

A Controlled Benchmark of Generative Models for ADS-B Aircraft Trajectories

Ruolan Ren,^{*,1} Tian Huang,¹ Ruixin Wang ¹ Wei Cong,² Christophe Hurter ^{3,4} and Xavier Olive ^{5,3}

¹Civil Aviation University of China, Tianjin, China

²Feeyo Technology Co., Ltd.

³Fédération ENAC ISAE-SUPAERO ONERA, Université de Toulouse, France

⁴Université de Toulouse, ENAC, Toulouse, France

⁵ONERA DTIS, Université de Toulouse, France

*Corresponding author: rowanren@qq.com

Abstract

Data-driven aircraft trajectory generation supports simulation and safety assessment, but resemblance to recorded traffic does not ensure physical plausibility. Differences in datasets and evaluation protocols also make published models difficult to compare. We present a controlled benchmark using OpenSky arrival trajectories at Zurich and Tianjin international airports. We examine latent sampling as a generation problem distinct from reconstructing observed trajectories. The comparison includes a temporal convolutional variational autoencoder with a learned VampPrior, direct diffusion, and latent diffusion through the same frozen decoder. A common experimental setup controls data preparation and documents training and sampling. Evaluation combines geographic and feature-profile comparisons with aircraft-performance-based replay. VampPrior captures the main arrival flows more convincingly in the visual comparison, whereas direct diffusion yields replay discrepancies closer to the real-data baseline. The findings distinguish the quality of a learned representation from that of its sampling mechanism, and show that geographic similarity and replayability can favour different generators.

Keywords: aircraft trajectory generation, ADS-B, variational autoencoders, diffusion models

Abbreviations: JOAS: Journal of Open Aviation Science, ATM: Air Traffic Management, ADS-B: Automatic Dependent Surveillance–Broadcast

1. Introduction

Aircraft trajectory generation produces time-indexed sequences of aircraft states for applications including airspace design, fast-time simulation, and Monte Carlo safety assessment [1]. Model-driven approaches combine equations of motion with aircraft-performance models, thereby enforcing explicit physical constraints [2]. Operational variability must then be introduced through model inputs or additional assumptions, and may be difficult to specify exhaustively [3]. Data-driven approaches instead estimate the joint distribution of recorded trajectories and can reproduce the variability observed in surveillance data [3, 4]. However, statistical similarity does not guarantee that a generated trajectory is physically or operationally plausible; both aspects must be evaluated separately [5].

Data-driven aircraft trajectory generation has progressed from multivariate density models [1, 4] to deep latent-variable models such as Variational Autoencoders (VAE) [3]. Physics-guided Neural Ordinary Differential Equations provide a complementary approach by incorporating domain knowledge into the generation of vertical flight profiles from open ADS-B data [6]. More recently, Diffusion Models (DM) have been applied to terminal-area trajectory generation [7]. Then, Flow Matching (FM) [8] has also been implemented for the generation of complete aircraft trajectories

[9]. The studies cited above evaluate their proposed approaches using different datasets, trajectory representations, preprocessing pipelines, and evaluation criteria. Consequently, it remains difficult to determine whether reported differences originate from the generative paradigm itself or from the experimental setting.

We address this gap through a controlled benchmark using ADS-B data from the OpenSky Network [10]. The emphasis is on reproducibility: documenting how trajectories are prepared and how models are trained and sampled, rather than comparing published scores obtained under different experimental settings. We compare a TCVAE with a learned VampPrior, diffusion in trajectory space, and diffusion in the latent space of that TCVAE. This comparison focuses on a practical limitation of latent-variable trajectory models: successful reconstruction does not by itself specify a suitable distribution from which to generate new latent vectors.

All model configurations use a common preprocessing pipeline, trajectory representation, and training, validation, and test partitions. Following the evaluation framework of Olive *et al.* [5], we compare geographic patterns and groundspeed and altitude profiles. An illustrative latent-space case examines irregular generation. Simulator replay at Zurich assesses route-following under an aircraft-performance model.

The paper is structured as follows. Section 2 reviews existing trajectory-generation methods. Section 3 describes the datasets and shared representation. Section 4 introduces the models and their implementation. Section 5 presents the comparison and examines the challenges of sampling in a learned latent space. Section 6 provides concluding remarks.

2. Literature review

Trajectory generation was first conducted using model-driven generation methods that require flight-dynamics equations and expert knowledge [11, 12]. These methods use physical equations to guarantee the physical realism of aircraft movement, but they require many precise aircraft and environmental characteristics as inputs and have difficulty capturing uncertainties during flight. With ADS-B providing increasing amounts of trajectory data in recent years, data-driven methods have received increasing attention because they can learn operational patterns directly from observed trajectories.

Many early data-driven studies used classical statistical models to learn the distribution of trajectory data. For instance, Barratt *et al.* [13] proposed a cluster-based Gaussian mixture model (GMM), which enabled accurate inference and realistic trajectory generation. Krauth *et al.* [1] compared multivariate normal distributions, Gaussian mixture models, and vine copulas for generating two-dimensional go-around flows at Zurich Airport. These methods do not require precise physical equations or detailed physical states of aircraft, but estimating complex dependencies becomes increasingly difficult as trajectory dimensionality increases.

Deep generative modelling (DGM), which has been successfully applied in various fields, provides a way to handle high-dimensional data distributions and uncertainty, making it well suited to the characteristics of aircraft trajectory data. Jarry *et al.* [14] explored Generative Adversarial Networks (GANs) for aircraft trajectory generation to learn the probability distribution of real approach flight paths and identify abnormal trajectories that do not follow this distribution. Wijnands *et al.* [15] used TimeGAN [16] to generate trajectories during the landing phase. It was capable of generating trajectories from clustered data; however, its performance was limited in sparse parts of the trajectory distribution, such as go-arounds and holding patterns. Although these studies show that GANs can be used for aircraft trajectory generation, their adversarial training process can be unstable, and their generation process has limited interpretability, making them less suitable for controllable

trajectory generation.

Variational Autoencoders (VAEs) provide another deep generative framework. Krauth et al. (2023) [3] adapted the VAE to aircraft trajectory modelling by using Temporal Convolutional Networks, forming a Temporal Convolutional VAE (TCVAE), and introduced a Variational Mixture of Posteriors (VampPrior) as the prior distribution. The generated trajectories were later used for collision-risk assessment [17]; the approach was then successfully applied to evaluate the risk arising from go-around and take-off procedures [18]. These works illustrate that the TCVAE performs well in generating four-dimensional aircraft trajectories, including rare operational scenarios such as go-arounds and collisions. Aref et al. [19, 20] also focused on collision-risk assessment and used VAEs to generate trajectories based on a chosen seed track. They then used multiple seed tracks [21, 22] to fill gaps in variability coverage. Murad and Ruocco [23] proposed a Time-Based Vector Quantized Variational Autoencoder (TimeVQVAE) for trajectory generation. By discretizing the latent space and integrating transformer priors, the model performed better than TCVAE in their experiments and could generate entire flight paths.

