A Gentle Introduction and Tutorial on Deep Generative Models in Transportation Research

Seongjin Choi[a,*], Zhixiong Jin[b,c,*], Seung Woo Ham[d], Jiwon Kim[e] and Lijun Sun[f]

[a]Department of Civil, Environmental, and Geo- Engineering, University of Minnesota, 500 Pillsbury Dr. SE, Minneapolis, MN 55455, USA
[b]Univ. Guatave Eiffel, ENTPE, LICIT-ECO7, Lyon, F-69675, France
[c]Urban Transport Systems Laboratory (LUTS), École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, CH 1015, Switzerland
[d]Department of Civil and Environmental Engineering, National University of Singapore, Singapore
[e]School of Civil Engineering, The University of Queensland, Brisbane St Lucia, Queensland, Australia
[f]Department of Civil Engineering, McGill University, 817 Sherbrooke Street West, Montreal, Quebec H3A 0C3, Canada

ARTICLE INFO

Keywords:
AI in Transportation
Deep Generative Models
Generative AI
Deep Learning
Machine Learning

ABSTRACT

Deep Generative Models (DGMs) have rapidly advanced in recent years, becoming essential tools in various fields due to their ability to learn data distributions and generate synthetic data. Their importance in transportation research is increasingly recognized, particularly for applications like traffic data generation, prediction, and feature extraction. This paper offers a comprehensive introduction and tutorial on DGMs, with a focus on their applications in transportation. It begins with an overview of generative models, followed by detailed explanations of fundamental models, a systematic review of the literature, and practical tutorial code to aid implementation. The paper also discusses current challenges and opportunities, highlighting how these models can be effectively utilized and further developed in transportation research. This paper serves as a valuable reference, guiding researchers and practitioners from foundational knowledge to advanced applications of DGMs in transportation research.

Contents

*Equal Contribution
**This survey is based on literature up to October 2024.

Email addresses: chois@umn.edu (S. Choi); zhixiong.jin@univ-eiffel.fr (Z. Jin); sw.ham@nus.edu.sg (S.W. Ham); jiwon.kim@uq.edu.au (J. Kim); lijun.sun@mcgill.ca (L. Sun)
ORCID(s): 0000-0001-7140-537X (S. Choi); 0000-0002-1370-781X (Z. Jin); 0000-0001-7531-9217 (S.W. Ham); 0000-0001-6380-3001 (J. Kim); 0000-0001-9488-0712 (L. Sun)

1. Introduction

The rapid growth of artificial intelligence in transportation research over recent years has provided innovative solutions to address both longstanding and emerging transportation issues. Among the spectrum of artificial intelligence techniques, Deep Generative Models (DGMs) are increasingly gaining attention in transportation research due to their ability to learn the underlying patterns in large datasets and model the data distribution. Using the learned distributional characteristics, DGMs can generate synthetic data that accurately replicate real-world scenarios, estimate and predict agent-level trajectories, link-level and network-level traffic states, and provide latent representations of complex transportation systems. The primary purpose of this paper is to review the state-of-the-art theories and investigate the applications, potential benefits, and challenges of using DGMs in transportation research. Additionally, we aim to offer a comprehensive tutorial for researchers in transportation engineering who are interested in incorporating DGMs into their works. By providing a detailed overview of the current state-of-the-art models and literature with practical tutorials, this paper seeks to facilitate the adoption of DGMs and inspire further innovations in transportation research.

1.1. Background

Conventional deep learning models in transportation research have predominantly utilized discriminative modeling approaches. Discriminative models, as the name suggests, are designed to discriminate or differentiate patterns in data. These models learn a direct mapping from input variables ($\mathbf{x}$) to output variables ($\mathbf{y}$), effectively estimating the conditional probability $p(\mathbf{y}|\mathbf{x})$. In practice, this translates into the model's ability to classify input data into predefined categories or predict target variables based on input features. For example, Convolutional Neural Networks (CNNs) and their variants are widely used to analyze spatial traffic patterns for classifying different traffic conditions; Recurrent Neural Networks (RNNs), including Long Short-Term Memory (LSTM) networks, model temporal dependencies for forecasting traffic variables over time; and Graph Neural Networks (GNNs) capture complex spatial relationships within transportation networks represented as graphs, facilitating tasks such as traffic prediction and network anomaly detection.

Even though discriminative models have achieved great success in transportation research, inherent limitations constrain their effectiveness. They require a structured form of extensive labeled (paired $\mathbf{x}$ and $\mathbf{y}$ data) datasets and are confined to the patterns present within the training data. Additionally, these models are unable to capture the underlying data distribution $p(\mathbf{x})$, which restricts their ability to understand complex patterns in the dataset fully. Consequently, these limitations hinder their capacity to generalize beyond the observed scenarios in the training dataset, i.e., the predictive ability is limited within what was given in the training dataset. Moreover, transportation data is inherently dynamic and complex, often characterized by uncertainty and rare events. Relying solely on discriminative models can thus impede the development of robust solutions capable of adapting to such complexities and predicting or generating rare events with non-zero probability.

Deep Generative Models (DGMs), on the other hand, are a class of machine learning models with the core objective of learning the joint probability $p(\mathbf{x}, \mathbf{y})$ or the underlying data distribution $p(\mathbf{x})$. This capability enables DGMs to handle both discriminative tasks (such as classification and prediction) and generative tasks (such as data generation) within a unified framework. In contrast to traditional discriminative models, which are often limited by their reliance on extensive labeled datasets, DGMs can learn from the inherent patterns in data to generate realistic synthetic examples. In transportation research, DGMs can be used to create realistic traffic scenarios, such as time-space speed contour diagrams, simulate traveler behaviors under different conditions while accounting for the characteristics of individual agents, and generate synthetic datasets to augment rare events. Their ability to capture the complexities of traffic dynamics and traffic agent behavior directly addresses data scarcity issues and enhances the robustness and generalization of predictive models. This versatility makes DGMs an essential tool for managing the dynamic and uncertain nature of transportation systems.

From our viewpoint, the development of effective DGMs represents one of the most promising opportunities in modern transportation research. DGMs have the potential to revolutionize how we simulate, analyze, and optimize transportation systems by learning complex patterns from vast amounts of data without relying on strict assumptions or extensive model-based parameterization. While traditional theory-based models have been invaluable in understanding transportation dynamics, they often face challenges such as oversimplified assumptions and a narrow focus on specific scenarios. Despite these limitations, theory-based models remain essential, as they provide a foundation upon which data-driven methods like DGMs can build. Specifically, instead of replacing traditional approaches, DGMs serve as complementary and supplementary tools, enhancing and expanding upon established models. Their flexibility allows for the modeling of intricate dynamics and the generation of realistic, unseen scenarios, ultimately improving traffic

simulations, scenario planning, and the implementation of digital twins in the transportation area.

1.2. Relationship with Existing Surveys

DGMs have played a pivotal role in advancing machine learning over the past decade by enabling the generation and approximation of joint distributions for targets and training data. Several notable surveys have highlighted the significance and versatility of DGMs across various fields. For example, Harshvardhan et al. (2020) provided a comprehensive guide tracing the evolution from traditional machine learning-based generative models to advanced DGMs. Similarly, Bond-Taylor et al. (2021) reviewed a wide range of DGMs, including energy-based models, Variational Autoencoders (VAEs), autoregressive models, and normalizing flows, discussing their strengths, weaknesses, and applications. Area-specific surveys further illustrate the adaptability and impact of DGMs. For instance, De et al. (2022) explored DGMs' applications in the Industrial Internet of Things (IIoT), while Guo and Zhao (2022) systematically reviewed DGMs for graph generation, focusing on their applications, taxonomy, and analysis. Additionally, Lopez et al. (2020) examined the role of DGMs in molecular biology, highlighting their utility in advancing scientific discoveries. Similarly, Anstine and Isayev (2023) provided a comprehensive review in the application of chemical sciences with discussing application challenges.

In transportation research, several review papers have similarly explored the potential of DGMs and highlighted their extensive applications in the field. For instance, Boquet et al. (2020) discussed the use of VAEs to address key transportation challenges, including traffic imputation, dimensionality reduction, and anomaly detection Similarly, Lin et al. (2023a) categorized the use of Generative Adversarial Networks (GANs) in autonomous driving, traffic flow research, and anomaly detection, while also identifying challenges and future research directions for integrating GANs into transportation research. However, these survey primarily focuses on typical models, with less coverage of other DGMs. Yan and Li (2023) systematically investigated the role of DGMs, or generative AI, in addressing key issues such as traffic perception, prediction, simulation, and decision-making within Intelligent Transportation Systems (ITS), while discussing existing challenges and future research directions. Even though this survey provides valuable insights, it would benefit from a more comprehensive explanation of fundamental DGMs and a deeper analysis of challenges and opportunities within the transportation context. Such enhancements could make DGMs more accessible and easier to adopt for a broader range of researchers in the field.

To address these gaps, this paper offers a comprehensive introduction and tutorial on DGMs in transportation applications. First, we provide a clear explanation of fundamental DGMs to ensure readers gain a solid understanding of core concepts. This is followed by a systematic review of state-of-the-art DGMs, with particular attention to their application in transportation research. We also include practical tutorial codes to guide researchers and practitioners in effectively applying these models to real-world transportation problems. Lastly, the paper concludes by emphasizing the importance of DGMs, discussing current research challenges, and exploring potential solutions, making it a valuable resource for those interested in exploring the use of DGMs in transportation.

1.3. Objectives of the Paper

This paper aims to provide a comprehensive survey of the application and potential of DGMs in the field of transportation research. The objectives of this survey paper are as follows:

- To provide an accessible and comprehensive introduction to DGMs for transportation researchers. We aim to establish this paper as a starting point for those interested in exploring the potential of DGMs in transportation research.
- To review the state-of-the-art in transportation research using DGMs, offering insights into the current landscape and practices across various transportation domains.
- To contribute practical value by offering a tutorial section, providing hands-on guidance and resources for implementing DGMs in transportation research.
- To identify and discuss the challenges, limitations, and opportunities of employing DGMs in transportation research.

With these objectives, we hope to contribute to the wider acceptance and understanding of DGMs within the field of transportation research. It is our belief that the adoption of these models can revolutionize the way we approach and solve transportation problems, leading to more robust, efficient, and sustainable systems.

Table 1
Comparison of previous survey papers

Reference	Contribution	Wide range of DGMs	Transportation Application	Tutorial Code
Harshvardhan et al. (2020)	provide a comprehensive survey and implementation guide for machine and deep learning-based DGMs.	✓	✗	✓
Bond-Taylor et al. (2021)	provide a comparative review of DGMs, analyzing their strengths, weaknesses, and applications	✓	✗	✗
De et al. (2022)	provide a comprehensive review of DGMs in Industrial Internet of Things (IIoT) applications	✓	✗	✗
Guo and Zhao (2022)	provide systematic surveys of DGMS for graph generation with their applications, taxonomy, and analysis.	✓	✗	✗
Lopez et al. (2020)	provide a comprehensive review in the application of DGMs in molecular biology, highlighting their role in advancing scientific discoveries	✓	✗	✗
Anstine and Isayev (2023)	provide a comprehensive review in the application of chemical sciences with discussing their application challenges	✓	✗	✗
Boquet et al. (2020)	review the Variational Autoencoders (VAE) models to address key challenges in transportation	✓ (partially)	✓	✗
Lin et al. (2023a)	provide comprehensive applications of Generative Adversarial Networks (GANs) in Intelligent Transportation Systems (ITSs)	✓ (partially)	✓	✗
Yan and Li (2023)	provide a comprehensive reviews of DGMs in the ITS applications	✓	✓	✗
Current Work	provide a comprehensive review and implementation tutorial on DGMs focusing on their applications in transportation research	✓	✓	✓

1.4. Structure of Paper

In Section 2, we introduce the capabilities and applications of several key DGMs in the literature, providing a structured overview along with detailed mathematical formulations to enhance understanding. Specifically, we introduce Variational Autoencoders (VAEs) in Section 2.3, Generative Adversarial Networks (GANs) in Section 2.4, Normalizing Flows (or flow-based generative models) in Section 2.5, Score-based generative models in Section 2.6 and Diffusion models in Section 2.7. In Section 3, we introduce state-of-the-art transportation research using DGMs. Particularly, we introduce applications of DGM 1) for generating realistic new data samples that can be applied in data synthesis, trajectory generation, and missing data imputation in Section 3.1, 2) for estimating and predicting distributions at three different levels of analyses in transportation research (agent-level, link-level, and region-level) in Section 3.2, and 3) for understanding underlying dynamics and learning unsupervised representations of data for applications like anomaly detection and mode choice analysis in Section 3.3. **Readers primarily interested in the current practices can begin with Section 3 for a review of the latest literature in transportation research using**

DGMs, and then refer back to Section 2 for an introduction to the models themselves. In Section 4, we provide a tutorial that offers practical guidance on implementing DGMs in transportation research. We introduce two examples: 1) generating travel survey data in Section 4.1, and 2) generating highway traffic speed contour data in Section 4.2. In Section 5, we identify and discuss the challenges and opportunities associated with using DGMs in the transportation domain, emphasizing the importance of addressing these challenges for the successful adoption of DGMs. Finally, in Section 6, we summarize and conclude the paper.

2. Introduction to Deep Generative Models

2.1. Overview of Deep Generative Models

2.1.1 Background

Generative models are a class of machine learning models that aim to understand and capture the underlying probability distribution of a dataset. By learning this distribution, these models can generate new data points that are similar to those in the original dataset. Unlike discriminative models, which focus on modeling the conditional probability of the output given the input, $p(\mathbf{y}|\mathbf{x})$, generative models aim to learn the joint distribution $p(\mathbf{x}, \mathbf{y})$ or $p(\mathbf{x})$. By understanding the joint distribution, generative models can also derive the conditional probability $p(\mathbf{y}|\mathbf{x})$ as $p(\mathbf{y}|\mathbf{x}) = \frac{p(\mathbf{x},\mathbf{y})}{p(\mathbf{x})}$. This means generative models can perform discriminative tasks effectively while also being capable of generating new data that aligns with the learned distribution. Therefore, generative models provide a broader framework that not only includes the functionalities of discriminative models but also extends beyond them to include data generation capabilities and model data generation process. Examples of earlier generative models include Gaussian Mixture Models, Hidden Markov Model, and Naive Bayes Classifier. However, they often rely on simplifying assumptions and may struggle with high-dimensional or complex data patterns.

Deep Generative Models (DGMs) build upon these concepts by leveraging deep learning architectures to model more complex, high-dimensional data without relying on such strong assumptions. By utilizing neural networks, DGMs can learn these complex, nonlinear relationships within data, allowing for more accurate modeling and generation of realistic data samples.

A key advantage of DGMs is their ability to generate data, which is essential in applications such as data augmentation, synthetic data creation, and data privacy enhancement. This capability allows the creation of new, synthetic datasets that can enhance the robustness and performance of machine-learning models, especially when real-world data is limited or sensitive. Another significant benefit of using DGMs is their capacity for probabilistic inference. It involves using the probability distributions learned by DGMs to do predictions and estimate uncertainties. The probabilistic inference is vital for understanding and modeling the uncertainties inherent in transportation systems, as it provides a range of possible outcomes along with their associated probabilities. Furthermore, DGMs are notable for their ability to extract deep insights from data and enable a wide range of applications through the manipulation and analysis of latent vectors. The latent vectors represent compressed, lower-dimensional forms of the input data that retain its essential features. This capability makes DGMs powerful tools for capturing and analyzing complex data relationships in transportation applications.

Some fundamental concepts and terminologies to understand the following sections are explained in Appendix A.

2.1.2 Model Training Objective

The objective of training DGMs is to minimize the discrepancy between the true data distribution $p_{\text{data}}(\mathbf{x})$ and the model's estimated distribution $p_{\text{model}}(\mathbf{x}; \theta)$. This discrepancy can be expressed using the KL divergence. The goal is to adjust the model parameters θ such that the model distribution closely approximates the true data distribution. This is expressed in the optimization problem:

$$\theta^* = \arg\min_{\theta} D_{KL}(p_{\text{data}}(\mathbf{x})||p_{\text{model}}(\mathbf{x}; \theta)). \tag{2.1}$$

However, in practice, we do not have direct access to $p_{\text{data}}(\mathbf{x})$, and hence we rely on an empirical dataset (or training dataset) to represent the true data distribution. Given that direct computation of KL divergence is not feasible without knowledge of $p_{\text{data}}(\mathbf{x})$, DGMs typically rely on maximizing the likelihood of the observed data under the model. This approach aligns with the principle of maximum likelihood estimation (MLE), which posits that the optimal model parameters θ are those that maximize the likelihood of the training data appearing under the model's assumed

probability distribution. Mathematically, this is represented as:

$$\theta^* = \arg\max_{\theta} \prod_{i=1}^{N} p_{\text{model}}(\mathbf{x}_i; \theta), \tag{2.2}$$

where $\mathbf{x}_i$ denotes an instance of the training data. To simplify calculations and enhance numerical stability, we typically work with the logarithm of the likelihood function, transforming the product into a summation. This is referred to as the log-likelihood, and it modifies our optimization problem to:

$$\theta^* = \arg\max_{\theta} \sum_{i=1}^{N} \log p_{\text{model}}(\mathbf{x}_i; \theta), \tag{2.3}$$

which is essentially the mean log-likelihood over all N data points in the training dataset. This is the empirical estimate of the expected log-likelihood under the data distribution $p_{\text{data}}(x)$:

$$\theta^* = \arg\max_{\theta} \mathbb{E}\left[\log p_{\text{model}}(\mathbf{x}; \theta)\right] = \arg\max_{\theta} \mathbb{E}\left[\log p_{\theta}(\mathbf{x})\right]. \tag{2.4}$$

2.2. DGM Classification

DGMs can be classified into different categories based on their approach to maximizing data likelihood. The first one is **explicit density models** that directly model the probability distribution $p_\theta(\mathbf{x})$. These models use the likelihood function during training to align the generated data distribution with the true data distribution. In contrast, **implicit density models** do not explicitly use $p_\theta(\mathbf{x})$ during training. Instead, they employ alternative methods that theoretically achieve the goal of maximizing data likelihood by implicitly capturing the data distribution. Explicit density models can be further categorized into **tractable density models** and **intractable density models** based on whether the likelihood computation during training is directly calculable or must be approximated.

2.2.1 Explicit Tractable Density Models

Explicit Tractable Density Models perform direct and explicit computations of the likelihood, making them highly interpretable and theoretically robust. Examples include:

- Autoregressive Models: Examples include PixelRNN (Van Den Oord et al., 2016), PixelCNN (Van den Oord et al., 2016), and NADE (Uria et al., 2016). In these models, each output is generated sequentially with each output conditioned on previous outputs. This sequential generation ensures that the likelihood of each output can be directly calculated.

- Normalizing Flows: These models utilize invertible transformations to map complex data distributions to simpler, known parametric distributions (such as Gaussian distribution). The invertibility of these transformations allows for the exact computation of the likelihood of any given input, making these models particularly powerful for tasks requiring detailed density estimation.

2.2.2 Explicit Intractable Density Models

Explicit Intractable Density Models aim to directly model $p_\theta(\mathbf{x})$, but they rely on approximation methods due to the infeasibility of exact computation in complex data scenarios. These models include:

- Variational Autoencoders (VAEs): VAEs use an encoder-decoder architecture where the encoder approximates the posterior distribution of latent variables, and the decoder approximates the data distribution conditioned on these latent variables. The likelihood is indirectly maximized through a lower bound estimate, known as the Evidence Lower Bound (ELBO).

- Diffusion Models: These models gradually convert data into a known noise distribution and learn to reverse this process. Training involves approximating the reverse of this diffusion process, which indirectly maximizes the data likelihood.

- Score-Based Generative Models: These models learn the score (gradient of log probability) of the data distribution and use it to generate samples by iteratively refining noise samples. The model indirectly maximizes the likelihood through the score-matching technique.

2.2.3 Implicit Density Models

Implicit Density Models do not explicitly use $p_\theta(\mathbf{x})$ during training. Instead, they employ alternative approaches that can theoretically align with the goal of maximizing data likelihood by implicitly capturing the data distribution.

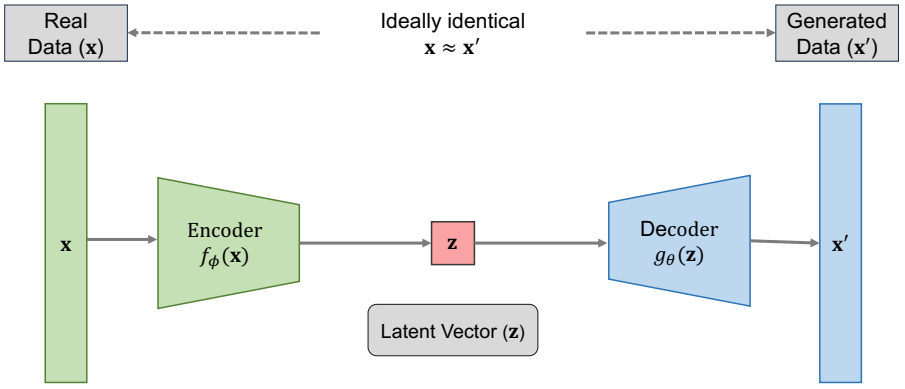
- Generative Adversarial Networks (GANs): GANs consist of two competing networks—a generator and a discriminator. The generator produces samples aimed at being indistinguishable from real data, while the discriminator evaluates their authenticity. Even though $p_\theta(\mathbf{x})$ is not explicitly computed, the adversarial training process theoretically leads to the generator capturing the true data distribution, resulting in implicitly maximizing the data likelihood.

2.3. Variational Autoencoder

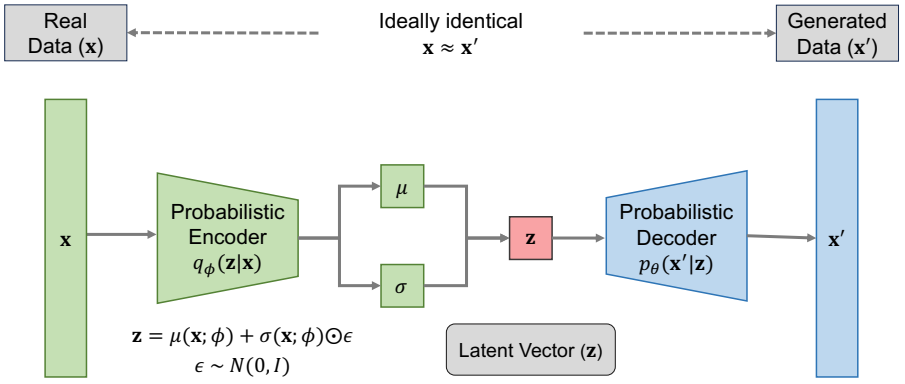
Variational Autoencoders (VAEs) are a class of DGMs that combine the structure of Autoencoders with variational inference. Since the introduction in Kingma and Welling (2013), VAEs have significantly influenced the overall field of generative models. As shown in Figure 1 (a), Autoencoders (AEs) typically comprise two main components: an encoder and a decoder. AEs were first introduced by Hinton and Salakhutdinov (2006) for data compression and dimensionality reduction. Specifically, AEs compress input data $\mathbf{x}$ into a lower-dimensional latent representation $\mathbf{z}$ using the ϕ-parameterized encoder $f_\phi(\mathbf{x}) = \mathbf{z}$ and reconstruct the input data from the latent representation with the θ-parameterized decoder $g_\theta(\mathbf{z}) = \mathbf{x}$. The objective function of an AE is typically:

$$\mathcal{L}_{AE} = \|\mathbf{x} - \mathbf{x}'\|_2^2 = \|\mathbf{x} - g_\theta(f_\phi(\mathbf{x}))\|_2^2, \tag{2.5}$$

where $\mathbf{x}'$ is the reconstructed data of $\mathbf{x}$.



(a) Model architecture of Autoencoder (AE)



(b) Model architecture of Variational Autoencoder (VAE)

Figure 1: Schematic overview of AE and VAE. Here, $\mathbf{x}$ is the real data and $\mathbf{x}'$ denotes the generated data, and $\mathbf{z}$ represents the latent vector.

In contrast, the primary objective of VAEs is data generation, focusing on training a high-performing decoder. The decoder samples from the latent space to generate new data that resemble the original training dataset. Unlike traditional AEs that learn deterministic functions, f_ϕ and g_θ, VAE model probabilistic (or generative) encoder and decoder, $q_\phi(\mathbf{z}|\mathbf{x})$ and $p_\theta(\mathbf{x}|\mathbf{z})$, by introducing intermediate latent variable z and using variational inference as shown in Figure 1 (b). The ϕ-parameterized encoder, represented by $q_\phi(\mathbf{z}|\mathbf{x})$, transforms input data $\mathbf{x}$ into a latent representation $\mathbf{z}$ in a lower-dimensional latent space. Notably, instead of learning a deterministic mapping to a single latent vector, VAE's encoder estimates the parameters of a probability distribution in the latent space. The θ-parameterized decoder, represented by $p_\theta(\mathbf{x}|\mathbf{z})$, takes the encoded latent vector $\mathbf{z}$ and reconstructs the original data point $\mathbf{x}$.

Following the detailed derivations in Appendix B, a new objective function called the Evidence Lower BOund (ELBO), $\mathcal{L}(\mathbf{x};\theta,\phi)$[1], can be defined as:

$$\log p_\theta(\mathbf{x}) \geq \mathcal{L}(\mathbf{x};\theta,\phi) = \mathbb{E}_\mathbf{z}\left[\log p_\theta(\mathbf{x}|\mathbf{z})\right] - D_{KL}\left(q_\phi\left(\mathbf{z}|\mathbf{x}\right)||p(\mathbf{z})\right). \tag{2.6}$$

Equation (2.6) demonstrates that the ELBO is a *'lower bound'* to the log-likelihood of the data. Therefore, maximizing the ELBO with respect to the parameters of the encoder and decoder, ϕ and θ, also maximizes the true log-likelihood ($\log p_\theta(\mathbf{x})$). The ELBO can be efficiently estimated using stochastic gradient-based optimization methods without involving the intractable posterior $p_\theta(\mathbf{z}|\mathbf{x})$. The first term of the right hand side of Equation (2.6), $\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x}|\mathbf{z})\right]$, is called *reconstruction error*, which measures the VAE's capability to accurately reconstruct the input data from its latent variable generated by the encoder network. The second term, $D_{KL}\left(q_\phi\left(\mathbf{z}|\mathbf{x}\right)||p(\mathbf{z})\right)$, is called *regularization*, which focuses on aligning the distribution of the latent variables with the assumed distribution.

A key challenge in training VAEs is estimating the gradient of the expected value term in the ELBO to the encoder network parameters ϕ. This involves differentiating through the random sampling operation involved in generating the latent variable $\mathbf{z}$, which is inherently non-deterministic and does not permit direct differentiation. The *reparameterization trick* addresses this challenge by reparameterizing the random variable $\mathbf{z}$ such that the randomness is independent of the parameters. This enables the model to backpropagate gradients through the deterministic part of the reparameterization, while the stochasticity is handled separately. Instead of sampling $\mathbf{z}$ directly from $q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mu(\mathbf{x};\phi), \sigma(\mathbf{x};\phi)^2)$, the model samples a standard Gaussian noise variable $\epsilon \sim \mathcal{N}(0, I)$ and reparameterizes $\mathbf{z}$ as follows:

$$\mathbf{z} = \mu(\mathbf{x};\phi) + \sigma(\mathbf{x};\phi) \odot \epsilon, \tag{2.7}$$

where $\odot$ denotes the element-wise multiplication. The gradients with respect to ϕ can be computed since the sampling operation is deterministic for ϕ, and all randomness is relegated to ϵ.

Conditional VAEs (CVAEs) (Sohn et al., 2015) is one of the extensions of the standard VAEs. The core innovation of CVAEs lies in their ability to incorporate additional conditional information, often in the form of labels or related data, directly into the generative process. This conditioning allows the model to generate more targeted and context-specific data. Unlike traditional VAEs, CVAEs leverage the extra conditional information to produce outputs that are not only high in quality but also relevant to the specified conditions. This is achieved by modifying both the encoder and decoder to accept and process the conditional information alongside the input data. As a result, CVAEs have a degree of control and specificity in the generated output, such as generating images of a certain class or style, customizing text generation, and enhancing recommendation systems. They also offer benefits in interpretability and the potential for disentangled representation learning. The CVAE framework has thus emerged as a powerful tool for controlling the generated output and improving the model performance. Applications of CVAE in transportation research varies from traffic data generation, prediction and classification.

2.4. Generative Adversarial Networks

Generative Adversarial Networks (GANs) are another class of generative models, introduced in Goodfellow et al. (2014). GANs have attracted considerable attention due to their capability to generate highly realistic data, and they have been widely used in a wide range of applications, including image synthesis, text generation, and data augmentation. Unlike other types of DGMs, GANs do not *explicitly* model data distribution or data likelihood. Instead, they employ a game-theoretical approach to train a model focused on generating realistic data.

As shown in Figure 2, GANs consist of two primary components: a *Generator* (G) and a *Discriminator* (D). The Generator's task is to produce realistic synthetic data, while the Discriminator focuses on distinguishing between real

[1] Typically, we train θ and ϕ jointly, but we differentiate the notation to distinguish parameters for encoder and decoder.

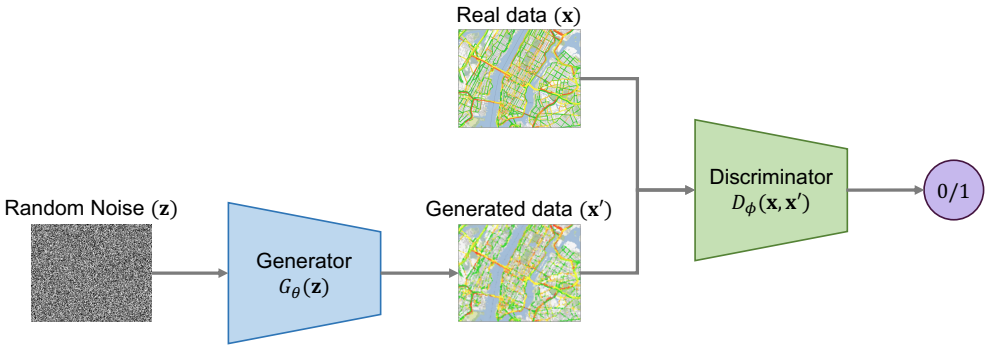


Figure 2: Schematic overview of Generative Adversarial Network (GAN). Here, $\mathbf{x}$ is the real data, $\mathbf{x}'$ denotes the generated data, and $\mathbf{z}$ represents the random noise.

data and synthetic data generated by the Generator. This competitive dynamic resembles a two-player min-max game, where the Generator attempts to deceive the Discriminator, and the Discriminator endeavors to avoid being deceived. Formally, let the parameters of the Generator G be denoted as θ and the parameters of the Discriminator D be denoted as ϕ. The Generator G takes as input a random noise vector $\mathbf{z}$, sampled from a known distribution (often a standard normal distribution), and transforms it into a data instance $\mathbf{x} = G_\theta(\mathbf{z})$. The Generator is similar to the decoder of VAE in that both aim to learn the mapping from random noise to the data sample. The Discriminator D takes as input a data instance, which can either be real data $\mathbf{x}$ or generated data $G_\theta(\mathbf{z})$, and outputs a scalar representing the probability that the input data is real, denoted as $D_\phi(\mathbf{x})$. The learning process is guided by a value function $V(\phi, \theta)$, defined as:

$$V(\phi, \theta) = \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}(\mathbf{x})}[\log D_\phi(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}[\log(1 - D_\phi(G_\theta(\mathbf{z})))], \tag{2.8}$$

where p_{data} represents the true data distribution, and $p(\mathbf{z})$ denotes the distribution of the input noise vectors. The first and second terms in the value function correspond to the log probability of the Discriminator correctly identifying real and generated data, respectively. The learning process involves finding optimal parameters, ϕ^* and θ^*, for both D and G, through gradient-based methods as shown below. This process alternates between the following two steps:

$$\min_\theta \max_\phi V(\phi, \theta), \tag{2.9}$$

1. Maximizing $V(\phi, \theta)$ with respect to ϕ: This step enhances the Discriminator's capability to distinguish between real and generated data.
2. Minimizing $V(\phi, \theta)$ with respect to θ: This step improves the Generator's ability to generate synthetic data that can deceive the Discriminator.

