text{A.3})$$

This loss encourages the generator to accurately model the step-by-step relationships in the time-series data. By alternating between open-loop and closed-loop training modes, TimeGAN captures both the overall data distribution and the temporal relationships.

A.2. TTS-CGAN

TTS-CGAN [85] is an advanced extension of the TTS-GAN framework designed to handle multi-class data efficiently. This model enhances the original TTS-GAN [84] by introducing conditional capabilities, allowing it to generate synthetic data across multiple categories from a single model. The section provides a description of the model architecture and the training and loss functions.

Architecture

As with all GANs, TTS-CGAN is comprised of two main components: the generator and the discriminator, where in this case each utilizes a transformer-based architectures (the model outline is depicted in Figure A.3). The generator in TTS-CGAN starts with a 1D vector of random noise and a categorical label. These

inputs are concatenated and subsequently processed through a linear mapping to adjust their dimensionality. The transformed vector then passes through a series of transformer encoder blocks. A given transformer block includes a multi-head self-attention layer and a multi-layer perceptron with GELU activation. Positional encodings are issued to include temporal ordering, and the output sequence is reshaped by a Conv2D layer to match the dimensions of real signals. This process enables the generator to produce similar synthetic time-series data in terms of structure of actual data. This is achieved by conditioning on both random noise and a target category label. Layer normalization is applied throughout to stabilize training and dropout is used after each block to prevent overfitting. The core transformer blocks of the model can be repeated and stacked by altering the model depth parameter.

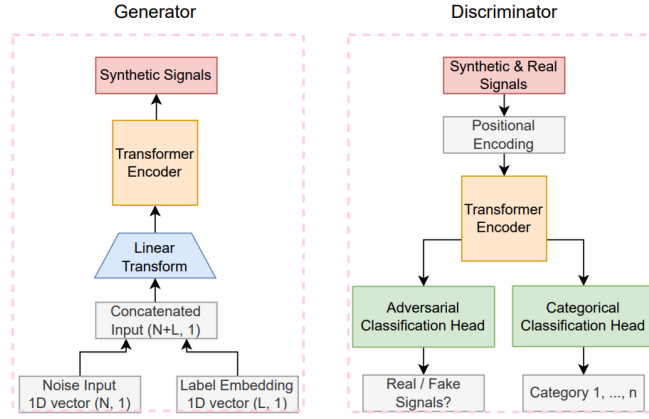


Figure A.3: TTS-CGAN model architecture, from [85]

The discriminator in the TTS-CGAN model evaluates the authenticity of the generated signals and classifies them into their respective categories, using similar transformer encoder blocks. The discriminator's architecture includes two classification heads: an adversarial head (D_{adv}) and a categorical head (D_{cls}). The adversarial head distinguishes between real and synthetic signals, while the categorical head classifies the signals into predefined categories. By incorporating positional encodings and dividing the sequence into patches, the discriminator effectively analyzes time-series data and assesses both authenticity and category. Time series are treated based on inspiration from image handling of convolutional networks. Time-series are input as a 2D image with a height of 1 and width equal to the number of time steps, visualized in Figure A.4.

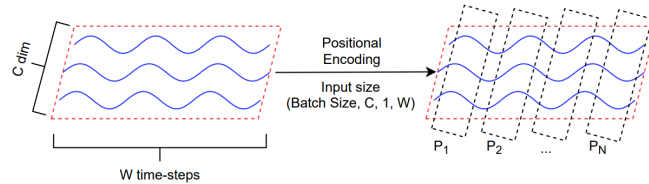


Figure A.4: Processing of multivariate time-series in TTS-CGAN, from [85]

Training & Loss Functions

Training the TTS-CGAN involves optimizing both the generator and the discriminator using a combination of adversarial and categorical losses. The objective functions of the generator (L_G) and the discriminator (L_D) are defined according to Equation (A.4).

$$\begin{aligned} L_D &= -L_{adv} + \lambda L_{cls}^r \\ L_G &= L_{adv} + \lambda L_{cls}^f \end{aligned} \quad (\text{A.4})$$

L_{adv} is the adversarial loss, which measures how well the discriminator (D) can distinguish between real and synthetic signals. The classification loss terms L_{cls}^r and L_{cls}^f measure the performance of the discriminator in classifying real and synthetic signals into their correct categories, respectively. The hyperparameter λ balances the classification and adversarial loss in terms of comparative importance (generally set to 1). The

adversarial loss L_{adv} ensures that synthetic signals are indistinguishable from real ones, defined according to Equation (A.5).

$$L_{adv} = E_x [\log D_{adv}(x)] + E_{z,c} [\log (1 - D_{adv}(G(z, c)))] \quad (\text{A.5})$$

$G(z, c)$ represents the generated signal based on both random noise z and the assigned categorical label c , and D_{adv} attempts to differentiate between real and synthetic signals. The generator seeks to minimize L_{adv} , while the discriminator aims to maximize it. With the goal of training stabilization and ensuring data quality, the authors opted to use a Wasserstein GAN objective with gradient penalty, defined according to Equation (A.6).

$$L_{adv} = E_x [D_{adv}(x)] - E_{z,c} [D_{adv}(G(z, c))] - \lambda_{gp} E_{\hat{x}} [(\|\nabla_{\hat{x}} D_{adv}(\hat{x})\|_2 - 1)^2] \quad (\text{A.6})$$

Therein, the sampling $\hat{x}$ is performed on a straight line between real and synthetic samples. λ_{gp} is a hyperparameter generally set to 1. This approach ensures high-fidelity synthetic signal generation and effective training dynamics. The categorical classification losses guide the discriminator to classify real signals correctly and ensure that the generator produces signals matching the target category. These losses are defined according to Equation (A.7).

$$\begin{aligned} L_{cls}^r &= E_{x,c'} [-\log D_{cls}(c' | x)] \\ L_{cls}^f &= E_{z,c} [-\log D_{cls}(c | G(z, c))] \end{aligned} \quad (\text{A.7})$$

Therein, L_{cls}^r and L_{cls}^f are the categorical classification losses for real and synthetic signals, respectively. c' is the real signal's label, and c is the target category for the synthetic signal.

B

Real & Synthetic Datasets

This appendix provides an overview of the datasets utilized in this study for training and evaluating the generative models. It includes both the real (unlabeled and labeled) and synthetic aircraft trajectory datasets, in [Appendix B.1](#) and [Appendix B.2](#), respectively. The real datasets are sourced from ADS-B data through the OpenSky Network [110] (directly provided in the Python *traffic* module [99]). Real datasets are varied in terms of trajectory complexity, runway labeling, cluster assignment, and operational type (normal, go-around, or holding pattern). Following the real datasets, several synthetic datasets are presented, generated by different configurations of the TimeGAN [138] model. In this report, the data is presented visually to ensure familiarity and understanding of the underlying trajectory distribution, and the effect of various labeling strategies. A selection of raw files themselves are directly available as *.npy* or *.npz* files in the repository.

B.1. Real Datasets

Real datasets are further divided in four categories (experimental use cases), namely unconditional, runway allocation, operational condition, and cluster assignment. In most cases, variations are provided in terms of the maximum cumulative track angle change filter used as a complexity proxy of trajectories. As an example, a maximum track angle of 180 degrees removes all trajectories from the dataset that exceed this threshold. This filter was applied to reduce the complexity of the sparser regions of the underlying trajectory distribution. The following subsections provide visuals of the prepared datasets.

Unconditional

In the unconditional case, trajectories are used without specific labeling or conditioning. This approach provides a baseline for evaluating the effectiveness of conditional models in generating realistic and diverse synthetic trajectories. Given that roughly 95% of landings involve runway 14, the unconditional set is solely on this runway. [Figure C.9](#) and [Figure C.10](#) depict the set of trajectories with a cumulative track angle maximum threshold of 180 and 270 degrees, demonstrating the widening/narrowing of the data distribution.

Labeled - Runway Allocation

Each trajectory can be labeled based on the specific runway used for landing, providing additional granularity in the dataset. This labeling allows for analysis of runway-specific landing behaviors and performance, potentially enhancing a model's adaptability. As is evident, the vast majority of landings are on runway 14. [Figure B.3](#) and [Figure B.4](#) show the distribution of trajectories per runway, for two cumulative track angle thresholds.

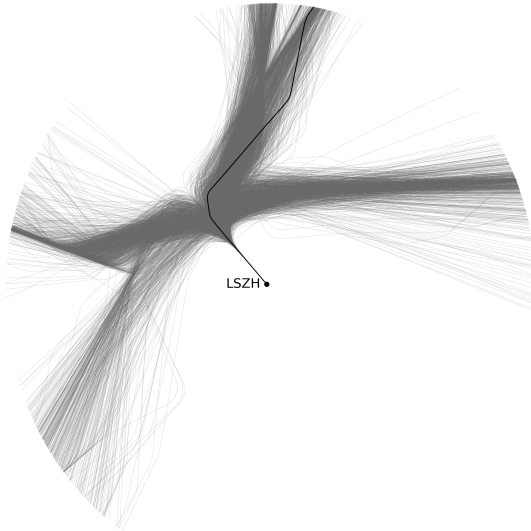


Figure B.1: Unconditional trajectories - 180 degree track angle

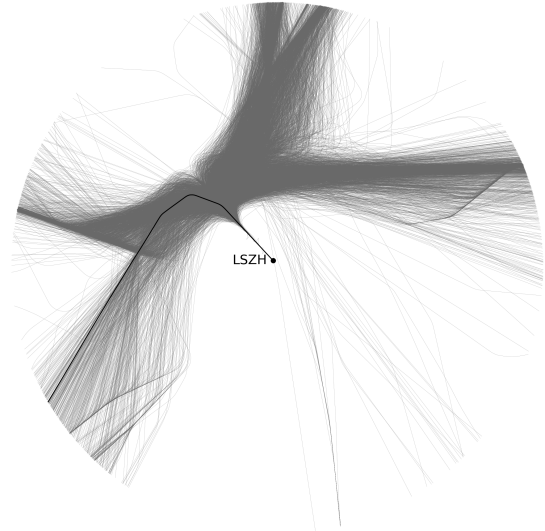


Figure B.2: Unconditional trajectories - 270 degree track angle

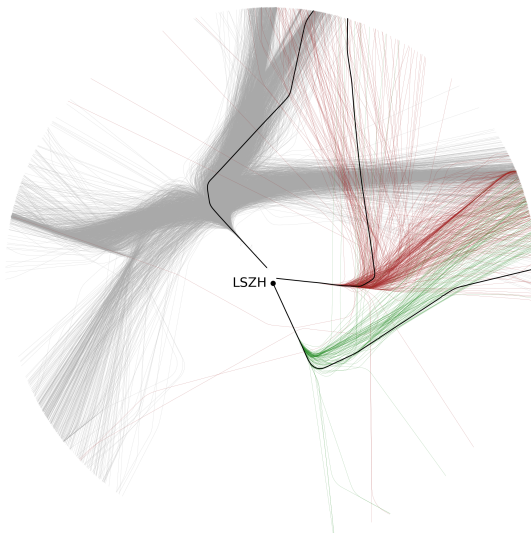


Figure B.3: Trajectories per runway - 180 degree track angle

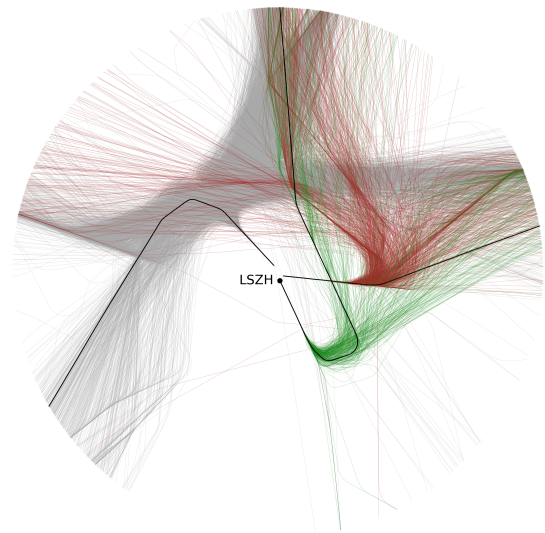


Figure B.4: Trajectories per runway - 270 degree track angle

Labeled - Operational Condition

The operational condition refers to the *nature* of the trajectory, either being normal, a go-around, or involving a holding pattern. Normal procedures refer to standard landing sequences that do not involve go-arounds or holding patterns. These trajectories serve as a baseline for comparison, highlighting deviations in more complex landing scenarios. Go-arounds are identified by detecting multiple Instrument Landing System (ILS) alignments of the aircraft, separated by a climbing phase. This labeling helps discern the unique characteristics of trajectories where the aircraft makes a second landing attempt, sometimes on a different runway. Holding patterns are recognized by the total change in track angle of the aircraft's flight path, using a minimum cumulative track angle threshold to label such trajectories. This requires unwrapping the track angle to handle angular discontinuities, ensuring accurate detection of holding patterns. [Figure B.5](#) and [Figure B.6](#) visualize the difference in trajectory types, both in terms of geographic and altitude profiles.