Some new generative methods have emerged in recent years. Diffusion Models (DMs) [24, 25, 26] have been applied to general trajectory generation. More recently, Flow Matching (FM) [8] has emerged as another generative modelling approach. These methods have also begun to be applied to aircraft trajectory generation. Zuo et al. [7] proposed the Air Trajectory Diffusion (AirTraj-Diff) framework to learn the distribution of aircraft trajectories in the terminal area. Larsen et al. [27] adapted diffusion- and flow-matching-based models to learn trajectory distributions from a data-rich airport and transfer the generation model to an airport with limited data. They also study latent diffusion, in which denoising is learned in a TCVAE representation and generated latent vectors are decoded into trajectories. Matsuda et al. [9] developed a conditional flow-matching model that generates realistic aircraft trajectories.

Alongside advances in generative modelling, conditional generation has increasingly been used to produce trajectories for specified operational scenarios. Motte et al. [28] extended the VampPrior-based VAE framework by conditioning trajectory generation on aircraft type. Zuo et al. [7] incorporated meteorological information into a conditional diffusion model, while Matsuda et al. [9] conditioned a flow-matching model on surrounding-traffic states. These studies demonstrate that conditional formulations can be applied across VAE-, diffusion-, and flow-matching-based models to generate trajectories corresponding to specified operational contexts.

Although these studies demonstrate the feasibility of conditional generation across different model families, they use different conditioning variables and address different application scenarios. They also use different inputs and evaluation metrics, making direct comparison across model families difficult, particularly for the recently introduced diffusion and flow-matching approaches. Therefore, a controlled benchmark using common datasets, a shared trajectory representation, and a consistent evaluation protocol is necessary for a fair comparison.

Accordingly, this study compares an established VAE-based approach with learned-prior and diffusion-based sampling strategies under a unified experimental setting. The TCVAE is retained as the reference baseline because it is one of the earliest VAE-based architectures developed specifically for four-dimensional aircraft trajectory generation and has subsequently been adopted as an established benchmark in aviation trajectory-generation studies.

3. Datasets and preprocessing

We use arrival trajectories at Zurich Airport (LSZH) and Tianjin Binhai International Airport (ZBTJ), reconstructed from OpenSky Network surveillance data [10]. The public landing_zurich_2019 col-

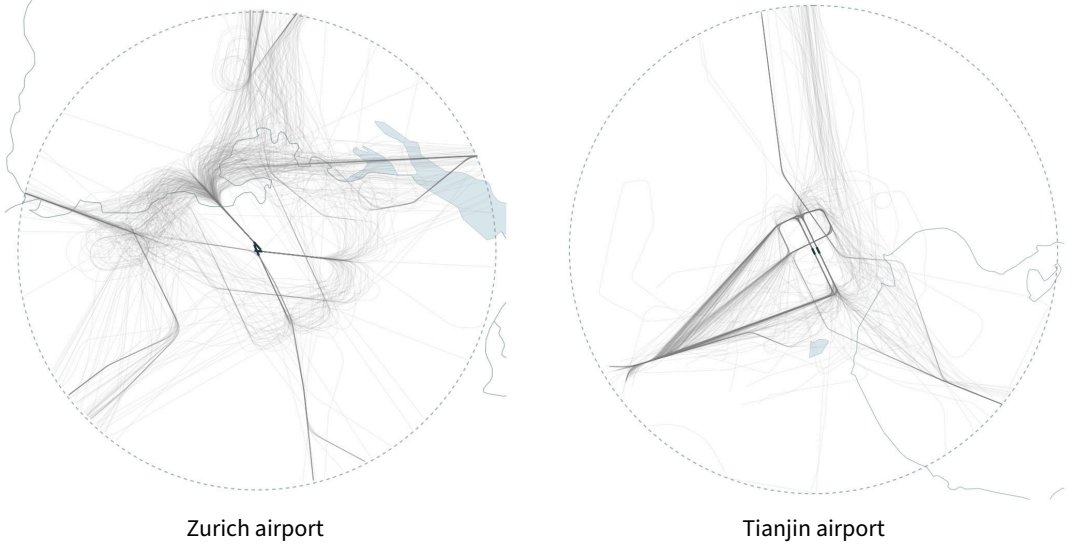


Figure 1. Example arrivals at Zurich (left) and Tianjin (right). Dashed circles indicate distances of 40 and 60 NM from the respective airports.

lection, distributed with the traffic library [29], covers 1 October to 30 November 2019 and has been used in previous trajectory-generation studies [3, 27]. The Tianjin dataset covers 1 August to 30 September 2026. Both datasets provide position, altitude, ground speed, and track, with runway and initial-flow labels available as metadata.

Before fixed-length resampling, the Tianjin data are filtered using the techniques described by Olive et al. [30]. We use the aggressive configuration in *traffic*, which combines rolling-window filtering with derivative- and cluster-based checks to address isolated outliers and consecutive erroneous reports. The filtered trajectories are resampled at one-second intervals before conversion to the common model representation.

We select a radius of 40 NM around Zurich and 60 NM around Tianjin to retain the main arrival patterns in their respective terminal manoeuvring areas. For Tianjin, we additionally exclude Airbus test flights associated with the nearby assembly plant, as their flight patterns can differ from those of regular arrivals. This exclusion keeps the dataset focused on regular arrival operations.

For modelling, each trajectory is resampled to 200 chronological points and represented by track in degrees, ground speed in knots, altitude in feet, and elapsed time in seconds. Geographic paths are recovered by backward integration from a fixed landing endpoint. Flights are partitioned into training, validation, and test sets. Normalization statistics are fitted separately for each timestep and feature using the training set only, then applied unchanged to validation and test data.

4. Methodology

4.1 Variational autoencoders for temporal trajectories

A classical autoencoder learns a deterministic encoder $z = f_\phi(x)$ and decoder $\hat{x} = g_\theta(z)$ by minimizing the reconstruction error between an observation x and its reconstruction $\hat{x}$. This method is commonly used for anomaly detection or for trajectory clustering, as distances in a compact latent space are not subject to the curse of dimensionality. Although the resulting bottleneck can provide a compact representation, the encoded points are not required to follow a known probability

distribution.