This process does not explicitly involve the evaluation of the likelihood function; however, it implicitly maximizes the data likelihood and minimizes the difference between true data distribution p_{data} and model distribution p_{model}.

A detailed derivation of how GAN formulation matches with maximizing likelihood is shown in Appendix C.

Conditional GANs (Mirza and Osindero, 2014), commonly known as cGANs, are an advanced adaptation of the traditional GAN framework. The primary motivation behind cGANs is to give more control over the data generation process. While traditional GANs are adept at generating data from a learned distribution, cGANs add an additional layer of conditioning, usually based on labels or external information. For example, in the case that the user wants to generate two distinctive classes of outputs, standard GANs would require two separate models to learn each distribution or, at best, generate a mix without explicit control over the output. On the other hand, cGANs allow the model to generate images of a specific category based on a label provided to it. The beauty of cGANs lies in the fact that both the generator and the discriminator are conditioned on this label. The generator utilizes this label to produce data, while the discriminator uses it to discern the authenticity of the generated data. The mathematical representation encapsulates this conditioning through additional input layers for the generator and discriminator. When producing an output sample, the generator does not merely rely on the random noise vector but also factors in the label, ensuring the generated output aligns with the desired category. On the flip side, the discriminator evaluates both the output and its

corresponding label, ensuring that fake (generated) data is not only realistic but is also consistent with its associated type.

Generative Adversarial Imitation Learning (Ho and Ermon, 2016), or GAIL, is an innovative fusion of imitation learning principles with the GAN framework. Imitation learning, at its core, revolves around the idea of learning a policy (or behavior) by using experts' demonstrations. GAIL captures this essence but employs the adversarial training approach of GANs to achieve it. In the GAIL framework, the generator is no longer producing static data; instead, it produces sequences of actions aiming to imitate an expert's behavior. Meanwhile, the discriminator's role is to distinguish between sequences produced by the generator and those exhibited by the expert. Traditional imitation learning often relies on methods like behavioral cloning, where the agent directly learns from expert trajectories. However, this method is prone to compounding errors and struggles with situations not encountered during training. GAIL, on the other hand, captures the underlying reward structure by using the adversarial training framework.

Similar to the concept of cGANs, which integrate additional conditioning information to guide the data generation process, conditional GAIL (cGAIL) (Zhang et al., 2020a) extends the GAIL framework by conditioning both the generator and the discriminator on extra contextual variables. These variables can include environmental factors, state features, or operational conditions that are critical in accurately capturing expert behavior. By incorporating such information, cGAIL is better equipped to model context-dependent behaviors, leading to improved performance and robustness in dynamic, real-world settings.

Adversarial Autoencoders (AAEs) (Makhzani et al., 2015) represent an innovative fusion of VAEs and GANs, designed to enhance the capabilities of generative models. The primary advantage of AAEs lies in their ability to impose arbitrary prior distributions on the latent space, going beyond the Gaussian priors typically used in VAEs. This flexibility allows for the modeling of complex data distributions more effectively. AAEs consist of a standard autoencoder architecture, complemented by an adversarial network that enforces the latent space to conform to the chosen prior distribution. To be more specific, the fundamental condition in VAE loss is that KL divergence between the distribution of encoded input data and prior can be calculated. The function of KL divergence in loss term is making the distribution of encoded input data $q_\phi(\mathbf{z}|\mathbf{x})$ the same as the prior distribution $p(\mathbf{z})$, which is the same logic of GAN. Therefore, AAE utilizes the discriminator term of GAN to replace the KL divergence of VAE, making it possible to use arbitrary prior distribution. Through adversarial training, AAEs generate sharper, more detailed outputs compared to traditional VAEs, especially noticeable in tasks like image generation. They are versatile in applications, ranging from semi-supervised learning to unsupervised clustering and anomaly detection. Additionally, AAEs offer better control over the generation process, including conditional generation and the learning of disentangled representations, making them suitable for tasks requiring precise control and interpretability. The integration of adversarial principles into autoencoders has thus positioned AAEs as a powerful and flexible tool in the realm of generative modeling, addressing key challenges of traditional VAEs and opening new avenues in machine learning research.

2.5. Normalizing Flows (Flow-based Generative Models)

A flow-based deep generative model learns the likelihood by using *normalizing flows*. A normalizing flow describes a method for constructing a complex distribution by using a series of invertible mapping functions (Rezende and Mohamed, 2015). The idea behind using normalizing flows to learn the likelihood of the real data is that a complex distribution can be learned by sequentially applying multiple normalizing flows to a simple base distribution (e.g., Gaussian distribution).

As shown in Figure 3, for a multivariate vector $\mathbf{x}$, the objective of normalizing flow is to learn the probability density function $p(\mathbf{x})$. The density $p(\mathbf{x})$ is transformed into a simple distribution $p(\mathbf{z}_0)$ (independent multivariate Gaussian distribution). This transformation f is composed of K invertible (bijective) functions $f_i, i \in \{1, ...K\}$. This transformation can be denoted as follows:

$$\mathbf{x} = \mathbf{z}_K = f_K \circ f_{K-1} \circ \cdots \circ f_1(\mathbf{z}_0) = f(\mathbf{z}_0), \tag{2.10}$$

where we denote $\mathbf{z}_i$ as latent vector after applying i invertible functions $(f_1, \cdots, f_i)$.

By definition,

$$\mathbf{z}_i = f_i(\mathbf{z}_{i-1}), \text{ thus } \mathbf{z}_{i-1} = f_i^{-1}(\mathbf{z}_i). \tag{2.11}$$

The log-likelihood of probability density function of i-th latent vector, $p(\mathbf{z}_i)$, can be calculated as follows:

$$\int p_i(\mathbf{z}_i)\, d\mathbf{z}_i = \int p_{i-1}(\mathbf{z}_{i-1})\, d\mathbf{z}_{i-1} = 1 \quad \Rightarrow \textit{ integration property of probability distribution}$$

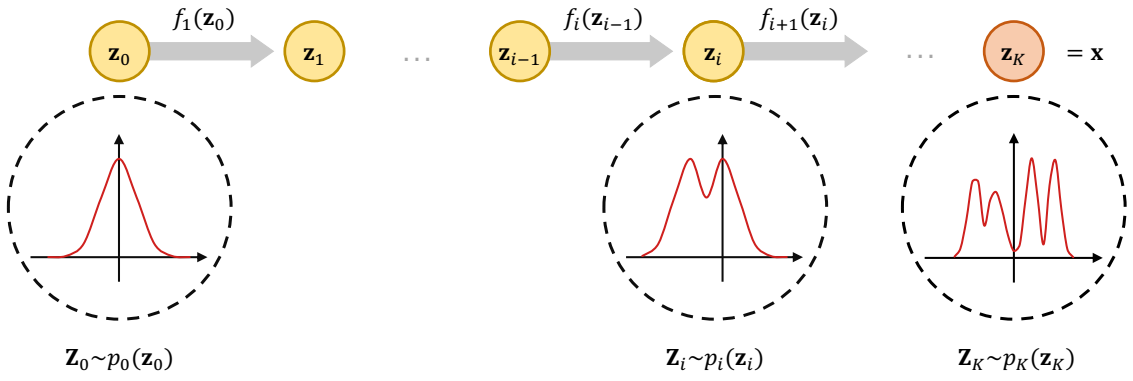


Figure 3: Schematic overview of normalizing flow. Here, $\mathbf{z}_0$ is a simple, known distribution (such as a standard Gaussian), $\mathbf{z}_i$ represents an intermediate distribution, and $\mathbf{z}_K$ denotes the target distribution.

$$
\begin{aligned}
p_i(\mathbf{z}_i) &= p_{i-1}\left(f_i^{-1}(\mathbf{z}_i)\right)\left|\det\left(\frac{df_i^{-1}}{d\mathbf{z}_i}\right)\right| && \Rightarrow \textit{ Change of Variable Theorem} \\
&= p_{i-1}\left(\mathbf{z}_{i-1}\right)\left|\det\left(\frac{df_i}{d\mathbf{z}_{i-1}}\right)^{-1}\right| && \Rightarrow \textit{ Inverse Function Theorem} \\
&= p_{i-1}\left(\mathbf{z}_{i-1}\right)\left|\det\left(\frac{df_i}{d\mathbf{z}_{i-1}}\right)\right|^{-1} && \Rightarrow \textit{ property of Jacobians of invertible function} \\
\log p_i(\mathbf{z}_i) &= \log p_{i-1}\left(\mathbf{z}_{i-1}\right) - \log\left|\det\left(\frac{df_i}{d\mathbf{z}_{i-1}}\right)\right|. && \Rightarrow \textit{ by applying logarithm to both sides}
\end{aligned}
\tag{2.12}
$$

As a results, the probability density of $\mathbf{x}$ can be calculated as follows:

$$
\begin{aligned}
\log p(\mathbf{x}) = \log \pi_K(\mathbf{z}_K) &= \log p_{K-1}(\mathbf{z}_{K-1}) - \log\left|\det\left(\frac{df_K}{d\mathbf{z}_{K-1}}\right)\right| \\
&= \log p_{K-2}(\mathbf{z}_{K-2}) - \log\left|\det\left(\frac{df_{K-1}}{d\mathbf{z}_{K-2}}\right)\right| - \log\left|\det\left(\frac{df_K}{d\mathbf{z}_{K-1}}\right)\right| \\
&= \cdots \\
&= \log p_0(z_0) - \sum_{i=1}^{K} \log\left|\det\left(\frac{df_i}{d\mathbf{z}_{i-1}}\right)\right|.
\end{aligned}
\tag{2.13}
$$

Finally, the objective of training for a generative model is maximizing the likelihood of a training set, $\mathcal{D}$, as follows:

$$
\mathcal{L}(\mathcal{D}) = -\frac{1}{|\mathcal{D}|}\sum_{x\in\mathcal{D}} \log p(\mathbf{x}). \tag{2.14}
$$

In practice, there are two conditions to consider when deciding proper functions for normalizing flows. The first condition is that by definition, the functions should be invertible. Also, computing the Jacobian determinant should be feasible since usually computing the Jacobian of functions and computing the determinant are both computationally expensive. As a result, properly defining the function f is the key to normalizing flows.

There are several feasible functions from previous studies that satisfy the constraints: linear function (Rezende and Mohamed, 2015), 1×1 convolution (Kingma and Dhariwal, 2018), and affine coupling layer (Dinh et al., 2014; Durkan et al., 2019). In this paper, we introduce the affine coupling layer as an example since this is one of the widely

used forms due to its ability to deal with high-dimensional data such as image and sound. The affine coupling layer from Dinh et al. (2016) can be formulated as follows:

$$\begin{aligned} \mathbf{y}_{1:d} &= \mathbf{x}_{1:d}, \\ \mathbf{y}_{d+1:D} &= \mathbf{x}_{d+1:D} \odot \exp\left(s(\mathbf{x}_{1:d})\right) + t(\mathbf{x}_{1:d}), \end{aligned} \tag{2.15}$$

where $\mathbf{y}_{1:d}$ is the first d elements of $\mathbf{y}$, $\mathbf{y}_{d+1:D}$ is the rest of $\mathbf{y}$, $\mathbf{x}_{1:d}$ is the first d elements of $\mathbf{x}$, $\mathbf{x}_{d+1:D}$ is the rest of $\mathbf{x}$, s and t are scale and translation functions, which are usually implemented as multi-layer perceptrons (MLP), and $\odot$ is the element-wise multiplication operator. As noted in Dinh et al. (2016), the affine coupling layer is an invertible function and guarantees fast computation of the determinant of the Jacobian matrix.

First, the inverse function of Equation (2.15) is as follows:

$$\begin{aligned} \mathbf{x}_{1:d} &= \mathbf{y}_{1:d}, \\ \mathbf{x}_{d+1:D} &= \left(\mathbf{y}_{d+1:D} - t\left(\mathbf{y}_{1:d}\right)\right) \odot \exp\left(-s\left(\mathbf{y}_{1:d}\right)\right). \end{aligned} \tag{2.16}$$

Also, the Jacobian matrix of this transformation is

$$\frac{\partial \mathbf{y}}{\partial \mathbf{x}^\top} = \begin{bmatrix} \mathbb{I}_d & 0 \\ \frac{\partial \mathbf{y}_{d+1:D}}{\partial \mathbf{x}_{1:d}^\top} & \operatorname{diag}\left(\exp\left[s(\mathbf{x}_{1:d})\right]\right) \end{bmatrix}, \tag{2.17}$$

where $\mathbb{I}_d$ is a $d \times d$ identity matrix, $\operatorname{diag}(*)$ is the diagonal matrix whose diagonal entries correspond to the given matrix and non-diagonal entries are zero.

As a result, the Jacobian matrix shown in Equation (2.17) is a lower-triangular matrix. Therefore, the determinant of the Jacobian matrix can be calculated as follows:

$$\left|\det\left(\frac{\partial \mathbf{y}}{\partial \mathbf{x}^\top}\right)\right| = \prod_{i=1}^{D}\left[\frac{\partial \mathbf{y}}{\partial \mathbf{x}^\top}\right]_{i,i} = \prod_{i=1}^{d}\exp\left[s(\mathbf{x}_{1:d})\right]_{i,i} = \exp\left(\sum_{i=1}^{d}\left[s(\mathbf{x}_{1:d})\right]_{i,i}\right), \tag{2.18}$$

and

$$\log\left|\det\left(\frac{\partial \mathbf{y}}{\partial \mathbf{x}^\top}\right)\right| = \sum_{i=1}^{d}\left[s(\mathbf{x}_{1:d})\right]_{i,i}, \tag{2.19}$$

where $[*]_{i,i}$ is i-th diagonal entry of the given matrix.

2.6. Score-Based Generative Models

Score-based generative models are a class of generative models that leverage the concept of "score" of a probability density function to generate new data samples. The term *score* is used in a statistical context, referring to the gradient (or intuitively, *slope*) of the log-likelihood function with respect to the data, $\mathbf{x}$. Formally, the score is denoted as $\nabla_{\mathbf{x}} \log p(\mathbf{x})$, as shown in Figure 4. Intuitively, the score tells us how to change $\mathbf{x}$ to increase the probability of $\mathbf{x}$ under the model's current estimate of the probability density function. A higher score indicates a direction in which the data point $\mathbf{x}$ could be adjusted to make it more likely under the model.

In score-based generative models, this concept is used to iteratively refine samples from an initial noise distribution, guiding them towards regions of higher probability under the target distribution (i.e., the distribution of the training data). This is achieved by estimating the score function at various points in the data space and using it to perform gradient ascent on the log-likelihood landscape. Essentially, by following the direction of the score, new data samples can be generated that are likely under the model's learned distribution, thus resembling the characteristics of the training data.

The goal of (deep) generative models is to learn the underlying data distribution $p(\mathbf{x})$ given training data samples drawn from true data distribution, $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N\}$ with $\mathbf{x}_i \sim p(\mathbf{x}), \forall i \in 1, \cdots, N$. The score-based generative models (Song and Ermon, 2019; Song et al., 2020b) focuses on the development of a "score network," denoted as $s_\theta(\mathbf{x})$, which is essentially a θ-parameterized neural network designed to approximate the score of the data distribution; that is, $s_\theta(\mathbf{x})$

aims to be a close estimate of $\nabla_{\mathbf{x}} \log p(\mathbf{x})$. The score network can be trained by minimizing the squared $L2$ distance between the ground-truth score and the estimated score by the score network as follows:

$$\mathbb{E}_{\mathbf{x}\sim p(\mathbf{x})} \|\nabla_{\mathbf{x}} \log p(\mathbf{x}) - s_\theta(\mathbf{x})\|_2^2. \tag{2.20}$$

A significant challenge with this approach is that it involves direct access to $\nabla_{\mathbf{x}} \log p(\mathbf{x})$, which is not practically feasible. To address this issue, a substantial body of research has been dedicated to developing alternative methodologies, collectively known as "score matching" (Hyvärinen and Dayan, 2005; Vincent, 2011; Song et al., 2020a) that minimizes Equation (2.20) without having to have access to the ground-truth data score.

One of the main concerns in score-based generative models is figuring out how the *score network* can be efficiently and accurately trained. Song and Ermon (2019) identified several challenges in training the score network. One of the main challenges is that since most training data is located in a small subspace of the high dimensional data space, training the score network in low data density regions can be challenging.

Therefore, Song and Ermon (2019) proposed to perturb the data space using a scheduled Gaussian noise and learn the score function using the perturbed data following Vincent (2011). This approach is also called "denoising score matching," and as shown in Vincent (2011), if the perturbation is small enough, learning the score function of a perturbed data distribution ($q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})$) is almost surely equivalent to learning the actual score function of $p(\mathbf{x})$, i.e., $\nabla_{\mathbf{x}} \log q_\sigma(\tilde{\mathbf{x}}|\mathbf{x}) \approx \nabla_{\mathbf{x}}$. Therefore, the updated learning objective is as follows:

$$\begin{aligned} &\mathbb{E}_{\mathbf{x}\sim p(\mathbf{x})}\left[\mathbb{E}_{\tilde{\mathbf{x}}\sim q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})}\left\|\nabla_{\mathbf{x}} \log q_\sigma(\tilde{\mathbf{x}}|\mathbf{x}) - s_\theta(\tilde{\mathbf{x}})\right\|_2^2\right] \\ =&\mathbb{E}_{\mathbf{x}\sim p(\mathbf{x})}\left[\mathbb{E}_{\tilde{\mathbf{x}}\sim \mathcal{N}(\mathbf{x},\sigma^2\mathbb{I})}\left\|\frac{\tilde{\mathbf{x}}-\mathbf{x}}{\sigma^2} - s_\theta(\tilde{\mathbf{x}})\right\|_2^2\right], \end{aligned} \tag{2.21}$$

where, $\tilde{\mathbf{x}}$ is the perturbed data with Gaussian distribution with standard deviation σ. Song and Ermon (2019) further proposed the use of variance scheduling for data perturbation. Formally, we can define a set of standard deviation of data perturbation, $\{\sigma_t\}_{t=1}^L$, that satisfies $\frac{\sigma_1}{\sigma_2} = \cdots = \frac{\sigma_{L-1}}{\sigma_L} > 1$. Specifically, we start from a large enough data perturbation that can cover (or generate data in) the low data density regions in the data space, and then we gradually decrease the perturbation so that the learned score function can match with the actual score function. Intuitively, we can interpret the large perturbation as learning the general trend or landscape of the score function. In contrast, we can interpret the small perturbation as learning the details of the actual score function.

Then, the score network is trained given the current noise level (σ_t) and this is called the *Noise Conditional Score Network*. For a given σ, the updated learning objective is:

$$\mathbb{E}_{\mathbf{x}\sim p(\mathbf{x})}\left[\mathbb{E}_{\tilde{\mathbf{x}}\sim \mathcal{N}(\mathbf{x},\sigma^2\mathbb{I})}\left\|\frac{\tilde{\mathbf{x}}-\mathbf{x}}{\sigma^2} - s_\theta(\tilde{\mathbf{x}},\sigma)\right\|_2^2\right]. \tag{2.22}$$

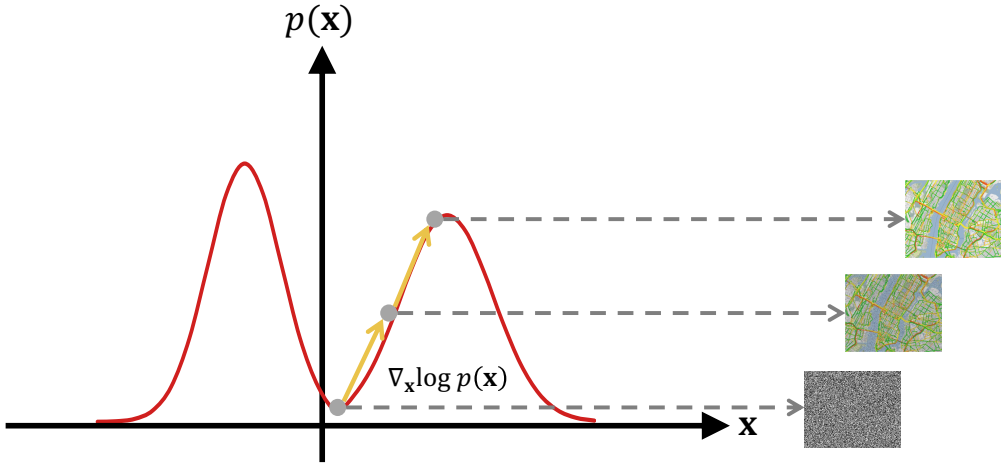


Figure 4: Schematic overview of score-based generative model. Here, $\mathbf{x}$ denotes a data sample from the underlying distribution, and $\nabla_{\mathbf{x}} \log p(\mathbf{x})$ represents the score function.

Once the score network is trained, data generation can be accomplished through an iterative method known as *Langevin dynamics*, or Langevin Monte Carlo (Parisi, 1981; Grenander and Miller, 1994). Langevin dynamics is originally formulated to describe the behavior of molecular systems, such as Brownian motion. A more detailed derivation of equations for Langevin dynamics in the generative model setting is presented in Appendix E. Given a fixed step size $\epsilon > 0$, and an initial random noise $\mathbf{x}$, the following iterative procedure can be used:

$$\mathbf{x}_{i+1} \leftarrow \mathbf{x}_i + \nabla_x \log p(\mathbf{x}) \cdot \epsilon + \sqrt{2\epsilon} \cdot \mathbf{z}_i, \tag{2.23}$$

where $\mathbf{z}_i \sim \mathcal{N}(0, I)$. Since the values from the trained score network should match the true score, we can use the following iterative procedure to draw samples from the distribution of interest:

$$\mathbf{x}_{i+1} \leftarrow \mathbf{x}_i + s_\theta(\mathbf{x}_i) \cdot \epsilon + \sqrt{2\epsilon} \cdot \mathbf{z}_i. \tag{2.24}$$

2.7. Diffusion Models

Diffusion models are a class of DGMs that have shown a remarkable ability to generate high-quality samples across various domains. The core concept behind diffusion models is inspired by the physical process of "diffusion," where particles move from areas of higher concentration to lower concentration until they reach an equilibrium. Diffusion models work by gradually adding noise to data until it becomes indistinguishable random noise (known as a forward diffusion process, or diffusion process), and then learning the reverse of this process of denoising the data (known as a reverse diffusion process, or generation process).

Figure 5 shows the general framework of the diffusion model. We start from $\mathbf{x}_0$ which is the real data. At each diffusion step, we add a small noise. After a sufficiently large number of steps (here we denote it as k), the data is nearly complete noise as shown in $\mathbf{x}_k = \mathbf{z}$. The forward process of adding noise is given, and the main 'learning' part of the diffusion model is to learn the reverse process which transforms the complete noise $\mathbf{z}$ to the data $\mathbf{x}_0$. This is similar to VAE, GAN, and Normalizing Flows that we start from a complete noise (or well-known distribution such as Gaussian distribution or uniform distribution) and we learn to transform the noise into the data distribution $p(\mathbf{x})$. The main difference is the idea inspired by the physical "diffusion" process and denoising process which recovers the data from a complete noise.

Ho et al. (2020) presents the *Denoising Diffusion Probabilistic Model*, or DDPM, which extends the diffusion model framework by incorporating a denoising autoencoder-like architecture. This architecture allows for more efficient learning of the reverse diffusion process, enabling the model to generate high-fidelity samples from complex data distributions.

The forward diffusion process is designed to corrupt the original data $\mathbf{x}_0$ over a series of time steps $t = 1, 2, \ldots, T$. At each step t, Gaussian noise is added to the data, which is mathematically described by the conditional distribution $q(\mathbf{x}_t|\mathbf{x}_{t-1})$. This distribution specifies how the data at step t is generated from the data at the previous step $t-1$.

Formally, the forward process is defined as:

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t \mathbb{I}), \tag{2.25}$$

where $q(\mathbf{x}_t|\mathbf{x}_{t-1})$ follows the conditional Gaussian distribution with mean $\sqrt{1-\beta_t}\mathbf{x}_{t-1}$ and covariance $\beta_t \mathbb{I}$, t is the diffusion timestep, and β_t is a time-dependent coefficient that controls the amount of data retained from the previous

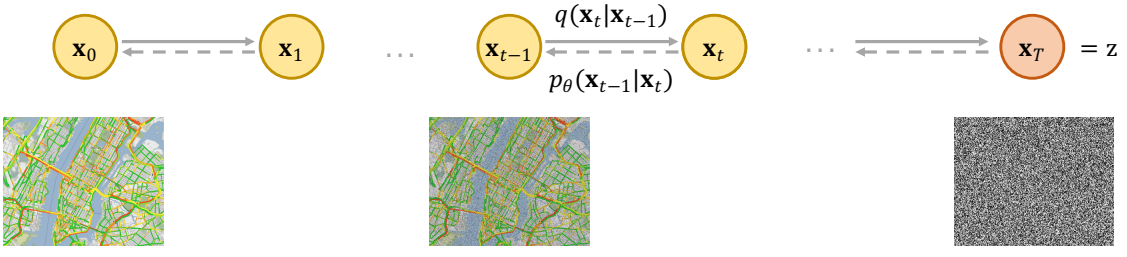


Figure 5: Schematic overview of diffusion model. Here, $\mathbf{x}_0$ represents the original data, $\mathbf{x}_t$ represents the intermediate noisy states, and $\mathbf{x}_T$ represents the noise data (e.g., Gaussian distribution).

step where $0 < \beta_t \ll 1$. The choice of $\sqrt{1-\beta_t}$ and β_t ensures that the variance of each diffusion step is maintained at 1. This is important because it stabilizes the diffusion process, preventing the variance from either exploding or vanishing over time.

Based on the reparameterization trick, the transition from $\mathbf{x}_{t-1}$ to $\mathbf{x}_t$ can be also expressed as:

$$\mathbf{x}_t = \sqrt{1-\beta_t}\mathbf{x}_{t-1} + \beta_t \boldsymbol{\epsilon}_t, \tag{2.26}$$

where $\boldsymbol{\epsilon}_t \sim \mathcal{N}(0, I)$ is Gaussian noise with mean 0 and identity covariance matrix.

From Equation (2.25), we can derive the equation for $q(\mathbf{x}_{0:T})$ representing the diffusion trajectory from the original data $\mathbf{x}_0$ to the complete noise $\mathbf{x}_T = z$ based on properties of Markov chain and chain rules as follows:

$$\begin{aligned} &q\left(\mathbf{x}_{0:T}\right) = q(\mathbf{x}_0)\prod_{t=1}^{T} q\left(\mathbf{x}_t|\mathbf{x}_{t-1}\right), \\ &\text{or,} \\ &q\left(\mathbf{x}_{1:T}|\mathbf{x}_0\right) = \prod_{t=1}^{T} q\left(\mathbf{x}_t|\mathbf{x}_{t-1}\right). \end{aligned} \tag{2.27}$$

The reverse diffusion process (the denoising process) represents the generative distribution parameterized with θ. The joint distribution $p_\theta\left(\mathbf{x}_{0:T}\right)$ is defined as a Markov chain with Gaussian transition distributions, $p_\theta\left(\mathbf{x}_{t-1}|\mathbf{x}_t\right)$ starting at $p\left(\mathbf{x}_T\right) = \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbb{I})$ as follows:

$$p_\theta\left(\mathbf{x}_{0:T}\right) = p\left(\mathbf{x}_T\right)\prod_{t=1}^{T} p_\theta\left(\mathbf{x}_{t-1}|\mathbf{x}_t\right), \tag{2.28}$$

where

$$p_\theta\left(\mathbf{x}_{t-1}|\mathbf{x}_t\right) = \mathcal{N}\left(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t)\right). \tag{2.29}$$

Similar to VAE and Normalizing Flows (models using explicit densities for training), the training objective is to maximize the likelihood or minimize the negative log-likelihood

$$\theta^* = \arg\min_\theta \mathbb{E}\left[-\log p_\theta\left(\mathbf{x}_0\right)\right]. \tag{2.30}$$

Instead of directly training the model with negative log-likelihood, we can consider q as the approximate posterior and use the variational bound on negative log-likelihood to train the model. A detailed derivation is presented in Appendix D and Ho et al. (2020). As a result, we can find an upper bound of the optimization

$$\mathbb{E}\left[-\log p_\theta\left(\mathbf{x}_0\right)\right] \leq \mathbb{E}\left[\underbrace{D_{KL}\left(q\left(\mathbf{x}_T|\mathbf{x}_0\right)||p\left(\mathbf{x}_T\right)\right)}_{L_T} + \sum_{t=2}^{T}\underbrace{D_{KL}\left(q\left(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0\right)||p_\theta\left(\mathbf{x}_{t-1}|\mathbf{x}_t\right)\right)}_{L_{t-1}} \underbrace{-\log p_\theta\left(\mathbf{x}_0|\mathbf{x}_1\right)}_{L_0}\right]. \tag{2.31}$$

Therefore, the overall loss function of minimizing the negative log-likelihood in Equation (2.30) is decomposed into several losses, L_T, L_{t-1}, and L_0. Here, L_T is constant since both $q\left(\mathbf{x}_T|\mathbf{x}_0\right)$ and $p\left(\mathbf{x}_T\right)$ are fixed, and therefore, we can ignore this term. Also, in Ho and Ermon (2016), L_0 is explicitly defined by using the characteristics of the image generation problem, and as a result, L_0 can be interpreted as a reconstruction loss of a problem-specific decoder. As a result, the actual learning process of the diffusion model is related to L_{t-1}.

L_{t-1} measures the KL-divergence of $q\left(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0\right)$ from $p_\theta\left(\mathbf{x}_{t-1}|\mathbf{x}_t\right)$. The diffusion process, q, represents the process of adding small noise to the data; i.e., given a less noisy data $\mathbf{x}_{t-1}$, the distribution of a more noisy data $\mathbf{x}_t$. The first term, $q\left(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0\right)$, represents the true denoising process which is derived from the definition of q given the true

data without noise, $\mathbf{x}_0$. What the diffusion models try to learn is the denoising process $p_\theta\left(\mathbf{x}_{t-1}|\mathbf{x}_t\right)$; i.e., given a more noisy data $\mathbf{x}_t$, the distribution of a less noisy data $\mathbf{x}_{t-1}$. As a result, L_{t-1} captures the distributional difference between the true denoising process (given the true data) and the approximated denoising process (without the true data).

Since the diffusion process follows Gaussian distribution, the true reverse process, $q\left(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0\right)$, can be assumed to follow a Gaussian distribution if T is sufficiently large, or $T \to \infty$. Let $q\left(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0\right) = \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0), \tilde{\beta}_t\mathbb{I})$. To derive explicit form of $\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0)$ and $\tilde{\beta}_t$, first, we should derive a closed-form equation for sampling $\mathbf{x}_t$ at an arbitrary timestep t from Equation (2.25) as shown in Appendix D.

Then, Ho et al. (2020) found that using L_{simple} results in better training in empirical testing.

$$L_{simple} = \mathbb{E}_{t,\mathbf{x}_0,\boldsymbol{\epsilon}}\left[\left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta\left(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t\right)\right\|_2^2\right]. \tag{2.32}$$

2.8. Discussion

Different classes of Deep Generative Models (DGMs) use various approaches to model data distributions. Since all models have their imperfections, it's crucial to understand the commonly known pros and cons of each class. Typically, we expect three key things from DGMs: **1) High-Quality Sample Generation** — The ability to produce samples that are indistinguishable from real data; **2) Mode Coverage and Sample Diversity** — The capacity to capture all the variations in the data, including minority modes, ensuring that the generated samples are diverse and representative of the entire data distribution; and **3) Fast and Computationally Efficient Sampling**— The ability to generate samples quickly without requiring extensive computational resources.