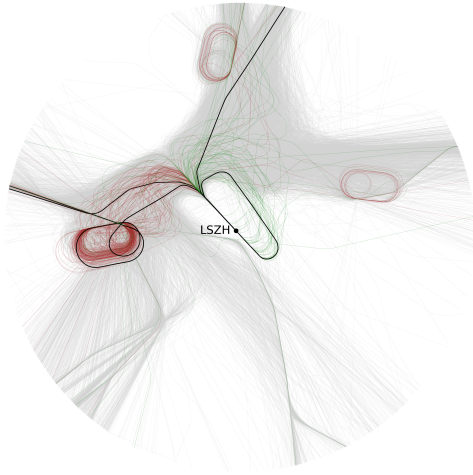


Figure B.5: Geographic profiles of various operational conditions

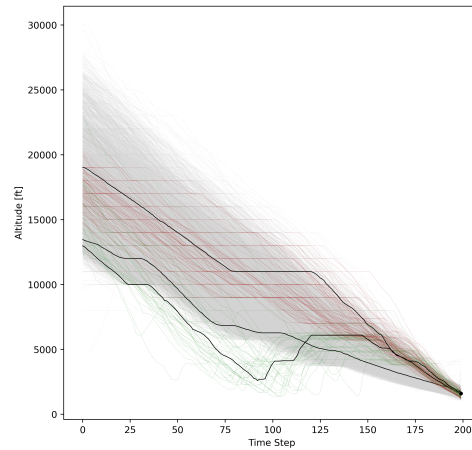
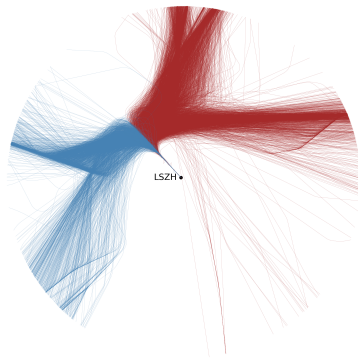
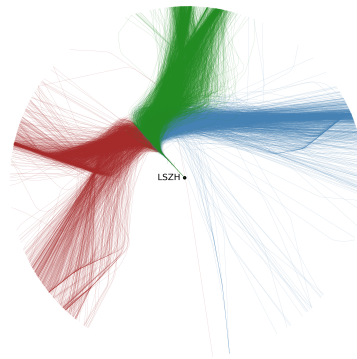
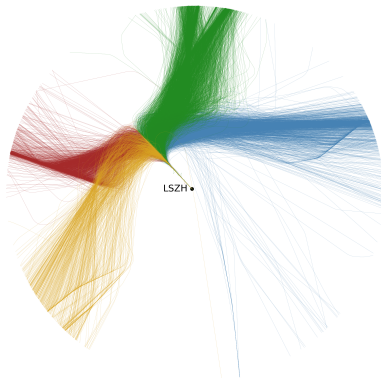
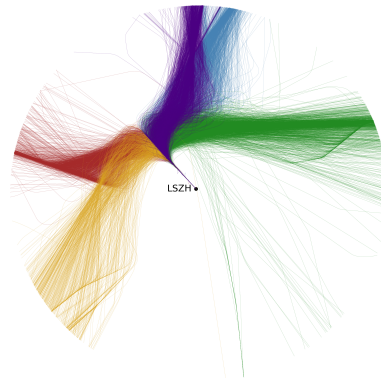


Figure B.6: Altitude profiles of various operational conditions

Labeled - Cluster Assignment

In addition to explicit labeling, trajectories are clustered using the k-means algorithm with Dynamic Time Warping (DTW) as the distance metric. DTW accounts for temporal distortions by aligning sequences to minimize distance, enabling accurate clustering of trajectories with similar patterns but varying temporal dynamics. For different track angle thresholds on runway 14, the number of clusters is varied between $N = 2$ and $N = 5$. The figures below depict the resulting labeled trajectories for an increasing cluster count.

Figure B.7: Clustered trajectories ($N = 2$)Figure B.8: Clustered trajectories ($N = 3$)Figure B.9: Clustered trajectories ($N = 4$)Figure B.10: Clustered trajectories ($N = 5$)

B.2. Synthetic Datasets

This section provides an explanation of the synthetic datasets that are made available in the open repository. The value proposition of these datasets is discussed (characteristics of the different scenarios/groupings), and insight is given into the temporal and spatial coverage of the key variables using heat maps.

Dataset Characteristics

The synthetic datasets generated in the TimeGAN experiments are designed to capture a comprehensive range of aircraft landing trajectories across different operational scenarios and environmental conditions. Each dataset consists of multiple groups representing distinct patterns or behaviors in the trajectory data. TimeGAN is trained for each group separately, after which groups can be combined to form a single dataset. For instance, one dataset is segmented into five clusters, which likely represent different trajectory profiles influenced by varying weather conditions, aircraft types, or operational strategies. Another dataset is divided into three clusters, focusing on broader groups of flight behaviors. Additionally, there is a dataset specifically categorized by runway usage, capturing variations in landing patterns tied to different runway configurations, which the results suggest is essential for understanding how different runway usage influence landing trajectories. The go-around dataset, where trajectories start when an initial landing attempt is aborted, presents unique challenges due to its highly dynamic nature and rarity in the dataset. While experimenting with these datasets, it was observed that increasing the number of training steps and expanding the number of hidden dimensions in the model led to improved coverage of the diverse data patterns. However, this also introduced more noise into the generated data, highlighting a crucial trade-off between model complexity and the quality of synthetic data. This balance is critical when training generative models like TimeGAN to ensure they generate realistic data that represent the range of operational scenarios. To alleviate some of the noise, a smoothing filter (such as a simple moving average or exponentially weighted moving average) was also applied. Both the raw and post-processed versions of datasets are provided in the repository.

Temporal & Spatial Coverage Verification

The datasets incorporate a range of variables critical for modeling aircraft landing behaviors, with distributions that reflect the complex dynamics of flight operations. Continuous variables such as longitude, latitude, and altitude exhibit multi-modal distributions, capturing different operational phases and conditions within each cluster. The data is stored in an array format with dimensions (number of samples, sequence length, feature dimensions). Here, the number of samples corresponds to the total number of recorded flight trajectories included in the dataset, while the sequence length represents the duration or the number of time steps over which each trajectory is recorded. The feature dimensions denote the various variables (state vector) measured at each time step, consisting of longitude, latitude and altitude. Categorical variables, such as runway identifiers and cluster labels, follow distributions that reflect operational frequencies, with certain clusters or runways being more common under specific conditions. The results highlighted that increasing the number of hidden dimensions allowed TimeGAN to better model the intricate relationships between these variables, resulting in synthetic data that more closely matched the distributions observed in the real datasets. However, this improvement in data coverage came at the cost of increased noise, particularly for less frequent or more dynamic scenarios, such as those captured in the go-around dataset. Moreover, the relationship between the input real dataset characteristics and the optimal number of clusters or grouping remains inadequately quantified. Establishing a more precise relationship is crucial, as it would enable better tailoring of the model architecture to the specific complexities of the dataset, improving both the fidelity and interpretability of the generated data. [Figure B.11](#) to [Figure B.18](#) contain heat-maps of the synthetic geographic and altitude profiles of aircraft (in the case of five and three clusters). It is evident that there is a concentration around typical profiles (increasingly for final approach), but that this is not at the expense of diversity.

Note on TTS-CGAN Results

The TTS-CGAN model struggled due to the mismatch between its complexity and the variability of the data. TTS-CGAN employs GANs and self-attention mechanisms to handle long-range dependencies, but these require large amounts of diverse, high-quality data to be effective. Given the high dimensionality and variability of aircraft trajectory data, the model could not train on the required dataset, resulting in outputs that were neither interpretable nor realistic. Efforts to simplify the task, such as reducing the sampling rate and focusing on more homogeneous subsets of data, were not successful. The model still failed to capture the essential temporal and spatial correlations required for generating realistic trajectories, and its conditional mechanism

was not robust enough to handle the large variations between different groups effectively. However, due to the lack of interpretability of the current results (exemplified in [Figure B.19](#)), understanding why the synthetic data quality is poor remains largely speculative.

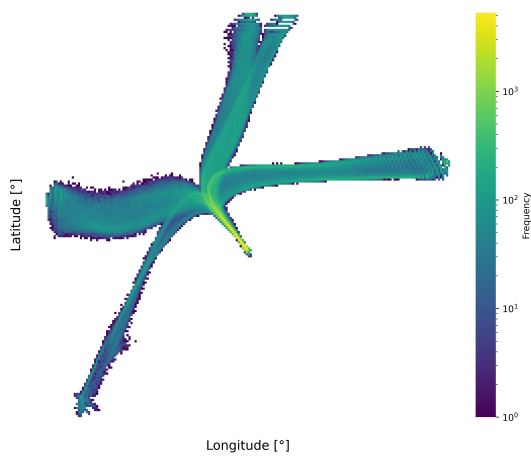


Figure B.11: Geographic profile heat map (5 clusters)

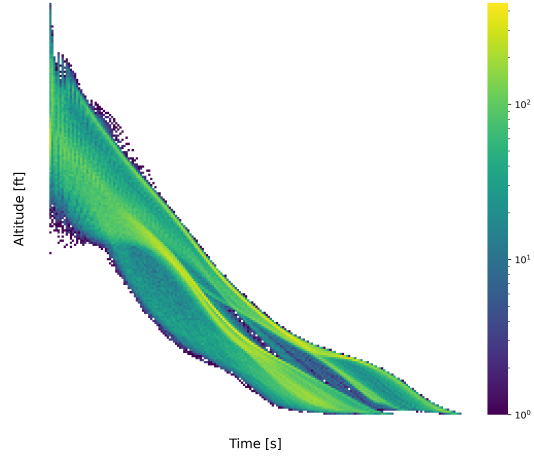


Figure B.12: Altitude profile heat map (5 clusters)