The model consequently provides no principled rule for drawing a new latent vector that can be decoded as a new observation. A Variational Autoencoder (VAE) instead defines a probabilistic latent-variable model [31]. Its encoder maps each observation to a distribution over latent variables, and training regularizes these distributions with respect to a prior from which new samples can be drawn. For an observed trajectory x , the encoder defines the approximate posterior

$$q_{\phi}(z | x) = \mathcal{N}\left(\mu_{\phi}(x), \text{diag}\left(\sigma_{\phi}^2(x)\right)\right), \quad (1)$$

while the decoder specifies $p_{\theta}(x | z)$. The parameters are estimated by maximizing the evidence lower bound,

$$\mathcal{L}(x) = \mathbb{E}_{q_{\phi}(z|x)}[\log p_{\theta}(x | z)] - D_{\text{KL}}(q_{\phi}(z | x) \| p(z)), \quad (2)$$

which trades reconstruction accuracy against agreement between the posterior and a chosen prior $p(z)$. After training, new trajectories are obtained by sampling z from the prior and decoding it. Figure 2 summarizes the two paths through the model: posterior inference and reconstruction during training, and prior sampling during generation.

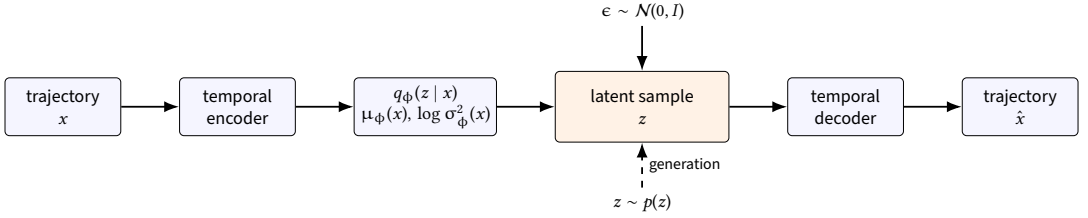


Figure 2. Variational autoencoder for trajectory reconstruction and generation. During training and reconstruction, the encoder parameterizes $q_{\phi}(z | x)$ and the reparameterization step produces z . During generation, this path is replaced by a sample from the prior $p(z)$; the same decoder then maps z to a trajectory.

Trajectory data are ordered sequences rather than independent feature vectors. We therefore use the Temporal Convolutional VAE (TCVAE) introduced in [3]. Its encoder and decoder replace fully connected mappings with temporal convolutional networks [32]. Dilated causal convolutions enlarge the receptive field across successive blocks, while residual connections support optimization of the deeper temporal network. The resulting model captures dependencies along the trajectory while retaining a fixed-dimensional latent representation.

Figure 3 follows 100 held-out Zurich trajectories through encoding and decoding. The posterior codes depend on those inputs; they are not prior draws. Comparing these reconstructions with the inputs tests representation and decoding, not generation.

The latent vectors have 64 components, so their distribution cannot be shown directly in two dimensions. We fit principal component analysis (PCA) to the training posterior samples and project both training and held-out codes onto its first two components. This projection is used only for visualization; encoding, decoding, and latent diffusion use all 64 dimensions.

4.2 Sampling the TCVAE latent space

The usefulness of the TCVAE as a generator depends on the distribution used to draw latent vectors. During reconstruction, an observed trajectory x is first passed through the encoder, and the decoder receives either a sample from its approximate posterior $q_{\phi}(z | x)$ or the posterior mean $\mu_{\phi}(x)$. Reconstruction therefore does not require sampling from the prior $p(z)$. During generation, by contrast, no observed trajectory is available to define a posterior, so the decoder must receive a

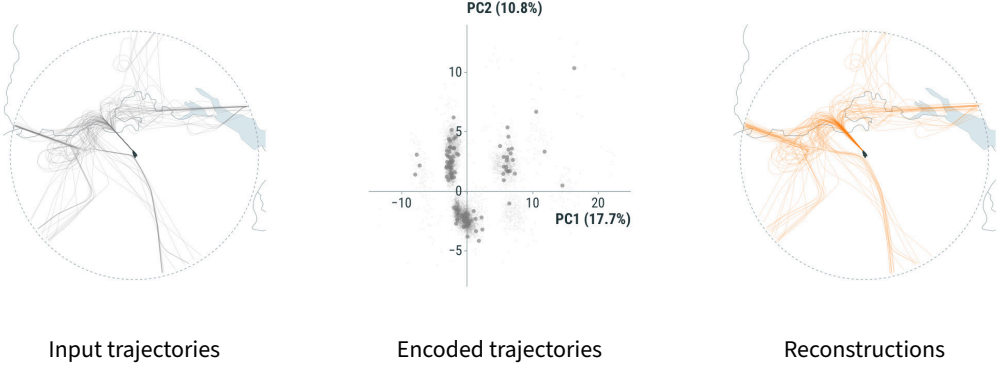


Figure 3. Reconstruction is not generation: the same 100 held-out Zurich runway-14 inputs, their posterior samples in a PCA projection, and their decoder reconstructions. The input and reconstruction paths use the same kinematic-to-geographic conversion. Good agreement here does not establish the quality of prior-generated trajectories.

latent vector drawn from the prior or from another learned latent generator. A low reconstruction error therefore shows that the encoder–decoder pair can represent the observations, but it does not establish that the sampling distribution used for generation covers the same regions of latent space.

For a fixed observation, drawing z from $q_\phi(z \mid x)$ is conditional. Selecting a training observation uniformly and then sampling its posterior instead defines an empirical mixture [33]:

$$q_{\text{emp}}(z) = \frac{1}{N} \sum_{i=1}^N q_\phi(z \mid x_i). \quad (3)$$

This is a valid sampling distribution, although it retains a dependence on the stored training population. We use posterior samples as a reconstruction diagnostic rather than adding this empirical-mixture generator to the benchmark.

The learned alternatives considered here are a VampPrior introduced for aircraft trajectories in [3] and a diffusion model fitted in latent space. Both define generation mechanisms that can be used without supplying a particular observed trajectory. Their agreement with the encoded population must be evaluated rather than assumed.

In a standard VAE, the aggregate posterior need not be standard Gaussian, particularly when the population contains several approach patterns. Figure 4 contrasts a fixed Gaussian with an illustrative learned mixture. It is a conceptual view, not a measurement of the present model’s latent density.

The first learned strategy is the Variational Mixture of Posteriors prior (VampPrior) [33], used for aircraft-trajectory generation in [3]. Rather than imposing a fixed Gaussian prior, the VampPrior defines

$$p_{\text{Vamp}}(z) = \frac{1}{K} \sum_{k=1}^K q_\phi(z \mid \chi_k), \quad (4)$$

where χ_k are learned pseudo-inputs and K is the number of mixture components. Each pseudo-input is passed through the same encoder as an observed trajectory and therefore parameterizes one Gaussian component in latent space. The pseudo-inputs and the encoder are optimized jointly with the decoder. At generation time, no observed trajectory is required: a component is selected from the learned mixture, a latent vector is sampled from that component, and the decoder maps it to a trajectory.