However, most DGMs cannot yet simultaneously satisfy these three key requirements, a challenge often referred to as the *Generative Learning Trilemma* (Xiao et al., 2021). This trilemma highlights the inherent trade-offs between high-quality sample generation, mode coverage and diversity, and fast, computationally efficient sampling. For example, Generative Adversarial Networks (GANs) have the advantage of generating high-quality samples rapidly. As discussed in Section 2.4, GANs are renowned for producing realistic samples through a single forward pass of the generator network, making them computationally efficient at inference time. However, common shortcomings of GANs include mode collapse, where the model fails to capture the full diversity of the data distribution, generating samples from only some modes but not all. On the other hand, Variational Autoencoders (VAEs) and Normalizing Flows usually perform well in terms of mode coverage and fast computation time. VAEs promote diverse and representative sampling by modeling the entire data distribution, and they allow for quick sample generation by direct decoding from latent variables. Normalizing Flows also enables efficient sampling and exact density estimation. However, the quality of the samples generated by VAEs and Normalizing Flows may not be as high as those produced by other models. Diffusion Models present another approach, capable of covering diverse modes and generating high-quality samples. Recent advances have enabled diffusion models to produce samples that rival or even surpass GANs in terms of quality and diversity. However, diffusion models typically consist of a large number of denoising steps (often ranging from 500 to 1000), which require substantial computation to generate a single sample. This makes the sampling process slow and computationally expensive compared to other models.

As a result, understanding these trade-offs of using each class of DGM is essential when applying DGMs in practice. Since no current model perfectly satisfies all three key requirements, researchers should choose the model class that best aligns with the specific needs of their application, whether that be sample quality, diversity, or computational efficiency. Additionally, some previous research has explored combining different modeling approaches from several DGMs, such as Xiao et al. (2021) which combined Diffusion models with GAN, Grover et al. (2018) which combined Normalizing Flows with GAN, and Zhang and Chen (2021) which combined Normalizing Flows with Diffusion model. These examples demonstrate that by integrating different DGMs, researchers can also address the limitations imposed by using a single DGM.

3. Deep Generative Models in Different Areas of Transportation Research

DGMs have become important in current research for their ability to model complex data distributions and generate data samples that closely mimic real-world observations. Particularly, in transportation research, DGMs offer tools to simulate, predict, and optimize transportation systems effectively (Huynh and Phung, 2021; Lin et al., 2023a; Yan and Li, 2023). Therefore, in this section, we provide a systematic review of the applications of DGMs in transportation research.

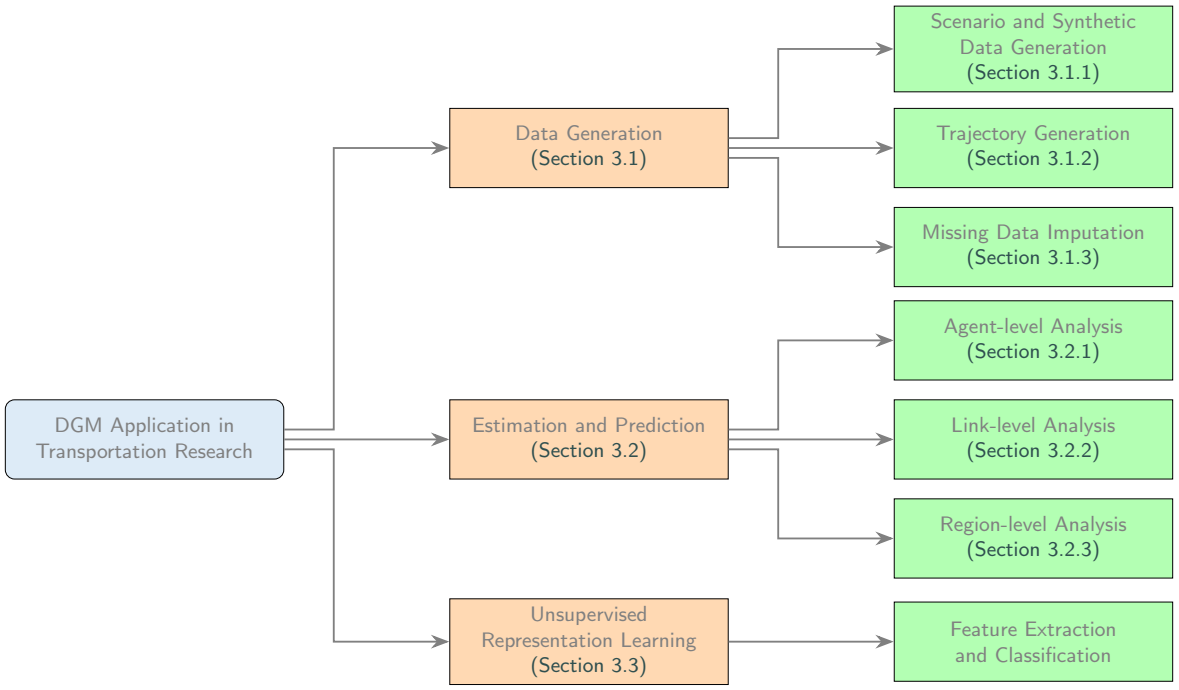


Figure 6: An overview of DGMs in transportation research.

This review focuses on three main areas where DGMs have significantly impacted transportation research: data generation, estimation and prediction, and unsupervised representation learning. By examining their applications in these areas, we gain a clear understanding of how DGMs impacted and will impact transportation research. An overview of this section is shown in Figure 6.

1. **DGM for data generation**: The core functionality of DGMs lies in their ability to generate realistic synthetic data that closely replicate real-world conditions. This capability is particularly valuable in transportation research, where collecting large, high-quality datasets can be challenging and costly. Through the generation of diverse traffic scenarios, DGMs enable researchers to investigate situations that are difficult to capture through traditional data collection methods.
2. **DGM for estimation and prediction**: Beyond generating data, DGMs are also highly effective for estimation and prediction tasks. DGMs learn the underlying distribution of the given (training) data, and as a result, they can model the inherent uncertainties and variations in traffic data. This capability allows DGMs to provide probabilistic estimations and forecasts of traffic states. The probabilistic approach enhances the accuracy, robustness, and reliability of predictions, which support better decision-making in transportation planning and operation.
3. **DGM for unsupervised representation learning**: Most DGMs are designed to learn the transformation function from a sample from a simple distribution (also known as a base distribution) to a target distribution. The 'noisy' data sample from the base distribution can be considered as a *latent representation* of the synthetic data sample. These latent representations can be used for various analyses, such as classifying transportation modes, detecting traffic anomalies, or understanding driving behaviors. This ability to derive insights without requiring labeled data makes DGMs powerful tools in unsupervised learning scenarios.

3.1. DGM for Data Generation

The rapid urbanization of society requires effective traffic management systems to reduce congestion, improve urban planning, and enhance transportation network efficiency (Xu et al., 2015). These systems are fundamentally built on large volumes of high-quality traffic data. Sufficient data volume is essential for accurate analyses and modeling of complex traffic behaviors and patterns, while data quality ensures precise and reliable insights and applications. Despite advancements in technology that have improved data collection capabilities in transportation, traffic sensors

often face issues of sparsity and unreliability, making it challenging to obtain adequate, high-quality data in real-world scenarios (Chen et al., 2003; Ni and Leonard, 2005). DGMs address these challenges by both increasing the volume and improving the quality of traffic datasets. They can generate synthetic data to supplement existing datasets, effectively expanding their size. Additionally, DGMs replicate real-world conditions with high fidelity, which enhances the dataset's qualitative aspects. These capabilities are particularly valuable for transportation research, where traditional data collection methods may be costly, time-consuming, and impractical, especially for capturing rare or non-reproducible scenarios. For example, synthetic data generated by DGMs can simulate a range of traffic conditions, from typical daily commutes to rare events like accidents or extreme weather. Consequently, the synthetic data from DGMs is crucial for developing robust traffic models and systems that can handle a wide range of scenarios. In the following sections, we will explore data generation through DGMs in three key areas: scenario and synthetic data generation, trajectory generation, and missing data imputation.

3.1.1 Scenario and Synthetic Data Generation

Scenario and synthetic data generation involve using DGMs to produce a wide range of synthetic data. This process includes creating diverse driving scenarios to test and evaluate autonomous vehicles, synthesizing demographic and travel behavior data to represent different populations in studies, and generating synthetic traffic data for training other machine learning models for traffic operation and management. Additionally, DGMs can generate rare or unexpected traffic situations to improve anomaly detection and response strategies. These capabilities allow researchers to simulate various conditions and events, which can help enhance the robustness and accuracy of analysis. This section highlights how the DGMs enable comprehensive testing, analysis, and improvement of transportation networks through effective scenario and synthetic data generation. We also categorize recent studies by generation type, task, model used, dataset, and evaluation metrics, as summarized in Table 3.

3.1.1.1 Driving Scenario Generation

Many studies in this section have focused on driving scenario generation, which involves creating diverse and realistic driving situations to test and evaluate autonomous vehicles and traffic management systems (Ghosh et al., 2016; Yun et al., 2019; Jin et al., 2023; Xu et al., 2023b; Huang et al., 2024; Li et al., 2024d; Jiang et al., 2024). The use of DGMs is beneficial in this context, as they generate scenarios that overcome the limited variety of real-world data and eliminate the need for costly, time-intensive data collection, thereby enabling scalable testing frameworks. Most of these studies utilize additional information as conditions for their generative models, such as car-following theory, safety critics, or physical restrictions, to generate more realistic and high-performing models. Zhong et al. (2023) presented a framework using a conditional diffusion model to generate realistic and controllable traffic simulations. The key benefit of the model is that it allows users to specify trajectory properties, such as reaching a goal or following speed limits, while ensuring that the generated trajectories are physically feasible and realistic through the guidance of signal temporal logic rules. Similarly, Sun et al. (2023) introduced a data-driven method, DriverSceneGen, for generating diverse and realistic driving scenarios to address data limitations. DriverSceneGen employed a diffusion model to create initial driving scenes in a rasterized Bird-Eye-View (BEV) format and used a simulation network to predict multiple future scenarios based on the initial setup. Furthermore, Xu et al. (2023a) developed DiffScene, a diffusion-based framework designed to generate safety-critical driving scenarios for evaluating and enhancing autonomous vehicle safety. DiffScene used a diffusion model to approximate the distribution of low-density areas in traffic data, creating realistic safety-critical situations. These scenarios were further optimized using guided adversarial objectives to maintain both realism and safety-criticality. Niedoba et al. (2024) introduced DJINN, a diffusion-based generative model that simulates joint traffic scenarios by diffusing the trajectories of all agents in a scene simultaneously. Rather than predicting trajectories for each agent independently, DJINN models the full joint distribution conditioned on flexible observation masks—capturing past, present, or even future state information—and maps context. This approach enables the generation of diverse, realistic traffic scenes, including rare and safety-critical events, while also allowing for test-time guidance and scenario editing. These studies have demonstrated the capability of using DGMs in driving scenario generation by providing more diverse and realistic scenarios for traffic management and Autonomous Vehicle development. Despite these promising developments, evaluating the quality of generated scenarios remains an open problem, as current metrics often fail to fully capture both realism and safety-critical aspects. In particular, there is no appropriate metric to quantitatively compare the distribution of generated driving scenarios to the ground truth. Moreover, existing metrics typically assess only specific features—for instance, verifying whether

Table 3
A summary of recent studies in scenario and synthetic data generation using DGM

Generation Type	Author	Task	Model	Dataset	Evaluation Metrics
Driving Scenario Generation	Zhong et al. (2023)	Driving Scenarios	Diffusion	nuScenes	Rule-specific Metrics, WD
	Sun et al. (2023)	Driving Scenarios	DriveSceneGen (Diffusion-based)	Waymo Motion Data	FID, Average Feature Distance, Precision, Recall, F1-Score
	Xu et al. (2023a)	Driving Scenarios	DiffScene (Diffusion-based)	CARLA Simulation	Rule-specific Metrics, SSPD, FID, DTW, WD, KLD
	Niedoba et al. (2024)	Driving Scenarios	DJINN (Diffusion-based)	Argoverse, INTERACTION	minADE_k, minFDE_k, MFD_k, minSceneADE_k, minSceneFDE_k
Population and Activity Synthesis	Borysov et al. (2019)	Population Synthesis	VAE	Danish National Travel Survey	SRMSE, R^2, Diversity Measures, Distribution Comparison, Pearson's Correlation Coefficient
	Garrido et al. (2020)	Population Synthesis	VAE, WGAN	Danish National Travel Survey	SRMSE, R^2, Sampling Zeros, Structural Zeros, Pearson's Correlation Coefficient
	Kim and Bansal (2023)	Population Synthesis	VAE, WGAN	Korea Household Travel Survey	SRMSE, Recall, Precision, F1-Score, Number of Unique Combinations
	Jutras-Dubé et al. (2024)	Population Synthesis	CTGAN, TVAE	American Community Survey	SRMSE, Recall, Precision, F1-Score, Sampling Zeros, Structural Zeros
	Jeong et al. (2021)	Activity-based Modeling	VAE	Seoul Household Travel Survey	Matching Rate, RMSE, Precision, Recall, F1-Score, Accscore
	Badu-Marfo et al. (2022)	Activity-based Modeling	CTGAN	2013 Montreal OD Travel Survey	SRMSE, R^2, Pearson's Correlation Coefficient, Distribution Comparison
	Lee et al. (2025)	Activity-based Modeling	CollaGAN	Seoul Household Travel Survey, Smart Card Data	SRMSE, Feasibility, Heterogeneity, F1-Score, Distribution Comparison
Synthetic Data and Model Training	Ozturk et al. (2020)	RL Agent Training	GAN	NGSIM	Crash Counts, Reward Comparison
	Dewi et al. (2022)	Recognition Models Training	DCGAN	Taiwan Prohibitory Signs Dataset	MSE, Structural Similarity Index, Intersection over Union, mAP, Detection time, Classification accuracy
	Zhu et al. (2022)	Prediction Model Training	STDGAN	PEMSD4, PEMSD7	MAE, RMSE, MAPE
	Zhou et al. (2023)	Urban Flow Generation	KSTDiff	Urban Flow Data from 4 Cities (Washington D.C., NYC, Beijing, Baltimore)	MAE, RMSE, SMAPE, MMD, NRMSE, JSD, Pearson's Correlation Coefficient
	Rong et al. (2023a)	OD Generation	DiffODGen	OD Data from 3 US City (Cook County, New York City, Seattle)	RMSE, NRMSE, CPC, JSD, Rate of Nonzero Flows, Accuracy, 1-Recall, Precision
	Rong et al. (2023b)	OD Generation	ODGN (cGAN)	OD Data from 8 US City (NYC, LA, Chicago, Houston, SF, Seattle, Washington D.C., Memphis)	JSD, RMSE, CPC
Anomaly Data Generation	Cai et al. (2020)	Crash Data Augmentation	DCGAN	Orlando Expressway SR 408 Data	AUC, Sensitivity, Specificity
	Huo et al. (2021)	Extreme Situations Generation	T^2GAN	Sina Weibo Traffic-related Text Data, Beijing Fourth Ring Road Passenger Flow Data	MAE, RMSE
	Chen et al. (2022d)	Crash Data Augmentation	GAM, cGAN, GDAGAN	Taiwan Provincial Highway Accident Records	Specificity, FPR, Precision, Recall, F1-Score, BA, G-mean
	Chen et al. (2024b)	Crash Data Augmentation	CTGAN-RU	Washington State Crash Data	Specificity, Recall, G-mean, Distribution Consistency, Parameter Recovery

scenarios obey traffic rules—without capturing the full spectrum of driving realism and safety. Furthermore, current DGM-based frameworks often fail to support long-duration simulations, unlike traditional simulators, which poses an additional challenge. In addition, current studies sometimes fall short in replicating extreme scenarios, such as sudden braking or rapid lane changes, that are critical in highly dynamic, localized environments. This gap underscores the need for future research to develop comprehensive, unified evaluation metrics and more robust simulation platforms that can support extended runs, thereby facilitating rigorous and reproducible testing of various driving systems.

3.1.1.2 Population and Activity Synthesis

Population synthesis is a critical component of transportation research that involves the generation of synthetic demographic and travel behavior data to accurately represent populations. The primary challenge is to capture the full joint distribution of the population while generating new, diverse samples that include rare or unobserved events. Accurately modeling this joint distribution becomes increasingly complex as the number of attributes grows. Simultaneously, during the sampling stage, the exponential increase in possible attribute combinations leads to the sampling

zero problem, making it difficult to generate diverse samples that include rare or unobserved events. Recent advances in DGMs, such as VAEs and GANs, offer promising solutions by approximating high-dimensional joint distributions and ensuring sample diversity. Several studies have demonstrated that these approaches can outperform traditional methods such as Gibbs sampling and Bayesian Networks(Borysov et al., 2018; Badu-Marfo et al., 2020; Mensah et al., 2022; Johnsen et al., 2022). For instance, Borysov et al. (2019) employed a VAE to learn latent representations of agents' attributes, thereby generating synthetic micro-agents that maintain the statistical properties of the original population. Similarly, Garrido et al. (2020) compared VAE and Wasserstein GAN (WGAN) for recovering sampling zeros in high-dimensional travel survey data and found that while WGANs can achieve higher predictive accuracy, VAEs tend to generate more diverse populations. Correspondingly, Kim and Bansal (2023) introduced custom loss functions for both VAE and WGAN to better handle structural-zeros issues, demonstrating improvements in both feasibility and diversity on a large-scale South Korean Household Travel Survey (HTS). Recently, Jutras-Dubé et al. (2024) introduced a copula-based generative framework that generates synthetic data for a target population using only known marginal totals with a sample from a similar population. The proposed method combines copula theory with DGMs (Conditional Tabular GAN (CTGAN) and Tabular VAE (TVAE)) to decouple dependency structures from marginal distributions, ensuring that the synthesized data accurately reflects both the structural relationships and the target population's marginals. Overall, these advances highlight the versatility of DGMs in addressing the complexities of population synthesis, balancing between achieving predictive accuracy and generating diverse, representative samples.

While synthetic populations form a strong base for transportation analysis, capturing dynamic human mobility requires Activity-Based Modeling (ABM) to simulate sequences of individual activities and predict their interconnections (Liao et al., 2024). The challenge in ABM lies in collecting sufficient data to represent the joint distribution of individual attributes—such as income, job type, gender, and age—which becomes even more complex when incorporating spatial dimensions. Adding spatial attributes increases the likelihood of rare combinations, thereby exacerbating the sampling zero problem. Recent advances in DGMs have been applied to address these challenges (Kim et al., 2022; Jeong et al., 2021; Badu-Marfo et al., 2022; Lee et al., 2025). For example, Jeong et al. (2021) integrated a VAE with a Hidden Markov Model (HMM) to infer and synthesize human activity sequences from mobile data. By embedding high-dimensional, mixed discrete, and continuous features into a lower-dimensional latent space, the proposed VAE-HMM mitigates the challenges of inconsistent clustering and overfitting commonly encountered in traditional HMMs. In another study, Badu-Marfo et al. (2022) proposed a dual-GAN structure, Composite Travel GAN (CTGAN), which simultaneously generates socioeconomic attributes and sequential mobility data learned from the joint distribution of both tabular attributes and sequential trip chain locations data. More recently, Lee et al. (2025) proposed Collaborative GAN (CollaGAN), a GAN-based data fusion method that combines household travel surveys with smart card data. CollaGAN utilizes a semi-supervised variational embedding to harmonize the two datasets within a shared, low-dimensional latent space, employing multiple loss functions—including boundary loss and expert-designed constraints—and dual discriminators to enhance both the feasibility and heterogeneity of the generated activity schedules. These advances demonstrate the potential of DGMs to overcome the challenges of data sparsity and high-dimensional complexity in ABM, ultimately improving the accuracy and diversity of synthesized activity patterns.

However, current DGMs for population and activity synthesis face several challenges. First, these models often suffer from a lack of interpretability, making it difficult to understand the underlying decision processes. Additionally, the characteristics of HTS including a relatively small data size compared to the number of attributes, infrequent updates, and lengthy collection durations, limit the effective utilization of these models. Moreover, the sequential nature of the generated outputs can lead to logically inconsistent travel patterns—such as implausible transitions between transportation modes or mismatches between socioeconomic status and residential characteristics—that necessitate extensive post-processing corrections. Addressing these gaps is essential for advancing DGMs toward more realistic, transparent and policy-relevant transportation models.

3.1.1.3 Synthetic Data and Model Training

The DGM can also be used to generate various types of synthetic traffic data for traffic modeling and training purposes. These synthetic datasets enable researchers to simulate traffic scenarios and train models without relying solely on real-world data, which can be expensive and difficult to collect. In related research, instead of using basic models, most researchers combine various deep learning architectures or incorporate additional information to further improve performance (Chen et al., 2019; Zhang et al., 2020a; Ozturk et al., 2020; Wu et al., 2020; Chen et al., 2021c; Zhou et al., 2023; Rong et al., 2023b; Kumar et al., 2023; Devadhas Sujakumari and Dassan, 2023). For example, Rong

et al. (2023a) introduced a large-scale OD generation method using a graph-denoising diffusion model. This approach simplifies the generation process by decomposing it into two stages: first, generating the network topology, and second, generating the edge weights. By tackling these stages separately, the model accurately generates realistic large-scale OD data. In a similar way, Rong et al. (2023b) introduced an Origin-Destination Generation Networks (ODGN), a physics-informed machine learning framework for generating OD by combining Multi-view Graph Attention Networks (MGAT) to extract urban features and a gravity-guided predictor to estimate mobility flows between regions. The model employs a cGAN training strategy with a random walk sampling discriminator to capture overall network topology, producing OD networks that closely match real-world properties. In addition, Zhou et al. (2023) introduced KSTDiff, a Knowledge-enhanced Spatio-temporal Diffusion model that generates dynamic urban flow data for regions lacking historical records by leveraging an Urban Knowledge Graph (UKG) and a region-customized diffusion process guided by a learnable volume estimator. The framework can also be adapted for urban flow prediction, achieving competitive performance compared to state-of-the-art methods.

For model training, Ozturk et al. (2020) introduced a GAN-based traffic simulator that generates realistic and stochastic vehicle trajectories from real data, which are then used to train reinforcement learning agents. These generated data help RL agents train in a more realistic environment and significantly improve their generalization capabilities compared to agents trained on simple rule-based simulators. Dewi et al. (2022) applied a Deep Convolutional GAN (DCGAN) to generate synthetic traffic sign images. This approach captures the distribution of real traffic sign images and produces new samples to augment the original dataset. The researcher further combined these synthetic images with real ones to improve the performance of deep recognition models, such as CNN and ResNet 50. Additionally, Zhu et al. (2022) employed the Spatio-Temporal Dependencies GAN (STDGAN) to generate high-quality synthetic traffic volume data for prediction tasks. Specifically, STDGAN captures implicit variation patterns in traffic volume based on information from traffic speed and occupancy data. Experiment results indicate that enriching the training dataset with the synthetic volume data led to more robust and accurate performance in prediction models.

The provided examples demonstrate the ability of DGMs to generate various traffic data that support traffic management and enhance model training, offering valuable resources for transportation research and applications. One key limitation of using DGMs is ensuring that the synthetic data accurately reflects the complex, dynamic, real-world environment. In general, the generated data fails to fully capture the intricate patterns, dynamic shifts, and non-stationary behaviors seen in real scenarios, which can introduce biases into the model training process. This limitation not only affects the quality of the synthetic data but also complicates its integration with real data. Determining the optimal balance is another critical issue in model training. Specifically, an excess of synthetic data may overwhelm the true data distribution, while too little might not adequately address data scarcity. In summary, these challenges can hinder the model's ability to generalize and perform effectively under dynamic conditions.

3.1.1.4 Anomaly Data Generation

Anomaly data generation is crucial for creating rare or unexpected traffic situations, which are essential for testing and evaluating systems under extreme conditions. Generating such data helps develop robust models that can effectively handle imbalanced datasets. In this research field, DGMs have been highly adapted to augment imbalanced traffic data. This augmentation not only addresses the scarcity of anomalous events in real-world datasets but also plays a pivotal role in enhancing the accuracy and resilience of prediction and classification models used in traffic safety and management applications.(Islam et al., 2021; Zarei and Hellinga, 2021; Li et al., 2024a; Chen et al., 2024b). For instance, Cai et al. (2020) utilized a DCGAN to generate synthetic crash data from learning the distribution of crash-related traffic data. The synthetic data was then combined with real data to create a balanced dataset, which was used to train various crash prediction algorithms, including Logistic Regression, SVM, ANN, and CNN. Similarly, Chen et al. (2022d) compared several data augmentation methods including Synthetic Minority Oversampling Technique (SMOTE), GAN, conditional GAN (cGAN), and Gaussian Discriminant Analysis GAN (GDAGAN) to address the imbalance in traffic collision datasets. These models generated additional samples for underrepresented classes, thereby balancing the traffic collision dataset, and enhancing the performance of classifiers trained on them. Furthermore, Chen et al. (2024b) developed a method that integrates Conditional Tabular GAN with Random Under-sampling (CTGAN-RU) to generate synthetic crash data. This approach accounts for both the continuous and discrete characteristics of imbalanced crash datasets by incorporating various conditions, thereby creating a more balanced training set and improving the accuracy and reliability of crash severity prediction models. Huo et al. (2021) introduced the Text-to-Traffic Generative Adversarial Network (T^2GAN), a novel framework that generates realistic traffic situations by

fusing traditional traffic data with semantic information extracted from social media. In T^2GAN, a pre-trained GloVe model encodes the traffic-related text into semantic features that are integrated into the GAN training process, with a global-local loss employed to align the modalities and produce varied, contextually accurate traffic scenarios. The current studies underscore the importance of using DGMs for generating anomaly data to address imbalanced datasets and enhance the robustness of traffic management and safety systems. However, current models face significant challenges. In particular, they struggle to effectively manage imbalanced crash data that often includes missing values, heteroscedasticity, noise, and small sample sizes—characteristics common in real-world datasets. This highlights the need for further research to refine these methods and ensure that synthetic data generation more accurately reflects the complexities of actual traffic anomalies.

3.1.2 Trajectory Generation

Trajectory generation creates realistic movement patterns for vehicles and pedestrians, which is essential for developing accurate traffic simulations and urban planning tools. By leveraging DGMs, researchers can produce synthetic trajectories that mimic real-world behaviors of traffic agents, enabling more robust analysis and testing of transportation systems. These generated trajectories help understand traffic dynamics, evaluate transportation policies, and enhance infrastructure design. These trajectories can be further classified based on their scale and scope, such as micro and macro scales. Micro-scale trajectories focus on detailed, short-term movement patterns of individual agents in localized areas, like intersections, while macro-scale trajectories represent broader, long-term traffic flow across extensive road networks or regions. The classification into micro and macro scales significantly impacts their use cases, such as collision avoidance or urban planning, and determines the complexity of the problems they address. Table 4 presents an overview of recent studies in trajectory generation, organized by generation scale, task, model employed, dataset, and evaluation metrics.

3.1.2.1 Micro-scale Trajectory Generation

Micro-scale trajectory generation involves producing detailed agent movements, closely focusing on behaviors within localized areas like intersections or specific road segments. It focuses on generating detailed vehicle dynamics, such as acceleration, deceleration, and lane-changing. The DGMs are particularly effective for this task as they efficiently learn complex and varied behaviors directly from real-world data, producing realistic trajectories. These accurate and detailed trajectories significantly enhance applications such as collision avoidance, precise maneuver planning, and road safety (Kuefler et al., 2017; Krajewski et al., 2018; Ding et al., 2019; Krajewski et al., 2019; Demetriou et al., 2020; Zhou et al., 2020b; Zhang et al., 2022d; Gong et al., 2023; Dong et al., 2023; Singh et al., 2023; Ma and Qu, 2023; Shi et al., 2024). For example, Krajewski et al. (2018) used two DGMS, Trajectory GAN(TraGAN) and VAE(TraVAE), which generate realistic lane change trajectories by learning intuitive, disentangled latent parameters from real-world driving data. These models can accurately reproduce observed maneuvers while also synthesizing new trajectories to fill gaps in simulation datasets. To further solve the smoothness problem, Krajewski et al. (2019) integrated a Bézier-curve output layer with additional loss terms into TraVAE, resulting in BézierVAE. This model generates smooth trajectories in both the position and speed domains, enhancing trajectory modeling for safety validation of highly automated vehicles. Ding et al. (2019) introduced a Multi-Vehicle Trajectory Generator (MTG) that integrates a VAE framework with bi-directional Gate Recurrent Units (GRUs) in the encoder and a multi-branch GRU decoder to simulate realistic vehicle-to-vehicle encounters. The authors also propose a novel disentanglement metric to assess the model's stability and interpretability. In another study, Bhattacharyya et al. (2022) employed GAIL with several extensions (PS-GAIL, RAIL, and Burn-InfoGAIL) to model and replicate human driving behavior in simulation. By addressing the challenges of stochasticity, multimodality, and latent variability in human driving behavior, the proposed methods generate realistic car-following and lane-changing behaviors, as validated on the NGSIM dataset. Chen et al. (2022a) developed a two-step hybrid driving model that combines model-based controllers with GAIL for traffic simulation. The model generates high-level driver traits used as parameters for low-level model-based controllers to simulate human-like driving in multi-agent traffic scenarios. Ma et al. (2023) developed the Physics-Informed Conditional GAN (PICGAN) to control Connected Autonomous Vehicles (CAVs) in mixed traffic environments. The proposed model integrates theoretical physics with a dual conditional GAN framework, which consists of an encoder-decoder generator and two discriminators. By training on both observed and simulated data, PICGAN demonstrates improved robustness and adaptability. Similarly, Yu et al. (2024) proposed a Theory-data Dual Driven Stochastically GAN (TDS-GAN) by integrating a physics-informed GAN with a two-dimensional Intelligent Driver Model (2D-IDM)

Table 4
A summary of recent studies in trajectory generation using DGM

Generation Scale	Author	Task	Model	Dataset	Evaluation Metrics
Micro	Krajewski et al. (2018)	GPS-level Generation (High Sampling)	TrajGAN, TrajVAE	HighD	MSE
	Krajewski et al. (2019)	GPS-level Generation (High Sampling)	BézierVAE	HighD	RMSE, Normalized Variance-of-Differences
	Ding et al. (2019)	GPS-level Generation (High Sampling)	MTG (β-VAE-based)	UMTRI Driving Encounter Dataset	Rule-specific Metrics, Disentanglement Metric
	Bhattacharyya et al. (2022)	GPS-level Generation (High Sampling)	GAIL	NGSIM	RMSE, Rule-specific Metrics
	Chen et al. (2022a)	GPS-level Generation (High Sampling)	GAIL	HighD	RMSE
	Ma et al. (2023)	GPS-level Generation (High Sampling)	PICGAN	NGSIM	MSE, Rule-specific Metrics
	Yu et al. (2024)	GPS-level Generation (High Sampling)	TDS-GAN	Closed Test Site from Highway Research Institute of the Ministry of Transport of China	RMSE, MAPE, MAE, Rule-specific Metrics
Macro	Choi et al. (2021a)	Link-level Generation	TrajGAIL	Aimsun Simulation, Seoul Taxi DTG	BLEU, METEOR, JSD
	Sun and Kim (2023)	Link-level Generation	MAGAIL-VL	pNEUMA	BLEU, JSD, MAPE, NMAE
	Xiong et al. (2023)	Grid-level Generation	TrajSGAN	MTL Trajet Dataset	JSD-based Metrics, Recall, R^2
	Zhu et al. (2024a)	GPS-level Generation (Low Sampling)	DiffTraj	Chengdu & Xi'an GPS Trajectory Dataset	Density Error (JSD-based), Trip Error (JSD-based), Length Error, Pattern Score (F1-Score-based), MAE, MSE, RMSE
	Wei et al. (2024)	Link-level Generation	Diff-RNTraj	Porto & Chengdu Taxi Datasets	JSD, Road Segment Connectivity

to capture the stochastic and heterogeneous car-following behavior of human-driven vehicles in mixed traffic. In this framework, the GAN captures the inherent randomness and uncertainty of car-following, generating realistic, varied trajectories that enhance the predictive accuracy of the theory-driven 2D-IDM for short-term forecasts and macroscopic traffic flow simulations. DGMs have demonstrated an ability to produce highly detailed and realistic micro-scale trajectories, which are crucial for applications that demand precise and immediate responses in traffic systems. Despite these promising results, several limitations remain to be solved. First, fine-tuning the guidance parameters of DGMs is often sensitive and requires extensive calibration to balance between trajectory diversity and fidelity. Specifically, even minor deviations can significantly affect the model's ability to replicate rare or extreme driving events, such as sudden braking or abrupt lane changes, which are critical for safety. Moreover, the performance of DGMs is highly dependent on the quality and representativeness of the training data and any deficiencies in data coverage can compromise the model's accuracy and its generalizability to unseen scenarios.