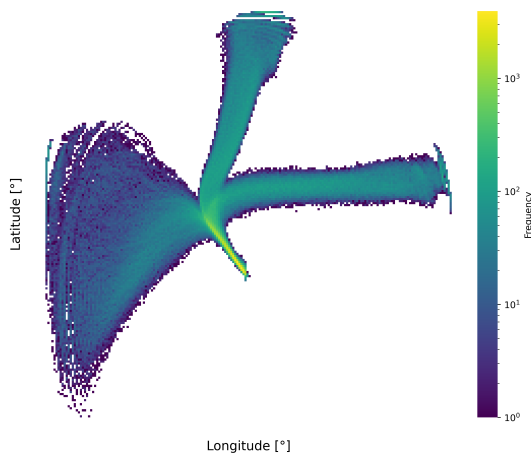


Figure B.13: Geographic profile heat map (3 clusters)

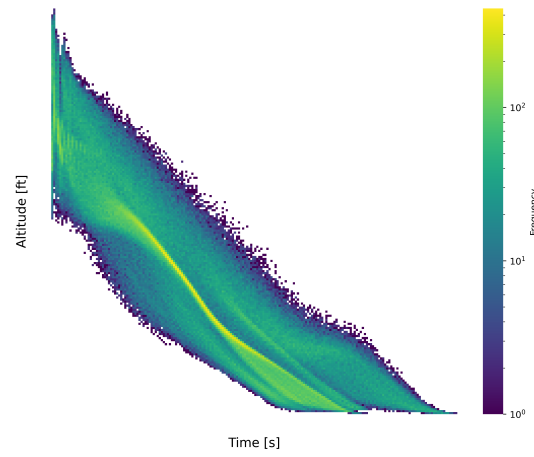


Figure B.14: Altitude profile heat map (3 clusters)

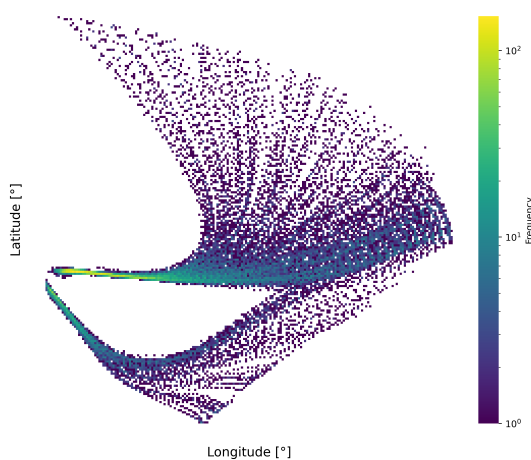


Figure B.15: Geographic profile heat map (runways 28 & 34)

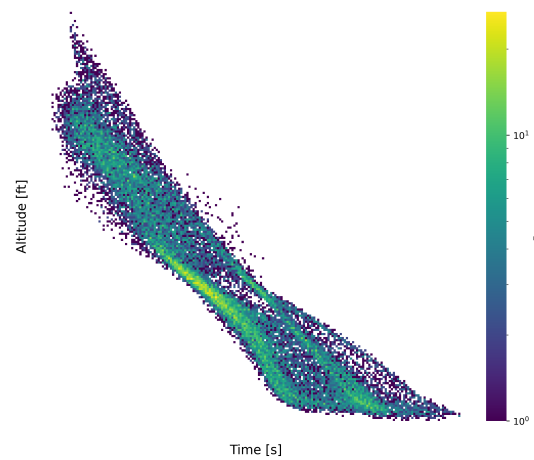


Figure B.16: Altitude profile heat map (runways 28 & 34)

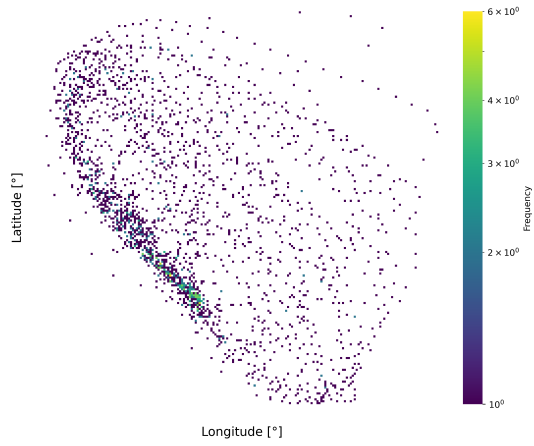


Figure B.17: Geographic profile heat map (go-arounds)

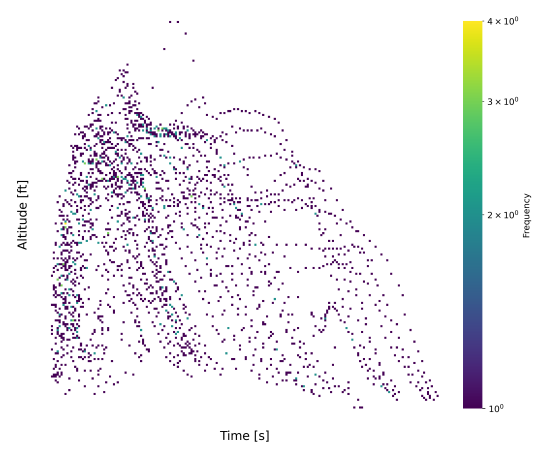


Figure B.18: Altitude profile heat map (go-arounds)

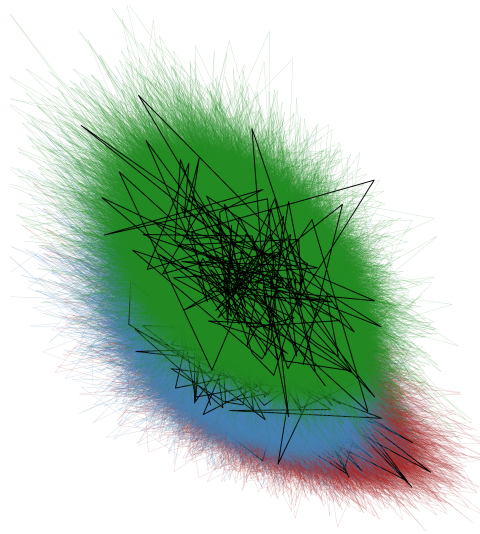


Figure B.19: Example of uninterpretable results of TTS-CGAN

C

TimeGAN Verification - Hyperparameter Variation & Model Behavior

This appendix provides an analysis of the effects of hyperparameter variations and notable behaviors regarding the TimeGAN model. The inherent unpredictability and volatility of the generation process necessitate careful observation of how variations in model parameters and datasets influence outcomes. This meticulous examination is crucial for verifying the reliability and validity of the synthetic data produced by TimeGAN. While GANs are notoriously unstable during training due to issues such as mode collapse and sensitivity to hyperparameters, the authors of TimeGAN claim that the introduction of a latent space that explicitly captures temporal relationships helps mitigate some of these stability challenges. [Appendix C.1](#) examines the impact of key hyperparameters on the generation process. [Appendix C.2](#) discusses notable model behaviors observed during training, specifically mode collapse, and the trade-off between mode coverage and noise in synthetic data.

C.1. Hyperparameter Variations

This section investigates the effects of varying several key hyperparameters on the performance of the TimeGAN model. These are the number of layers in the model components, the module type (GRU or LSTM), the hidden dimensionality, and the number of iteration steps (epochs). To limit the scope, the analysis focuses on a single cluster of trajectories. The baseline values to conduct the variational studies are: number of layers (3), module type (GRU), hidden dimensionality (24), and the number of iteration steps (1000). These values were based on initial tests and the requirement of limiting computational overhead. The starting point anomaly (further discussed in [Appendix C.2](#)) occurs due to lower training steps, but does not dismiss the effects of the variation in parameters. Overall, this section clearly demonstrates the variability in performance with varying parameter configurations. Gray indicates real trajectories, and red indicate synthetic ones.

Number of Layers - *The number of RNN layers stacked in the network components* - In the context of aircraft trajectories, increasing the number of layers enhances the models ability to capture intricate, long-term dependencies in flight patterns, such as multi-step maneuvers and interactions. Too many layers may complicate training and risk overfitting, affecting the models generalization. Too few layers are seen to lead to cause jagged flight paths. The number of layers has little to no effect on the mode coverage of the data.

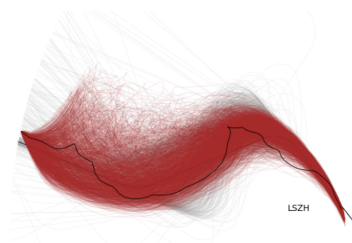


Figure C.1: Two layers

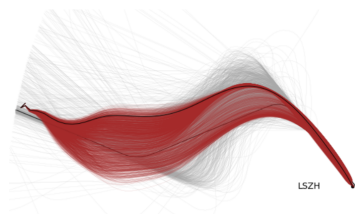


Figure C.2: Three layers

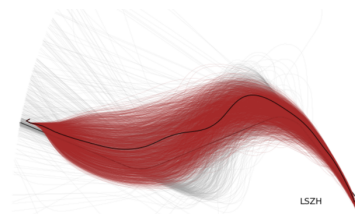


Figure C.3: Four layers

RNN Module Type - Specifies the type of RNN cell used, either as LSTM or GRU - For aircraft trajectory generation, LSTM units are effective in capturing long-term dependencies in flight patterns, whereas GRU units offer a faster and less complex alternative. An LSTM network is more flexible/complex, and requires more training data to reap its rewards. For the same dataset (under identical hyperparameter configurations), the GRU model performance superior. The LSTM model overfits to noise and is projected to increase in performance with more data and higher sampling rates (longer sequences).

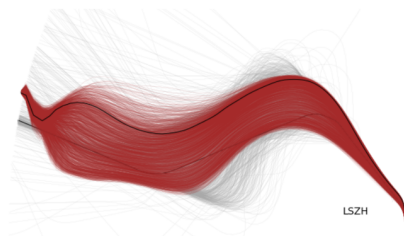


Figure C.4: GRU Modules

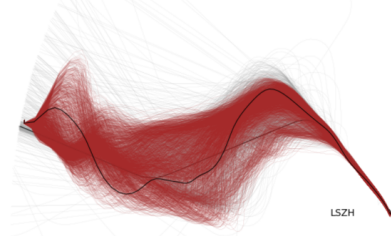


Figure C.5: LSTM Modules

Hidden Dimensionality - Specifies the dimensionality of the hidden states in the RNN layers - For aircraft trajectory generation, a larger hidden dimensionality enables the model to capture detailed flight patterns and subtle variations in altitude, speed, and direction. However, overly large values might lead to overfitting, where synthetic trajectories become too closely aligned with the training data. With an increase in dimensionality, trajectories become smoother but more localized according to a multimodal distributions. This is most clearly seen in the case of a hidden dimensionality of 48.