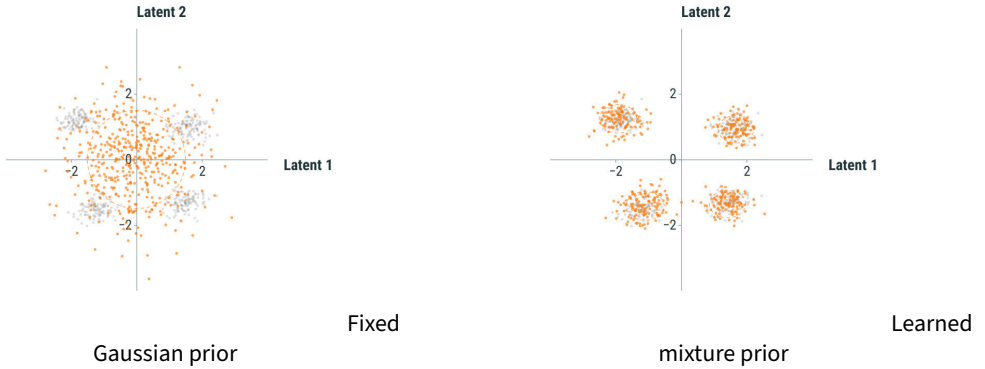


Figure 4. The prior schematics used in the slides. Grey points illustrate an encoded population; orange points illustrate prior draws. The fixed Gaussian and mixture are conceptual sampling distributions, not empirical results or a matched trained-prior ablation.

4.3 Diffusion-based trajectory generation

Diffusion models generate samples by learning to reverse a predefined process that progressively corrupts data with Gaussian noise [24]. For a clean trajectory representation x_0 , a noisy sample at diffusion step t is constructed as

$$x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon, \quad \epsilon \sim \mathcal{N}(0, I), \quad (5)$$

where $\bar{\alpha}_t$ is the cumulative signal-retention coefficient determined by the noise schedule. The index t denotes the diffusion step and therefore the noise level, rather than elapsed time along a flight.

During training, we sample a clean trajectory x_0 , a diffusion step t uniformly from $\{1, \dots, T\}$, and Gaussian noise ϵ . The resulting noisy trajectory x_t and the diffusion step t are supplied to a neural network $\epsilon_\theta(x_t, t)$, which predicts the added noise. The network is trained by minimizing the simplified noise-prediction objective

$$\mathcal{L}_{\text{DM}}(\theta) = \mathbb{E}_{x_0, t, \epsilon} [\|\epsilon - \epsilon_\theta(x_t, t)\|_2^2]. \quad (6)$$

This objective penalizes the difference between the sampled noise and the network prediction, enabling the model to learn denoising across different noise levels. The noisy training sample can be constructed directly at the selected step using the forward-process equation, without executing the entire noising sequence.

Generation starts from a fresh Gaussian noise sample with the same dimensions as the trajectory representation. At each sampling step, the network predicts the noise in the current state, and a prescribed reverse update uses this prediction to obtain a state at a lower noise level. Repeating these updates produces a new trajectory representation without requiring an observed trajectory as input. Figure 5 summarizes the distinction between constructing noisy examples for training and generating new samples through iterative denoising.

The direct diffusion baseline operates on the common normalized trajectory tensor defined in Section 3. Its noise prediction has the same dimensions as the input tensor, covering all trajectory points and feature channels. The model therefore learns to generate the trajectory features jointly, without relying on an autoencoder representation.

Figure 6 illustrates the evolution of generated trajectory features during sampling.

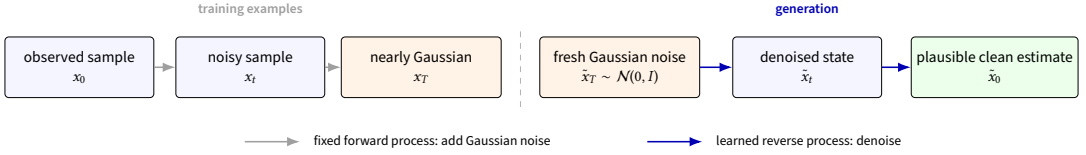


Figure 5. Training and generation with a diffusion model. The fixed forward process constructs noisy training examples from observed samples. Generation is a separate process: it starts from a fresh Gaussian draw and applies the learned denoiser repeatedly to produce $\tilde{x}_0$, an estimate of a plausible clean sample. The dashed line separates the training and generation paths; the generated sample is not a reconstruction of the particular observation x_0 .

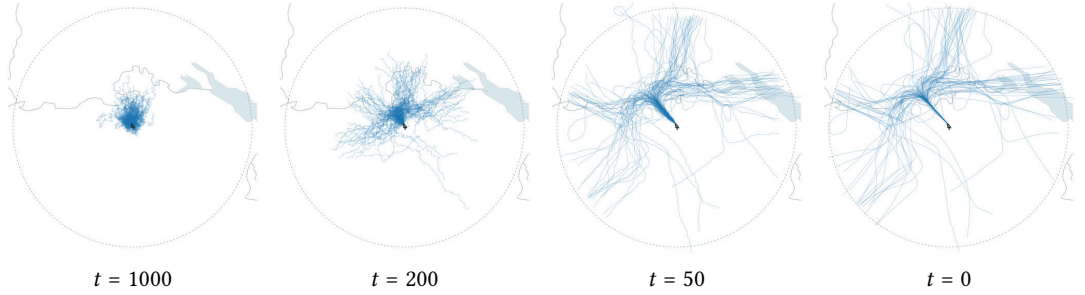


Figure 6. Zurich runway-14 denoising stages for 100 samples, using DDIM sampling [34]. DDIM applies a different sampling update to the DDPM-trained noise predictor; with $\eta = 0$, the updates are deterministic given the initial noise sample. The geographic display uses fixed timing and groundspeed bounded to 30–600 kt. Noisy intermediate states are not physical flights; the panels illustrate generation rather than a sequence of observed aircraft positions.

4.4 Latent diffusion

We fit a diffusion model to the latent space of a VampPrior-trained TCVAE, following the encode–generate–decode approach used by Larsen et al. [27]. The diffusion model generates new latent vectors, which the existing TCVAE decoder maps to trajectories. Denoising therefore takes place in a 64-dimensional latent space rather than in the trajectory tensor.