3.1.2.2 Macro-scale Trajectory Generation

Macro-scale trajectory generation covers broad spatial and temporal scopes, including extensive areas such as entire road networks or large regions over longer periods. Such trajectories are essential for strategic planning, traffic management, and long-term behavioral analysis. By examining OD pairs, travel time distributions, and overall traffic flow patterns, macro-scale trajectories provide valuable insights into larger trends and systemic issues. The information is crucial for urban planning, optimizing public transportation systems, and enhancing the efficiency of traffic networks. Numerous studies have applied DGMs to generate such macro-scale trajectories, offering valuable insights for strategic decision-making in transportation engineering.(Chen et al., 2021c,b; Zhang et al., 2022b; Zhu et al., 2024a; Wei et al., 2024; Wang and Kankanhalli, 2024; Zhu et al., 2024b). For example, Choi et al. (2021a) developed TrajGAIL, a GAIL framework for generating link sequences of urban vehicle trajectories. The model combines the capabilities of GAIL with RNNs to learn the underlying distributions of urban vehicle trajectory data. TrajGAIL allows for the generation

of synthetic vehicle trajectories that closely resemble real-world patterns, even from limited observations. Similarly, Sun and Kim (2023) introduced MAGAIL-VL, a data-driven, network-wide traffic simulation framework considering both Vehicle and Link agents. Specifically, MAGAIL-VL learns the distribution of vehicle movements and link states from observed data, enabling the generation of realistic traffic scenarios across an entire network. In another study, Xiong et al. (2023) introduced TrajSGAN, a semantic-guiding adversarial network for urban trajectory generation that synthesizes human mobility trajectories at a grid scale. It integrates an attention-based generator for trajectory location prediction with a rollout module and a CNN-based discriminator to assess the spatial structure of the generated trajectories. This integrated framework effectively reduces divergence in spatial metrics and has been successfully applied to epidemic diffusion studies with high accuracy. Recently, diffusion-based models have gained popularity for generating fine-level urban trajectories. Recently, diffusion-based models have become popular for generating fine-level urban trajectories(Zhu et al., 2024a; Wei et al., 2024; Wang and Kankanhalli, 2024; Zhu et al., 2024b). For example, Zhu et al. (2024a) introduced DiffTraj, a novel approach for urban-scale trajectory generation that produces high-quality synthetic GPS trajectories using a spatial-temporal diffusion probabilistic model. DiffTraj leverages a U-Net architecture enhanced with residual blocks and multi-scale feature fusion to accurately estimate noise levels during the reverse denoising process. This design effectively captures complex spatial-temporal dependencies, enabling the generation of realistic synthetic trajectories that are well-suited for urban mobility analysis and preserving the statistical properties of real-world data. Similarly, Wei et al. (2024) presented Diff-RNTraj, a diffusion-based model designed to generate trajectories that are geographically accurate and adhere to road network constraints. The model uses a continuous diffusion framework combined with a pre-training strategy to handle the hybrid nature of road network–constrained trajectory data, resulting in realistic and usable synthetic trajectories. Even though the current studies have demonstrated significant advances in macro-scale trajectory generation, several challenges remain to be addressed. First, integrating external conditions—such as dynamic traffic states, diverse regional behaviors, and factors like departure times and local events—into the generative process is still limited, which restricts the models' ability to accurately reflect real-time traffic dynamics. Second, ensuring that generated trajectories are physically plausible and strictly adhere to road network connectivity is challenging, often resulting in unrealistic or disconnected routes. Third, capturing the inherent stochasticity and heterogeneity of human mobility remains a major hurdle. Finally, the absence of unified evaluation metrics further complicates objective assessment and comparison among models, making it difficult to benchmark progress in urban-level trajectory generation using DGMs.

3.1.3 Missing Data Imputation

The missing data imputation in transportation aims to reconstruct original data X from missing or corrupted data $\hat{X}$ with conditional information ϕ such as known historical data, geometry configuration, and external factors ($\hat{X} = f(X, \phi)$). This process inherently involves data generation, as it creates new values to fill gaps in existing datasets, thereby enhancing the volume and quality of data available for analysis. In traditional imputation methods, missing data is often challenging to reconstruct due to the complex spatio-temporal dependencies in traffic data, which limit the accuracy and robustness of simple statistical methods. In recent years, DGMs have become powerful tools for handling the complexities of traffic data imputation. Their ability to model the underlying distribution of traffic data enables them to generate realistic and accurate reconstructions of missing entries across various scenarios. DGMs can also integrate a wide range of conditional information, including the spatio-temporal dependencies of traffic data, to enhance imputation performance (Zhang et al., 2024a; Cai et al., 2023; Yang et al., 2021; Zhang et al., 2021c; Peng et al., 2023; Shin et al., 2023; Duan et al., 2024; Chen et al., 2022b). Traditionally, missing data has been categorized into three types: Missing Completely at Random (MCAR), Missing at Random (MAR), and Missing Not at Random (MNAR) (Lin and Tsai, 2020; Hasan et al., 2021). However, the unique spatio-temporal characteristics of traffic data require a reevaluation of these categories. To address this, Chan et al. (2023) proposed a new classification specifically for transportation research, dividing missing data into Fiber Missing Data, Block Missing Data, and Random Missing Data. In this paper, we adopt this updated categorization to better address the specific challenges of traffic data imputation. Table 5 summarizes recent studies and classifies them according to missing type, imputation task, model, dataset, missing rate, and evaluation metrics.

3.1.3.1 Fiber Missing Data

Fiber Missing Data occurs when there is a sudden, temporary failure of data-acquisition devices, leading to gaps in data collection that can be short-term or long-term. Many advanced DGMs have been developed to address this issue

Table 5
A summary of recent studies in missing data imputation using DGM

Missing Type	Author	Imputation Task	Model	Dataset	Missing Rate	Evaluation Metrics
Fiber	Han et al. (2020)	Speed	CA-GAN	PEMS	10–100%	MRE, Time Cost Comparison
	Xu et al. (2021)	Speed	GA-GAN	PEMS-BAY, Seattle Dataset	10–70%	MAE, RMSE, MAPE, Residual Analysis
	Zhang et al. (2021b)	Lane-level Speed	GaGAN	Hangzhou Signalized Road Data	20–100%	MAE, RMSE, MAPE, Pearson's Correlation Coefficient
	Liu et al. (2023)	Speed	PriSTI (Diffusion-based)	AQI-36, METR-LA, PEMS-BAY	10–90%	MAE, MSE, CRPS
Block	Li et al. (2018)	Flow	3DConvGAN	Beijing Taxi Dataset	20–80%	RSE
	Zhang et al. (2021c)	Volume	SA-GAIN (GAN-based)	Seattle I-5 Highway	10–80%	MAE, MMD, RMSE
	Yuan et al. (2022)	Speed Passenger Flow	STGAN	Beijing Road Dataset Beijing Subway Dataset	20–80%	RMSE, NMAE, MAE
	Hou et al. (2023)	Flow	MissII (GAN-based)	Beijing Taxi Dataset	20–60%	MAE, RMSE, CosineSim
Random	Chen et al. (2019)	Flow	GAN	PEMS	30–80%	MAE, RMSE, MRE
	Boquet et al. (2019)	Speed	VAE	PEMS	10–40%	RMSE, MAPE
	Yang et al. (2021)	In&Outflow	ST-LBAGAN	Beijing Taxi, NYC Bike Datasets	10–60%	RMSE, MAE
	Shin et al. (2023)	Speed	AAE	Korean ITS Data, PEMSD7	10–50%	RMSE, MAPE
	Li et al. (2023a)	Volume Speed	TGAIN (GAN-based)	I90 Dataset, Changchun Speed Dataset	10–90%	RMSE, MAPE
	Zheng et al. (2024)	Speed	DPRDDM (Diffusion-based)	Zen & Beijing Traffic Data	10–30%	RMSE, MAPE

effectively (Li et al., 2018; Zhang et al., 2021c,b; Huang and Chen, 2022; Yuan et al., 2022; Shen et al., 2022; Cai et al., 2023; Zhang et al., 2023b; Hou et al., 2023; Cai et al., 2023; Li et al., 2024c; Duan et al., 2024; Zhang et al., 2024a). For instance, Han et al. (2020) developed the Content-Aware GAN (CA-GAN) to handle missing traffic data in time series. In this method, traffic data were modeled as tensors to capture the inherent spatial-temporal correlations. The CA-GAN learns the underlying traffic distribution and generates realistic traffic patterns, effectively completing gaps in speed data, particularly in cases with consecutive data losses. Similarly, Xu et al. (2021) introduced the Graph Aggregate GAN (GA-GAN), which combines the strengths of Graph Sample and Aggregate (GraphSAGE) with a GAN to impute missing traffic speed data. In this approach, GraphSAGE aggregates information from neighboring nodes in the road network to capture spatial-temporal correlations, which are then used by a Wasserstein GAN (WGAN) to generate the missing data. The GA-GAN has been tested in both fiber and random missing data environments, demonstrating its versatility. Zhang et al. (2021b) proposed the Gated GAN (GaGAN) to address missing speed data on signalized roads. The GaGAN integrates attention mechanisms within graph convolutional operations to capture spatial correlations and enhances GRUs with Self-Attention (SA-GRU) to learn temporal dependencies across signal cycles. This approach is effective for both fiber and random missing data scenarios, ensuring that the imputed values retain the inherent spatio-temporal patterns observed in lane-level traffic measurements. In another study, Liu et al. (2023) introduced PriSTI, a conditional diffusion framework for spatio-temporal data imputation. Specifically, the PriSTI uses a diffusion-based noise estimation module and a spatio-temporal feature extraction process to enhance its performance under high missing rates and diverse spatial configurations. It has been evaluated in environments with both fiber and random missing data, showing significant robustness. These models demonstrate the capabilities of DGMs in addressing the challenges posed by fiber missing data, ensuring the continuity and quality of traffic datasets even in the face of temporary data collection failures.

3.1.3.2 Block Missing Data

Block Missing Data occurs when there are no data-acquisition detectors in the area of interest, leading to complete loss in the dataset for the region. This type of missing data is particularly challenging to address because of the

absence of any observations over an extended spatial and temporal range. Advanced DGMs have been developed to tackle this challenge, though most studies do not focus exclusively on block missing data. Instead, they often evaluate model performance under various conditions, including fiber and random missing data scenarios (Li et al., 2018; Zhang et al., 2021c; Yuan et al., 2022; Hou et al., 2023; Li et al., 2023b; Zhang et al., 2024a). For example, Li et al. (2018) combined a 3D Convolutional Neural Network with GAN (3DConvGAN) to address missing traffic data. Unlike traditional methods that may not fully exploit the spatial-temporal features of historical data, 3DConvGAN uses a fractional strided 3D CNN in both the generator and discriminator to enhance imputation performance. The model's effectiveness was also tested in environments with Fiber, Block, and Random Missing Data, highlighting its versatility. In another study, Zhang et al. (2021c) introduced SA-GAIN, a Self-Attention Generative Adversarial Imputation Network, designed to impute missing traffic flow data. By incorporating a self-attention mechanism in GAN, SA-GAIN effectively captures important features across spatial and temporal dimensions. The model was tested in both fiber and block missing data scenarios, demonstrating robust performance. In Yuan et al. (2022), the authors enhanced the performance of GAN in traffic flow imputation by designing generative and center losses. These losses enable the model to more accurately reconstruct missing data while preserving local spatio-temporal correlations. This approach was evaluated across all three types of missing data environments: Fiber, Block, and Random Missing Data. The proposed method demonstrated significant improvements in imputing missing traffic data under these conditions. Hou et al. (2023) developed MissII, a novel theory-guided deep learning framework for traffic data imputation. The approach first estimates traffic flow between points of interest using mobility models that leverage environmental and social factors to capture complex traffic dynamics. The estimated traffic flow data are then used as real samples to guide the GAN's training process to improve the imputation accuracy. These studies illustrate the effectiveness of DGMs in addressing block missing data, which presents unique challenges due to the complete absence of data in specific areas. By leveraging spatio-temporal modeling and integrating advanced neural network architectures, these models provide comprehensive solutions to ensure continuity and quality in traffic data even when entire sections of data are missing.

3.1.3.3 Random Missing Data

Random Missing Data occurs when the data-acquisition detectors fail unpredictably in the spatio-temporal domain, which results in gaps in data with little to no correlation among the missing entries. Due to its unpredictable nature, this scenario presents significant challenges and is commonly used to evaluate the performance of imputation models (Li et al., 2018; Chen et al., 2021a, 2022b; Wu et al., 2022a; Qin et al., 2021; Kazemi and Meidani, 2021; Tu et al., 2021; Zhang et al., 2021b; Xu et al., 2021; Shen et al., 2022; Yuan et al., 2022; Yang et al., 2022; Zhang et al., 2022a; Wang et al., 2023; Liu et al., 2023; Li et al., 2023b; Huang et al., 2023; Zhang et al., 2023b; Cai et al., 2023; Hou et al., 2023; Cai et al., 2023; Li et al., 2024c; Zheng et al., 2024; Duan et al., 2024; Zhang et al., 2024b,a). For example, Chen et al. (2019) developed a GAN-based method to impute missing traffic flow data by combining real and synthetic data. This approach utilizes a parallel data paradigm where the GAN generates synthetic data that are used alongside real data to train the imputation model, enhancing its ability to fill in gaps accurately. Boquet et al. (2019, 2020) used a VAE-based method for imputing missing traffic data, aimed at improving the accuracy of traffic forecasting systems impacted by sensor or system failures. The unsupervised approach learns the underlying data distribution from a latent space, resulting in improved post-imputation performance and effective data augmentation. Yang et al. (2021) introduced the Spatio-Temporal Learnable Bidirectional Attention GAN (ST-LBAGAN) that combines a U-Net generator with bidirectional attention to capture spatio-temporal dependencies for missing traffic data imputation. The model uses multi-channel inputs with a mask to focus on missing regions and is trained with a composite loss to ensure realistic outputs, achieving high accuracy even at high missing rates. To fully exploit the benefits of both VAE and GAN in data imputation, Shin et al. (2023) presented an Adversarial Autoencoder (AAE)-based model that leverages spatio-temporal feature extraction to address missing traffic data. AAE combines the principles of VAE and GAN to capture the complex dependencies within traffic data, providing robust imputation capabilities even in scenarios with high rates of missing data. Additionally, Li et al. (2023a) proposed the Time Series Generative Adversarial Imputation Network (TGAIN) to tackle the challenge of imputing time series data. This model is designed to learn the multi-state distribution of missing time series data under conditional vector constraints. By employing a multiple imputation strategy, it effectively handles the uncertainty inherent in the imputation process. Recently, Zheng et al. (2024) used traffic domain knowledge and a denoising diffusion model to develop the Doubly Physics-Regularized Denoising Diffusion Model (DPRDDM) for recovering corrupted traffic speed data. This model demonstrates robustness in handling various types of noise, including Gaussian white noise, random corrupted noise, spatially correlated noise,

temporally correlated noise, and a mixture of these noise types. The current studies in random missing scenario illustrate the diverse applications and robust capabilities of DGMs in addressing the challenges posed by random missing data. By effectively capturing and utilizing spatio-temporal dependencies, these models ensure accurate and reliable data imputation even in the most unpredictable scenarios.

DGMs show promising solutions for handling missing data in traffic systems; however, some key challenges still persist at this stage. A primary concern is limited transferability: models tend to perform well on their training datasets but struggle to generalize across diverse traffic conditions and sensor networks, limiting their practical applicability. Additionally, high missing rates and irregular patterns substantially affect imputation quality, compounded by the difficulty of capturing complex spatiotemporal dependencies in multidimensional traffic data. Furthermore, the lack of standardized evaluation protocols—such as consistent missing rate ranges, evaluation metrics, and unified datasets—makes it difficult to compare and validate different approaches effectively. These limitations highlight the need for further research to improve the robustness and adaptability of DGMs in addressing missing data in traffic systems.

3.2. DGM for Estimation and Prediction

One of the significant aspects of DGMs in transportation studies is their capability to learn and accurately model the distribution of input data. This foundational property enables DGMs to handle complex datasets and is critical for advanced analytical tasks such as traffic data generation, prediction, and classification. Another critical yet less explored advantage of DGMs is their ability to perform density estimation—modeling the probability distribution of potential outcomes based on input data. This capability is a key function in predictive analytics and estimation tasks within transportation systems. Specifically, "prediction" refers to forecasting future traffic conditions using historical and real-time data, while "estimation" focuses on determining the current status of the system, which often requires filling in data gaps or refining data accuracy. The ability to generate and evaluate multiple potential scenarios from DGMs is invaluable for effective traffic management. It allows transportation planners and engineers to anticipate and respond to a range of possible current and future conditions, thereby improving the efficiency and safety of the traffic network. By providing a probabilistic view of potential outcomes, DGMs empower decision-makers to accommodate the inherent variability and dynamic nature of traffic flows, significantly enhancing both the precision and reliability of transportation systems planning.

Building on the foundational understanding of DGMs and their application in traffic estimation and prediction, the following sections will discuss current research from three distinct scale levels: agent, link, and region. At the agent level, we will explore how these models are applied in individual vehicle and pedestrian prediction and estimation, critical for autonomous driving and safety systems. At the link level, the focus will shift to how DGMs are used for modeling traffic flows and speed on specific roadway segments, which is crucial for daily traffic management and dynamic routing. Finally, at the regional level, we will discuss the role of DGMs in shaping strategic planning and operational decisions by analyzing and predicting wide-area traffic patterns. Through this comprehensive examination, we aim to highlight the versatility and impact of DGMs in addressing the complexities of modern transportation challenges, thereby showcasing their essential role in the evolution of smart mobility solutions.

3.2.1 Agent-level Analysis

Agent-level analysis focuses on understanding and predicting the behaviors of individuals, such as vehicles, cyclists, or pedestrians. This detailed analysis is crucial for various applications like autonomous vehicle navigation, pedestrian safety systems, and dynamic routing. DGMs are particularly critical in this context since they can learn complex agent behavior patterns directly from data while inherently accounting for uncertainty in the estimation or prediction process. In other words, DGMs can reflect the inherent variability of real-world actions and results in more robust and reliable analysis. Table 6 summarizes recent studies, categorizing them by model, observation/prediction length, sampling time, prediction samples, dataset, and evaluation metrics.

Notably, a significant portion of research in agent-level traffic analysis using DGMs primarily emphasizes learning the underlying distributions of traffic data and produces single outcome(Roy et al., 2019; Zhao et al., 2020; Hegde et al., 2020; Zhou et al., 2021; Wang and He, 2021; Jagadish et al., 2021, 2022; Chen et al., 2022c; Hsu et al., 2023; Westny et al., 2024). This learning process is essential because it allows models to capture the intricate patterns and correlations within data that traditional models might miss. For example, Li et al. (2019a) proposed the Conditional Generative Neural System (CGNS), which combines the cVAE with GAN to generate feasible, realistic, and diverse future trajectories for multiple agents by leveraging both static context and dynamic interactions. Kim et al. (2021)

Table 6
A summary of recent studies in agent-level analysis using DGM

Author	Model	Observation / Prediction Length	Sampling Time	Prediction Samples	Dataset	Evaluation Metrics
Gupta et al. (2018)	SocialGAN	3.2s/4.8s	0.4s	1-100	ETH, UCY	minADE_k, minFDE_k
Kosaraju et al. (2019)	Social-BiGAT (GAN-based)	3.2s/4.8s	0.4s	1-20	ETH, UCY	minADE_k, minFDE_k
Li et al. (2019a)	GAN + cVAE	2s/5s 3.2s/4.8s	0.5s 0.4s	1	INTERACTION ETH, UCY, Stanford Drone Dataset	ADE, FDE
Salzmann et al. (2020)	Trajectron++ (VAE-based)	3.2s/4.8s 2s/1-4s	0.4s 0.5s	20	ETH, UCY nuScenes	minADE_k, minFDE_k, KDENLL_k
Wang et al. (2020c)	TS-GAN	3s/1-5s	0.1s	1	NGSIM	RMSE
Kim et al. (2021)	cVAE	1s/1s	0.1s	1	CarMaker HILS	RMSE, MAE
Sun et al. (2021a)	Flow	3.2s/0.4-4.8s	0.4s	1-20	ETH, UCY, Stanford Drone Dataset	minADE_k, minFDE_k, NLL
Gómez-Huélamo et al. (2022)	GAN	2s/3s	0.1s	1	Argoverse	ADE, FDE
Chen et al. (2023)	EquiDiff	3s/1-5s	0.2s	1	NGSIM	RMSE
Vishnu et al. (2023)	TS-GAN, TS-CVAE	3.2s/4.8s	0.4s	1-20	EyeonTraffic, INTERACTION	minADE_k, minFDE_k, KDENLL_k
Li et al. (2023c)	cVAE + Diffusion	2s/6s 2s/3s	0.5s 0.1s	1-15	nuScenes Argoverse	minADE_k, minFDE_k
Liu et al. (2024)	Diffusion	3s/1-5s	0.2s	1	NGSIM	RMSE

improved the accuracy of ego vehicle trajectory prediction using a driving style-based cVAE. Their model integrates a DeepConvLSTM network to recognize driving styles from in-vehicle sensor data. This approach not only predicts a single trajectory but also estimates a probability distribution over future trajectories, effectively capturing the inherent uncertainty in driving behavior. Additionally, Wang et al. (2020c) presented a GAN-based framework TS-GAN for vehicle trajectory prediction that uses multi-vehicle collaborative learning. Their method combines an auto-encoder social convolution module to capture spatial interactions and a recurrent social mechanism to model temporal relationships among surrounding vehicles. These fused features feed into a conditional GAN that generates a multi-modal probability distribution over future trajectories for a target vehicle. Similarly, Gómez-Huélamo et al. (2022) explored the use of attention mechanisms with GANs for vehicle trajectory prediction. The model generates feasible and realistic trajectories by considering both social interactions and the physical constraints of the road network. Recently, Chen et al. (2023) introduced EquiDiff, a conditional equivariant diffusion model that combines a denoising diffusion probabilistic model with an SO(2)-equivariant transformer to effectively manage uncertainties in vehicle trajectory predictions Liu et al. (2024) aimed to predict the distribution of endpoints of multi-agent trajectories using denoising diffusion and Transformer models, capturing both the spatio-temporal dynamics and the intrinsic intent of vehicles to enhance overall prediction quality.

Another critical aspect of using DGMs in agent-level analysis is the ability to perform density estimation of outputs. This capability allows for generating multiple potential outcomes, offering a comprehensive probabilistic view that supports robust decision-making under uncertainty. Research has increasingly focused on this feature, enabling traffic managers and autonomous vehicle systems to adapt dynamically to varied and unpredictable conditions (Feng et al., 2019; Zhao et al., 2019; Bhattacharyya et al., 2019; Cheng et al., 2020; Neumeier et al., 2021; Wang et al., 2020a; Eiffert et al., 2020; Agarwal et al., 2020; Dendorfer et al., 2021; Rossi et al., 2021b; Choi et al., 2021b; Li et al., 2021; Rossi et al., 2021a; Oh and Peng, 2022; Wu et al., 2022b; Zhong et al., 2022; Guo et al., 2023; Xing et al., 2022; Jagadish et al., 2022; De Miguel et al., 2022; Gui et al., 2022; Yao et al., 2023; Rempe et al., 2023; Tang et al., 2024; Yang et al., 2024). For example, Gupta et al. (2018) proposed the Social GAN to predict socially acceptable future trajectories for pedestrians in crowded scenes. The GAN model captures complex dynamics and social behaviors, generating a wide range of potential trajectories. Extending this idea, Kosaraju et al. (2019) introduced Social-BiGAT,

a trajectory forecasting model that integrates Bicycle-GAN with Graph Attention Networks (GATs) and LSTMs. This model generates multimodal predictions of pedestrian trajectories by leveraging complex social interactions and physical context cues. Other studies have adopted different strategies to model uncertainty. Sun et al. (2021a) introduced a normalizing flow-based prediction model designed to model the exact probability distribution of future human trajectories. In a multi-agent context, Salzmann et al. (2020) developed Trajectron++, an advanced VAE-based model that forecasts the trajectories of multiple agents in dynamic environments. Trajectron++ integrates a CVAE with Gaussian Mixture Models (GMMs) within a graph-based recurrent neural network framework. This structure allows Trajectron++ to generate multiple potential trajectories in a probabilistic manner, providing a comprehensive representation of possible future scenarios. Vishnu et al. (2023) incorporated traffic states into Transformer, GAN, and CVAE models to enhance multi-agent trajectory predictions. These models generate diverse plausible trajectory outcomes that are contextually relevant and highly predictive. Recent work has further explored hybrid approaches. Li et al. (2023c) developed a framework combining cVAE and conditional diffusion models for multi-modal vehicle trajectory prediction. This approach addresses the challenge of predicting highly uncertain future vehicle trajectories in urban environments by first generating trajectories with cVAE and refining them using a diffusion model.

These examples underscore the diverse and robust capabilities of DGMs in agent-level analysis, which are crucial for intelligent transportation systems. However, as models incorporate richer contextual data and accommodate a variable number of agents, their computational complexity increases, posing challenges for real-time scalability. Moreover, effectively integrating complex dynamics and heterogeneous contextual cues remains a significant difficulty.

3.2.2 Link-level Analysis

Effective traffic management at the link level is essential for maintaining smooth traffic flow and ensuring safety on individual road links within transportation networks. This process includes accurate predictions and estimations of traffic conditions such as flow, speed, and travel time. DGMs play a pivotal role at this scale by providing advanced insights that support real-time traffic management, routing optimization, and congestion control. By leveraging the power of DGMs, traffic managers can respond more effectively to changing conditions on each link and anticipate future variations with greater precision. Table 7 provides an overview of recent studies, classifying them based on model type, observation/prediction length, sampling time, prediction samples, dataset, and evaluation metrics.

As discussed earlier, one of the benefits of using DGMs in estimation and prediction tasks is their ability to learn the characteristics and distributions from training data, enabling them to generate realistic and reliable data for effective traffic decision-making (Lin et al., 2018; Chen et al., 2018; Zhang et al., 2019a,b; Impedovo et al., 2019; Zang et al., 2019; Xu et al., 2020a; Li et al., 2020b; Zhou et al., 2020a; Yu et al., 2020a; Aibin et al., 2021; Sun et al., 2021b; Zhang et al., 2021a; Song et al., 2021; Jin et al., 2022; Wang et al., 2022a; Zhao et al., 2022; Mo et al., 2022a; Xu et al., 2022; Khaled et al., 2022). For example, Yu and Gu (2019) proposed a deep neural network architecture called the Graph Convolutional Generative Autoencoder (GCGA) to address real-time speed estimation problems. In GCGA, the Graph Convolutional Network (GCN) extracts spatial characteristics, and an autoencoder-based GAN uses these features to generate traffic speed maps from incomplete data. Similarly, Li et al. (2019b) presented DeepGTT, a VAE-based model designed to predict travel time distributions by integrating real-time traffic data and spatial features. The VAE enables the model to generate realistic travel time distributions even under conditions of data sparsity and variability. In another study, Xu et al. (2020b) developed the Graph Embedding GAN (GE-GAN), which selects adjacent links to estimate road traffic speed and volume more accurately. To be more specific, it uses Graph Embedding (GE) techniques to represent the spatial characteristics of road networks, while a Wasserstein GAN (WGAN) is employed to generate traffic state data. Yu et al. (2020b) introduced a framework combining clustering and generative models to predict taxi hotspots. This model LSTM-CGAN, which integrates LSTM with CGAN, learns the spatiotemporal distribution of taxi hotspots to generate accurate predictions. Moreover, Zhou et al. (2020a) proposed a Bayesian framework that combines VAEs with GNNs for robust traffic prediction. The model benefits from the generative capabilities of VAEs and Normalizing Flow to capture multimodal traffic data distributions, addressing the uncertainty and complexity of road sensor networks.

Many studies of DGMs focus on learning complex data distributions to generate accurate and realistic outcomes. However, an equally important but less explored aspect is the density estimation of outputs. Without probabilistic modeling, it is challenging to differentiate forecasts between low and high-noise scenarios, which is essential for offering a full probabilistic view and enhancing decision-making under uncertainty (Arnelid et al., 2019; Rasul et al., 2020, 2021; Wen et al., 2023; Lin et al., 2024). Rasul et al. (2020) introduced an autoregressive deep learning framework for multivariate probabilistic time series forecasting that leverages conditioned normalizing flow. Their model learns the

Table 7
A summary of recent studies in link-level analysis using DGM

Author	Model	Task	Input / Prediction Length	Sampling Time	Prediction Samples	Dataset	Evaluation Metrics
Yu and Gu (2019)	GCGA (GAN-based)	Speed Estimation	-	-	1	Cologne Speed Data	MAPE
Li et al. (2019b)	DeepGTT (VAE-based)	Travel Time Distribution Estimation	-	-	1	Chinese Provincial Capital City Taxi Dataset	RMSE, MAE
Xu et al. (2020b)	GE-GAN	Volume Estimation Speed Estimation	-	-	1	PEMS Seattle Dataset	RMSE, MAE, MAPE
Yu et al. (2020b)	LSTM-CGAN	Taxi Hotspots Prediction	60min/10min	10min	1	Beijing Taxi trajectory Taximeter Data	1-Recall, FPR, ROC, Section Consistency
Zhou et al. (2020a)	VGRAN (VAE + Flow)	Speed Prediction	60min/5-60min	5min	1	METR, PEMS	RMSE, MAE, MAPE
Rasul et al. (2020)	Flow	Flow Prediction Volume Prediction	Flexible/24hr Flexible/12hr	1hr 0.5hr	100	PEMS-SF NYC Taxi Data	$CRPS_{sum}$, MSE
Rasul et al. (2021)	TimeGrad (Diffusion-based)	Flow Prediction Volume Prediction	Flexible/24hr Flexible/12hr	1hr 0.5hr	100	PEMS-SF NYC Taxi Data	$CRPS_{sum}$
Wen et al. (2023)	DiffSTG (Diffusion-based)	Flow Prediction	1hr/1hr	5min	100	PEMS08	CRPS, RMSE, MAE
Feng et al. (2024)	LDT (Diffusion-based)	Flow Prediction Volume Prediction	Flexible/24hr Flexible/12hr	1hr 0.5hr	100	PEMS-SF NYC Taxi Data	$CRPS_{sum}$, MSE

joint distribution of future observations by conditioning on historical data, enabling it to capture complex dependencies among multiple time series. This approach not only improves prediction accuracy but also provides robust uncertainty estimates. Building on this work, Rasul et al. (2021) proposed TimeGrad, which employs a denoising diffusion model for the same forecasting task. TimeGrad generates forecasts by progressively transforming white noise into meaningful data through a learned Markov chain. At each time step, it estimates the gradient of the data distribution and uses Langevin sampling to refine the predictions, further improving performance and uncertainty quantification. Similarly, Wen et al. (2023) developed DiffSTG, a framework for probabilistic Spatio-Temporal Graph (STG) forecasting that combines Spatio-Temporal GNNs (ST-GNNs) with DDPMs. DiffSTG captures both spatial and temporal dependencies in STG data while modeling inherent uncertainties. This model addresses the inefficiencies of TimeGrad in long-term forecasting by using a non-autoregressive approach to predict multiple future horizons simultaneously. In another study, Feng et al. (2024) introduced the Latent Diffusion Transformer (LDT) for high-dimensional multivariate probabilistic time series forecasting. LDT employs a latent space approach combined with a diffusion-based conditional generator and a symmetric statistics-aware autoencoder to enhance the expressiveness and manageability of forecasting complex time series data. Building on previous studies, DGMs have demonstrated robust capabilities in link-level analysis, providing essential tools for predicting and managing traffic conditions on road networks and thereby contributing to more efficient and responsive traffic management systems.