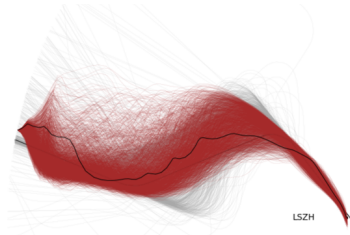


Figure C.6: 24 hidden dimensions

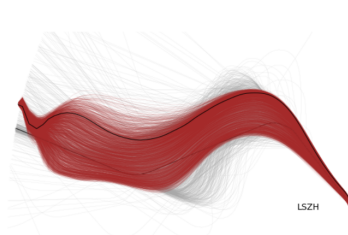


Figure C.7: 36 hidden dimensions

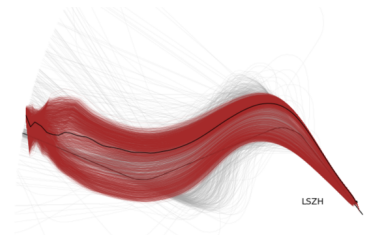


Figure C.8: 48 hidden dimensions

Number of Iteration Steps - The total number of training steps or epochs - More iterations allow the model to better learn the complex patterns in aircraft movements, resulting in more realistic synthetic data. However, excessive iterations can lead to overfitting, where the synthetic data may become too similar to the training data and lose generalizability. This results data that does cover the original distribution to a greater extent, but at the expense of flight path smoothness.

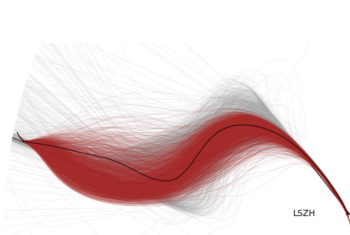


Figure C.9: 500 training iterations

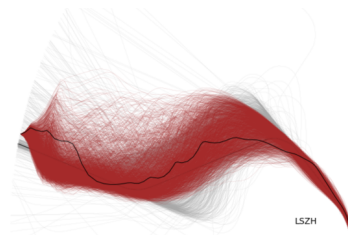


Figure C.10: 1,000 training iterations

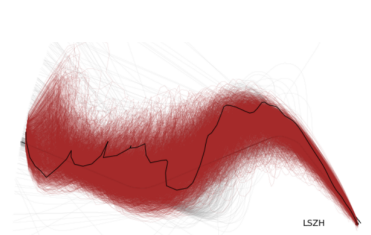


Figure C.11: 2,500 training iterations

C.2. Notable Model Behavior

This section focuses on specific behavioral issues that arose during the training and generation phases of the TimeGAN model. One prominent issue discussed is mode collapse, a phenomenon in which the generator produces highly repetitive or limited outputs, thus reducing diversity in the generated data. Other notable behavior concerns the trade-off between observed noise and coverage of the trajectory distribution, with an

increasing number of training iterations (epochs).

Model Collapse at the Start of Trajectories - Mode collapse occurred in the TimeGAN model when it became overwhelmed by the variability in the trajectory data relative to the amount of available training data. Specifically, the model exhibited a tendency to generate synthetic trajectories that all commenced from the same initial point. This phenomenon was particularly pronounced when clustering techniques were not applied to the data and with high sampling rates of the training data. The underlying cause of this mode collapse is that the model struggled to effectively capture and replicate the diverse starting conditions of the trajectories due to insufficient data coverage. The excessive variation in trajectory patterns relative to the training data led to a situation where the model defaulted to a limited set of outputs. The curse of dimensionality played a key role, with sparse data in such a large input space, the model defaulted to generating a narrow set of initial states, failing to adequately represent the full range of trajectory variations. A lower batch size exacerbated this issue as is evident in [Figure C.12](#).

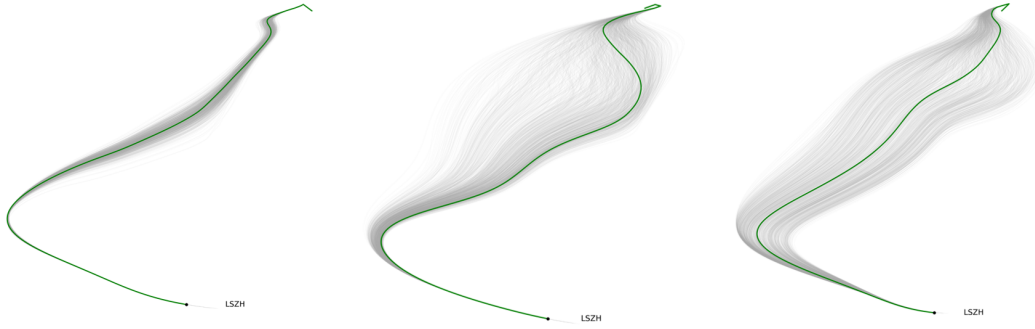


Figure C.12: Mode collapse of the TimeGAN model for batch sizes 16, 32, and 64

Trade-Off Between Training Iterations, Observed Noise, and Coverage - As the number of training iterations increased, the TimeGAN model demonstrated improved coverage of the original data distribution, leading to more comprehensive representation of trajectory patterns. However, this improvement came at the cost of increased noise, particularly concentrated further upstream in the generated trajectories. This phenomenon is attributed to the nature of aircraft trajectories, where variation is typically greater further from the airport and decreases as the trajectory approaches the airport. Consequently, extended training iterations amplified noise predominantly in the upstream segments of the trajectories, where the model's capacity to capture finer details was challenged. To mitigate this issue, smoothing filters were applied during post-processing, as detailed in the paper, to reduce the noise in the upstream regions and enhance the overall quality and coherence of the synthetic trajectories. [Figure C.13](#) also depicts the larger step size in geographic location of the trajectories further upstream - an expected consequence of a fixed sampling rate with a monotonously decreasing airspeed.

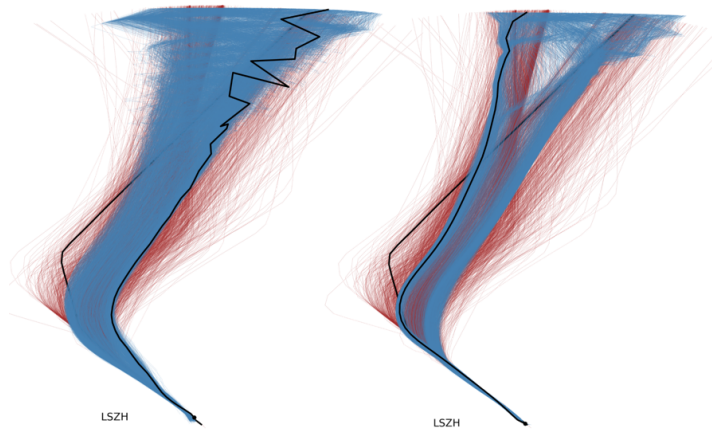


Figure C.13: Trade-off between the trajectory distribution coverage and noise, for 5,000 (left) and 1,000 (right) training iterations

D

Software User Guide & Repository Structure

This appendix provides the reader and potential user of the software with instructions in the form of a user guide and code repository structure outline. [Appendix D.1](#) presents an overview of the three modules contained in the repository. The three modules resemble the main parts of an ML workflow, each consisting of multiple Python files with bundled functionality. These are briefly outlined alongside the required input and corresponding outputs. [Appendix D.2](#) presents a more detailed explanation in the form of a user guide. Following this should allow the reader to quickly experiment with any of the available datasets and model configurations. This appendix has been written to promote transparency and encourage further investigation of the TimeGAN model behavior in this context.

D.1. Code Repository Structure

The Python files containing all functionality are bundled according to three modules, reflecting the key steps in any ML project workflow. These are the following: Module I (data loading, preprocessing, labeling and clustering), Module II (TimeGAN model training, post-processing and plotting) and Module III (evaluation framework - data diversity, fidelity and usefulness). [Figure D.1](#) outlines to which Module each file belongs, and the inputs and outputs of the three Modules.

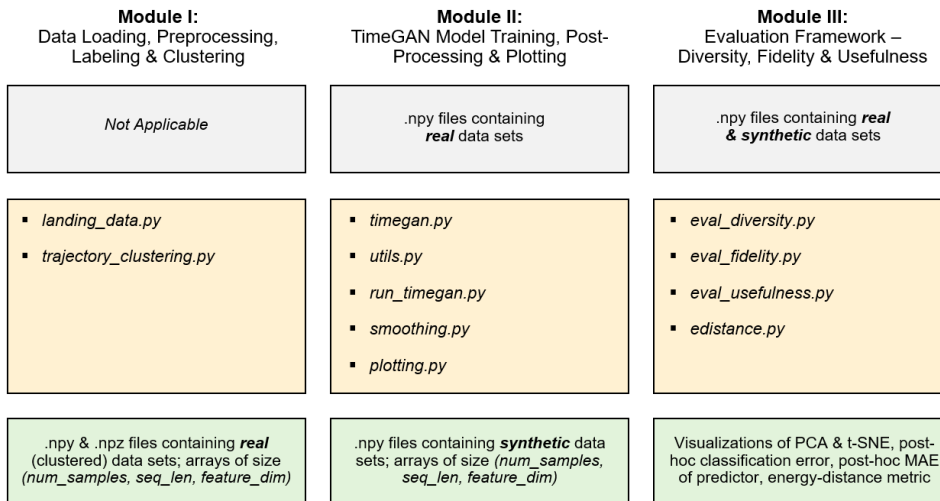


Figure D.1: Code Repository Structure

The key dependencies for this project are: *cartes*, *geopandas*, *itertools*, *matplotlib*, *numpy*, *os*, *pandas*, *random*, *scipy*, *shapely*, *scikit-learn*, *tensorflow 1.x*, *tqdm*, *traffic* and *tslearn*.

D.2. Software User Guide

For the user guide, the same Module structuring is followed (outlined above). For the most granular information on individual functions and variables, the reader is recommended to inspect the files. All functions have detailed doc-strings and the code in general contains ample comments.

D.2.1. Module I: Data Loading, Preprocessing, Labeling & Clustering

These scripts are designed for preprocessing and labeling Zurich (LSZH) aircraft landing trajectories data from October 1, 2019, to November 30, 2019. It provides several functions to clean, process, visualize, and analyze aircraft landing trajectories. It supports identifying different landing scenarios, such as go-arounds, holding patterns, and detecting arrival runways. The user is in control of the sampling rate, complexity threshold, and what type of data files to generate (and save) in .npz/.npz file formats.

- **Preprocess Landing Data:** Load and preprocess Zurich (LSZH) landing trajectories from October 1, 2019, to November 30, 2019.
- **Convert Timestamps:** Convert date-time columns to Unix timestamps and calculate elapsed time (since the start of a trajectory) for analysis.
- **Detect and Filter Events:** Identify go-arounds, holding patterns, and runway-specific landings (and assign labels accordingly).
- **Visualize Trajectories:** Plot aircraft trajectories and landing patterns on a map for visual analysis.
- **Data Formatting:** Prepare trajectory data for machine learning by converting to NumPy arrays and splitting into train/test sets.
- **Perform K-Means Clustering using DTW:** Performs k-means clustering on landing trajectories using Dynamic Time Warping (DTW) as the dedicated distance metric.
- **Elbow Method Heuristic:** Uses the elbow method to determine the optimal number of clusters.
- **Visualize Clusters Centers:** Visualizes the cluster centers or representative trajectories for each cluster.

D.2.2. Module II: TimeGAN Model Training, Post-Processing & Plotting

These scripts pertain to the training of the TimeGAN model, and the post-processing and visualization of the synthetic samples. The TimeGAN model implementation is taken from the main author's official repository (timegan.py and utils.py). The remaining files work with this model and the resulting output data.

- **TimeGAN Architecture:** Model architecture as created by the original authors. Key command inputs requiring specification are sequence length, module, hidden dimensions, number of layers, iterations and batch size.
- **Training TimeGAN:** Specify TimeGAN hyperparameter configurations and train the model to create synthetic aircraft landing trajectories from real data, then save the generated data and plot training losses.
- **Data Post-Processing:** Apply smoothing filters to the first three features (longitude, latitude, altitude) of trajectory data and save the processed data to separate files.
- **Synthetic Data Visualization:** Plot 2D (altitude profiles) and 3D (geographic coordinate profiles) visualizations of aircraft trajectories for comparison between different datasets, including real and synthetic data.