The TCVAE is trained first, and its encoder and decoder are then frozen. We draw one posterior sample for each training trajectory and standardize the resulting latent vectors using training-set statistics. The latent denoiser is trained with the noise-prediction objective in Equation 6: Gaussian noise is added at a randomly selected diffusion step, and the network learns to predict it by minimizing the mean squared error. Only the denoiser parameters are updated during this stage. Its training data come from encoded trajectories rather than the VampPrior pseudo-inputs.

During generation, sampling starts from Gaussian noise in the standardized latent space. Iterative denoising produces a new latent vector, which is transformed back to the original latent scale before decoding. This allows us to compare diffusion-based latent generation with direct VampPrior sampling while retaining the same frozen decoder. Figure 7 illustrates the latent denoising process.

4.5 Implementation framework

The experiments are implemented in a Python framework named *chabuduo*, available on GitHub at <https://github.com/Rowan4399/chabuduo>. The framework uses separate command-line stages for data acquisition, preprocessing, tensor preparation, validation, training, sampling, and plotting. Its architecture has three components:

- **Data preparation.** Surveillance trajectories are processed with traffic and converted into fixed-length tensors. Trajectory identifiers, representation metadata, data partitions, and training-set

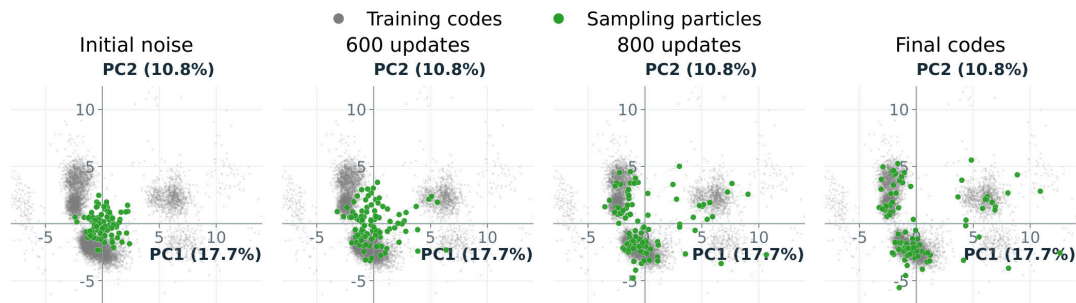


Figure 7. Illustration of latent diffusion sampling at updates 0, 600, 800, and 1000 (left to right). Encoded training trajectories (grey) and sampling particles (green) are shown in the same fixed two-dimensional PCA projection. Denoising takes place in the full 64-dimensional latent space.

normalization statistics are saved alongside these tensors, allowing different models to reuse the same prepared inputs.

- **Models and training.** Neural models operate on normalized tensors rather than aviation-domain objects. The training code handles optimization and validation-based checkpoint selection, with configurations, environment records, and loss histories retained in run directories.
- **Sampling and plotting.** Sampling reads a checkpoint and writes a generated bank with its random seed and sampling settings. Geographic conversion and plotting are separate from the neural models, so saved samples can be inspected without repeating training or generation.

We provide on the GitHub repository the YAML configuration files for each execution, specifying data preparation, model and training parameters, random seeds, and sampling settings.

5. Results

The results below compare the VampPrior TCVAE, direct diffusion, and latent diffusion through the frozen VampPrior decoder, with held-out real trajectories as the baseline. Models are trained separately for each airport.

To keep the training manageable and focus the comparison, we use runway-specific arrival subsets rather than the complete airport collections: runway 14 at Zurich and runway 16L at Tianjin. Table 1 gives the partitions used for the model comparisons. All map panels contain 100 trajectories per source, taken from saved sample banks without filtering for visual quality. Checkpoints are selected on validation data. VampPrior sampling and latent diffusion share the same frozen decoder within each dataset, allowing their latent generation mechanisms to be compared.

Table 1. Trajectory counts for the runs shown in the Results figures.

Dataset	Training	Validation	Test
Zurich runway 14	9,814	2,103	2,103
Tianjin runway 16L	2,261	484	486

5.1 Geographic samples

Figure 8 provides a first visual assessment of variability around the main arrival flows. For Zurich (top row), VampPrior reproduces these flows most clearly, although some paths are irregular on

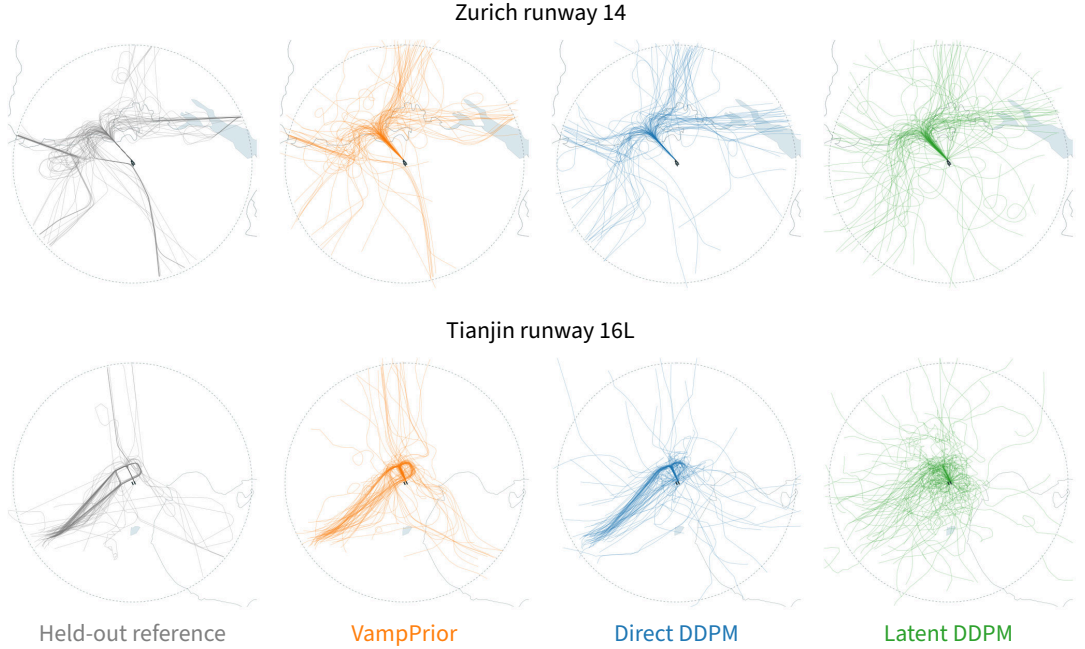


Figure 8. Geographic comparison for Zurich runway 14 (top) and Tianjin runway 16L (bottom), with 100 trajectories per panel. Columns show the held-out reference, VampPrior TCVAE, direct DDPM, and latent DDPM through the frozen VampPrior decoder. All four panels within each dataset share the same map extent.

final approach. Direct DDPM is also visually promising on the later approach segments, but its initial paths are more dispersed. Convergence at the landing point itself is imposed by construction. Latent diffusion, also explored by Larsen et al. [27], is less convincing here, with more diffuse patterns and looping paths.