One of the limitations is that, compared to agent-level studies, there is a notable gap in research on density estimation at the link level. Furthermore, while some probabilistic models excel at quantifying uncertainty, they often show less performance in point forecast accuracy relative to deterministic methods. Their reliance on variational inference can also lead to inaccurate posterior estimates when data is limited, underscoring a trade-off between capturing uncertainty and achieving precise forecasts.

3.2.3 Region-level Analysis

At the regional level, traffic management includes estimating and predicting traffic conditions across large geographic areas encompassing multiple road networks and transportation systems. This type of analysis is essential for understanding broader traffic patterns and making informed decisions that impact urban planning, resource allocation, and emergency response strategies. Key tasks at the region level include predicting and estimating average speeds across the network, estimating OD flows to understand travel patterns, forecasting traffic demand to anticipate future needs, analyzing overall traffic flow to identify congestion areas, and performing crash analysis to enhance network safety. The complexity of managing traffic at the regional level arises from the need to integrate data from diverse

sources and understand the dynamic interactions between various components of the transportation network. From this perspective, DGMs are particularly well-suited due to their capability to learn from extensive datasets, capture complex spatial and temporal patterns, and model uncertainties of output results(Liang et al., 2018; Saxena and Cao, 2019; Zhang et al., 2020c; Li et al., 2020a; Kakkavas et al., 2021; Wu et al., 2021; Feng et al., 2021; Kakkavas et al., 2021; Naji et al., 2021; Wang et al., 2022b; Mo et al., 2022b; Li et al., 2022c,b; Yuan et al., 2023; Lin et al., 2023b; Zhang et al., 2023a; Zarei et al., 2024; Li et al., 2024b; Shao et al., 2024). In Table 8, we summarize recent studies by considering the model used, task, observation/prediction length, sampling time, dataset, and evaluation metrics.

Yu et al. (2019) integrated a modified DBSCAN algorithm with a CGAN model built on LSTM to predict taxi-passenger demand. The model accounts for spatial, temporal, and external dependencies of input data, thereby providing accurate demand predictions. In another study, Wang et al. (2020b) introduced the SeqST-GAN, a sequence-to-sequence GAN model comprising LSTM and CNN for multi-step urban crowd flow prediction. This model enhances long-term prediction accuracy by treating crowd flow data as "image frames" and incorporating contextual features such as weather, holidays, and points of interest. Furthermore, Zhang et al. (2020b) developed Curb-GAN, a conditional GAN-based model for estimating urban traffic conditions under various travel demand scenarios. Curb-GAN integrates dynamic convolutional layers to capture local spatial correlations along irregular road networks and self-attention mechanisms to model temporal dependencies, producing accurate and realistic traffic estimations. Li et al. (2022a) proposed the Attentive Dual-Head Spatial-Temporal GAN (ADST-GAN) to predict crowd flows. The model integrates attentive temporal and spatial mechanisms by combining ConvLSTM, self-attention, and a dual-head discriminator within the GAN framework. These components capture complex spatial-temporal dependencies in crowd flow data and mitigate overfitting, ensuring that the synthetic data generated closely mimics real-world traffic patterns. Moreover, Huang et al. (2022b) presented the Dynamic Multi-Graph Convolutional Network with GAN framework (DMGC-GAN) for OD-based ride-hailing demand prediction. The framework builds dynamic, directed OD graphs—capturing geographic adjacency, mutual attraction, and mobility associations—to tackle data sparsity and complex spatio-temporal dependencies. These graphs are processed by a Temporal Multi-Graph Convolutional Network (TMGCN) with a GRU and integrated into a GAN framework to refine predictions.

Previous research has shown the effectiveness of DGMs in managing and predicting traffic conditions at the regional level, providing valuable insights for strategic planning and operational decision-making across large transportation networks. However, they also face several limitations. Many models depend heavily on high-quality, abundant training data, which may not always be available—especially when sudden or atypical demand scenarios occur. Additionally, many approaches rely on grid-based or clustering partitioning of urban areas, a method that can oversimplify and fail to capture the complex geometries of real-world road networks. Finally, despite the use of DGMs to improve output quality, there remains a risk of generating overly smooth or “blurry” predictions that obscure critical localized variations.

3.3. DGM for Unsupervised Representation Learning

Unsupervised representation learning involves training a model to learn useful features or representations of the data without requiring labeled inputs. This method is valuable in environments where labeled data is sparse or expensive, which is often the case in transportation datasets. DGMs, such as VAEs, are adept at discovering intricate structures in unlabeled data by learning to compress data and reconstruct it back into the original space. In other words, DGMs learn to encode data into a compact, latent space—a lower-dimensional representation of the original data—which preserves much of the information but in a more compressed form. The latent vector is a compact representation of the input data that captures its most critical features. These vectors are the output of the model’s encoder component and serve as a compressed knowledge base of the data, containing the essential information needed to reconstruct or generate new data points. In transportation research, the manipulation and analysis of these latent vectors allow researchers to infer traffic patterns, and anomalies that might not be apparent in the high-dimensional original data.

One primary research direction is feature extraction and classification. It gains popularity in the field of transportation when employing DGMs, particularly because of their potent capability to discern and categorize complex, multi-dimensional data without direct supervision. Many studies have used the latent space in various classification tasks such as transportation mode identification, driving behavior analysis, and anomaly detection (Krajewski et al., 2018; Rákos et al., 2021, 2020; Yao and Bekhor, 2022; Rakos et al., 2021; De Miguel et al., 2022; Xie et al., 2023; Santhosh et al., 2021; Islam et al., 2021; Ding et al., 2019; Yuan et al., 2023; Kim et al., 2021). For clarity, Table 9 summarizes these recent studies, detailing the models used, latent space functions, and datasets. One notable study by Boquet et al. (2020) used the VAE to represent complex, high-dimensional traffic data. The VAE learns a low-

Table 8
A summary of recent studies in region-level analysis using DGM

Author	Model	Task	Observation / Prediction Length	Sampling Time	Dataset	Evaluation Metrics
Yu et al. (2019)	cGAN	Taxi-Passenger Demand Prediction	60min/10min	10min	Beijing Taxi Dataset, New York City Taxi Dataset	MSE, MAE, MAPE
Zhang et al. (2020b)	Curb-GAN	Speed Prediction Inflow Prediction	12hr/1-12hr	1h	Shenzhen Speed Dataset Taxi Inflow Dataset	RMSE, MAPE
Wang et al. (2020b)	SeqST-GAN	Crowd Flow Prediction	24hr/1-20hr	1hr	BikeNYC, TaxiNYC	RMSE, MAE
Li et al. (2022a)	ADST-GAN	Crowd Flow Prediction	24h/1-4h	1h	BikeNYC, TaxiNYC	RMSE
Huang et al. (2022b)	DMGC-GAN	OD Ride-Hailing Demand Prediction	200min/20min	20min	New York City Taxi, Limousine Commission Ride-Hailing Dataset	RMSE, MAE, MAPE, R^2

dimensional latent space representation that captures the underlying patterns and dependencies within the traffic data. This latent space can be used in various tasks such as missing-data imputation, anomaly detection, traffic forecasting, Likewise, Neumeier et al. (2021) proposed an unsupervised model based on the VAE architecture to predict vehicle trajectories with an interpretable latent space. The latent space can be analyzed to understand and predict lane-changing maneuvers, enhancing the model's utility in studying vehicle behaviors. Moreover, Santhosh et al. (2021) developed a novel approach combining CNN and VAE to classify vehicle trajectories and detect anomalies. This hybrid model addresses the challenges of classifying time-series data of varying lengths and identifying anomalies such as lane violations, sudden speed changes, and vehicles moving in the wrong direction. The visualization of the latent space in their model provides a clear understanding of how the VAE encodes trajectory data and distinguishes between normal and anomalous patterns. Chen et al. (2021c) proposed a method for generating realistic traffic flow data using GANs with semantic latent code manipulation. By exploring and manipulating the semantic representations in the latent space, their model generates traffic flow data that accurately reflects realistic patterns and conditions. Additionally, Zhang et al. (2022c) introduced the Dirichlet VAE (DirVAE) to identify transportation modes from GPS trajectory data. The latent space of the DirVAE visualizes and classifies different transportation modes, demonstrating the utility of the latent space in distinguishing between various modes of transportation.

These studies demonstrate that DGMs are effective for feature extraction and classification in transportation. Leveraging the latent space allows researchers to reveal intricate patterns and dependencies, leading to more precise and insightful analysis of transportation data. Future work should aim to develop methods that efficiently extract and utilize the information in latent representations, thereby enhancing model robustness and interpretability and deepening our understanding of transportation phenomena.

3.4. Summary

This section reviewed the significant role of DGMs in transportation research, focusing on their applications in data generation, estimation and prediction, and unsupervised representation learning. In data generation, DGMs are invaluable for producing synthetic datasets that replicate real-world traffic conditions. This capability is essential for training models, testing new scenarios, and supplementing real data, particularly when data collection is challenging or costly. DGMs can generate diverse traffic scenarios, realistic trajectories, and even complete missing data, thereby ensuring comprehensive and robust datasets for traffic analysis and management. For estimation and prediction, DGMs excel in modeling the uncertainties and variations inherent in traffic data. By learning from extensive datasets, they provide probabilistic forecasts of traffic conditions and enable effective decision-making for traffic management and planning. These models support the anticipation and adaptation to probable future traffic patterns, contributing to more efficient and responsive transportation systems. Lastly, in unsupervised representation learning, DGMs facilitate the analysis of high-dimensional traffic data by encoding it into compact, interpretable latent spaces. This enables the classification of transportation modes, analysis of driving behaviors, and detection of anomalies, providing deeper insights without the need for labeled data. The ability of DGMs to reveal hidden structures within data is particularly

Table 9
A summary of recent studies in unsupervised representation learning using DGM

Author	Model	Latent Space Function	Dataset
Boquet et al. (2020)	VAE	Missing-data Imputation, Anomaly Detection, Traffic Forecasting	PEMS, UKM1, UKM4
Neumeier et al. (2021)	DVAE	Understanding Lane-Changing Maneuvers	HighD
Santhosh et al. (2021)	CNN-VAE	Distinguish Normal and Anomalous Patterns	T15, QMUL, 4WAY datasets
Chen et al. (2021c)	GAN	Linear Interpolation, Manipulation Modifies Traffic-Flow Properties	PEMS
Zhang et al. (2022c)	DirVAE	Transportation Mode Classification	Geolife V1.3, OSMNX MTL Trajet 206,2017

valuable for understanding and improving transportation systems.

Despite these advances in transportation studies, future research must address several key challenges. Standardizing evaluation methods is crucial to ensure consistency and comparability across studies. Future studies should also consider the dynamic characteristics of traffic data, which vary significantly across different spatial and temporal contexts. Additionally, understanding and incorporating the causal relationships within traffic data, as well as addressing ethical and privacy concerns associated with synthetic data generation, are critical areas that require further exploration. These challenges, along with other emerging trends and opportunities in the use of DGMs for transportation research, will be discussed in more detail in Section 5.

4. Tutorial

In this section, we present practical examples of how DGMs can be applied in transportation research. To reach a broader audience, we provide two hands-on tutorials: 1) Generating Household Travel Survey Data in Section 4.1, and 2) Generating Highway Traffic Speed Contour (Time-Space Diagram) in Section 4.2. To keep the paper concise, detailed explanations of the model structure and its loss functions in the code are provided in Appendices F and G. Importantly, all data and code used in these tutorials—including preprocessing scripts, model training, inference code, and pre-trained model parameters—are available in our GitHub repository: `https://github.com/UMN-Choi-Lab/DGMinTransportation`. The tutorial code is implemented in Python, with PyTorch serving as the primary library for the tutorials. Additional requirements are detailed in the associated GitHub repository.

This tutorial aims to equip researchers with the necessary tools to explore the application of DGMs in transportation studies. Accordingly, we structured the tutorial and developed the accompanying code using one representative discrete dataset and one representative continuous dataset. Given the scope of this paper, it is impractical to address every possible data type; however, by covering these key examples, we expect to offer a wide range of valuable insights. Moreover, our goal is not to identify the top-performing models within each research domain. To this end, although we performed numerical evaluations, the results can vary significantly depending on the numerous hyperparameters and optimizers used for each model. Consequently, we want to emphasize that readers should not judge any particular model solely based on the results presented in this paper. Instead, the focus is on providing qualitative insights, which is a common practice in DGM research. For readers interested in rigorous performance evaluations, including numerical or analytical assessments, we suggest referring to the existing literature. Nevertheless, the code provided in our repository offers a robust foundation for initiating DGM-based research, enabling researchers to extend these models for specific transportation study requirements.

The code implementations for the tutorial and experiments are conducted in a Python-based environment. Key libraries used include `torch` (tested both on v1.13 and v2.4) for deep learning model development, leveraging GPU acceleration to handle large-scale data and neural networks efficiently, and `numpy` and `pandas` for numerical computations and data manipulation. All development is done in a notebook environment to allow for interactive data exploration, experimentation, and real-time feedback. This setup also ensures compatibility with widely used tools, including Jupyter Notebook and Google Colab, offering flexibility for development and testing.

4.1. Generating Household Travel Survey Data

Household Travel Survey (HTS) data provides a comprehensive overview of individuals' daily travel patterns, modes of transportation, and trip purposes within sampled households. These datasets often include sociodemographic and household-level characteristics, making them indispensable for transportation planners, policymakers, and researchers. HTS data is crucial for understanding mobility patterns, forecasting travel demands, and informing infrastructure developments, public transit planning, and sustainable transportation initiatives (Zhang and Li, 2022). However, traditional methods of collecting HTS data, such as face-to-face interviews or telephone surveys, typically yield sample sizes representing only 1–5% of the population in a given region (Stopher and Greaves, 2007), raising concerns about their representativeness and accuracy. Additionally, low sample rates, high survey costs, and safety concerns of interviewers further complicate the data collection process. While some countries, like Singapore, have adopted the online HTS collection system, questions about participant fidelity remain. These limitations underscore the need for innovative approaches, such as synthetic HTS data generation, to more accurately represent population-wide travel behaviors.

Advanced artificial intelligence techniques, particularly DGMs, have emerged as a promising solution to address the constraints of traditional HTS data collection. While prior studies have explored the use of GAN or VAE for synthetic data generation (Kim and Bansal, 2023; Garrido et al., 2020; Borysov et al., 2019), other DGMs remain underexplored. Therefore, this section offers a tutorial on generating a simplified version of HTS data using DGMs, demonstrating their potential to overcome traditional data collection limitations and laying the groundwork for future research and applications in this field.

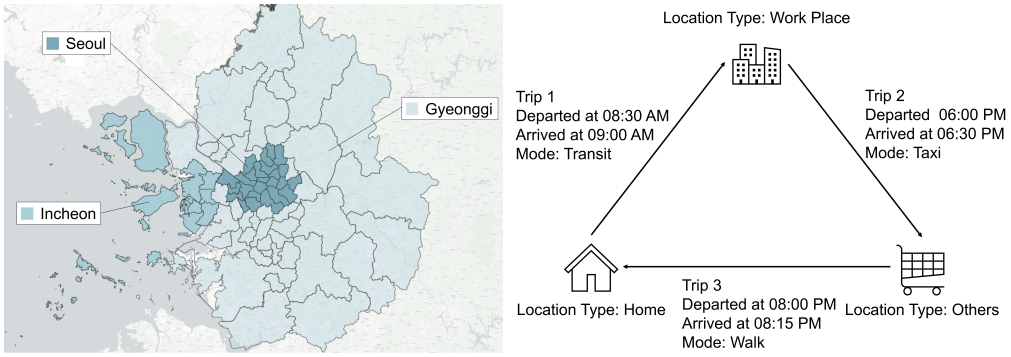
4.1.1 Data and Preprocessing

In this tutorial, we applied DGMs to the 2010 HTS data from South Korea. This dataset provides comprehensive details on household travel patterns, including trip purposes, transportation modes, and travel times across various cities and districts in the region, making it a valuable resource for investigating urban mobility dynamics. While the original HTS dataset covers the entire country, this study only focuses specifically on the Seoul Metropolitan Area, as illustrated in Figure 7a. The Seoul Metropolitan Area encompasses Seoul, Incheon, and Gyeonggi-do, located in the northwest region of South Korea. With a population of approximately 26 million, this area represents more than half of the country's total population.

The original HTS includes both sociodemographic and travel data for households. For simplicity, we exclude the sociodemographic data and focus solely on travel data. An example of the HTS travel data used in this study is shown in Figure 7b. This travel data, which comprises three different trips, can be transformed into tabular data as shown in Table 10. To simplify the problem, we selected the types of origin, activity, mode choice, and destination as target features for data generation. The detailed explanation about each column in HTS data is illustrated in Appendix H

In our tutorial example, each DGM is designed to generate each row that contains the data in Table 10 by learning the joint distribution of the selected features. The data includes origin and destination types divided into five categories: home, work, school, transfer points, and other locations. Transportation modes are categorized into nine discrete integers: 0 for the start of the day, 1 for the end of the day, 2 for staying at a location, and 3 through 8 representing various modes of transport such as walking, public transit, driving (driver and passenger), cycling, and taxis. Lastly, activity types include eight categories: work, education, leisure, shopping, and escort activities. It is important to note that the goal of this tutorial is not generating the complete daily travel, but rather focusing on individual trips. This allows the model to capture the relationships between features. During model evaluation, we present qualitative assessments with marginal distributions of each feature. This qualitative approach prioritizes interpretability and understanding over direct numerical performance metrics. Nevertheless, the model is trained to capture the joint probability distribution, emphasizing its ability to generate realistic combinations of attributes.

Most DGMs are typically designed to operate in continuous spaces, learning to map the probability distribution of data. This design presents challenges when applied to discrete datasets, such as HTS data. When dealing with discrete variables, the probability distribution only has values at specific discrete points, while the probability is zero everywhere else. Additionally, to normalize the probabilities to sum to 1, the probability at these specific data points can become infinitely large. This causes DGMs to assign infinite likelihood to certain points, which makes the model unstable. To address this issue, adding noise to the discrete variables, i.e., relaxed one hot encoding, was suggested from the literature. (Jang et al., 2016) presented Categorical VAEs, which incorporate noise from Gumbel-Softmax distribution. (Maddison et al., 2016) showed a similar approach using Concrete distribution to approximate the discrete value into continuous value.



(a) (b)

Figure 7: Visual Description of Household Travel Survey dataset. (a) Study Area: Seoul Metropolitan Area (b) Example daily travel in the HTS data

Table 10
Trip chain data generated from example daily travel in Figure 7b

	Origin Type	Origin District Num.	Departure Time	Activity Type	Mode Type	Destination Type	Destination District Num.	Arrival Time
1	Home	Gyeonggi-1	00:00 AM	Start	Stay	Home	Gyeonggi-1	08:30 AM
2	Home	Gyeonggi-1	08:30 AM	Travel	Transit	Work Place	Seoul-5	09:00 AM
3	Work Place	Seoul-5	09:00 AM	Work	Stay	Work Place	Seoul-5	06:00 PM
4	Work Place	Seoul-5	06:00 PM	Travel	Taxi	Others	Gyeonggi-2	06:30 PM
5	Others	Gyeonggi-2	06:30 PM	Shopping	Stay	Others	Gyeonggi-2	08:15 PM
6	Others	Gyeonggi-2	08:15 PM	Travel	Walk	Home	Gyeonggi-1	08:30 PM
7	Home	Gyeonggi-1	08:30 PM	End	Stay	Home	Gyeonggi-1	12:00 PM

In this tutorial, we use a simple dequantization technique to address the challenges of generating discrete data using DGMs. Rather than relying on complex methods such as Gumbel-Softmax to model discrete or categorical distribution, we add *uniform random noise* to each discrete data sample before training the model, effectively transforming the discrete data into continuous space. After data generation using DGMs, we apply a *floor* operation to convert it back to the discrete form. Mathematically, this process transforms the generation of a discrete value $\mathbf{x}_i$ into generating a continuous value within the range $[\mathbf{x}_i, \mathbf{x}_i + 1)$. Consequently, computing the probability $p(\mathbf{x})$ is expressed as:

$$p(\mathbf{x}) = \int p(\mathbf{x}+\mathbf{u})d\mathbf{u} = \mathbb{E}_{\mathbf{u}\sim q(\mathbf{u}|\mathbf{x})}\left[\frac{p(\mathbf{x}+\mathbf{u})}{q(\mathbf{u}|\mathbf{x})}\right], \tag{4.1}$$

where $q(\mathbf{u}|\mathbf{x})$ is the noise distribution. Since we assume it to be uniform, this can be rewritten as:

$$p(\mathbf{x}) = \mathbb{E}_{\mathbf{u}\sim U[0,1)^D}\left[p(\mathbf{x}+\mathbf{u})\right], \tag{4.2}$$

where $U[0,1)^D$ is a D-dimensional (data dimension) random uniform vector. This method effectively smooths the distribution and ensures stable training and generation of discrete variables. This approach works well across various DGMs, as we demonstrate in this tutorial. The Python implementation of this dequantization process is included in the accompanying codebase. All five DGMs presented in this tutorial follow the same dequantization process, ensuring consistency and comparability across models.

Table 11
Performance comparison across different DGMs in HTS Data

	VAE	GAN	NF	NCSN	DDPM
SRMSE	0.53550	0.48710	**0.32318**	1.39829	1.09457
MAE	0.03031	0.02350	**0.01432**	0.04501	0.03346
KLD	0.04774	0.05254	**0.01605**	0.07657	0.05494

4.1.2 Results Comparison

In this section, we share the empirical insights gained during model training. Further details about each model can be found in Appendix F. Figure 8 presents the results of generating HTS data.

Similar to how VAEs tend to generate slightly blurry images in image generation, in this experiment with discrete values, the probability distribution also appeared relatively uniform rather than being overly skewed towards one side. However, it often struggled to accurately match the activity type and mode type, depending on the specified conditions. This may be attributed to the increased complexity resulting from the higher number of attributes. In contrast, the GAN frequently encountered mode collapse, leading to cases where all probabilities were assigned to a particular activity or mode. However, this issue was less pronounced in cases involving fewer data attributes, such as problems related to origin location and destination location type. The Normalizing Flow model demonstrated superior performance, with robustness across a wide range of hyperparameters. The loss reduction pattern followed the typical behavior expected in learning models, with an initial sharp decrease followed by a gradual decline in later stages. Throughout the training process and in the results, the model consistently performed well. The Diffusion model, which is commonly built using a U-Net architecture, was instead implemented with a simpler three-layer neural network in our approach. Despite initial concerns regarding its effectiveness, the model performed relatively well due to the simplicity of the dataset. However, occasional loss spikes occurred, which led the model to failure of training. Overall, the learning process remained stable, but future experiments could benefit from more sophisticated neural network architectures to further enhance performance. The NCSN model exhibited high sensitivity to hyperparameters. For instance, when the max and min sigma values were low, the model displayed characteristics of high entropy or temperature, leading to unpredictable performance. Conversely, higher sigma values resulted in one-hot-like behavior. Additionally, the model's performance varied across different categories: when it performed well on generating the location type, performance on generating types of action and mode became poor, and vice versa. Balancing hyperparameters effectively remains a key challenge for ensuring consistent performance across all attributes.

Table 11 presents the numerical results obtained by comparing the distribution of the ground truth data with that of the generated data using the SRMSE, MAE, and KLD[2] metrics. For SRMSE, we computed the distribution for each unique combination of two columns and then calculated the SRMSE from these distributions of ground truth and generated data; this procedure was repeated for all combinations of two columns, and then we averaged the sum of it. As we mentioned, results presented herein are provided solely as examples, and readers are encouraged to modify various hyperparameters and optimizers using the code available on GitHub to obtain alternative outcomes. In the Table 11, the VAE and GAN models yielded SRMSE, MAE, and KLD values of 0.53550/0.03031/0.04774 and 0.48710/0.02350/0.05254, respectively, indicating relatively higher error levels. Notably, the Normalizing Flow model demonstrated superior performance with values of 0.32318, 0.01432, and 0.01605 for SRMSE, MAE, and KLD, respectively, thereby exhibiting higher accuracy and robustness across varied hyperparameter configurations. In contrast, the NCSN and DDPM models showed comparatively larger error metrics, with the NCSN model displaying significant sensitivity to hyperparameter adjustments. These results emphasize that model performance is intricately linked to the selected architecture and training strategies.

[2] See Appendix A

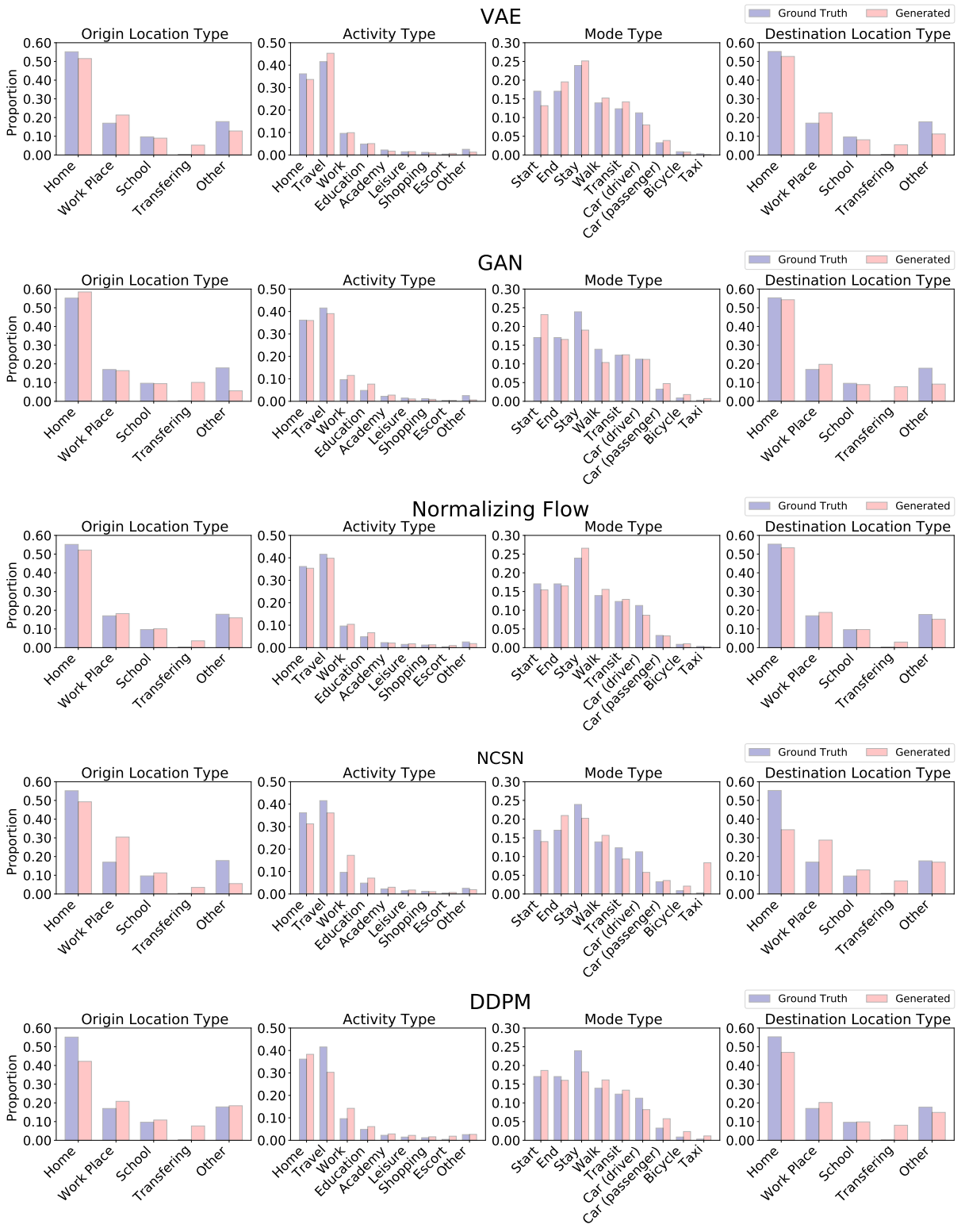


Figure 8: Results of each DGM in HTS data generation

4.2. Generating Highway Traffic Speed Contour

The generation of highway traffic speed contours plays a crucial role in analyzing and visualizing traffic dynamics across both time and space. Traditionally, traffic flow has been represented using a time-space diagram of vehicles, which depicts the movement of vehicles along a road segment over time, providing insight into congestion patterns and traffic bottlenecks. However, converting this into a time-space speed contour offers several advantages. While traditional time-space diagrams focus on the position of vehicles at specific times, speed contours emphasize variations in traffic speed across both temporal and spatial dimensions, offering a clearer representation of traffic conditions such as slowdowns, jams, or free-flowing traffic across broader time periods. In this tutorial, we generated the traffic speed contour data based on the vehicle trajectory data.

Speed contours offer a continuous and intuitive visualization of traffic speeds across different locations and time intervals, making them particularly useful for traffic management and planning. They enable transportation professionals to quickly identify and quantify areas of concern, such as sections of highways prone to congestion or inconsistent traffic flow, thereby facilitating better decision-making. Moreover, these contours can serve as input for traffic forecasting models, aiding in the development of responsive traffic control systems, infrastructure improvements, and policy interventions aimed at reducing congestion and improving road safety. In this section, we aim to generate highway traffic speed contours from the input noise. Through the learning process, the model captures the spatial and temporal joint distribution of traffic dynamics, resulting in a detailed speed contour that reflects variations in traffic conditions.

4.2.1 Data and Preprocessing

In our tutorial section, we apply the DGMs to the Tennessee Department of Transportation's I-24 Mobility Technology Interstate Observation Network (MOTION) data (Gloudemans et al., 2023)[3]. I-24 MOTION data contains the vehicle trajectories in a four-mile section of I-24 in the Nashville-Davidson County Metropolitan area. It captures the trajectory in a high spatial and temporal resolution by using 294 ultra-high definition cameras and AI algorithms from Vanderbilt University. The AI algorithm transforms the raw videos into detailed 2-dimensional trajectories which is suitable to study vehicle behavior and traffic flow. The data collection occurred from Monday, November 21, 2022, to Friday, December 2, 2022, encompassing all eastbound and westbound lanes. The collection was done during peak morning hours, from 6 AM to 10 AM. This dataset is significant as it includes a variety of days as well as trajectories during an accident that occurred on November 21. However, it is important to note that when a single vehicle appears multiple times across different cameras, each instance is treated as a separate vehicle. Therefore, the whole trajectory will be sliced into multiple short sequences of trajectories. Consequently, this data needs a vehicle-matching algorithm to obtain the full trajectories of the single vehicle, but it can be effectively utilized for constructing time-space traffic contours without matching the vehicles.

The raw data from the MOTION project comprises segments of vehicle trajectory information. In this tutorial, our focus is on the reconstruction of traffic speed contours. To facilitate this, the trajectory data is transformed into velocity data corresponding to each specific time and position. Given that the calculation of velocity data based on trajectories inevitably results in the presence of zero or NaN values (except in the unlikely event of a vehicle being present at every possible position throughout the entire timespan), preprocessing was necessary. This preprocessing predominantly involved interpolation. The left and right of Figure 9 show the time-space traffic contours diagram before and after the interpolation.