D.2.3. Module III: Evaluation Framework - Diversity, Fidelity & Usefulness

These scripts collectively evaluate the quality and usefulness of synthetic data compared to real data. They assess the similarity and diversity between datasets, train models to differentiate between real and synthetic data, and measure the effectiveness of synthetic data in predictive tasks. The evaluations are performed using various methods, including dimensionality reduction, model training, and statistical metrics. In all cases, the required inputs are real and synthetic datasets in the required array format of (number of samples, sequence length, feature dimension).

- **Data Diversity:** Assess the diversity of synthetic data by comparing its distribution to real data using dimensionality reduction techniques like PCA and t-SNE. It visualizes how well the synthetic data represents the original data distribution (optionally flattening the data arrays along the temporal or feature dimensions).
- **Data Fidelity:** Train a discriminator model to differentiate between original and synthetic data. Evaluate the model's performance using metrics such as accuracy, confusion matrix, and true positive/negative rates to determine how convincingly the synthetic data mimics the real data.
- **Data Usefulness:** Evaluates the usefulness of synthetic data by comparing the predictive performance of models trained on synthetic versus real data. Relies on an LSTM-based model for time series regression to measure and compare mean absolute errors on test data.
- **Energy Distance:** Calculates the energy distance metric to assess the similarity and diversity between true and generated data. Measure how close or different the distributions of the real and synthetic datasets are by computing energy distances across multiple samples.

All source files and datasets (both real and synthetic) are available in the online repository.

Bibliography

- [1] Tobias Albrecht, Robert Keller, Dominik Rebholz, and Maximilian Röglinger. Fake it till you make it: Synthetic data for emerging carsharing programs. *Transportation Research Part D: Transport and Environment*, 127:104067, 2024. ISSN 1361-9209. doi: 10.1016/j.trd.2024.104067. URL <https://www.sciencedirect.com/science/article/pii/S1361920924000245>.
- [2] Erik Altman, Jovan Blanuša, Luc von Niederhäusern, Beni Egressy, Andreea Anghel, and Kubilay Atasü. Realistic synthetic financial transactions for anti-money laundering models. In *Advances in Neural Information Processing Systems*, volume 36, pages 29851–29874, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/5f38404edff6f3f642d6fa5892479c42-Paper-Datasets_and_Benchmarks.pdf.
- [3] Konstantin Artiouchine, Philippe Baptiste, and Christoph Dürr. Runway sequencing with holding patterns. *European Journal of Operational Research*, 189(3):1254–1266, September 2008. ISSN 0377-2217. doi: 10.1016/j.ejor.2006.06.076. URL <https://www.sciencedirect.com/science/article/pii/S0377221707005978>.
- [4] Ehsan Asadi, Jan Evler, Henning Preis, and Hartmut Fricke. Coping with uncertainties in predicting the aircraft turnaround time at airports. In *Operations Research Proceedings 2019*, pages 773–780. Springer International Publishing, September 2020. ISBN 978-3-030-48439-2. URL https://link.springer.com/chapter/10.1007/978-3-030-48439-2_94.
- [5] Samuel A. Assefa, Danial Dervovic, Mahmoud Mahfouz, Robert E. Tillman, Prashant Reddy, and Manuela Veloso. Generating synthetic data in finance: opportunities, challenges and pitfalls. In *Proceedings of the First ACM International Conference on AI in Finance*, ICAIF '20. Association for Computing Machinery, October 2021. ISBN 9781450375849. doi: 10.1145/3383455.3422554. URL <https://dl.acm.org/doi/10.1145/3383455.3422554>.
- [6] Rohit Babbar and Bernard Schölkopf. Data scarcity, robustness and extreme multi-label classification. *Machine Learning*, 108(8):1329–1351, March 2019. URL <https://sloanreview.mit.edu/article/the-real-deal-about-synthetic-data/>.
- [7] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271 [cs]*, March 2018. URL <https://arxiv.org/abs/1803.01271>.
- [8] Donald J. Berndt and James Clifford. Using dynamic time warping to find patterns in time series. In *KDD Workshop*, pages 359–370. AAAI Press, April 1994. ISBN 0-929280-73-3. URL <http://dblp.uni-trier.de/db/conf/kdd/kdd94.html#BerndtC94>.
- [9] Philippe C. Besse, Brendan Guillouet, Jean-Michel Loubes, and François Royer. Review and perspective for distance-based clustering of vehicle trajectories. *IEEE Transactions on Intelligent Transportation Systems*, 17(11):3306–3317, May 2016. doi: 10.1109/TITS.2016.2547641.
- [10] Fred Bryant and Paul Yarnold. Principal-component analysis and exploratory and confirmatory factor analysis. January 1995. URL <https://psycnet.apa.org/record/1995-97110-004>.
- [11] Simon Calder. Flight chaos that hit 700,000 passengers took 90 minutes to fix, report finds — independent.co.uk. <https://www.independent.co.uk/travel/news-and-advice/air-traffic-control-bank-holiday-flights-holiday-cancelled-b2512417.html>, 2024.
- [12] Paola Natalia Canas, Juan Diego Ortega, Marcos Nieto, and Oihana Otaegui. Virtual passengers for real car solutions: synthetic datasets. *arXiv preprint arXiv:2205.06556 [cs]*, May 2022. URL <https://arxiv.org/abs/2205.06556>.

- [13] Daniel Carvajal-Patiño and Raul Ramos-Pollán. Synthetic data generation with deep generative models to enhance predictive tasks in trading strategies. *Research in International Business and Finance*, 62:101747, December 2022. ISSN 0275-5319. doi: <https://doi.org/10.1016/j.ribaf.2022.101747>. URL <https://www.sciencedirect.com/science/article/pii/S0275531922001313>.
- [14] Leonardo Carvalho, Alice Sternberg, Leandro Maia Gonçalves, Ana Beatriz Cruz, Jorge A. Soares, Diego Brandão, Diego Carvalho, and Eduardo Ogasawara. On the relevance of data science for flight delay research: a systematic review. *arXiv preprint arXiv:1703.06118 [cs]*, March 2017. URL <https://arxiv.org/abs/1703.06118>.
- [15] Bin Chen, Xing Zhao, and Jin Wu. Evaluating prediction models for airport passenger throughput using a hybrid method. *Applied Sciences*, 13(4), 2023. ISSN 2076-3417. doi: 10.3390/app13042384. URL <https://www.mdpi.com/2076-3417/13/4/2384>.
- [16] Gang Chen. A simple text to video model via transformer. *arXiv preprint arXiv:2309.14683 [cs]*, September 2023. URL <https://arxiv.org/abs/2309.14683>.
- [17] Lei Chen and Raymond Ng. On the marriage of lp-norms and edit distance. In *Proceedings of the Thirtieth International Conference on Very Large Data Bases - Volume 30*, page 792803. VLDB Endowment, 2004. ISBN 0120884690. URL <https://dl.acm.org/doi/10.5555/1316689.1316758>.
- [18] Lei Chen, M. Tamer Özsu, and Vincent Oria. Robust and fast similarity search for moving object trajectories. In *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data*, page 491502. Association for Computing Machinery, 2005. ISBN 1595930604. doi: 10.1145/1066157.1066213. URL <https://dl.acm.org/doi/10.1145/1066157.1066213>.
- [19] Xinyu Chen, Jiajie Xu, Rui Zhou, Wei Chen, Junhua Fang, and Chengfei Liu. Trajvae: A variational autoencoder model for trajectory generation. *Neurocomputing*, 428:332–339, March 2021. ISSN 0925-2312. doi: 10.1016/j.neucom.2020.03.120. URL <https://www.sciencedirect.com/science/article/pii/S0925231220312017>.
- [20] Andrea Coletta, Sriram Gopalakrishnan, Daniel Borrajo, and Svitlana Vyetrenko. On the constrained time-series generation problem. *arXiv preprint arXiv:2307.01717 [cs]*, July 2023. URL <https://arxiv.org/abs/2307.01717>.
- [21] Yu Cui, Qing He, and Ling Bian. Generating a synthetic probabilistic daily activity-location schedule using large-scale, long-term and low-frequency smartphone gps data with limited activity information. *Transportation Research Part C: Emerging Technologies*, 132:103408, November 2021. ISSN 0968-090X. doi: 10.1016/j.trc.2021.103408. URL <https://www.sciencedirect.com/science/article/pii/S0968090X21004034>.
- [22] Ramon Dalmau and Gilles Gawinowski. Learning with confidence the likelihood of flight diversion due to adverse weather at destination. *IEEE Transactions on Intelligent Transportation Systems*, 24(5): 56155624, May 2023. ISSN 1524-9050. doi: 10.1109/TITS.2023.3235741. URL <https://doi.org/10.1109/TITS.2023.3235741>.
- [23] Ramon Dalmau and Gilles Gawinowski. The effectiveness of supervised clustering for characterising flight diversions due to weather. *Expert Systems with Applications*, 237:121652, March 2024. ISSN 0957-4174. doi: 10.1016/j.eswa.2023.121652. URL <https://www.sciencedirect.com/science/article/pii/S0957417423021541>.
- [24] Ayan Das, Yongxin Yang, Timothy Hospedales, Tao Xiang, and Yi-Zhe Song. Chirodiff: Modelling chi-rographic data with diffusion models. *arXiv preprint arXiv:2304.03785 [cs]*, April 2023. URL <https://arxiv.org/abs/2304.03785>.
- [25] Peter Davis, Roy Lay-Yee, and Janet Pearson. Using micro-simulation to create a synthesised data set and test policy options: The case of health service effects under demographic ageing. *Health Policy*, 97(2):267–274, 2010. ISSN 0168-8510. doi: 10.1016/j.healthpol.2010.05.014. URL <https://www.sciencedirect.com/science/article/pii/S0168851010001387>.