The same broad pattern appears at Tianjin (bottom row). The held-out trajectories are concentrated around the main approach flow, whereas the generated trajectories introduce more spread than is present in the baseline. For Tianjin, latent DDPM does not reproduce the concentration of the held-out arrival flow.

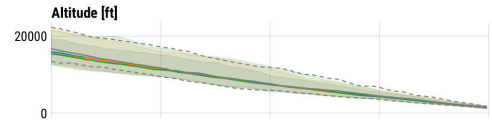
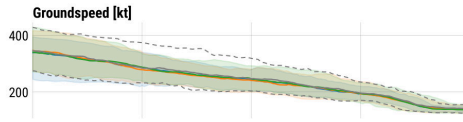
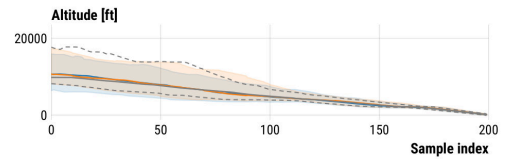
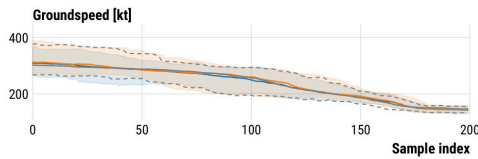
5.2 Feature distributions

Figure 9 shows generated groundspeed and altitude profiles generally centred close to the held-out profiles, with broadly similar or slightly narrower spread.

Sampling can also produce irregular paths, such as the example in Figure 10. The observed routes displayed as the closest latent neighbours bear some broad resemblance to the generated trajectory, but do not reproduce its full geometry. In this example, the irregular trajectory corresponds to a code in a sparsely populated region of the PCA projection.

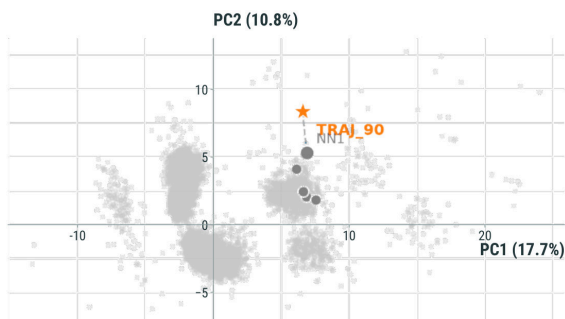
5.3 Simulator replay

The physical evaluation proposed by Olive et al. [5] measures how closely an aircraft simulator can reproduce an intended trajectory under an aircraft-performance model. We follow this principle using MiniSky, a lightweight version of BlueSky [35], with OpenAP. Each observed or generated trajectory supplies route, speed, and altitude instructions to the autopilot. We compare the intended and replayed trajectories using two distance measures: symmetrized segment-path distance (SSPD),

Zurich runway 14**Tianjin runway 16L**

--- Held-out Direct DDPM VampPrior Latent DDPM

Figure 9. Groundspeed and altitude profiles for 100 trajectories per source, represented by 200 points. Lines show medians; generated bands and dashed reference boundaries show the 5th–95th percentile intervals. The top row is Zurich runway 14 and the bottom row is Tianjin runway 16L.



Latent space (PCA)



Observed and generated routes

Figure 10. Archived Zurich latent-space and route diagnostic. Left: a generated code (orange star), observed codes (grey), and highlighted neighbours in a two-component PCA projection. Right: observed and generated routes retained from the presentation.

which compares path geometry, and dynamic time warping (DTW), which aligns ordered position sequences. The archived procedure selects a complete replay candidate by minimum altitude mean absolute error before reporting these horizontal distances.

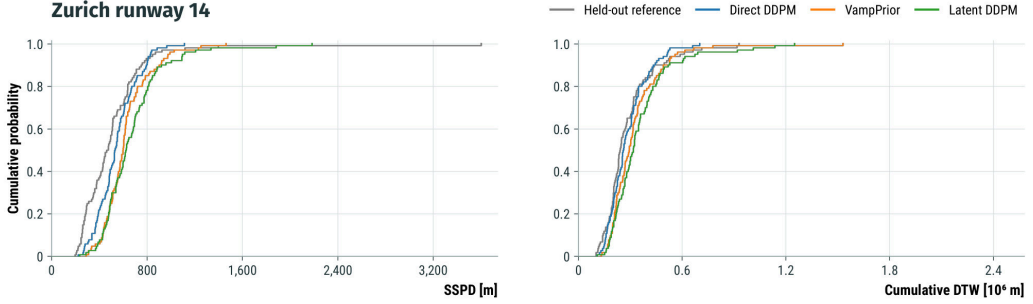


Figure 11. Archived Zurich runway-14 replay discrepancies for the held-out reference and three generators. Left: symmetrized segment-path distance (SSPD), in metres. Right: cumulative, unnormalized dynamic time warping (DTW), displayed in 10^6 metres. Each distance compares an intended trajectory with its own MiniSky replay. Lower discrepancies shift a curve towards the left.

Even historical trajectories are not replayed exactly. Discrepancies can arise from observational uncertainty, unknown flight parameters and meteorological conditions, and approximations in the simulator and instruction conversion. With perfect replay, every distance would be zero and each cumulative distribution would jump to one at zero distance. We therefore compare generated trajectories with the held-out replay baseline in Figure 11, asking how closely their distance distributions approach the reference curve.

Direct DDPM is generally closer to this reference than VampPrior, most clearly for SSPD; the DTW curves are less separated. This suggests that its generated paths are easier to follow under the chosen replay procedure, despite the more convincing geographic flows produced by VampPrior. Latent DDPM remains less satisfactory in this comparison. Visual similarity and simulator replay thus provide different assessments of the same generated trajectories.

5.4 Recorded computational cost

Table 2. Recorded training time and completed epochs for the Zurich runway-14 and Tianjin runway-16L runs.