The interpolation process is composed of two main steps. Firstly, we interpolate the zero and NaN values using linear and nearest-neighbor interpolation methods, respectively. Secondly, we apply a smoothing technique based on Edie's definition of traffic flow dynamics (Treiber and Helbing, 2003). The initial interpolation step is crucial; without it, if we attempt to interpolate every zero or NaN value directly using Edie's box method, the required size of Edie's box would become excessively large, leading to significant distortion of the dataset. Before interpolation, most of the regions are colored in red because the 0 and NaN values are depicted as red. The actual values of these regions are much higher than what is shown in the figure. After interpolation, we can observe a much clearer propagation of congestion and its subsequent dissipation. Free flow is also clearly illustrated after interpolation. The blue region forming a straight line after 3,600 seconds indicates an increase in traffic speed following the morning rush hour as the time passes 9 AM.

The detailed pseudocode for the interpolation process, based on Edie's definition of traffic flow dynamics, is provided in Appendix I. The smoothing process, known as the "adaptive smoothing method," is designed to filter out noisy

[3] https://i24motion.org/data

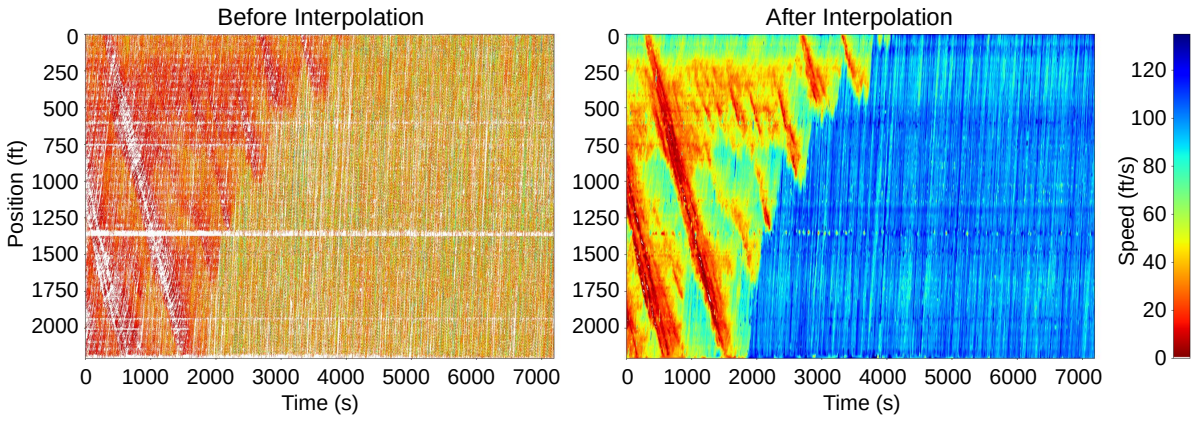


Figure 9: Comparison of traffic speed contours before and after the interpolation

fluctuations while considering the primary direction of flow propagation. This method employs spatiotemporal correlation analysis to identify the dominant characteristic line. The adaptive smoothing method uses a spatio-temporal low-pass filter that allows only low-frequency Fourier components to pass through, smoothing out high-frequency components. The filter eliminates high-frequency noises over a timescale shorter than τ and spatial noise over a length scale shorter than σ. The values for τ and σ were determined through empirical trials to ensure the data is not overly blurred while effectively passing the main propagation. The filter captures two main propagation types: free flow and congested traffic flow. These two propagations have different coefficients of velocity and direction, reflecting the distinct properties of each traffic flow type. The property of each traffic flow is reflected based on the different values of each coefficient.

4.2.2 Results Comparison

We discuss the key findings from our model training in this section. As with the HTS data generation, a comprehensive description of each model is available in Appendix G. Figure 10 presents the 64 images of ground truth speed contour data and outputs of 64 samples generated by each DGM. It is worth noting that the results generated by the DGMs in Figure 10 do not directly match the ground truth in the same figure. These images are instead "plausible" representations generated from noise. Therefore, readers should keep in mind that good images are those that closely resemble the ground truth, with clear and well-defined traffic speed patterns across the contours. In contrast, poor images may appear overly blurred, and noisy, or suffer from mode collapse, where the generated outputs fail to capture the underlying traffic dynamics.

It is evident that the VAE outputs are noticeably more blurred compared to those produced by the other models. Throughout the training process, we experimented with a wide range of neural network architectures for the VAE. Despite attempting to train the VAE with neural networks that had four to eight times more parameters than the current configuration, we were unable to eliminate the inherent blurriness of the VAE outputs. Unlike other algorithms, which exhibited some variability in performance across training sessions, the VAE consistently produced stable results both numerically and visually. However, the VAE outputs remained uniformly blurred, with some generated images appearing severely noisy and closely resembling the mean of the dataset.

In contrast, GANs demonstrated superior performance, especially given the simplicity of their neural network architecture. Even when using a neural network architecture almost identical to that of the VAE, the GAN produced clearer and more diverse images. Although the training process occasionally showed abrupt fluctuations in the loss function, as long as the generator's training did not lag significantly behind that of the discriminator, the model eventually achieved a balance. Considering the simplicity of the code, training time, and generation speed, the strong performance of such a straightforward algorithm is very encouraging.

RealNVP displayed the most variability among all the algorithms. Occasionally, it produced excellent results, but this occurred in only about 1 in 20 training runs. Since the loss function consists of four terms, the model's training trajectory varied depending on the coefficients applied to each term. The coefficients we proposed in the code were the result of extensive trial and error. It was observed that both log-likelihood and regularization were crucial, and

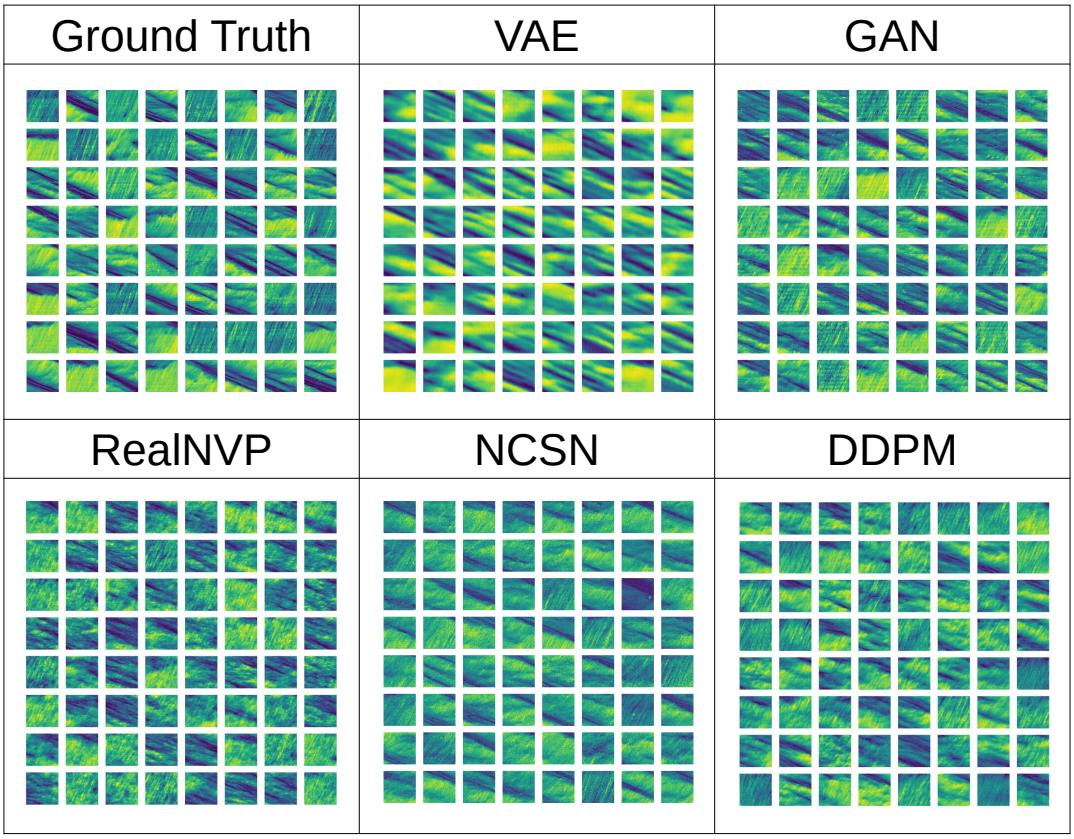


Figure 10: Ground truth data and generated samples from each DGM in highway traffic speed contour generation (Ground truth data and generated sample do not have a matching relationship. Readers should primarily focus on examining whether the visual patterns of the two datasets are similar.)

the performance was highly sensitive to the numerical value of the first term in each component. Additionally, once RealNVP fell into a local minimum during training, it was unable to escape, resulting in poor performance across all samples and mode collapse, where the images appeared more like homogeneous noise. The generated images resembled ground truth data with slightly mixed boundaries between data values.

Compared to RealNVP, DDPM demonstrated relatively stable performance. It consistently produced uniform models with good image quality across different training sessions. The resulting images resembled ground truth data with the addition of white noise, similar to photographs taken at a very high ISO setting. NCSN, which follows a mechanism fundamentally similar to that of DDPM, exhibited comparable performance. Despite the uniform resolution of 64x64, the images conveyed a sense of being highly fine-grained.

To complement our qualitative analysis, we introduced two additional metrics beyond the mean squared error (MSE) of the image distribution. The first is that we incorporated the Learned Perceptual Image Patch Similarity (LPIPS) metric to assess perceptual similarity between real and generated images(Zhang et al., 2018). LPIPS is a metric that leverages deep features extracted from pretrained convolutional neural networks, such as AlexNet, which we used in the current study, to capture both low-level details and high-level semantic information. Rather than relying solely on pixel-wise differences, LPIPS computes feature maps across multiple layers, which are first L2-normalized. The spatial and channel-wise differences between corresponding feature maps are then aggregated using spatial averaging and weighted by learned parameters that reflect each channel's importance. The final perceptual distance, obtained by summing these weighted differences across all layers, aligns closely with human perceptual judgments. In our evaluation framework, a pretrained LPIPS model—trained with human evaluation data (e.g., BAPPS) was utilized to ensure that the computed distances reliably mirror subjective image quality assessments.

Table 12
Performance comparison across different DGMs in speed contour data

	VAE	GAN	RealNVP	NCSN	DDPM
MSE	0.06646	0.08569	**0.06421**	0.10583	0.23108
LPIPS	0.38957	**0.14988**	0.20113	0.36131	0.23849
Sobel-Edge-MSE	0.00169	**0.00147**	0.00237	0.00198	0.00201

In addition to LPIPS, we employed the Sobel-Edge-MSE metric to quantitatively evaluate the structural fidelity of the generated images. This metric harnesses the classical Sobel operator to extract edge information by computing the gradient magnitude in both horizontal and vertical directions, effectively capturing the structural content of an image. By applying the Sobel operator to both the real and generated images, we obtained corresponding edge maps, and the MSE between these maps was calculated. A lower Sobel-Edge-MSE value indicates that the generated image more accurately replicates the edge features of the real image, thereby preserving crucial structural details. Shock waves in time-space diagrams are typically manifested as distinct linear or planar features, characterized by clear and well-defined boundaries that delineate abrupt transitions in traffic flow. Consequently, when the generated images exhibit low Sobel-Edge-MSE, it indicates that the model effectively captures the essential dynamics of shockwave propagation by replicating these sharp transitions and geometrical features.

We generated 600 images and randomly sampled 600 real images. Subsequently, LPIPS and Sobel-Edge scores were computed for every one of the 600×600 pairs. We then solved a matching problem to minimize the mean of the total sum of these scores. This approach was adopted because selecting only the lowest LPIPS and Sobel-Edge scores among the real samples could yield falsely high-performance metrics when a model exclusively targets a single traffic state (image) among multiple states. In other words, a model might achieve high precision by accurately generating a specific traffic state but at the expense of generation diversity (i.e., low recall), which is undesirable in image generation tasks where diversity is critical. Consequently, we established the current metric framework, which effectively serves as a surrogate for the F1 score by balancing both precision and recall. This approach might be challenged on the basis of using only 600 images, and such concerns are valid. The more images generated and compared, the better the model's actual performance can be evaluated. However, since this comparison method scales quadratically ($O(n^2)$), 600 images were empirically the maximum number of images for this evaluation framework.

Table 12 summarizes the quantitative evaluation of the generative models using the MSE, LPIPS, and Sobel-Edge-MSE metrics. Notably, the GAN model achieved the lowest LPIPS score (0.14988) and the lowest Sobel-Edge-MSE (0.00147), indicating superior perceptual similarity and structural fidelity compared to the other models. In contrast, the VAE exhibited the highest LPIPS (0.38957) and a relatively higher Sobel-Edge-MSE (0.00169), reflecting its tendency to produce blurred and less detailed outputs. The RealNVP, NCSN, and DDPM models demonstrated intermediate performance with respect to LPIPS and Sobel-Edge-MSE, suggesting a balanced capability in capturing both perceptual and structural characteristics of the ground truth data. It is important to note that MSE is not well-suited for reflecting visual similarity; in our results, the most prominent difference between the VAE and the other models is not captured by the MSE values (VAE: 0.06646, RealNVP: 0.06421, etc.). Overall, the quantitative results appear to be largely consistent with the visual outcomes: lower LPIPS scores indicate that the overall appearance of the generated images more closely resembles the real images, while lower Sobel-Edge-MSE scores suggest that the edge structures—particularly the shockwaves—exhibit greater similarity to those in the ground truth. These findings not only demonstrate the utility of these metrics in capturing essential qualitative aspects but also highlight the inherent challenges in their evaluation, given that each metric quantifies a subjective, qualitative feature.

Figure 11 illustrates averaged feature maps extracted from AlexNet for two distinct datasets: 600 ground truth images and 600 images generated by various models. Specifically, the feature map from the first convolutional layer was selected, and among its multiple channels, the first 64 channels were visualized. For each dataset, the feature maps across all 600 images were summed and then averaged to produce a representative visualization.

Overall, the general pattern of activations appears similar between the ground truth and the generated images, but qualitative differences are evident in specific regions. For instance, notable discrepancies between the ground truth and VAE outputs are observed in the feature map corresponding to the first row, fifth column, and in the seventh row, third and fourth column. In contrast, GAN-generated shows consistent results with their superior LPIPS performance. Their feature maps closely resemble those of the ground truth, albeit with a detectable difference in the second row, seventh

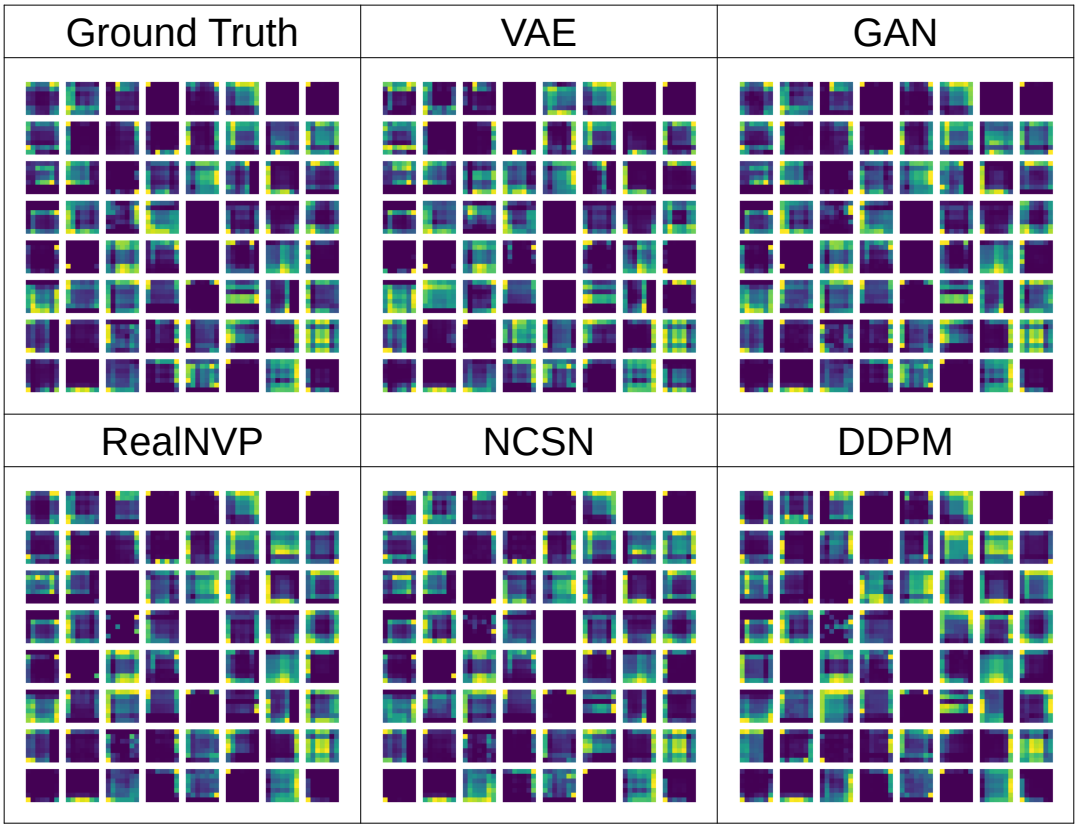


Figure 11: Feature map from ground truth data and generated samples from each DGM in highway traffic speed contour (This is a visualization of the 64 channels, from one random image from each set)

column. Similarly, RealNVP demonstrates differences in the image at the sixth row and sixth column. NCSN shows significant differences in the sixth row, last column and at the last row, third column, while DDPM exhibits significant deviations in the third column of the second row, the last column of the second row, and the last column of the third row. It should be emphasized that these observations are qualitative in nature. Nonetheless, the qualitative analysis aligns with the LPIPS evaluation, particularly highlighting the high degree of feature similarity in GAN outputs. This indicates that while all generative models generally capture the underlying structure of the ground truth images, subtle, localized discrepancies in feature representation may affect perceptual similarity metrics.

In overall, it is crucial to approach model performance evaluations with caution. Qualitative and quantitative comparisons do not provide a comprehensive assessment of model efficacy. The performance of these models can vary significantly based on hyperparameter tuning and architectural choices, making it challenging to definitively claim the superiority of any single neural network configuration or hyperparameter set. Interestingly, our empirical observations during implementation revealed instances where simpler network structures yielded superior results, underscoring the complexity of model optimization. This phenomenon highlights the importance of thorough experimentation and the potential pitfalls of overgeneralization in neural network performance assessment. Future research should focus on developing more robust quantitative metrics for model comparison, considering the multifaceted nature of generative model performance across various tasks and datasets.

Figure 12: Challenges and opportunities of applying DGMs in transportation research

5. Challenges and Opportunities

Figure 12 provides a structured overview of the challenges and opportunities in applying Deep Generative Models (DGMs) to transportation research as discussed in this paper. The left side highlights key challenges, including evaluation and validation (Section 5.1) and dynamic characteristics of transportation data (Section 5.2). These challenges stem from the difficulties in assessing generative models in complex transportation systems and the evolving nature of mobility data, which can introduce distribution shifts over time.

On the right side, the figure presents opportunities for advancing DGMs in transportation research. These include leveraging DGMs for privacy preservation (Section 5.3), ensuring equitable data generation to mitigate bias (Section 5.4), fostering trustworthy AI through transparency and fairness (Section 5.5), and integrating causal generative models to enhance interpretability and generalizability (Section 5.6). By addressing the challenges and capitalizing on these opportunities, DGMs can become powerful tools for transportation research.

5.1. Evaluation and Validation of Generative Models

Deep generative models (DGMs) aim to capture the underlying probabilistic distributions of complex, high-dimensional data. Evaluating these models is inherently multifaceted, requiring a blend of rigorous quantitative metrics and nuanced qualitative assessments. For transportation applications, this strategy must address several key dimensions: ensuring fidelity through robust metrics to confirm that synthetic data closely replicate real-world statistical properties; maintaining realism by incorporating physical laws that govern traffic dynamics; and preserving diversity to capture the full spectrum of observed behaviors, thereby avoiding pitfalls like mode collapse. Additionally, it is essential to thoughtfully model the complex, structured nature of transportation data. Integrating these elements is crucial for generating synthetic data that are not only statistically robust but also practically valuable for simulation, decision support, and further analysis.

Evaluating DGMs involves assessing how well these models generate data that accurately reflect the underlying distribution of observed data using quantitative metrics. Achieving high fidelity in transportation data analysis is both critical and challenging, largely because available datasets are often limited, increasing the risk that deep learning models may overfit if not properly validated. One principled approach is to use scoring rules (Gneiting and Raftery, 2007) that quantify how well the predicted probabilities align with actual outcomes. For example, the Log Score measures the negative log-likelihood of the synthetic data under the probabilistic distribution produced by the model. In addition, the Continuously Ranked Probabilistic Score (CRPS) and the Energy Score (ES) provide measures of how well the cumulative distribution functions capture the nuances of continuous univariate and multivariate data, respectively. These robust scoring rules have been used as loss functions to optimize generative neural networks without the need for adversarial training (Bouchacourt et al., 2016; Pacchiardi et al., 2024; Chen et al., 2024a; Shen and Meinshausen, 2024; De Bortoli et al., 2025). Another key challenge lies in ensuring realism. Traffic data are governed by physical laws, and DGMs must generate outputs that adhere to these constraints. When creating synthetic vehicle trajectories, it is essential that the models incorporate realistic driving dynamics—such as smooth acceleration, deceleration, and turning behaviors—to replicate actual traffic flow patterns. If models rely solely on spatial data without integrating temporal dynamics (like acceleration or turning angles), the resulting trajectories may appear statistically plausible but fail to capture the underlying physical realities, reducing their practical utility. This issue is similarly evident in the generation of traffic speed data as we introduced in Sec 4.2, which are fundamentally produced by the dynamics of

microscopic traffic flow. Treating such data merely as images or numerical arrays can strip away their inherent physical context, leading to unrealistic outputs that violate known traffic flow patterns.

In addition to fidelity and realism, it is crucial for DGMs to capture the full range of variability inherent in transportation data. Real-world traffic exhibits a broad spectrum of behaviors—ranging from individual driving styles to complex vehicle interactions. A common pitfall is mode collapse, where the generated outputs lack diversity and fail to represent rare but significant events. Ensuring diversity is essential for reproducing realistic collective behaviors and is especially important in risk-sensitive applications such as training models for autonomous driving. A further challenge in the transportation domain is the definition of an appropriate probability distribution for complex, structured data. Consider travel survey data presented in Section 4.1 as an example, where multiple constraints exist at different hierarchical levels (Sun et al., 2018): household composition follows specific structural patterns; individuals within the same household often share correlated socio-demographic features; and the sequence of trips must adhere to logical orderings, with interdependencies such as an adult's trip to drop off a child. Simplifying these relationships—often by assuming independence among household members—can ease computational challenges but may significantly compromise the quality and representativeness of the synthetic data.

5.2. Dynamic Characteristics of Transportation data

Mobility patterns and traffic data are continually evolving due to factors such as seasonal variations, changes in travel behavior, infrastructure updates, and technological advancements. These evolving conditions can lead to shifts in the underlying data distribution, posing challenges for DGMs trained on historical data. Such shifts can decrease the performance and reliability of the trained DGMs, especially in long-term deployments or applications where transportation patterns change over time. For instance, in tasks like driving scenario generation or population synthesis, the introduction of new vehicle types (e.g., electric or autonomous vehicles) or emerging transportation modes (e.g., shared micro-mobility) can alter driving behaviors and mode choice patterns, making the synthetic datasets generated by DGMs outdated. In anomaly data generation tasks, where anomalies are outliers in the data distribution (e.g., incidents, road blockages, or sudden traffic surges), dynamic changes in transportation can cause shifts in both normal and anomaly data distributions. This makes it harder to detect shifts in anomaly patterns and further complicates the distinction between normal and anomalous data. In estimation and prediction tasks, distribution shifts at different levels pose their own challenges. At the agent level, shifts in pedestrian trajectory data may occur due to behavioral changes driven by public health events, such as the COVID-19 pandemic. During such events, pedestrians tend to maintain larger physical distances, altering typical movement patterns and interactions in public spaces. Similarly, vehicle trajectory distributions can shift due to the introduction of autonomous vehicles or vehicle-to-vehicle communications that affect the way vehicles interact with traffic and can alter driving patterns. At the link level, new road construction or traffic control strategies can change the flow, speed, or density of links, making models trained on previous data less accurate. At the regional level, gradual shifts in population demographics, land use, or public transit services can alter travel demand, leading to outdated models for OD estimation or demand forecasting.

To enable DGMs to effectively manage and adapt to data distribution shifts in evolving transportation environments, strategies can be approached in two key steps: detecting changes in data distribution and updating the model to incorporate new datasets. The first step, detecting changes in data distribution, involves identifying when and how the characteristics of the data have evolved from what was seen during the model's training phase. This can be achieved using statistical tests, such as the Kolmogorov-Smirnov test for numerical features or the Chi-square test for categorical features, which assess whether significant differences exist between the distribution of new data and the original training dataset. However, these tests are typically used for single or pairs of features, and applying these tests to high-dimensional datasets can be computationally expensive and may not effectively capture interactions between features. In high-dimensional datasets, distance metrics like Wasserstein Distance, Jensen-Shannon Divergence, Kullback-Leibler Divergence, or Cosine Similarity can be used to quantitatively measure how far apart the distributions are. Additionally, visualization techniques such as histogram analysis and dimensionality reduction methods like t-distributed Stochastic Neighbor Embedding (t-SNE) or Principal Component Analysis (PCA) can provide visual insights into changes in feature distributions. DGMs themselves can also be leveraged to detect changes in data distributions by comparing the distributions of new data to those of the training data using the DGM's data generation and density estimation capabilities.

A key challenge in this process is the availability of labeled data to validate and monitor distribution shifts, as transportation datasets often suffer from label scarcity, especially in emerging behaviors and rare anomaly detection scenarios. Unsupervised methods and self-supervised learning may provide alternative solutions.

Once a shift in data distribution has been detected, the next step is to update the model to incorporate new trends. One crucial aspect is determining the appropriate time scale for model updates by considering the analysis time horizon. Some data shifts occur over short periods (e.g., traffic disruptions from special events), while others evolve gradually over years (e.g., shifts in travel demand due to changing land use or autonomous vehicle adoption). To address this, adaptive time-window analysis can be implemented, allowing DGMs to adjust their update frequency based on detected changes. For example, event-driven retraining enables rapid model updates following sudden changes (e.g., a major policy change), while periodic retraining (e.g., every six months) ensures models remain accurate in the face of gradual changes. Retraining the model from scratch with the new data can be computationally intensive and time-consuming, particularly for large datasets and complex models. An alternative approach is to employ incremental learning or continual learning methods, which update the model progressively as new data arrives, avoiding the need to retrain the entire dataset. Fine-tuning through transfer learning is another strategy, where a pre-trained model is adjusted to better align with the new distribution while retaining previously learned features. Fine-tuning involves updating a subset of the model parameters using a small amount of labeled data from the new distribution, providing a computationally efficient means to adapt to evolving data. However, model durability in evolving contexts remains a challenge, as continual updates can lead to catastrophic forgetting of prior knowledge. Techniques such as replay-based learning, memory-augmented models, or meta-learning approaches may offer potential solutions to improve long-term model stability.

5.3. Deep Generative Models for Enhancing Transportation Data Privacy

Synthetic data generation is one of the core applications of DGMs, as discussed in Section 3.1.1. The use of synthetic data generated by DGMs offers several significant advantages for privacy protection in transportation, highlighting the potential of synthetic data to balance the need for data utility with stringent privacy requirements. One of the primary benefits of synthetic data generated by DGMs is its ability to preserve the statistical or distributional properties and patterns of real-world transportation and mobility data while excluding personally identifiable information. This means that the generated synthetic data can reflect the trends, behaviors, and characteristics found in actual data without risking individual privacy. For instance, in trajectory data containing personal origin-destination pairs and commuting patterns (Choi et al., 2018), synthetic data can accurately represent peak travel times, popular routes, and average journey durations without including any specific details about individual travelers. This capability ensures that the generated synthetic data is useful for analysis, model training, and validation. Researchers and practitioners can derive meaningful insights for transportation policy and develop effective models for traffic management without compromising the quality and accuracy of the data. For example, urban planners can use synthetic data to simulate the impact of new infrastructure projects on traffic flow, while data scientists can train machine learning models to predict congestion or optimize public transport schedules. Furthermore, synthetic data enables safe data sharing and collaboration between different organizations, stakeholders, and researchers. For instance, transportation agencies who collected privacy-sensitive trajectory data, activity sequence, and time use data, can share synthetic datasets generated by DGMs with third-party developers, policymakers, and academic institutions without exposing sensitive information. This can foster innovation, accelerate research, and support the development of new solutions and services, all while maintaining privacy.

5.4. Addressing Data Bias for Equitable Data Generation

The performance of DGMs heavily depends on the quality and representativeness of the training data. Ideally, Generative AIs can generate realistic and unbiased data if the training data itself is unbiased. However, in practice, training data often contains inherent biases, whether related to socioeconomic factors, geographic disparities, or historical inequalities. For example, if a generative model for urban planning is trained on data predominantly from affluent neighborhoods, the synthetic data it produces might overemphasize characteristics specific to those areas, neglecting the needs and patterns of lower-income regions. This could result in urban planning decisions that disproportionately benefit wealthier communities, thereby perpetuating or even amplifying existing disparities. As a result, ensuring that generative models do not amplify these biases is crucial for producing fair and representative synthetic data. This requires a multifaceted approach, beginning with the careful selection and preprocessing of training data. Data preprocessing steps might include normalizing data distributions, balancing underrepresented groups, and removing explicit bias indicators. For instance, transportation data, it might involve ensuring that data from various demographics and geographic regions are equally represented.

5.5. Towards Trustworthy AI

Beyond implementing technical measures, securing public and stakeholder acceptance and building trust in synthetic data presents significant challenges. As the use of synthetic data becomes more prevalent in AI systems, ensuring that these systems embody **trustworthy AI** principles is essential. Concerns may arise about the reliability and credibility of policies and decisions based on synthetic data. To promote trustworthy AI, it is crucial to maintain transparency in the documentation of data generation processes, open-sourcing code for reproducibility, and explaining the methodologies and safeguards used making them accessible to non-technical stakeholders. Additionally, demonstrating the validity of synthetic data through rigorous testing and comparison with real-world outcomes reinforces trustworthiness. For example, when using synthetic traffic data for new policy design, models should be able to replicate real traffic patterns under various conditions to demonstrate their adaptability and reliability. Fairness in synthetic data generation is equally important, as DGMs must be designed to avoid perpetuating biases inherent in training data. Employing fairness-aware algorithms and conducting regular audits of the synthetic data can help identify and mitigate biases, ensuring ethical and equitable outcomes. Moreover, involving stakeholders—such as community representatives and subject matter experts—in the development and implementation process further enhances credibility and acceptance. Their insights can help identify potential biases in data and modeling, as well as practical implications to ensure that synthetic data applications align with societal needs and ethical standards, thereby fostering trustworthy AI.

5.6. Towards Causal Generative Models

Causality refers to the relationship between cause and effect, where one event (the cause) directly influences another event (the effect). Assuming that event $\mathbf{x}$ is the direct cause of event $\mathbf{y}$, the causal relationship $\mathbf{x} \rightarrow \mathbf{y}$ signifies that altering X can lead to a change in $\mathbf{y}$, whereas the reverse cannot hold true (Pearl, 1995, 2009). The concept is fundamental in understanding how and why certain phenomena occur, especially in complex systems such as transportation networks. Different from traditional methods that rely on statistical inference, which usually calculates the conditional probability $p(\mathbf{y}|\mathbf{x})$ and cannot consider confounding bias, causal models focus on uncovering the true cause-and-effect relationships. In other words, these models can isolate the impact of specific variables and provide a deeper understanding of how changes in $\mathbf{x}$ directly lead to changes in $\mathbf{y}$ (Wagner, 1999; Pearl and Bareinboim, 2014; Bagi et al., 2023). In transportation research, causality has been actively considered in many areas such as traffic flow (Queen and Albers, 2009; Li et al., 2015; Du et al., 2023), urban planning (Huang et al., 2022a; Akbari et al., 2023), and traffic safety (Davis, 2000; Elvik, 2011; Mannering et al., 2020) to understand the impact of various interventions on outcomes. Studies in these areas evaluate the effects of variables that contribute to issues like traffic bottlenecks, vulnerabilities, and severe accidents. These applications demonstrate the value of causal inference in uncovering the true triggers of transportation phenomena, leading to more effective interventions and policy decisions.