- [26] Gustavo H. de Rosa and João P. Papa. A survey on text generation using generative adversarial networks. *Pattern Recognition*, 119:108098, November 2021. ISSN 0031-3203. doi: 10.1016/j.patcog.2021.108098. URL <https://www.sciencedirect.com/science/article/pii/S0031320321002855>.
- [27] D. Delahaye, S. Puechmorel, P. Tsiotras, and E. Feron. Mathematical models for aircraft trajectory design: A survey. In *Air Traffic Management and Systems*, pages 205–247, 2014. ISBN 978-4-431-54475-3. URL https://link.springer.com/chapter/10.1007/978-4-431-54475-3_12.
- [28] Delft University of Technology, Sintef, Deep Blue, and Eurocontrol. D.2.1 state of the art. Technical report, Publications Office of the European Union, January 2024.
- [29] Abhyuday Desai, Cynthia Freeman, Zuhui Wang, and Ian Beaver. Timevae: A variational auto-encoder for multivariate time series generation. *arXiv preprint arXiv:2111.08095 [cs]*, November 2021. URL <https://arxiv.org/abs/2111.08095>.
- [30] Claudio Di Ciccio, Han van der Aa, Cristina Cabanillas, Jan Mendling, and Johannes Prescher. Detecting flight trajectory anomalies and predicting diversions in freight transportation. *Decision Support Systems*, 88:1–17, August 2016. ISSN 0167-9236. doi: 10.1016/j.dss.2016.05.004. URL <https://www.sciencedirect.com/science/article/pii/S016792361630077X>.
- [31] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929 [cs]*, October 2022. URL <https://arxiv.org/abs/2010.11929>.
- [32] A Eckstein. Data driven modeling for the simulation of converging runway operations. In *ICRAT 2020, 9th International Conference for Research in Air Transportation*, June 2010. URL https://www.icrat.org/previous-conferences/4th-international-conference/icrat_2010_book_of_abstracts.pdf.
- [33] Cristóbal Esteban, Stephanie L. Hyland, and Gunnar Rätsch. Real-valued (medical) time series generation with recurrent conditional gans. *arXiv preprint arXiv:1706.02633 [cs]*, June 2017. URL <https://arxiv.org/abs/1706.02633>.
- [34] Eurocontrol. Aviation data for research. URL <https://www.eurocontrol.int/dashboard/rnd-data-archive>.
- [35] Eurocontrol and Partners. Fly ai report - demystifying and accelerating ai in aviation/atm. Technical report, European Aviation High Level Group on AI, 2020. URL <https://www.eurocontrol.int/publication/fly-ai-report>.
- [36] European Parliament and of the Council. General data protection regulation (gdpr), 2016. URL <https://eur-lex.europa.eu/eli/reg/2016/679/oj>.
- [37] European University Institute of Florence. The impact of the general data protection regulation (gdpr) on artificial intelligence. Technical report, European Parliamentary Research Service, 2020. URL <https://www.eurocontrol.int/publication/fly-ai-report>.
- [38] *Federal Aviation Administration*. FAA, Washington, D.C., USA, Year of Publication. URL: <https://www.faa.gov>.
- [39] Otto Fabius and Joost R. van Amersfoort. Variational recurrent auto-encoders. *arXiv preprint arXiv:1412.6581 [cs]*, December 2014. URL <https://arxiv.org/abs/1412.6581>.
- [40] D. Foster and K.J. Friston. *Generative Deep Learning: Teaching Machines to Paint, Write, Compose, and Play*. O'Reilly Media, Incorporated, 2023. ISBN 9781098134181. URL <https://books.google.nl/books?id=tZxazwEACAAJ>.
- [41] Maurice Fréchet. Sur quelques points du calcul fonctionnel. *Rendiconti del Circolo Matematico di Palermo (1884-1940)*, 22:1–72, 1906. URL <https://api.semanticscholar.org/CorpusID:123251660>.

- [42] Nivesh Gadipudi, Irraivan Elamvazuthi, Mahindra Sanmugam, Lila Iznita Izhar, Tindyo Prasetyo, R Jegadeeshwaran, and Syed Saad Azhar Ali. Synthetic to real gap estimation of autonomous driving datasets using feature embedding. In *5th International Symposium in Robotics and Manufacturing Automation (ROMA)*, pages 1–5, 2022. doi: 10.1109/ROMA55875.2022.9915679. URL <https://ieeexplore.ieee.org/document/9915679>.
- [43] Aldren Gonzales, Guruprabha Guruswamy, and Scott R. Smith. Synthetic data in health care: A narrative review. *PLOS Digital Health*, 2(1):1–16, 01 2023. doi: 10.1371/journal.pdig.0000082. URL <https://journals.plos.org/digitalhealth/article?id=10.1371/journal.pdig.0000082>.
- [44] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., June 2014. URL https://proceedings.neurips.cc/paper_files/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf.
- [45] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. <http://www.deeplearningbook.org>.
- [46] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Commun. ACM*, 63(11):139144, October 2020. ISSN 0001-0782. doi: 10.1145/3422622. URL <https://dl.acm.org/doi/10.1145/3422622>.
- [47] Thomas Gräupl. Validation of simulated synthetic air traffic against recorded air traffic in germany. In *Integrated Communications Navigation and Surveillance (ICNS)*, pages 2063–2068, January 2017. URL <https://elib.dlr.de/111344/>.
- [48] Thomas Gräupl. Validation of simulated synthetic air traffic against recorded air traffic in germany. In *Integrated Communications Navigation and Surveillance (ICNS) Conference*, January 2017. URL <https://elib.dlr.de/111344/>.
- [49] Brendan Guillouet and James Van Hinsbergh. traj-dist. <https://github.com/bguillouet/traj-dist>, 2016.
- [50] Stephen Haben, Marcus Voss, and William Holderbaum. *Time Series Forecasting: Core Concepts and Definitions*. Springer International Publishing, 2023. ISBN 978-3-031-27852-5. doi: 10.1007/978-3-031-27852-5_5. URL https://link.springer.com/chapter/10.1007/978-3-031-27852-5_5.
- [51] Lintao Han, Yuchen Zhao, Hengyi Lv, Yisa Zhang, Hailong Liu, Guoling Bi, and Qing Han. Enhancing remote sensing image super-resolution with efficient hybrid conditional diffusion model. *Remote Sensing*, 15(13), June 2023. ISSN 2072-4292. doi: 10.3390/rs15133452. URL <https://www.mdpi.com/2072-4292/15/13/3452>.
- [52] F Hausdorff. *Grundz uge der mengenlehre*. 1914.
- [53] GaoYuan He, YongXiang Zhao, and ChuLiang Yan. Application of tabular data synthesis using generative adversarial networks on machine learning-based multiaxial fatigue life prediction. *International Journal of Pressure Vessels and Piping*, 199:104779, 2022. ISSN 0308-0161. doi: 10.1016/j.ijpvp.2022.104779. URL <https://www.sciencedirect.com/science/article/pii/S030801612200165X>.
- [54] Mikel Hernandez, Gorka Epelde, Ane Alberdi, Rodrigo Cilla, and Debbie Rankin. Synthetic data generation for tabular health records: A systematic review. *Neurocomputing*, 493:28–45, 2022. ISSN 0925-2312. doi: 10.1016/j.neucom.2022.04.053. URL <https://www.sciencedirect.com/science/article/pii/S0925231222004349>.
- [55] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/4c5bcfec8584af0d967f1ab10179ca4b-Paper.pdf.

- [56] Jacco M. Hoekstra and Joost Ellerbroek. Bluesky atc simulator project: an open data and open source approach. In *Proceedings of the 7th international conference on research in air transportation*, volume 131, page 132, 2016. URL <https://repository.tudelft.nl/record/uuid:d1131a90-f0ea-4489-a217-ad29987689a1>.
- [57] Felix Hoffmann, Ulrich Epple, Michael Schnell, and Uwe-Carsten Fiebig. Feasibility of Idacs1 cell planning in european airspace. In *2012 IEEE/AIAA 31st Digital Avionics Systems Conference (DASC)*, pages 1–17, October 2012. doi: 10.1109/DASC.2012.6383053. URL <https://ieeexplore.ieee.org/document/6383053>.
- [58] *International Civil Aviation Organization*. ICAO, Montreal, Canada, Year of Publication. URL: <https://www.icao.int>.
- [59] Damien Jacquemart and Jérôme Morio. Conflict probability estimation between aircraft with dynamic importance splitting. *Safety Science*, 51(1):94–100, 2013. ISSN 0925-7535. doi: <https://doi.org/10.1016/j.ssci.2012.05.010>. URL <https://www.sciencedirect.com/science/article/pii/S0925753512001300>.
- [60] Alexandre Jacquillat and Amedeo R. Odoni. Endogenous control of service rates in stochastic and dynamic queuing models of airport congestion. *Transportation Research Part E: Logistics and Transportation Review*, 73:133–151, January 2015. ISSN 1366-5545. doi: 10.1016/j.tre.2014.10.014. URL <https://www.sciencedirect.com/science/article/pii/S1366554514001872>.
- [61] S. Jansen. *Machine Learning for Algorithmic Trading: Predictive Models to Extract Signals from Market and Alternative Data for Systematic Trading Strategies with Python*. Expert Insight. Packt Publishing, 2020. ISBN 9781839217715. URL <https://books.google.nl/books?id=cki6zQEACAAJ>.
- [62] Stef Janssen, Arjan van den Berg, and Alexei Sharpanskykh. Agent-based vulnerability assessment at airport security checkpoints: A case study on security operator behavior. *Transportation Research Interdisciplinary Perspectives*, 5:100139, May 2020. ISSN 2590-1982. doi: 10.1016/j.trip.2020.100139. URL <https://www.sciencedirect.com/science/article/pii/S2590198220300506>.
- [63] Stef Janssen, Régis van der Sommen, Alexander Dilweg, and Alexei Sharpanskykh. Data-driven analysis of airport security checkpoint operations. *Aerospace*, 7(6), 2020. ISSN 2226-4310. doi: 10.3390/aerospace7060069. URL <https://www.mdpi.com/2226-4310/7/6/69>.
- [64] G. Jarry, N. Couellan, and D. Delahaye. On the use of generative adversarial networks for aircraft trajectory generation and atypical approach detection. In *Air Traffic Management and Systems IV*, pages 227–243, 2021. ISBN 978-981-33-4669-7. URL https://link.springer.com/chapter/10.1007/978-981-33-4669-7_13.
- [65] Gabriel Jarry, Daniel Delahaye, Florence Nicol, and Eric Féron. Aircraft Atypical Approach Detection using Functional Principal Component Analysis. In *SID 2018, 8th SESAR Innovations Days*, December 2018. URL <https://enac.hal.science/hal-01944595>.
- [66] Gabriel Jarry, Almoutar Hassoumi, Daniel Delahaye, and Christophe Hurter. Interactive trajectory modification and generation with fpca. In *ICRAT 2020, 9th International Conference for Research in Air Transportation*, June 2020. URL https://enac.hal.science/hal-02873429/file/Interactive_Trajectory_Modification_and_Generation_with_FPCA.pdf.
- [67] Seungho Jeon and Jung Taek Seo. A synthetic time-series generation using a variational recurrent autoencoder with an attention mechanism in an industrial control system. *Sensors*, 24(1), 2024. ISSN 1424-8220. doi: 10.3390/s24010128. URL <https://www.mdpi.com/1424-8220/24/1/128>.
- [68] Yifan Jiang, Shiyu Chang, and Zhangyang Wang. Transgan: Two pure transformers can make one strong gan, and that can scale up. In *Advances in Neural Information Processing Systems*, volume 34, pages 14745–14758, 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/7c220a2091c26a7f5e9f1cfb099511e3-Paper.pdf.
- [69] James Jordon, Lukasz Szpruch, Florimond Houssiau, Mirko Bottarelli, Giovanni Cherubin, Carsten Maple, Samuel N. Cohen, and Adrian Weller. Synthetic data – what, why and how? *arXiv preprint arXiv:2205.03257 [cs]*, May 2022. URL <https://arxiv.org/abs/2205.03257>.