Training stage	Zurich runway 14		Tianjin runway 16L	
	Time (h)	Epochs	Time (h)	Epochs
VampPrior TCVAE	64.73	251	17.32	273
Latent DDPM denoiser	0.89	500	0.21	500
Direct DDPM	2.39	700	0.64	700

Under our experimental settings, direct DDPM required substantially less recorded training time per epoch than VampPrior TCVAE on both datasets. For Zurich, direct DDPM averaged 12.29 s per epoch, compared with 15.47 min for TCVAE. For Tianjin, the corresponding averages were 3.28 s and 3.81 min. The latent DDPM denoiser required even less time per epoch, averaging 6.40 s for Zurich and 1.54 s for Tianjin.

The preceding VAE training dominated the total cost of latent diffusion. Including this stage, the recorded two-stage totals were 65.62 h for Zurich and 17.53 h for Tianjin. Direct DDPM therefore

had the lowest total recorded training cost among the three approaches in these experiments. This comparison reflects the configured architectures, batch sizes, and training budgets.

6. Concluding remarks

This study compared a temporal convolutional variational autoencoder with a learned VampPrior, direct DDPM, and latent DDPM using ADS-B arrival trajectories at Zurich and Tianjin airports. Shared preprocessing, trajectory representations, and data partitions supported comparisons of geographic patterns and feature profiles, complemented by aircraft-performance-based simulator replay for Zurich.

VampPrior reproduced the main arrival flows more clearly than latent DDPM, which generated more dispersed and irregular paths despite sharing the same frozen decoder within each dataset. Under the configurations examined, diffusion-based latent sampling did not improve generation over VampPrior sampling. This comparison illustrates why the quality of a learned representation and the mechanism used to sample it must be assessed separately.

Geographic resemblance and simulator replay favoured different generators. For Zurich, direct DDPM generally produced replay discrepancies closer to the held-out reference than VampPrior, particularly for SSPD, despite the clearer geographic flows obtained with VampPrior. Direct DDPM also required less recorded training time per epoch. These findings motivate further investigation of direct diffusion, while showing that visual agreement with observed traffic provides only part of the assessment of generated trajectories.

Future work will examine how diffusion model design and sampling strategies affect geographic variability and replayability. The relatively low recorded training cost of direct DDPM makes these experiments more practical. Extending the benchmark to conditional generation would also allow models to be compared on specified operational settings, such as an arrival route or aircraft type. Such comparisons should assess whether models satisfy the requested conditions while retaining realistic variability and consistency with aircraft performance.

Author contributions

Conceptualization (*all*), Methodology (R.R., T.H., R.W., C.H., X.O.), Software (R.R., T.H., X.O.), Data Curation (R.R., X.O.), Writing – Original Draft (R.R., X.O.), Writing – Review & Editing (*all*), Visualization (R.R., T.H., X.O.), Project administration (R.W.).

Open data statement

The Zurich landing dataset is publicly available as `landing_zurich_2019` through the traffic library [29]. The dataset is derived from OpenSky Network surveillance data and can be retrieved through the dataset interface provided by traffic. The Tianjin landing dataset was derived from OpenSky arrival histories; access information for the processed artifact and its machine-readable exclusion report will be provided with the final research release, subject to the applicable OpenSky data conditions.

Reproducibility statement

The preprocessing, model training, sampling, evaluation, and figure-generation code is developed in the open-source chabuduo repository: <https://github.com/Rowan4399/chabuduo>. Experiment configurations record the dataset selection, feature representation, data split, model parameters, and

sampling settings. The final paper will identify the release or commit used for the reported experiments and provide the corresponding processed-artifact and checkpoint metadata.

References

- [1] Timoth  Krauth, J r me Morio, Xavier Olive, Benoit Figuet, and Raphael Monstein. "Synthetic Aircraft Trajectories Generated with Multivariate Density Models". In: *Engineering Proceedings* 13.1 (2021), p. 7. DOI: [10.3390/engproc2021013007](https://doi.org/10.3390/engproc2021013007).
- [2] Daniel Delahaye, St phane Puechmorel, Panagiotis Tsiotras, and  ric F ron. "Mathematical Models for Aircraft Trajectory Design: A Survey". In: *Air Traffic Management and Systems*. Ed. by Electronic Navigation Research Institute. Springer Japan, 2014, pp. 205–247. DOI: [10.1007/978-4-431-54475-3_12](https://doi.org/10.1007/978-4-431-54475-3_12).
- [3] Timoth  Krauth, Adrien Lafage, J r me Morio, Xavier Olive, and Manuel Waltert. "Deep Generative Modelling of Aircraft Trajectories in Terminal Maneuvering Areas". In: *Machine Learning with Applications* 11 (2023), p. 100446. DOI: [10.1016/j.mlwa.2022.100446](https://doi.org/10.1016/j.mlwa.2022.100446).
- [4] Mayara Conde Rocha Mur a 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)*. IEEE, 2020, pp. 1–7. DOI: [10.1109/DASC50938.2020.9256644](https://doi.org/10.1109/DASC50938.2020.9256644).
- [5] Xavier Olive, Junzi Sun, Mayara Conde Rocha Mur a, and Timoth  Krauth. "A Framework to Evaluate Aircraft Trajectory Generation Methods". In: *Proceedings of the 14th USA/Europe Air Traffic Management Research and Development Seminar*. New Orleans, LA, USA, 2021.
- [6] Gabriel Jarry and Xavier Olive. "Generation of Vertical Profiles with Neural Ordinary Differential Equations Trained on Open Trajectory Data". In: *Journal of Open Aviation Science* 4 (2026). DOI: [10.59490/joas.2026.8457](https://doi.org/10.59490/joas.2026.8457).
- [7] Di Zuo, Kaiquan Cai, Meng Li, Yanbo Zhu, and Peng Zhao. "AirTraj-Diff: Generating Aircraft Trajectory With Conditional Diffusion Probabilistic Model". In: *IEEE Transactions on Intelligent Transportation Systems* 26.12 (2025), pp. 21667–21683. DOI: [10.1109/TITS.2025.3622626](https://doi.org/10.1109/TITS.2025.3622626).
- [8] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. "Flow Matching for Generative Modeling". In: *Proceedings of the 11th International Conference on Learning Representations*. 2023.
- [9] Haruki Matsuda, Naoya Takeishi, and Takehisa Yairi. "High-Fidelity Aircraft Trajectory Generation Using a Flow-based Generative Model". In: *Transactions of the Japan Society for Aeronautical and Space Sciences* 69.1 (2026), pp. 9–18. DOI: [10.2322/tjsass.69.9](https://doi.org/10.2322/tjsass.69.9).
- [10] Matthias Sch fer, Martin Strohmeier, Vincent Lenders, Ivan Martinovic, and Matthias Wilhelm. "Bringing up OpenSky: A large-scale ADS-B sensor network for research". In: *IPSN-14 proceedings of the 13th international symposium on information processing in sensor networks*. IEEE, 2014, pp. 83–94. DOI: [10.1109/IPSN.2014.6846743](https://doi.org/10.1109/IPSN.2014.6846743).
- [11] Thomas Pecsvaradi. *Four-Dimensional Guidance Algorithms for Aircraft in an Air Traffic Control Environment*. NASA Technical Note D-7829. National Aeronautics and Space Administration, 1975.
- [12] David B. Quanbeck. *Methods for Generating Aircraft Trajectories*. Tech. rep. AD-A122386/CNA-PP-361. Center for Naval Analyses, 1982.
- [13] Shane T. Barratt, Mykel J. Kochenderfer, and Stephen P. Boyd. "Learning Probabilistic Trajectory Models of Aircraft in Terminal Airspace From Position Data". In: *IEEE Transactions on Intelligent Transportation Systems* 20.9 (2019), pp. 3536–3545. DOI: [10.1109/TITS.2018.2877572](https://doi.org/10.1109/TITS.2018.2877572).
- [14] Gabriel Jarry, Nicolas Couellan, and Daniel Delahaye. "On the use of generative adversarial networks for aircraft trajectory generation and atypical approach detection". In: *ENRI International Workshop on ATM/CNS*. Springer, 2019, pp. 227–243.