In transportation applications, integrating causality into DGMs can significantly enhance their effectiveness. Conventional generative models primarily rely on training data, which limits their effectiveness in new or changing environments. In contrast, causal generative models leverage underlying cause-effect relationships, enhancing their ability to generalize across different settings and maintain robustness in Out-of-Distribution (OOD) scenarios. By learning causality, these models can adapt to covariate shifts—where input data distributions vary between training and deployment—with fewer samples, reusing many of their components without the need for retraining. For example, a causal generative model trained to understand the relationship between traffic density and travel time in one city can adapt to another city with different traffic patterns. Additionally, causal generative models maintain reliable predictions under OOD conditions such as major social events, natural disasters, or unexpected road closures by understanding and leveraging causal structures. This dual capability of improved generalization and OOD robustness makes causal generative models particularly valuable for developing scalable and effective transportation solutions in diverse and dynamically changing environments, as discussed in Section 5.2. Moreover, considering causality can improve the interpretability of DGMs. This capability not only helps stakeholders understand and trust the model's outputs but also allows them to control the generative process to produce targeted data. Traditional deep learning models are often perceived as black boxes, providing limited insights into the reasons behind certain outputs. Conversely, causal generative models focus on causal mechanisms, which offer clearer explanations for transportation phenomena. For example, in the context of traffic safety, a causal generative model can not only predict the likelihood of accidents at a particular intersection but also explain contributing factors to output such as poor road design, inadequate road signs, or high pedestrian activity. This interpretability is essential for developing trustworthy AI, as discussed in Section 5.5 Since the model is interpretable based on these causal factors, it also allows for the control of the generative process by directly manipulating these variables. This kind of controllable generative model, grounded in causality, can be used to generate data

representing specific scenarios, enabling transportation planners to simulate and evaluate the effects of interventions. This combination of interpretability and controllability makes causal generative models highly valuable in practical transportation applications.

6. Conclusion

This review paper offers a comprehensive review of state-of-the-art DGMs, which are transforming how to understand the underlying patterns of increasingly large amount of data. This paper also offers a review of state-of-art research in various transportation-related topics using DGMs, examining the current landscape and practices in transportation research with DGMs. Furthermore, this paper includes an open-sourced tutorial, offering practical guidance and resources for those looking to explore and implement DGMs in transportation research and beyond. Finally, this paper offers discussions on potential challenges and opportunities related to utilizing, implementing, and developing DGMs in the transportation domain. The authors hope that this paper contributes to the advancement of the transportation academic field and promotes further research in DGMs.

CRediT authorship contribution statement

Seongjin Choi: Conceptualization of this study, Data Curation, Investigation, Methodology, Software, Validation, Formal analysis, Supervision, Project administration, Writing - original draft, Writing - review and editing. **Zhixiong Jin:** Conceptualization of this study, Investigation, Methodology, Software, Validation, Formal analysis, Visualization, Writing - original draft, Writing - review and editing. **Seung Woo Ham:** Software, Data Curation, Visualization, Writing - original draft, Writing - review and editing. **Jiwon Kim:** Conceptualization of this study, Supervision, Writing - review and editing. **Lijun Sun:** Conceptualization of this study, Supervision, Writing - review and editing.

Declaration of generative AI and AI-assisted technologies in the writing process

During the preparation of this work, the authors used ChatGPT and Grammarly for language assistance. After utilizing these tools, the author(s) thoroughly reviewed and revised the content as necessary and take full responsibility for the final version of the published article.

Data and Code Availability Statement

All data and code associated with this work are available at

- https://github.com/UMN-Choi-Lab/DGMinTransportation.

Acknowledgment

S. Choi was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (RS-2025-00520858). L. Sun was supported by the Natural Sciences and Engineering Research Council (NSERC) of Canada Discovery Grant (c).

References

Agarwal, S., Sikchi, H., Gulino, C., Wilkinson, E., Gautam, S., 2020. Imitative planning using conditional normalizing flow. arXiv preprint arXiv:2007.16162 .

Aibin, M., Chung, N., Gordon, T., Lyford, L., Vinchoff, C., 2021. On short-and long-term traffic prediction in optical networks using machine learning, in: 2021 International Conference on Optical Network Design and Modeling (ONDM), IEEE. pp. 1–6.

Akbari, K., Winter, S., Tomko, M., 2023. Spatial causality: A systematic review on spatial causal inference. Geographical Analysis 55, 56–89.

Anstine, D.M., Isayev, O., 2023. Generative models as an emerging paradigm in the chemical sciences. Journal of the American Chemical Society 145, 8736–8750.

Arjovsky, M., Chintala, S., Bottou, L., 2017. Wasserstein generative adversarial networks, in: International conference on machine learning, PMLR. pp. 214–223.

Arnelid, H., Zec, E.L., Mohammadiha, N., 2019. Recurrent conditional generative adversarial networks for autonomous driving sensor modelling, in: 2019 IEEE Intelligent transportation systems conference (ITSC), IEEE. pp. 1613–1618.

Badu-Marfo, G., Farooq, B., Patterson, Z., 2020. A differentially private multi-output deep generative networks approach for activity diary synthesis. arXiv preprint arXiv:2012.14574 .

Badu-Marfo, G., Farooq, B., Patterson, Z., 2022. Composite travel generative adversarial networks for tabular and sequential population synthesis. IEEE Transactions on Intelligent Transportation Systems 23, 17976–17985.

Bagi, S.S.G., Gharaee, Z., Schulte, O., Crowley, M., 2023. Generative causal representation learning for out-of-distribution motion forecasting, in: International Conference on Machine Learning, PMLR. pp. 31596–31612.

Bhattacharyya, A., Hanselmann, M., Fritz, M., Schiele, B., Straehle, C.N., 2019. Conditional flow variational autoencoders for structured sequence prediction. arXiv preprint arXiv:1908.09008 .

Bhattacharyya, R., Wulfe, B., Phillips, D.J., Kuefler, A., Morton, J., Senanayake, R., Kochenderfer, M.J., 2022. Modeling human driving behavior through generative adversarial imitation learning. IEEE Transactions on Intelligent Transportation Systems 24, 2874–2887.

Bond-Taylor, S., Leach, A., Long, Y., Willcocks, C.G., 2021. Deep generative modelling: A comparative review of vaes, gans, normalizing flows, energy-based and autoregressive models. IEEE transactions on pattern analysis and machine intelligence 44, 7327–7347.

Boquet, G., Morell, A., Serrano, J., Vicario, J.L., 2020. A variational autoencoder solution for road traffic forecasting systems: Missing data imputation, dimension reduction, model selection and anomaly detection. Transportation Research Part C: Emerging Technologies 115, 102622.

Boquet, G., Vicario, J.L., Morell, A., Serrano, J., 2019. Missing data in traffic estimation: A variational autoencoder imputation method, in: ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE. pp. 2882–2886.

Borysov, S.S., Rich, J., Pereira, F.C., 2018. Scalable population synthesis with deep generative modeling. arXiv preprint arXiv:1808.06910 .

Borysov, S.S., Rich, J., Pereira, F.C., 2019. How to generate micro-agents? a deep generative modeling approach to population synthesis. Transportation Research Part C: Emerging Technologies 106, 73–97.

Bouchacourt, D., Mudigonda, P.K., Nowozin, S., 2016. Disco nets: Dissimilarity coefficients networks. Advances in Neural Information Processing Systems 29.

Cai, L., Sha, C., He, J., Yao, S., 2023. Spatial–temporal data imputation model of traffic passenger flow based on grid division. ISPRS International Journal of Geo-Information 12, 13.

Cai, Q., Abdel-Aty, M., Yuan, J., Lee, J., Wu, Y., 2020. Real-time crash prediction on expressways using deep generative models. Transportation research part C: emerging technologies 117, 102697.

Chan, R.K.C., Lim, J.M.Y., Parthiban, R., 2023. Missing traffic data imputation for artificial intelligence in intelligent transportation systems: Review of methods, limitations, and challenges. IEEE Access 11, 34080–34093.

Chen, C., Kwon, J., Rice, J., Skabardonis, A., Varaiya, P., 2003. Detecting errors and imputing missing data for single-loop surveillance systems. Transportation Research Record 1855, 160–167.

Chen, H., Ji, T., Liu, S., Driggs-Campbell, K., 2022a. Combining model-based controllers and generative adversarial imitation learning for traffic simulation, in: 2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC), IEEE. pp. 1698–1704.

Chen, J., Janke, T., Steinke, F., Lerch, S., 2024a. Generative machine learning methods for multivariate ensemble postprocessing. The Annals of Applied Statistics 18, 159–183.

Chen, J., Pu, Z., Zheng, N., Wen, X., Ding, H., Guo, X., 2024b. A novel generative adversarial network for improving crash severity modeling with imbalanced data. Transportation Research Part C: Emerging Technologies 164, 104642.

Chen, J., Zhang, S., Chen, X., Jiang, Q., Huang, H., Gu, C., 2021a. Learning traffic as videos: a spatio-temporal vae approach for traffic data imputation, in: International Conference on Artificial Neural Networks, Springer. pp. 615–627.

Chen, J., Zhang, S., Zhang, W., Chen, J., Gu, C., Huang, H., 2022b. Mtsvae: A traffic data imputation model considering different periodic temporal and global spatial features, in: 2022 International Joint Conference on Neural Networks (IJCNN), IEEE. pp. 1–8.

Chen, K., Chen, X., Yu, Z., Zhu, M., Yang, H., 2023. Equidiff: A conditional equivariant diffusion model for trajectory prediction, in: 2023 IEEE 26th International Conference on Intelligent Transportation Systems (ITSC), IEEE. pp. 746–751.

Chen, L., Zhou, Q., Cai, Y., Wang, H., Li, Y., 2022c. Cae-gan: A hybrid model for vehicle trajectory prediction. IET Intelligent Transport Systems 16, 1682–1696.

Chen, M.Y., Chiang, H.S., Huang, W.K., 2022d. Efficient generative adversarial networks for imbalanced traffic collision datasets. IEEE Transactions on Intelligent Transportation Systems 23, 19864–19873.

Chen, X., Xu, J., Zhou, R., Chen, W., Fang, J., Liu, C., 2021b. Trajvae: A variational autoencoder model for trajectory generation. Neurocomputing 428, 332–339.

Chen, Y., Lv, Y., Wang, F.Y., 2019. Traffic flow imputation using parallel data and generative adversarial networks. IEEE Transactions on Intelligent Transportation Systems 21, 1624–1630.

Chen, Y., Lv, Y., Wang, X., Wang, F.Y., 2018. Traffic flow prediction with parallel data, in: 2018 21st International Conference on Intelligent Transportation Systems (ITSC), IEEE. pp. 614–619.

Chen, Y., Lv, Y., Zhu, F., 2021c. Traffic flow synthesis using generative adversarial networks via semantic latent codes manipulation, in: 2021 IEEE International Intelligent Transportation Systems Conference (ITSC), IEEE. pp. 1451–1456.

Cheng, H., Yang, W., Sester, M., Rosenhahn, B., 2020. Context conditional variational autoencoder for predicting multi-path trajectories in mixed traffic. arXiv preprint arXiv:2002.05966 .

Choi, S., Kim, J., Yeo, H., 2021a. Trajgail: Generating urban vehicle trajectories using generative adversarial imitation learning. Transportation Research Part C: Emerging Technologies 128, 103091.

Choi, S., Kweon, N., Yang, C., Kim, D., Shon, H., Choi, J., Huh, K., 2021b. Dsa-gan: Driving style attention generative adversarial network for vehicle trajectory prediction, in: 2021 IEEE International Intelligent Transportation Systems Conference (ITSC), IEEE. pp. 1515–1520.

Choi, S., Yeo, H., Kim, J., 2018. Network-wide vehicle trajectory prediction in urban traffic networks using deep learning. Transportation Research Record 2672, 173–184.

Davis, G.A., 2000. Accident reduction factors and causal inference in traffic safety studies: a review. Accident Analysis & Prevention 32, 95–109.

De, S., Bermudez-Edo, M., Xu, H., Cai, Z., 2022. Deep generative models in the industrial internet of things: a survey. IEEE Transactions on Industrial Informatics 18, 5728–5737.

De Bortoli, V., Galashov, A., Guntupalli, J.S., Zhou, G., Murphy, K., Gretton, A., Doucet, A., 2025. Distributional diffusion models with scoring

rules. arXiv preprint arXiv:2502.02483 .
De Miguel, M.Á., Armingol, J.M., García, F., 2022. Vehicles trajectory prediction using recurrent vae network. IEEE Access 10, 32742–32749.
Demetriou, A., Allsvåg, H., Rahrovani, S., Chehreghani, M.H., 2020. Generation of driving scenario trajectories with generative adversarial networks, in: 2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC), IEEE. pp. 1–6.
Dendorfer, P., Elflein, S., Leal-Taixé, L., 2021. Mg-gan: A multi-generator model preventing out-of-distribution samples in pedestrian trajectory prediction, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 13158–13167.
Devadhas Sujakumari, P., Dassan, P., 2023. Generative adversarial networks (gan) and hdfs-based realtime traffic forecasting system using cctv surveillance. Symmetry 15, 779.
Dewi, C., Chen, R.C., Liu, Y.T., Tai, S.K., 2022. Synthetic data generation using dcgan for improved traffic sign recognition. Neural Computing and Applications 34, 21465–21480.
Ding, W., Wang, W., Zhao, D., 2019. A multi-vehicle trajectories generator to simulate vehicle-to-vehicle encountering scenarios, in: 2019 International Conference on Robotics and Automation (ICRA), IEEE. pp. 4255–4261.
Dinh, L., Krueger, D., Bengio, Y., 2014. Nice: Non-linear independent components estimation. arXiv preprint arXiv:1410.8516 .
Dinh, L., Sohl-Dickstein, J., Bengio, S., 2016. Density estimation using real nvp. arXiv preprint arXiv:1605.08803 .
Dong, J., Chen, S., Labi, S., 2023. Transfusor: Transformer diffusor for controllable human-like generation of vehicle lane changing trajectories. arXiv preprint arXiv:2308.14943 .
Du, W., Chen, S., Li, Z., Cao, X., Lv, Y., 2023. A spatial-temporal approach for multi-airport traffic flow prediction through causality graphs. IEEE Transactions on Intelligent Transportation Systems .
Duan, Y., Wang, C., Wang, C., Tang, J., Chen, Q., 2024. Traffic volume imputation using attention-based spatiotemporal generative adversarial imputation network. Transportation Safety and Environment , tdae008.
Durkan, C., Bekasov, A., Murray, I., Papamakarios, G., 2019. Neural spline flows. Advances in neural information processing systems 32.
Eiffert, S., Li, K., Shan, M., Worrall, S., Sukkarieh, S., Nebot, E., 2020. Probabilistic crowd gan: Multimodal pedestrian trajectory prediction using a graph vehicle-pedestrian attention network. IEEE Robotics and Automation Letters 5, 5026–5033.
Elvik, R., 2011. Assessing causality in multivariate accident models. Accident Analysis & Prevention 43, 253–264.
Feng, F., Zhang, J., Liu, C., Li, W., Jiang, Q., 2021. Short-term railway passenger demand forecast using improved wasserstein generative adversarial nets and web search terms. IET Intelligent Transport Systems 15, 432–445.
Feng, S., Miao, C., Zhang, Z., Zhao, P., 2024. Latent diffusion transformer for probabilistic time series forecasting, in: Proceedings of the AAAI Conference on Artificial Intelligence, pp. 11979–11987.
Feng, X., Cen, Z., Hu, J., Zhang, Y., 2019. Vehicle trajectory prediction using intention-based conditional variational autoencoder, in: 2019 IEEE Intelligent Transportation Systems Conference (ITSC), IEEE. pp. 3514–3519.
Fokker, A.D., 1914. Die mittlere energie rotierender elektrischer dipole im strahlungsfeld. Annalen der Physik 348, 810–820.
Garrido, S., Borysov, S.S., Pereira, F.C., Rich, J., 2020. Prediction of rare feature combinations in population synthesis: Application of deep generative modelling. Transportation Research Part C: Emerging Technologies 120, 102787.
Ghosh, A., Bhattacharya, B., Chowdhury, S.B.R., 2016. Sad-gan: Synthetic autonomous driving using generative adversarial networks. arXiv preprint arXiv:1611.08788 .
Gloudemans, D., Wang, Y., Ji, J., Zachar, G., Barbour, W., Hall, E., Cebelak, M., Smith, L., Work, D.B., 2023. I-24 motion: An instrument for freeway traffic science. Transportation Research Part C: Emerging Technologies 155, 104311.
Gneiting, T., Raftery, A.E., 2007. Strictly proper scoring rules, prediction, and estimation. Journal of the American statistical Association 102, 359–378.
Gómez-Huélamo, C., Conde, M.V., Ortiz, M., Montiel, S., Barea, R., Bergasa, L.M., 2022. Exploring attention gan for vehicle motion prediction, in: 2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC), IEEE. pp. 4011–4016.
Gong, L., Liu, Z., Hu, Y., Qu, T., Chen, H., Gong, X., 2023. Interactive generation of dynamically feasible vehicle trajectories using dual-vae. IFAC-PapersOnLine 56, 2214–2219.
Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y., 2014. Generative adversarial nets. Advances in neural information processing systems 27.
Grenander, U., Miller, M.I., 1994. Representations of knowledge in complex systems. Journal of the Royal Statistical Society: Series B (Methodological) 56, 549–581.
Grover, A., Dhar, M., Ermon, S., 2018. Flow-gan: Combining maximum likelihood and adversarial learning in generative models, in: Proceedings of the AAAI Conference on Artificial Intelligence. URL: https://doi.org/10.1609/aaai.v32i1.11829, doi:10.1609/aaai.v32i1.11829.
Gui, N., Zeng, T., Hu, J., Zhang, Y., 2022. Visual-angle attention predictor: A multi-agent trajectory predictor based on variational auto-encoder, in: CICTP 2022, pp. 866–877.
Guo, H., Meng, Q., Zhao, X., Liu, J., Cao, D., Chen, H., 2023. Map-enhanced generative adversarial trajectory prediction method for automated vehicles. Information Sciences 622, 1033–1049.
Guo, X., Zhao, L., 2022. A systematic survey on deep generative models for graph generation. IEEE Transactions on Pattern Analysis and Machine Intelligence 45, 5370–5390.
Gupta, A., Johnson, J., Fei-Fei, L., Savarese, S., Alahi, A., 2018. Social gan: Socially acceptable trajectories with generative adversarial networks, in: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 2255–2264.
Han, L., Zheng, K., Zhao, L., Wang, X., Wen, H., 2020. Content-aware traffic data completion in its based on generative adversarial nets. IEEE Transactions on Vehicular Technology 69, 11950–11962.
Harshvardhan, G., Gourisaria, M.K., Pandey, M., Rautaray, S.S., 2020. A comprehensive survey and analysis of generative models in machine learning. Computer Science Review 38, 100285.
Hasan, M.K., Alam, M.A., Roy, S., Dutta, A., Jawad, M.T., Das, S., 2021. Missing value imputation affects the performance of machine learning:

A review and analysis of the literature (2010–2021). Informatics in Medicine Unlocked 27, 100799.
Hegde, C., Dash, S., Agarwal, P., 2020. Vehicle trajectory prediction using gan, in: 2020 Fourth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC), IEEE. pp. 502–507.
Hinton, G.E., Salakhutdinov, R.R., 2006. Reducing the dimensionality of data with neural networks. science 313, 504–507.
Ho, J., Ermon, S., 2016. Generative adversarial imitation learning. Advances in neural information processing systems 29.
Ho, J., Jain, A., Abbeel, P., 2020. Denoising diffusion probabilistic models. Advances in neural information processing systems 33, 6840–6851.
Hou, M., Tang, T., Xia, F., Sultan, I., Kaur, R., Kong, X., 2023. Missii: missing information imputation for traffic data. IEEE Transactions on Emerging Topics in Computing .
Hsu, C.C., Kang, L.W., Chen, S.Y., Wang, I.S., Hong, C.H., Chang, C.Y., 2023. Deep learning-based vehicle trajectory prediction based on generative adversarial network for autonomous driving applications. Multimedia Tools and Applications 82, 10763–10780.
Huang, J., Chen, S., Xu, Q., Chen, Y., Hu, J., 2022a. Relationship between built environment characteristics of tod and subway ridership: A causal inference and regression analysis of the beijing subway. Journal of Rail Transport Planning & Management 24, 100341.
Huang, T., Chakraborty, P., Sharma, A., 2023. Deep convolutional generative adversarial networks for traffic data imputation encoding time series as images. International Journal of Transportation Science and Technology 12, 1–18.
Huang, Y., Chen, F., 2022. Data interpolation of traffic flow algorithm using wavelet transform for traffic generative modeling. IEEE Journal of Radio Frequency Identification 6, 739–742.
Huang, Z., Zhang, W., Wang, D., Yin, Y., 2022b. A gan framework-based dynamic multi-graph convolutional network for origin–destination-based ride-hailing demand prediction. Information Sciences 601, 129–146.
Huang, Z., Zhang, Z., Vaidya, A., Chen, Y., Lv, C., Fisac, J.F., 2024. Versatile scene-consistent traffic scenario generation as optimization with diffusion. arXiv preprint arXiv:2404.02524 .
Huo, G., Zhang, Y., Wang, B., Hu, Y., Yin, B., 2021. Text-to-traffic generative adversarial network for traffic situation generation. IEEE Transactions on Intelligent Transportation Systems 23, 2623–2636.
Huynh, V., Phung, D., 2021. Optimal transport for deep generative models: State of the art and research challenges, in: International Joint Conference on Artificial Intelligence 2021, Association for the Advancement of Artificial Intelligence (AAAI). pp. 4450–4457.
Hyvärinen, A., Dayan, P., 2005. Estimation of non-normalized statistical models by score matching. Journal of Machine Learning Research 6.
Impedovo, D., Dentamaro, V., Pirlo, G., Sarcinella, L., 2019. Trafficwave: Generative deep learning architecture for vehicular traffic flow prediction. Applied Sciences 9, 5504.
Islam, Z., Abdel-Aty, M., Cai, Q., Yuan, J., 2021. Crash data augmentation using variational autoencoder. Accident Analysis & Prevention 151, 105950.
Jagadish, D., Chauhan, A., Mahto, L., 2021. Autonomous vehicle path prediction using conditional variational autoencoder networks, in: Pacific-Asia Conference on Knowledge Discovery and Data Mining, Springer. pp. 129–139.
Jagadish, D., Chauhan, A., Mahto, L., 2022. Conditional variational autoencoder networks for autonomous vehicle path prediction. Neural Processing Letters 54, 3965–3978.
Jang, E., Gu, S., Poole, B., 2016. Categorical reparameterization with gumbel-softmax. arXiv preprint arXiv:1611.01144 .
Jeong, S., Kang, Y., Lee, J., Sohn, K., 2021. Variational embedding of a hidden markov model to generate human activity sequences. Transportation research part C: emerging technologies 131, 103347.
Jiang, C.M., Bai, Y., Cornman, A., Davis, C., Huang, X., Jeon, H., Kulshrestha, S., Lambert, J., Li, S., Zhou, X., et al., 2024. Scenediffuser: Efficient and controllable driving simulation initialization and rollout. arXiv preprint arXiv:2412.12129 .
Jin, J., Rong, D., Zhang, T., Ji, Q., Guo, H., Lv, Y., Ma, X., Wang, F.Y., 2022. A gan-based short-term link traffic prediction approach for urban road networks under a parallel learning framework. IEEE Transactions on Intelligent Transportation Systems 23, 16185–16196.
Jin, Y., Shen, X., Peng, H., Liu, X., Qin, J., Li, J., Xie, J., Gao, P., Zhou, G., Gong, J., 2023. Surrealdriver: Designing generative driver agent simulation framework in urban contexts based on large language model. arXiv preprint arXiv:2309.13193 .
Johnsen, M., Brandt, O., Garrido, S., Pereira, F., 2022. Population synthesis for urban resident modeling using deep generative models. Neural Computing and Applications , 1–16.
Jutras-Dubé, P., Al-Khasawneh, M.B., Yang, Z., Bas, J., Bastin, F., Cirillo, C., 2024. Copula-based transferable models for synthetic population generation. Transportation Research Part C: Emerging Technologies 169, 104830.
Kadanoff, L.P., 2000. Statistical physics: statics, dynamics and renormalization. World Scientific.
Kakkavas, G., Kalntis, M., Karyotis, V., Papavassiliou, S., 2021. Future network traffic matrix synthesis and estimation based on deep generative models, in: 2021 International Conference on Computer Communications and Networks (ICCCN), IEEE. pp. 1–8.
Kazemi, A., Meidani, H., 2021. Igani: Iterative generative adversarial networks for imputation with application to traffic data. IEEE Access 9, 112966–112977.
Khaled, A., Elsir, A.M.T., Shen, Y., 2022. Tfgan: Traffic forecasting using generative adversarial network with multi-graph convolutional network. Knowledge-Based Systems 249, 108990.
Kim, D., Shon, H., Kweon, N., Choi, S., Yang, C., Huh, K., 2021. Driving style-based conditional variational autoencoder for prediction of ego vehicle trajectory. IEEE Access 9, 169348–169356.
Kim, E.J., Bansal, P., 2023. A deep generative model for feasible and diverse population synthesis. Transportation Research Part C: Emerging Technologies 148, 104053.
Kim, E.J., Kim, D.K., Sohn, K., 2022. Imputing qualitative attributes for trip chains extracted from smart card data using a conditional generative adversarial network. Transportation Research Part C: Emerging Technologies 137, 103616.
Kingma, D.P., Dhariwal, P., 2018. Glow: Generative flow with invertible 1x1 convolutions. Advances in neural information processing systems 31.
Kingma, D.P., Welling, M., 2013. Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114 .
Kosaraju, V., Sadeghian, A., Martín-Martín, R., Reid, I., Rezatofighi, H., Savarese, S., 2019. Social-bigat: Multimodal trajectory forecasting using bicycle-gan and graph attention networks. Advances in neural information processing systems 32.

Krajewski, R., Moers, T., Meister, A., Eckstein, L., 2019. Béziervae: Improved trajectory modeling using variational autoencoders for the safety validation of highly automated vehicles, in: 2019 IEEE Intelligent Transportation Systems Conference (ITSC), IEEE. pp. 3788–3795.

Krajewski, R., Moers, T., Nerger, D., Eckstein, L., 2018. Data-driven maneuver modeling using generative adversarial networks and variational autoencoders for safety validation of highly automated vehicles, in: 2018 21st International Conference on Intelligent Transportation Systems (ITSC), IEEE. pp. 2383–2390.

Kuefler, A., Morton, J., Wheeler, T., Kochenderfer, M., 2017. Imitating driver behavior with generative adversarial networks, in: 2017 IEEE Intelligent Vehicles Symposium (IV), IEEE. pp. 204–211.

Kumar, R.S., Krishnamurthy, V., Podile, V., Priyanka, G.Y., Neha, V., 2023. Generative adversarial networks to improve the nature of training in autonomous vehicles, in: 2023 International Conference on Disruptive Technologies (ICDT), IEEE. pp. 161–164.

Lee, H., Bansal, P., Vo, K.D., Kim, E.J., 2025. Collaborative generative adversarial networks for fusing household travel survey and smart card data to generate heterogeneous activity schedules in urban digital twins. Available at SSRN 4993529 .

Li, C., Feng, G., Li, Y., Liu, R., Miao, Q., Chang, L., 2024a. Difftad: Denoising diffusion probabilistic models for vehicle trajectory anomaly detection. Knowledge-Based Systems 286, 111387.

Li, C., Zheng, L., Jia, N., 2024b. Network-wide ride-sourcing passenger demand origin-destination matrix prediction with a generative adversarial network. Transportmetrica A: Transport Science 20, 2109774.

Li, H., Cao, Q., Bai, Q., Li, Z., Hu, H., 2023a. Multistate time series imputation using generative adversarial network with applications to traffic data. Neural Computing and Applications 35, 6545–6567.

Li, J., Li, H., Cui, G., Kang, Y., Hu, Y., Zhou, Y., 2020a. Gacnet: A generative adversarial capsule network for regional epitaxial traffic flow prediction. Computers, Materials & Continua 64.

Li, J., Li, R., Huang, Z., Wu, P., Xu, L., 2023b. Dynamic adaptive generative adversarial networks with multi-view temporal factorizations for hybrid recovery of missing traffic data. Neural Computing and Applications 35, 7677–7696.

Li, J., Li, R., Xu, L., Liu, J., 2024c. Self-supervised generative adversarial learning with conditional cyclical constraints towards missing traffic data imputation. Knowledge-Based Systems 284, 111233.

Li, J., Ma, H., Tomizuka, M., 2019a. Conditional generative neural system for probabilistic trajectory prediction, in: 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE. pp. 6150–6156.

Li, J., Xiao, Y., Wu, J., Chen, Y., Liu, J., 2022a. Attentive dual-head spatial-temporal generative adversarial networks for crowd flow generation, in: 2022 IEEE 33rd Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC), IEEE. pp. 800–806.

Li, L., Bi, J., Yang, K., Luo, F., 2022b. Spatial-temporal semantic generative adversarial networks for flexible multi-step urban flow prediction, in: International Conference on Artificial Neural Networks, Springer. pp. 763–775.

Li, L., Bi, J., Yang, K., Luo, F., Yang, L., 2022c. Mgc-gan: Multi-graph convolutional generative adversarial networks for accurate citywide traffic flow prediction, in: 2022 IEEE International Conference on Systems, Man, and Cybernetics (SMC), IEEE. pp. 2557–2562.

Li, L., Su, X., Wang, Y., Lin, Y., Li, Z., Li, Y., 2015. Robust causal dependence mining in big data network and its application to traffic flow predictions. Transportation Research Part C: Emerging Technologies 58, 292–307.

Li, W., Wang, M., Li, L., Wang, K., Hu, Y., 2024d. Game-generative adversarial imitation learning for pedestrian simulation during pedestrian-vehicle interaction. IEEE Transactions on Intelligent Vehicles , 1–12doi:10.1109/TIV.2024.3420943.

Li, X., Cong, G., Cheng, Y., 2020b. Spatial transition learning on road networks with deep probabilistic models, in: 2020 IEEE 36th International Conference on Data Engineering (ICDE), IEEE. pp. 349–360.

Li, X., Cong, G., Sun, A., Cheng, Y., 2019b. Learning travel time distributions with deep generative model, in: The World Wide Web Conference, pp. 1017–1027.

Li, X., Rosman, G., Gilitschenski, I., Vasile, C.I., DeCastro, J.A., Karaman, S., Rus, D., 2021. Vehicle trajectory prediction using generative adversarial network with temporal logic syntax tree features. IEEE Robotics and Automation Letters 6, 3459–3466.

Li, Z., Liang, H., Wang, H., Zheng, X., Wang, J., Zhou, P., 2023c. A multi-modal vehicle trajectory prediction framework via conditional diffusion model: A coarse-to-fine approach. Knowledge-Based Systems 280, 110990.

Li, Z., Zheng, H., Feng, X., 2018. 3d convolutional generative adversarial networks for missing traffic data completion, in: 2018 10th International Conference on Wireless Communications and Signal Processing (WCSP), IEEE. pp. 1–6.

Liang, Y., Cui, Z., Tian, Y., Chen, H., Wang, Y., 2018. A deep generative adversarial architecture for network-wide spatial-temporal traffic-state estimation. Transportation Research Record 2672, 87–105.