- [70] Katikapalli Subramanyam Kalyan. A survey of gpt-3 family large language models including chatgpt and gpt-4. *Natural Language Processing Journal*, 6:100048, March 2024. ISSN 2949-7191. doi: 10.1016/j.nlp.2023.100048. URL <https://www.sciencedirect.com/science/article/pii/S2949719123000456>.
- [71] Maryam Kamgarpour, Vera Dadok, and Claire Tomlin. Trajectory generation for aircraft subject to dynamic weather uncertainty. In *49th IEEE Conference on Decision and Control (CDC)*, pages 2063–2068, December 2010. doi: 10.1109/CDC.2010.5717889. URL <https://ieeexplore.ieee.org/document/5717889>.
- [72] Dina Kampouraki, Robert Riemann, Vitor Bernardo, Maria Enescu, Marek Jessen, Kostantina Vemou, and Stefano Leucci. Techsonar report. Technical report, European Data Protection Supervisor, December 2020. URL https://www.edps.europa.eu/data-protection/our-work/publications/reports/techsonar-2021-2022-report_en.
- [73] Afaq Khattak, Jianping Zhang, Pak-Wai Chan, Feng Chen, and Caroline Mongina Matara. Ai-supported estimation of safety critical wind shear-induced aircraft go-around events utilizing pilot reports. *Heliyon*, 10(7):e28569, January 2024. ISSN 2405-8440. doi: 10.1016/j.heliyon.2024.e28569. URL <https://www.sciencedirect.com/science/article/pii/S2405844024046000>.
- [74] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114 [cs]*, December 2013. URL <https://arxiv.org/abs/1312.6114>.
- [75] Diederik P. Kingma and Max Welling. An introduction to variational autoencoders. *Found. Trends Mach. Learn.*, 12(4):307392, November 2019. ISSN 1935-8237. doi: 10.1561/22000000056. URL <https://dl.acm.org/doi/10.1561/22000000056>.
- [76] Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. Diffwave: A versatile diffusion model for audio synthesis. *arXiv preprint arXiv:2009.09761 [cs]*, September 2020. URL <https://arxiv.org/abs/2009.09761>.
- [77] Timoth  Krauth, J r me Morio, Xavier Olive, Benoit Figuet, and Raphael Monstein. Synthetic aircraft trajectories generated with multivariate density models. *Engineering Proceedings*, 13(1), 2021. ISSN 2673-4591. doi: 10.3390/engproc2021013007. URL <https://www.mdpi.com/2673-4591/13/1/7>.
- [78] Timoth  Krauth, Adrien Lafage, J r me Morio, Xavier Olive, and Manuel Waltert. Deep generative modelling of aircraft trajectories in terminal maneuvering areas. *Machine Learning with Applications*, 11:100446, March 2023. ISSN 2666-8270. doi: 10.1016/j.mlwa.2022.100446. URL <https://www.sciencedirect.com/science/article/pii/S2666827022001219>.
- [79] Siddique Latif, Abdullah Shahid, and Junaid Qadir. Generative emotional ai for speech emotion recognition: The case for synthetic emotional speech augmentation. *Applied Acoustics*, 210:109425, July 2023. ISSN 0003-682X. doi: 10.1016/j.apacoust.2023.109425. URL <https://www.sciencedirect.com/science/article/pii/S0003682X23002232>.
- [80] Daesoo Lee, Sara Malacarne, and Erlend Aune. Vector quantized time series generation with a bidirectional prior model. In Francisco Ruiz, Jennifer Dy, and Jan-Willem van de Meent, editors, *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*, volume 206 of *Proceedings of Machine Learning Research*, pages 7665–7693. PMLR, 25–27 Apr 2023. URL <https://proceedings.mlr.press/v206/lee23d.html>.
- [81] Daniel Leiker, Ashley Ricker Gyllen, Ismail Eldesouky, and Mutlu Cukurova. Generative ai for learning: Investigating the potential of learning videos with synthetic virtual instructors. In *Artificial Intelligence in Education. Posters and Late Breaking Results, Workshops and Tutorials, Industry and Innovation Tracks, Practitioners, Doctoral Consortium and Blue Sky*, pages 523–529. Springer Nature Switzerland, 2023. ISBN 978-3-031-36336-8.
- [82] Hongming Li, Shujian Yu, and Jose Principe. Causal recurrent variational autoencoder for medical time series generation. In *Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence and Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence and Thirteenth Symposium on Educational Advances in Artificial Intelligence*. AAAI Press, 2023. ISBN 978-1-57735-880-0. doi: 10.1609/aaai.v37i7.26031. URL <https://doi.org/10.1609/aaai.v37i7.26031>.

- [83] Huazhi Li, Toon Haer, Anaïs Couasnon, Alejandra R. Enríquez, Sanne Muis, and Philip J. Ward. A spatially-dependent synthetic global dataset of extreme sea level events. *Weather and Climate Extremes*, 41:100596, September 2023. ISSN 2212-0947. doi: 10.1016/j.wace.2023.100596. URL <https://www.sciencedirect.com/science/article/pii/S221209472300049X>.
- [84] Xiaomin Li, Vangelis Metsis, Huangyingrui Wang, and Anne Hee Hiong Ngu. Tts-gan: A transformer-based time-series generative adversarial network. In *Artificial Intelligence in Medicine: 20th International Conference on Artificial Intelligence in Medicine, AIME 2022, Halifax, NS, Canada, June 1417, 2022, Proceedings*, page 133143, June 2022. ISBN 978-3-031-09341-8. doi: 10.1007/978-3-031-09342-5_13. URL https://dl.acm.org/doi/10.1007/978-3-031-09342-5_13.
- [85] Xiaomin Li, Anne Hee Hiong Ngu, and Vangelis Metsis. Tts-cgan: A transformer time-series conditional gan for biosignal data augmentation. *arXiv preprint arXiv:2206.13676 [cs]*, June 2022. URL <https://arxiv.org/abs/2206.13676>.
- [86] Shujian Liao, Hao Ni, Lukasz Szpruch, Magnus Wiese, Marc Sabate-Vidales, and Baoren Xiao. Conditional sig-wasserstein gans for time series generation. *arXiv preprint arXiv:2006.05421 [cs]*, June 2020. URL <https://arxiv.org/abs/2006.05421>.
- [87] Haksoo Lim, Minjung Kim, Sewon Park, and Noseong Park. Regular time-series generation using sgm. *arXiv preprint arXiv:2301.08518 [cs]*, January 2023. URL <https://arxiv.org/abs/2301.08518>.
- [88] Lequan Lin, Zhengkun Li, Ruikun Li, Xuliang Li, and Junbin Gao. Diffusion models for time series applications: A survey. *arXiv preprint arXiv:2305.00624 [cs]*, May 2013. URL <https://arxiv.org/abs/2305.00624>.
- [89] Chunli Liu, Carmine Ventre, and Maria Polukarov. Synthetic data augmentation for deep reinforcement learning in financial trading. In *Proceedings of the Third ACM International Conference on AI in Finance, ICAIF '22*, page 343351, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450393768. doi: 10.1145/3533271.3561704. URL <https://doi.org/10.1145/3533271.3561704>.
- [90] Yingzhou Lu, Minjie Shen, Huazheng Wang, Xiao Wang, Capucine van Rechem, and Wenqi Wei. Machine learning for synthetic data generation: A review. *arXiv preprint arXiv:2302.0406 [cs]*, February 2023. URL <https://arxiv.org/abs/2302.04062>.
- [91] Fernando Lucini. The real deal about synthetic data. *MIT Sloan Management Review*, 2022. URL <https://sloanreview.mit.edu/article/the-real-deal-about-synthetic-data/>.
- [92] Mingchuan Luo, Michael Schultz, Hartmut Fricke, Bruno Desart, Floris Herrema, and Rocío Barragán Montes. Agent-based simulation for aircraft stand operations to predict ground time using machine learning. In *2021 IEEE/AIAA 40th Digital Avionics Systems Conference (DASC)*, pages 1–8, October 2021. doi: 10.1109/DASC52595.2021.9594325. URL <https://ieeexplore.ieee.org/document/9594325>.
- [93] Rogelio A. Mancisidor, Michael Kampffmeyer, Kjersti Aas, and Robert Jenssen. Discriminative multi-modal learning via conditional priors in generative models. *Neural Networks*, 169:417–430, January 2024. ISSN 0893-6080. doi: 10.1016/j.neunet.2023.10.048. URL <https://www.sciencedirect.com/science/article/pii/S089360802300610X>.
- [94] Shaun McKnight, S. Gareth Pierce, Ehsan Mohseni, Christopher MacKinnon, Charles MacLeod, Tom O'Hare, and Charalampos Loukas. A comparison of methods for generating synthetic training data for domain adaption of deep learning models in ultrasonic non-destructive evaluation. *NDT E International*, 141:102978, 2024. ISSN 0963-8695. doi: 10.1016/j.ndteint.2023.102978. URL <https://www.sciencedirect.com/science/article/pii/S0963869523001937>.
- [95] Olof Mogren. C-rnn-gan: Continuous recurrent neural networks with adversarial training. *arXiv preprint arXiv:1611.09904 [cs]*, November 2016. URL <https://arxiv.org/abs/1611.09904>.

- [96] Andrew Ng and Michael Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *Advances in Neural Information Processing Systems*, volume 14. MIT Press, 2001. URL https://proceedings.neurips.cc/paper_files/paper/2001/file/7b7a53e239400a13bd6be6c91c4f6c4e-Paper.pdf.
- [97] Angela Nuic, Damir Poles, and Vincent Mouillet. Bada: An advanced aircraft performance model for present and future atm systems. *International Journal of Adaptive Control and Signal Processing*, 24(10):850–866, May 2010. doi: 10.1002/acs.1176. URL <https://onlinelibrary.wiley.com/doi/abs/10.1002/acs.1176>.
- [98] Gyutaek Oh and Jong Chul Ye. Chapter 8 - medical image synthesis and reconstruction using generative adversarial networks. In *Deep Learning for Medical Image Analysis (Second Edition)*, The MICCAI Society book Series, pages 225–245. Academic Press, second edition edition, 2024. ISBN 978-0-323-85124-4. doi: 10.1016/B978-0-32-385124-4.00018-0. URL <https://www.sciencedirect.com/science/article/pii/B9780323851244000180>.
- [99] Xavier Olive. traffic, a toolbox for processing and analysing air traffic data. *Journal of Open Source Software*, 4:1518, 2019. ISSN 2475-9066. doi: 10.21105/joss.01518.
- [100] Xavier Olive, Junzi Sun, Mayara Condé Rocha Murça, and Timothe Krauth. A framework to evaluate aircraft trajectory generation methods. In *14th USA/Europe Air Traffic Management Seminar, ATM2021*, September 2021. URL <https://research.tudelft.nl/en/publications/a-framework-to-evaluate-aircraft-trajectory-generation-methods>.
- [101] Laura Pahren, Paul Thomas, Xiaodong Jia, and Jay Lee. A novel method in intelligent synthetic data creation for machine learning-based manufacturing quality control. *IFAC-PapersOnLine*, 55(19):73–78, 2022. ISSN 2405-8963. doi: 10.1016/j.ifacol.2022.09.186. URL <https://www.sciencedirect.com/science/article/pii/S2405896322014008>.
- [102] Leo L. Pipino, Yang W. Lee, and Richard Y. Wang. Data quality assessment. *Commun. ACM*, 45(4):211218, April 2002. ISSN 0001-0782. doi: 10.1145/505248.506010. URL <https://dl.acm.org/doi/10.1145/505248.506010>.
- [103] Vladimir Polyakov, Aleksandr Mezhenin, Vladimir Vereskun, Anatoly Korobeynikov, and Vasilii Shapovalov. Synthetic datasets for testing video security systems. In *Proceedings of the Fifth International Scientific Conference “Intelligent Information Technologies for Industry” (IITI’21)*, pages 672–680. Springer International Publishing, 2022. ISBN 978-3-030-87178-9.
- [104] Matthias Poppe and Jörg Buxbaum. Clustering climb profiles for vertical trajectory analysis. In *Proceedings of the 10th SESAR Innovation Days*, December 2020. URL https://www.sesarju.eu/sites/default/files/documents/sid/2020/papers/SIDs_2020_paper_66red.pdf.
- [105] Mayara Condé Rocha Murça and McWilliam de Oliveira. A data-driven probabilistic trajectory model for predicting and simulating terminal airspace operations. In *2020 AIAA/IEEE 39th Digital Avionics Systems Conference (DASC)*, pages 1–7, October 2020. doi: 10.1109/DASC50938.2020.9256644. URL <https://ieeexplore.ieee.org/document/9256644>.
- [106] Yuji Roh, Geon Heo, and Steven Euijong Whang. A survey on data collection for machine learning: A big data - ai integration perspective. *IEEE Transactions on Knowledge and Data Engineering*, 33(4):1328–1347, April 2021. ISSN 1558-2191. doi: 10.1109/TKDE.2019.2946162. URL <https://ieeexplore.ieee.org/document/8862913>.
- [107] C.-H. Rokitansky, M. Ehammer, and Th. Graupl. Newsky - building a simulation environment for an integrated aeronautical network architecture. In *2007 IEEE/AIAA 26th Digital Avionics Systems Conference*, pages 4.B.4–1–4.B.4–11, 2007. doi: 10.1109/DASC.2007.4391905. URL <https://ieeexplore.ieee.org/document/4391905>.
- [108] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *arXiv preprint arXiv:1505.04597 [cs]*, May 2015. URL <https://arxiv.org/abs/1505.04597>.