- [15] Sebastiaan Wijnands, Alexei Sharpanskykh, and Karim Aly. "Generation of synthetic aircraft landing trajectories using generative adversarial networks". In: *Proceedings of the 14th SESAR Innovation Days, Rome, Italy* (2024), pp. 12–15.
- [16] Jinsung Yoon, Daniel Jarrett, and Mihaela Van der Schaar. "Time-series generative adversarial networks". In: *Advances in neural information processing systems* 32 (2019).
- [17] Timothé Krauth, Benoit Figueet, Xavier Olive, and Jérôme Morio. "Collision risk assessment in Terminal Manoeuvring Areas based on trajectory generation methods". In: *15th Air Traffic Management Research and Development Seminar*. 2023.
- [18] Timothé Krauth, Jérôme Morio, Xavier Olive, and Benoit Figueet. "Advanced collision risk estimation in terminal manoeuvring areas using a disentangled variational autoencoder for uncertainty quantification". In: *Engineering Applications of Artificial Intelligence* 133 (2024), p. 108137.
- [19] Shahab Aref, John Shortle, and Lance Sherry. "Using variational autoencoders to generate 4d synthetic flight tracks for collision risk safety analysis". In: *2024 AIAA DATC/IEEE 43rd Digital Avionics Systems Conference (DASC)*. IEEE. 2024, pp. 1–10.
- [20] Shahab Aref, John Shortle, and Lance Sherry. "Generating synthetic flight tracks for collision risk safety analysis: Variational autoencoders with a single seed track". In: *2024 Integrated Communications, Navigation and Surveillance Conference (ICNS)*. IEEE. 2024, pp. 1–9.
- [21] Shahab Aref, John Shortle, and Lance Sherry. "A Methodology for Multi-Seed Generation of 4-D Synthetic Flight Tracks for Air-to-Air Collision Risk Modelling using Variational Autoencoder". In: *Internal Report, center for Air Transportation Systems Research* (2024).
- [22] Shahab Arefa, John Shortlea, and Lance Sherrya. "Synthetic Flight Track Generation for Collision Risk Analysis: An Iterative Multi-Seed Approach". In: *23rd Annual Conference on Systems Engineering Research, Arlington, VA*. 2026.
- [23] Abdulmajid Murad and Massimiliano Ruocco. "Synthetic aircraft trajectory generation using time-based vq-vae". In: *2025 Integrated Communications, Navigation and Surveillance Conference (ICNS)*. IEEE. 2025, pp. 1–10.
- [24] Jonathan Ho, Ajay Jain, and Pieter Abbeel. "Denoising Diffusion Probabilistic Models". In: *Advances in Neural Information Processing Systems*. Vol. 33. 2020, pp. 6840–6851.
- [25] Yuanshao Zhu, Yongchao Ye, Shiyao Zhang, Xiangyu Zhao, and James Yu. "Difftraj: Generating gps trajectory with diffusion probabilistic model". In: *Advances in Neural Information Processing Systems* 36 (2023), pp. 65168–65188.
- [26] Yuanshao Zhu, James Jianqiao Yu, Xiangyu Zhao, Qidong Liu, Yongchao Ye, Wei Chen, Zijian Zhang, Xuetao Wei, and Yuxuan Liang. "Controltraj: Controllable trajectory generation with topology-constrained diffusion model". In: *Proceedings of the 30th ACM SIGKDD conference on knowledge discovery and data mining*. 2024, pp. 4676–4687.
- [27] Olav Finne Præsteng Larsen, Massimiliano Ruocco, Michail Spitieris, Abdulmajid Murad, and Martina Ragosta. "Learning to Land Anywhere: Transferable Generative Models for Aircraft Trajectories". In: *arXiv preprint arXiv:2511.04155* (2025).
- [28] Arnault Motte, Xavier Olive, and Jérôme Morio. "Conditional Variational Autoencoders for aircraft type-specific trajectory generation". In: *SESAR Innovation Days 2025*. 2025.
- [29] Xavier Olive. "traffic, a toolbox for processing and analysing air traffic data". In: *Journal of Open Source Software* 4.39 (2019), p. 1518. DOI: 10.21105/joss.01518.
- [30] Xavier Olive, Jan Krummen, Benoit Figueet, and Richard Alligier. "Filtering Techniques for ADS-B Trajectory Preprocessing". In: *Journal of Open Aviation Science* 2.2 (Mar. 2025). DOI: 10.59490/joas.2024.7882. URL: <https://journals.open.tudelft.nl/joas/article/view/7882>.
- [31] Diederik P Kingma and Max Welling. "Auto-encoding variational bayes". In: *arXiv preprint arXiv:1312.6114* (2013).

- [32] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. “An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling”. In: *arXiv preprint arXiv:1803.01271* (2018). 450 451 452
- [33] Jakub M. Tomczak and Max Welling. “VAE with a VampPrior”. In: *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*. Vol. 84. Proceedings of Machine Learning Research. PMLR, 2018, pp. 1214–1223. 453 454 455
- [34] Jiaming Song, Chenlin Meng, and Stefano Ermon. *Denoising Diffusion Implicit Models*. 2020. doi: 10.48550/arXiv.2010.02502. arXiv: 2010.02502 [cs.LG]. 456 457
- [35] 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 (ICRAT)*. Philadelphia, PA, USA, June 2016. URL: <https://resolver.tudelft.nl/uuid:d1131a90-f0ea-4489-a217-ad29987689a1>. 458 459 460 461