- [109] Daniil Ryabko. *Universal Time-Series Forecasting with Mixture Predictors*. Springer Cham, 2020. doi: 10.1007/978-3-030-54304-4. URL <https://link.springer.com/book/10.1007/978-3-030-54304-4>.
- [110] Matthias Schäfer, Martin Strohmeier, Vincent Lenders, Ivan Martinovic, and Matthias Wilhelm. Bringing up opensky: a large-scale ads-b sensor network for research. In *Proceedings of the 13th International Symposium on Information Processing in Sensor Networks*, IPSN '14, page 8394. IEEE Press, April 2014. ISBN 9781479931460.
- [111] Kristian Schultz, Saptarshi Bej, Waldemar Hahn, Markus Wolfien, Prashant Srivastava, and Olaf Wolkenhauer. Convgen: A convex space learning approach for deep-generative oversampling and imbalanced classification of small tabular datasets. *Pattern Recognition*, 147:110138, 2024. ISSN 0031-3203. doi: 10.1016/j.patcog.2023.110138. URL <https://www.sciencedirect.com/science/article/pii/S003132032300835X>.
- [112] Michael Schultz and Hartmut Fricke. Improving aircraft turnaround reliability. In *Proceedings of the Third International Conference on Research in Air Transportation*, pages 335–343, 06 2008. URL https://www.researchgate.net/publication/263038959_Improving_Aircraft_Turnaround_Reliability.
- [113] SESAR Joint Undertaking. Automation in air traffic management: Long-term vision and initial research roadmap. Technical report, Publications Office of the European Union, 2020. URL <https://www.sesarju.eu/automation>.
- [114] SESAR Joint Undertaking. European atm master plan - digitalising europe's aviation infrastructure. Technical report, Publications Office of the European Union, 2020. URL <https://www.sesarju.eu/masterplan2020>.
- [115] SESAR Joint Undertaking. Sesar innovation pipeline - air traffic management research and innovation. Technical report, Publications Office of the European Union, 2023. URL <https://www.sesarju.eu/innovation-pipeline>.
- [116] Valerie Sessions and Marco Valtorta. The effects of data quality on machine learning algorithms. In *Proceedings of the 11th International Conference on Information Quality, MIT, Cambridge, MA, USA, November 10-12, 2006*, pages 485–498. MIT, 2006. URL <http://mitiq.mit.edu/iciq/iqdownload.aspx?ICIQYear=2006&File=The+Effects+of+Data+Quality+on+Machine+Learning+Algorithms.pdf>.
- [117] Sakib Shahriar. Gan computers generate arts? a survey on visual arts, music, and literary text generation using generative adversarial network. *Displays*, 73:102237, July 2022. ISSN 0141-9382. doi: 10.1016/j.displa.2022.102237. URL <https://www.sciencedirect.com/science/article/pii/S0141938222000658>.
- [118] M Sklar. Fonctions de répartition à n dimensions et leurs marges. *Publications de l'Institut de Statistique de l'Université de Paris*, 1959.
- [119] Junzi Sun, Jacco M. Hoekstra, and Joost Ellerbroek. Openap: An open-source aircraft performance model for air transportation studies and simulations. *Aerospace*, 7(8), July 2020. ISSN 2226-4310. doi: 10.3390/aerospace7080104. URL <https://www.mdpi.com/2226-4310/7/8/104>.
- [120] Siao Sun, Fusheng Wang, Sina Rashidian, Tahsin Kurc, Kayley Abell-Hart, Janos Hajagos, Wei Zhu, Mary Saltz, and Joel Saltz. Generating longitudinal synthetic ehr data with recurrent autoencoders and generative adversarial networks. page 153165, Berlin, Heidelberg, 2021. Springer-Verlag. ISBN 978-3-030-93662-4. doi: 10.1007/978-3-030-93663-1_12. URL https://doi.org/10.1007/978-3-030-93663-1_12.
- [121] Gabor Székely and Maria Rizzo. Testing for equal distributions in high dimension. *InterStat*, 5, November 2004. URL https://www.researchgate.net/publication/228918499_Testing_for_equal_distributions_in_high_dimension.

- [122] Doroteja Timotic and Fedja Netjasov. Automation in air traffic control: Trust, teamwork, resilience, safety. *Transportation Research Procedia*, 65:13–23, November 2022. ISSN 2352-1465. doi: 10.1016/j.trpro.2022.11.003. URL <https://www.sciencedirect.com/science/article/pii/S2352146522006706>.
- [123] Jakub Tomczak and Max Welling. Vae with a vampprior. In Amos Storkey and Fernando Perez-Cruz, editors, *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, volume 84 of *Proceedings of Machine Learning Research*, pages 1214–1223. PMLR, 09–11 Apr 2018. URL <https://proceedings.mlr.press/v84/tomczak18a.html>.
- [124] SESAR Joint Undertaking. Techniques to determine trajectory uncertainty and modelling. Technical report, Publications Office of the European Union, 2018. URL <https://ec.europa.eu/research/participants/documents/downloadPublic?documentIds=080166e5b893f53e&appId=PPGMS>.
- [125] U.S. Department of Transportation. Bureau of transportation statistics. URL <https://www.bts.gov/>.
- [126] Vibeke Binz Vallevik, Aleksandar Babic, Serena E. Marshall, Severin Elvatun, Helga M.B. Brøgger, Sharmini Alagaratnam, Bjørn Edwin, Narasimha R. Veeraragavan, Anne Kjersti Befring, and Jan F. Nygård. Can i trust my fake data a comprehensive quality assessment framework for synthetic tabular data in healthcare. *International Journal of Medical Informatics*, 185:105413, 2024. ISSN 1386-5056. doi: 10.1016/j.ijmedinf.2024.105413. URL <https://www.sciencedirect.com/science/article/pii/S1386505624000765>.
- [127] Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/7a98af17e63a0ac09ce2e96d03992fbc-Paper.pdf.
- [128] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605, August 2008. URL <http://jmlr.org/papers/v9/vandermaten08a.html>.
- [129] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.
- [130] M. Vlachos, G. Kollios, and D. Gunopulos. Discovering similar multidimensional trajectories. In *Proceedings 18th International Conference on Data Engineering*, pages 673–684, February 2002. doi: 10.1109/ICDE.2002.994784. URL <https://ieeexplore.ieee.org/document/994784>.
- [131] M.C. Waller, J.G. Rigopoulos, D.R. Blackman, and T.F. Berreen. Considerations in the application of dynamic programming to optimal aircraft trajectory generation. In *IEEE Conference on Aerospace and Electronics*, pages 574–579 vol.2, 1990. doi: 10.1109/NAECON.1990.112828. URL <https://ieeexplore.ieee.org/document/112828>.
- [132] Teng Wang, Yufan Zheng, and Haiwen Xu. A review of flight delay prediction methods. In *2022 2nd International Conference on Big Data Engineering and Education (BDEE)*, pages 135–141, August 2022. doi: 10.1109/BDEE55929.2022.00029. URL <https://ieeexplore.ieee.org/document/9980875>.
- [133] Gerald Woo, Chenghao Liu, Akshat Kumar, Caiming Xiong, Silvio Savarese, and Doyen Sahoo. Unified training of universal time series forecasting transformers. *arXiv preprint arXiv:2402.02592 [cs]*, February 2024. URL <https://arxiv.org/abs/2402.02592>.
- [134] Zhisheng Xiao, Karsten Kreis, and Arash Vahdat. Tackling the generative learning trilemma with denoising diffusion gans. *arXiv preprint arXiv:2112.07804 [cs]*, December 2021. URL <https://arxiv.org/abs/2112.07804>.
- [135] Zhisheng Xiao, Karsten Kreis, and Arash Vahdat. Tackling the generative learning trilemma with denoising diffusion gans. *arXiv preprint arXiv:2112.07804 [cs]*, December 2022. URL <https://arxiv.org/abs/2112.07804>.

- [136] Bilgi Yilmaz and Ralf Korn. Synthetic demand data generation for individual electricity consumers : Generative adversarial networks (gans). *Energy and AI*, 9:100161, August 2022. ISSN 2666-5468. doi: <https://doi.org/10.1016/j.egyai.2022.100161>. URL <https://www.sciencedirect.com/science/article/pii/S2666546822000209>.
- [137] Huaquan Ying, Rafael Sacks, and Amir Degani. Synthetic image data generation using bim and computer graphics for building scene understanding. *Automation in Construction*, 154:105016, October 2023. ISSN 0926-5805. doi: 10.1016/j.autcon.2023.105016. URL <https://www.sciencedirect.com/science/article/pii/S0926580523002765>.
- [138] Jinsung Yoon, Daniel Jarrett, and Mihaela van der Schaar. Time-series generative adversarial networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/c9efe5f26cd17ba6216bbe2a7d26d490-Paper.pdf.
- [139] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. Seqgan: Sequence generative adversarial nets with policy gradient. *Proceedings of the AAAI Conference on Artificial Intelligence*, 31(1), February 2017. doi: 10.1609/aaai.v31i1.10804. URL <https://ojs.aaai.org/index.php/AAAI/article/view/10804>.
- [140] Xinyu Yuan and Yan Qiao. Diffusion-ts: Interpretable diffusion for general time series generation. *arXiv preprint arXiv:2403.01742 [cs]*, March 2024. URL <https://arxiv.org/abs/2403.01742>.
- [141] Marta Zaniolo, Sarah Fletcher, and Meagan Mauter. Find: A synthetic weather generator to control drought frequency, intensity, and duration. *Environmental Modelling Software*, 172:105927, January 2024. ISSN 1364-8152. doi: <https://doi.org/10.1016/j.envsoft.2023.105927>. URL <https://www.sciencedirect.com/science/article/pii/S1364815223003134>.
- [142] Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander J. Smola. *Dive into Deep Learning*. Cambridge University Press, 2023. <https://D2L.ai>.
- [143] Yuanshao Zhu, Yongchao Ye, Ying Wu, Xiangyu Zhao, and James Yu. Synmob: Creating high-fidelity synthetic gps trajectory dataset for urban mobility analysis. In *Advances in Neural Information Processing Systems*, volume 36, pages 22961–22977, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/4786c0d1b9687a841bc579b0b8b01b8e-Paper-Datasets_and_Benchmarks.pdf.