Liao, X., Jiang, Q., He, B.Y., Liu, Y., Kuai, C., Ma, J., 2024. Deep activity model: A generative approach for human mobility pattern synthesis. arXiv preprint arXiv:2405.17468 .

Lin, H., Liu, Y., Li, S., Qu, X., 2023a. How generative adversarial networks promote the development of intelligent transportation systems: A survey. IEEE/CAA Journal of Automatica Sinica .

Lin, L., Shi, D., Han, A., Gao, J., 2024. Specstg: A fast spectral diffusion framework for probabilistic spatio-temporal traffic forecasting. arXiv preprint arXiv:2401.08119 .

Lin, W.C., Tsai, C.F., 2020. Missing value imputation: a review and analysis of the literature (2006–2017). Artificial Intelligence Review 53, 1487–1509.

Lin, Y., Dai, X., Li, L., Wang, F.Y., 2018. Pattern sensitive prediction of traffic flow based on generative adversarial framework. IEEE Transactions on Intelligent Transportation Systems 20, 2395–2400.

Lin, Y., Wan, H., Hu, J., Guo, S., Yang, B., Lin, Y., Jensen, C.S., 2023b. Origin-destination travel time oracle for map-based services. Proceedings of the ACM on Management of Data 1, 1–27.

Liu, M., Huang, H., Feng, H., Sun, L., Du, B., Fu, Y., 2023. Pristi: A conditional diffusion framework for spatiotemporal imputation, in: 2023 IEEE 39th International Conference on Data Engineering (ICDE), IEEE. pp. 1927–1939.

Liu, Y., Wang, R., Cao, Y., Sheng, Q.Z., Yao, L., 2024. Multi-agent traffic prediction via denoised endpoint distribution. arXiv preprint arXiv:2405.07041 .

Lopez, R., Gayoso, A., Yosef, N., 2020. Enhancing scientific discoveries in molecular biology with deep generative models. Molecular systems biology 16, e9198.

Ma, L., Qu, S., 2023. Application of conditional generative adversarial network to multi-step car-following modeling. Frontiers in Neurorobotics 17, 1148892.

Ma, L., Qu, S., Song, L., Zhang, Z., Ren, J., 2023. A physics-informed generative car-following model for connected autonomous vehicles. Entropy 25, 1050.

Maddison, C.J., Mnih, A., Teh, Y.W., 2016. The concrete distribution: A continuous relaxation of discrete random variables. arXiv preprint arXiv:1611.00712 .

Makhzani, A., Shlens, J., Jaitly, N., Goodfellow, I., Frey, B., 2015. Adversarial autoencoders. arXiv preprint arXiv:1511.05644 .

Mannering, F., Bhat, C.R., Shankar, V., Abdel-Aty, M., 2020. Big data, traditional data and the tradeoffs between prediction and causality in highway-safety analysis. Analytic methods in accident research 25, 100113.

Mao, X., Li, Q., Xie, H., Lau, R.Y., Wang, Z., Paul Smolley, S., 2017. Least squares generative adversarial networks, in: Proceedings of the IEEE international conference on computer vision, pp. 2794–2802.

Mensah, D.O., Badu-Marfo, G., Farooq, B., 2022. Robustness analysis of deep learning models for population synthesis. arXiv preprint arXiv:2211.13339 .

Mirza, M., Osindero, S., 2014. Conditional generative adversarial nets. arXiv preprint arXiv:1411.1784 .

Mo, Z., Fu, Y., Di, X., 2022a. Quantifying uncertainty in traffic state estimation using generative adversarial networks, in: 2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC), IEEE. pp. 2769–2774.

Mo, Z., Fu, Y., Xu, D., Di, X., 2022b. Trafficflowgan: Physics-informed flow based generative adversarial network for uncertainty quantification, in: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer. pp. 323–339.

Naji, H.A., Xue, Q., Zhu, H., Li, T., 2021. Forecasting taxi demands using generative adversarial networks with multi-source data. Applied Sciences 11, 9675.

Neumeier, M., Botsch, M., Tollkühn, A., Berberich, T., 2021. Variational autoencoder-based vehicle trajectory prediction with an interpretable latent space, in: 2021 IEEE International Intelligent Transportation Systems Conference (ITSC), IEEE. pp. 820–827.

Ni, D., Leonard, J.D., 2005. Markov chain monte carlo multiple imputation using bayesian networks for incomplete intelligent transportation systems data. Transportation research record 1935, 57–67.

Niedoba, M., Lavington, J., Liu, Y., Lioutas, V., Sefas, J., Liang, X., Green, D., Dabiri, S., Zwartsenberg, B., Scibior, A., et al., 2024. A diffusion-model of joint interactive navigation. Advances in Neural Information Processing Systems 36.

Oh, G., Peng, H., 2022. Cvae-h: Conditionalizing variational autoencoders via hypernetworks and trajectory forecasting for autonomous driving. arXiv preprint arXiv:2201.09874 .

Van den Oord, A., Kalchbrenner, N., Espeholt, L., Vinyals, O., Graves, A., et al., 2016. Conditional image generation with pixelcnn decoders. Advances in neural information processing systems 29.

Ozturk, A., Gunel, M.B., Dal, M., Yavas, U., Ure, N.K., 2020. Development of a stochastic traffic environment with generative time-series models for improving generalization capabilities of autonomous driving agents, in: 2020 IEEE Intelligent Vehicles Symposium (IV), IEEE. pp. 1343–1348.

Pacchiardi, L., Adewoyin, R.A., Dueben, P., Dutta, R., 2024. Probabilistic forecasting with generative networks via scoring rule minimization. Journal of Machine Learning Research 25, 1–64.

Parisi, G., 1981. Correlation functions and computer simulations. Nuclear Physics B 180, 378–384.

Pearl, J., 1995. Causal diagrams for empirical research. Biometrika 82, 669–688.

Pearl, J., 2009. Causality. Cambridge university press.

Pearl, J., Bareinboim, E., 2014. External validity: From do-calculus to transportability across populations. Statistical Science 29, 579–595. URL: https://doi.org/10.1214/14-STS486, doi:10.1214/14-STS486.

Peng, W., Lin, Y., Guo, S., Tang, W., Liu, L., Wan, H., 2023. Generative-contrastive-attentive spatial-temporal network for traffic data imputation, in: Pacific-Asia Conference on Knowledge Discovery and Data Mining, Springer. pp. 45–56.

Planck, V., 1917. Über einen satz der statistischen dynamik und seine erweiterung in der quantentheorie. Sitzungberichte der .

Qin, H., Zhan, X., Li, Y., Yang, X., Zheng, Y., 2021. Network-wide traffic states imputation using self-interested coalitional learning, in: Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining, pp. 1370–1378.

Queen, C.M., Albers, C.J., 2009. Intervention and causality: forecasting traffic flows using a dynamic bayesian network. Journal of the American Statistical Association 104, 669–681.

Rákos, O., Aradi, S., Bécsi, T., Szalay, Z., 2020. Compression of vehicle trajectories with a variational autoencoder. Applied Sciences 10, 6739.

Rakos, O., Becsi, T., Aradi, S., 2021. Adversarial autoencoder for trajectory generation and maneuver classification, in: 2021 IEEE 25th International Conference on Intelligent Engineering Systems (INES), IEEE. pp. 000013–000018.

Rákos, O., Bécsi, T., Aradi, S., Gáspár, P., 2021. Learning latent representation of freeway traffic situations from occupancy grid pictures using variational autoencoder. Energies 14, 5232.

Rasul, K., Seward, C., Schuster, I., Vollgraf, R., 2021. Autoregressive denoising diffusion models for multivariate probabilistic time series forecasting, in: International Conference on Machine Learning, PMLR. pp. 8857–8868.

Rasul, K., Sheikh, A.S., Schuster, I., Bergmann, U., Vollgraf, R., 2020. Multivariate probabilistic time series forecasting via conditioned normalizing flows. arXiv preprint arXiv:2002.06103 .

Rempe, D., Luo, Z., Bin Peng, X., Yuan, Y., Kitani, K., Kreis, K., Fidler, S., Litany, O., 2023. Trace and pace: Controllable pedestrian animation via guided trajectory diffusion, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 13756–13766.

Rezende, D., Mohamed, S., 2015. Variational inference with normalizing flows, in: International conference on machine learning, PMLR. pp. 1530–1538.

Rong, C., Ding, J., Liu, Z., Li, Y., 2023a. Complexity-aware large scale origin-destination network generation via diffusion model. arXiv preprint arXiv:2306.04873 .

Rong, C., Wang, H., Li, Y., 2023b. Origin-destination network generation via gravity-guided gan. arXiv preprint arXiv:2306.03390 .

Rossi, L., Ajmar, A., Paolanti, M., Pierdicca, R., 2021a. Vehicle trajectory prediction and generation using lstm models and gans. Plos one 16, e0253868.

Rossi, L., Paolanti, M., Pierdicca, R., Frontoni, E., 2021b. Human trajectory prediction and generation using lstm models and gans. Pattern Recognition 120, 108136.

Roy, D., Ishizaka, T., Mohan, C.K., Fukuda, A., 2019. Vehicle trajectory prediction at intersections using interaction based generative adversarial networks, in: 2019 IEEE Intelligent transportation systems conference (ITSC), IEEE. pp. 2318–2323.

Salzmann, T., Ivanovic, B., Chakravarty, P., Pavone, M., 2020. Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data, in: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVIII 16, Springer. pp. 683–700.

Santhosh, K.K., Dogra, D.P., Roy, P.P., Mitra, A., 2021. Vehicular trajectory classification and traffic anomaly detection in videos using a hybrid cnn-vae architecture. IEEE Transactions on Intelligent Transportation Systems 23, 11891–11902.

Saxena, D., Cao, J., 2019. D-gan: Deep generative adversarial nets for spatio-temporal prediction. arXiv preprint arXiv:1907.08556 .

Shahbazi, M., Danelljan, M., Paudel, D.P., Van Gool, L., 2022. Collapse by conditioning: Training class-conditional gans with limited data. arXiv preprint arXiv:2201.06578 .

Shao, F., Shao, H., Wang, D., Lam, W.H., Tam, M.L., 2024. A generative adversarial network-based framework for network-wide travel time reliability prediction. Knowledge-Based Systems 283, 111184.

Shen, G., Liu, N., Liu, Y., Zhou, W., Kong, X., 2022. Traffic flow imputation based on multi-perspective spatiotemporal generative adversarial networks, in: Proceedings of CECNet 2022. IOS Press, pp. 62–73.

Shen, X., Meinshausen, N., 2024. Engression: extrapolation through the lens of distributional regression. Journal of the Royal Statistical Society Series B: Statistical Methodology , qkae108.

Shi, H., Dong, S., Wu, Y., Nie, Q., Zhou, Y., Ran, B., 2024. Generative adversarial network for car following trajectory generation and anomaly detection. Journal of Intelligent Transportation Systems , 1–14.

Shin, D.H., Lee, S.E., Jeon, B.U., Chung, K., 2023. Missing value imputation model based on adversarial autoencoder using spatiotemporal feature extraction. Intelligent Automation & Soft Computing 37.

Singh, A.K., Shrestha, J., Albarella, N., 2023. Bi-level optimization augmented with conditional variational autoencoder for autonomous driving in dense traffic, in: 2023 IEEE 19th International Conference on Automation Science and Engineering (CASE), IEEE. pp. 1–8.

Sohn, K., Lee, H., Yan, X., 2015. Learning structured output representation using deep conditional generative models. Advances in neural information processing systems 28.

Song, X., Zhang, C., James, J., 2021. Learn travel time distribution with graph deep learning and generative adversarial network, in: 2021 IEEE International Intelligent Transportation Systems Conference (ITSC), IEEE. pp. 1385–1390.

Song, Y., Ermon, S., 2019. Generative modeling by estimating gradients of the data distribution. Advances in neural information processing systems 32.

Song, Y., Garg, S., Shi, J., Ermon, S., 2020a. Sliced score matching: A scalable approach to density and score estimation, in: Uncertainty in Artificial Intelligence, PMLR. pp. 574–584.

Song, Y., Sohl-Dickstein, J., Kingma, D.P., Kumar, A., Ermon, S., Poole, B., 2020b. Score-based generative modeling through stochastic differential equations. arXiv preprint arXiv:2011.13456 .

Stopher, P.R., Greaves, S.P., 2007. Household travel surveys: Where are we going? Transportation Research Part A: Policy and Practice 41, 367–381.

Sun, J., Kim, J., 2023. Toward data-driven simulation of network-wide traffic: A multi-agent imitation learning approach using urban vehicle trajectory data. IEEE Transactions on Intelligent Transportation Systems .

Sun, J., Wang, Z., Li, J., Lu, C., 2021a. Unified and fast human trajectory prediction via conditionally parameterized normalizing flow. IEEE Robotics and Automation Letters 7, 842–849.

Sun, L., Erath, A., Cai, M., 2018. A hierarchical mixture modeling framework for population synthesis. Transportation Research Part B: Methodological 114, 199–212.

Sun, S., Gu, Z., Sun, T., Sun, J., Yuan, C., Han, Y., Li, D., Ang Jr, M.H., 2023. Drivescenegen: Generating diverse and realistic driving scenarios from scratch. arXiv preprint arXiv:2309.14685 .

Sun, T., Sun, B., Jiang, Z., Hao, R., Xie, J., 2021b. Traffic flow online prediction based on a generative adversarial network with multi-source data. Sustainability 13, 12188.

Tang, Y., He, H., Wang, Y., Wu, Y., 2024. Utilizing a diffusion model for pedestrian trajectory prediction in semi-open autonomous driving environments. IEEE Sensors Journal .

Treiber, M., Helbing, D., 2003. An adaptive smoothing method for traffic state identification from incomplete information, in: Interface and transport dynamics: Computational Modelling. Springer, pp. 343–360.

Tu, J., Mei, G., Piccialli, F., 2021. Incomplete vehicle information completion using generative adversarial network to enhance the safety of autonomous driving, in: Second International Conference on Industrial IoT, Big Data, and Supply Chain, SPIE. pp. 61–66.

Uria, B., Côté, M.A., Gregor, K., Murray, I., Larochelle, H., 2016. Neural autoregressive distribution estimation. Journal of Machine Learning Research 17, 1–37.

Van Den Oord, A., Kalchbrenner, N., Kavukcuoglu, K., 2016. Pixel recurrent neural networks, in: International conference on machine learning, PMLR. pp. 1747–1756.

Vincent, P., 2011. A connection between score matching and denoising autoencoders. Neural computation 23, 1661–1674.

Vishnu, C., Abhinav, V., Roy, D., Mohan, C.K., Babu, C.S., 2023. Improving multi-agent trajectory prediction using traffic states on interactive driving scenarios. IEEE Robotics and Automation Letters 8, 2708–2715.

Wagner, A., 1999. Causality in complex systems. Biology and Philosophy 14, 83–101.

Wang, E., Cui, H., Yalamanchi, S., Moorthy, M., Djuric, N., 2020a. Improving movement predictions of traffic actors in bird's-eye view models using gans and differentiable trajectory rasterization, in: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 2340–2348.

Wang, J., Wang, W., Liu, X., Yu, W., Li, X., Sun, P., 2022a. Traffic prediction based on auto spatiotemporal multi-graph adversarial neural network. Physica A: Statistical Mechanics and its Applications 590, 126736.

Wang, J., Zhang, Y., Xing, X., Zhan, Y., Chan, W.K.V., Tiwari, S., 2022b. A data-driven system for cooperative-bus route planning based on generative adversarial network and metric learning. Annals of Operations Research , 1–27.

Wang, N., Kankanhalli, M., 2024. Dptraj-pm: Differentially private trajectory synthesis using prefix tree and markov process. arXiv preprint arXiv:2404.14106 .

Wang, N., Zhang, K., Zheng, L., Lee, J., Li, S., 2023. Network-wide traffic state reconstruction: An integrated generative adversarial network framework with structural deep network embedding. Chaos, Solitons & Fractals 174, 113830.

Wang, S., Cao, J., Chen, H., Peng, H., Huang, Z., 2020b. Seqst-gan: Seq2seq generative adversarial nets for multi-step urban crowd flow prediction. ACM Transactions on Spatial Algorithms and Systems (TSAS) 6, 1–24.

Wang, S., He, Z., 2021. A prediction model of vessel trajectory based on generative adversarial network. The Journal of Navigation 74, 1161–1171.

Wang, Y., Zhao, S., Zhang, R., Cheng, X., Yang, L., 2020c. Multi-vehicle collaborative learning for trajectory prediction with spatio-temporal tensor fusion. IEEE Transactions on Intelligent Transportation Systems 23, 236–248.

Wei, T., Lin, Y., Guo, S., Lin, Y., Huang, Y., Xiang, C., Bai, Y., Ya, M., Wan, H., 2024. Diff-rntraj: A structure-aware diffusion model for road network-constrained trajectory generation. arXiv preprint arXiv:2402.07369 .

Wen, H., Lin, Y., Xia, Y., Wan, H., Wen, Q., Zimmermann, R., Liang, Y., 2023. Diffstg: Probabilistic spatio-temporal graph forecasting with denoising diffusion models, in: Proceedings of the 31st ACM International Conference on Advances in Geographic Information Systems, pp. 1–12.

Westny, T., Olofsson, B., Frisk, E., 2024. Diffusion-based environment-aware trajectory prediction. arXiv preprint arXiv:2403.11643 .

Wu, C., Chen, L., Wang, G., Chai, S., Jiang, H., Peng, J., Hong, Z., 2020. Spatiotemporal scenario generation of traffic flow based on lstm-gan. IEEE Access 8, 186191–186198.

Wu, L., Wang, H., Gao, T., Li, B., Kong, F., 2022a. A traffic video completion model based on generative adversarial networks, in: Proceedings of 2021 Chinese Intelligent Automation Conference, Springer. pp. 658–669.

Wu, Q., He, K., Chen, X., Yu, S., Zhang, J., 2021. Deep transfer learning across cities for mobile traffic prediction. IEEE/ACM Transactions on Networking 30, 1255–1267.

Wu, X., Yang, H., Chen, H., Hu, Q., Hu, H., 2022b. Long-term 4d trajectory prediction using generative adversarial networks. Transportation Research Part C: Emerging Technologies 136, 103554.

Xiao, Z., Kreis, K., Vahdat, A., 2021. Tackling the generative learning trilemma with denoising diffusion gans. arXiv preprint arXiv:2112.07804 .

Xie, L., Guo, T., Chang, J., Wan, C., Hu, X., Yang, Y., Ou, C., 2023. A novel model for ship trajectory anomaly detection based on gaussian mixture variational autoencoder. IEEE Transactions on Vehicular Technology .

Xing, H., Hu, J., Zhang, Z., 2022. Multi-modal vehicle trajectory prediction via attention-based conditional variational autoencoder, in: 2022 37th Youth Academic Annual Conference of Chinese Association of Automation (YAC), IEEE. pp. 1367–1373.

Xiong, G., Li, Z., Zhao, M., Zhang, Y., Miao, Q., Lv, Y., Wang, F.Y., 2023. Trajsgan: A semantic-guiding adversarial network for urban trajectory generation. IEEE Transactions on Computational Social Systems .

Xu, C., Zhao, D., Sangiovanni-Vincentelli, A., Li, B., 2023a. Diffscene: Diffusion-based safety-critical scenario generation for autonomous vehicles, in: The Second Workshop on New Frontiers in Adversarial Machine Learning. URL: https://openreview.net/forum?id=hclEbdHida.

Xu, D., Lin, Z., Zhou, L., Li, H., Niu, B., 2022. A gats-gan framework for road traffic states forecasting. Transportmetrica B: transport dynamics 10, 718–730.

Xu, D., Peng, H., Wei, C., Shang, X., Li, H., 2021. Traffic state data imputation: An efficient generating method based on the graph aggregator. IEEE Transactions on Intelligent Transportation Systems 23, 13084–13093.

Xu, D., Peng, P., Wei, C., He, D., Xuan, Q., 2020a. Road traffic network state prediction based on a generative adversarial network. IET Intelligent Transport Systems 14, 1286–1294.

Xu, D., Wei, C., Peng, P., Xuan, Q., Guo, H., 2020b. Ge-gan: A novel deep learning framework for road traffic state estimation. Transportation Research Part C: Emerging Technologies 117, 102635.

Xu, D.W., Dong, H.H., Li, H.J., Jia, L.M., Feng, Y.J., 2015. The estimation of road traffic states based on compressive sensing. Transportmetrica B: Transport Dynamics 3, 131–152.

Xu, M., Niyato, D., Chen, J., Zhang, H., Kang, J., Xiong, Z., Mao, S., Han, Z., 2023b. Generative ai-empowered simulation for autonomous driving in vehicular mixed reality metaverses. IEEE Journal of Selected Topics in Signal Processing .

Yan, H., Li, Y., 2023. A survey of generative ai for intelligent transportation systems. arXiv preprint arXiv:2312.08248 .

Yang, B., Kang, Y., Yuan, Y., Huang, X., Li, H., 2021. St-lbagan: Spatio-temporal learnable bidirectional attention generative adversarial networks for missing traffic data imputation. Knowledge-Based Systems 215, 106705.

Yang, B., Kang, Y., Yuan, Y., Li, H., Wang, F., 2022. St-fvgan: filling series traffic missing values with generative adversarial network. Transportation Letters 14, 407–415.

Yang, C., Tian, A.X., Chen, D., Shi, T., Heydarian, A., 2024. Wcdt: World-centric diffusion transformer for traffic scene generation. arXiv preprint arXiv:2404.02082 .

Yao, R., Bekhor, S., 2022. A variational autoencoder approach for choice set generation and implicit perception of alternatives in choice modeling. Transportation Research Part B: Methodological 158, 273–294.

Yao, Y., Liu, Y., Dai, X., Chen, S., Lv, Y., 2023. A graph-based scene encoder for vehicle trajectory prediction using the diffusion model, in: 2023 International Annual Conference on Complex Systems and Intelligent Science (CSIS-IAC), IEEE. pp. 981–986.

Yu, B., Lee, Y., Sohn, K., 2020a. Forecasting road traffic speeds by considering area-wide spatio-temporal dependencies based on a graph convo-

lutional neural network (gcn). Transportation research part C: emerging technologies 114, 189–204.
Yu, H., Chen, X., Li, Z., Zhang, G., Liu, P., Yang, J., Yang, Y., 2019. Taxi-based mobility demand formulation and prediction using conditional generative adversarial network-driven learning approaches. IEEE Transactions on Intelligent Transportation Systems 20, 3888–3899.
Yu, H., Li, Z., Zhang, G., Liu, P., Wang, J., 2020b. Extracting and predicting taxi hotspots in spatiotemporal dimensions using conditional generative adversarial neural networks. IEEE Transactions on Vehicular Technology 69, 3680–3692.
Yu, J.J.Q., Gu, J., 2019. Real-time traffic speed estimation with graph convolutional generative autoencoder. IEEE Transactions on Intelligent Transportation Systems 20, 3940–3951.
Yu, Z., Zhao, J., Jiang, R., Shen, J., Wu, D., Zheng, S., 2024. Theory-data dual driven car following model in traffic flow mixed of avs and hdvs. Transportation Research Part C: Emerging Technologies 165, 104747.
Yuan, X., Qiao, Y., Zhao, P., Hu, R., Zhang, B., 2023. Traffic matrix estimation based on denoising diffusion probabilistic model, in: 2023 IEEE Symposium on Computers and Communications (ISCC), IEEE. pp. 316–322.
Yuan, Y., Zhang, Y., Wang, B., Peng, Y., Hu, Y., Yin, B., 2022. Stgan: Spatio-temporal generative adversarial network for traffic data imputation. IEEE Transactions on Big Data 9, 200–211.
Yun, Y., Jeong, D., Lim, S., 2019. Data-driven human-like cut-in driving model using generative adversarial network. Electronics Letters 55, 1288–1290.
Zang, D., Fang, Y., Wei, Z., Tang, K., Cheng, J., 2019. Traffic flow data prediction using residual deconvolution based deep generative network. IEEE Access 7, 71311–71322.
Zarei, M., Hellinga, B., 2021. Crash data augmentation using conditional generative adversarial networks (cgan) for improving safety performance functions. arXiv preprint arXiv:2112.12263 .
Zarei, M., Hellinga, B., Izadpanah, P., 2024. Application of conditional deep generative networks (cgan) in empirical bayes estimation of road crash risk and identifying crash hotspots. International journal of transportation science and technology 13, 258–269.
Zhang, B., Miao, R., Chen, Z., 2024a. Spatial-temporal traffic data imputation based on dynamic multi-level generative adversarial networks for urban governance. Applied Soft Computing 151, 111128.
Zhang, J., Li, H., Zhang, S., Yang, L., Jin, G., Qi, J., 2023a. A spatiotemporal graph generative adversarial networks for short-term passenger flow prediction in urban rail transit systems. International Journal of General Systems 52, 694–721.
Zhang, K., Feng, X., Jia, N., Zhao, L., He, Z., 2022a. Tsr-gan: Generative adversarial networks for traffic state reconstruction with time space diagrams. Physica A: Statistical Mechanics and its Applications 591, 126788.
Zhang, L., Wu, J., Shen, J., Chen, M., Wang, R., Zhou, X., Xu, C., Yao, Q., Wu, Q., 2021a. Satp-gan: Self-attention based generative adversarial network for traffic flow prediction. Transportmetrica B: Transport Dynamics 9, 552–568.
Zhang, L., Zhao, L., Pfoser, D., 2022b. Factorized deep generative models for end-to-end trajectory generation with spatiotemporal validity constraints, in: Proceedings of the 30th International Conference on Advances in Geographic Information Systems, pp. 1–12.
Zhang, M., Li, Y., 2022. Generational travel patterns in the united states: New insights from eight national travel surveys. Transportation research part A: policy and practice 156, 1–13.
Zhang, Q., Chen, Y., 2021. Diffusion normalizing flow. Advances in neural information processing systems 34, 16280–16291.
Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O., 2018. The unreasonable effectiveness of deep features as a perceptual metric, in: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 586–595.
Zhang, S., Hu, X., Chen, J., Zhang, W., Huang, H., 2024b. An effective variational auto-encoder-based model for traffic flow imputation. Neural Computing and Applications 36, 2617–2631.
Zhang, S., Wang, S., Tan, X., Liu, R., Zhang, J., Wang, J., 2023b. sasdim: self-adaptive noise scaling diffusion model for spatial time series imputation. arXiv preprint arXiv:2309.01988 .
Zhang, T., Wang, J., Liu, J., 2021b. A gated generative adversarial imputation approach for signalized road networks. IEEE Transactions on Intelligent Transportation Systems 23, 12144–12160.
Zhang, W., Zhang, P., Yu, Y., Li, X., Biancardo, S.A., Zhang, J., 2021c. Missing data repairs for traffic flow with self-attention generative adversarial imputation net. IEEE Transactions on Intelligent Transportation Systems 23, 7919–7930.
Zhang, X., Gao, Y., Wang, X., Feng, J., Shi, Y., 2022c. Geosdva: A semi-supervised dirichlet variational autoencoder model for transportation mode identification. ISPRS International Journal of Geo-Information 11, 290.
Zhang, X., Li, Y., Zhou, X., Luo, J., 2020a. cgail: Conditional generative adversarial imitation learning—an application in taxi drivers' strategy learning. IEEE transactions on big data 8, 1288–1300.
Zhang, Y., Chen, X., Wang, J., Zheng, Z., Wu, K., 2022d. A generative car-following model conditioned on driving styles. Transportation research part C: emerging technologies 145, 103926.
Zhang, Y., Li, Y., Zhou, X., Kong, X., Luo, J., 2020b. Curb-gan: Conditional urban traffic estimation through spatio-temporal generative adversarial networks, in: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 842–852.
Zhang, Y., Li, Y., Zhou, X., Kong, X., Luo, J., 2020c. Off-deployment traffic estimation—a traffic generative adversarial networks approach. IEEE transactions on big data 8, 1084–1095.
Zhang, Y., Wang, S., Chen, B., Cao, J., 2019a. Gcgan: Generative adversarial nets with graph cnn for network-scale traffic prediction, in: 2019 International Joint Conference on Neural Networks (IJCNN), IEEE. pp. 1–8.
Zhang, Y., Wang, S., Chen, B., Cao, J., Huang, Z., 2019b. Trafficgan: Network-scale deep traffic prediction with generative adversarial nets. IEEE Transactions on Intelligent Transportation Systems 22, 219–230.
Zhao, H., Luo, R., Yao, B., Wang, Y., Hu, S., Su, R., 2022. Graphsage-based generative adversarial network for short-term traffic speed prediction problem, in: 2022 17th International Conference on Control, Automation, Robotics and Vision (ICARCV), IEEE. pp. 837–842.
Zhao, L., Liu, Y., Al-Dubai, A.Y., Zomaya, A.Y., Min, G., Hawbani, A., 2020. A novel generation-adversarial-network-based vehicle trajectory prediction method for intelligent vehicular networks. IEEE Internet of Things Journal 8, 2066–2077.
Zhao, T., Xu, Y., Monfort, M., Choi, W., Baker, C., Zhao, Y., Wang, Y., Wu, Y.N., 2019. Multi-agent tensor fusion for contextual trajectory

prediction, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 12126–12134.
Zheng, Z., Wang, Z., Hu, Z., Wan, Z., Ma, W., 2024. Recovering traffic data from the corrupted noise: A doubly physics-regularized denoising diffusion model. Transportation Research Part C: Emerging Technologies 160, 104513.
Zhong, Z., Luo, Y., Liang, W., 2022. Stgm: Vehicle trajectory prediction based on generative model for spatial-temporal features. IEEE Transactions on Intelligent Transportation Systems 23, 18785–18793.
Zhong, Z., Rempe, D., Xu, D., Chen, Y., Veer, S., Che, T., Ray, B., Pavone, M., 2023. Guided conditional diffusion for controllable traffic simulation, in: 2023 IEEE International Conference on Robotics and Automation (ICRA), IEEE. pp. 3560–3566.
Zhou, D., Wang, H., Li, W., Zhou, Y., Cheng, N., Lu, N., 2021. Sa-sgan: A vehicle trajectory prediction model based on generative adversarial networks, in: 2021 IEEE 94th Vehicular Technology Conference (VTC2021-Fall), IEEE. pp. 1–5.
Zhou, F., Yang, Q., Zhong, T., Chen, D., Zhang, N., 2020a. Variational graph neural networks for road traffic prediction in intelligent transportation systems. IEEE Transactions on Industrial Informatics 17, 2802–2812.
Zhou, Y., Fu, R., Wang, C., Zhang, R., 2020b. Modeling car-following behaviors and driving styles with generative adversarial imitation learning. Sensors 20, 5034.
Zhou, Z., Ding, J., Liu, Y., Jin, D., Li, Y., 2023. Towards generative modeling of urban flow through knowledge-enhanced denoising diffusion, in: Proceedings of the 31st ACM International Conference on Advances in Geographic Information Systems, pp. 1–12.
Zhu, K., Zhang, S., Li, J., Zhou, D., Dai, H., Hu, Z., 2022. Spatiotemporal multi-graph convolutional networks with synthetic data for traffic volume forecasting. Expert Systems with Applications 187, 115992.
Zhu, Y., Ye, Y., Zhang, S., Zhao, X., Yu, J., 2024a. Difftraj: Generating gps trajectory with diffusion probabilistic model. Advances in Neural Information Processing Systems 36.
Zhu, Y., Yu, J.J., Zhao, X., Liu, Q., Ye, Y., Chen, W., Zhang, Z., Wei, X., Liang, Y., 2024b. Controltraj: Controllable trajectory generation with topology-constrained diffusion model. arXiv preprint arXiv:2404.15380 .

A. Preliminaries

- **Bayes' Rule:** This is a fundamental theorem in probability theory that describes how to update the probability of a hypothesis based on new evidence. It combines prior knowledge (prior probability) with new evidence (likelihood) to form a posterior probability, which is a revised probability given the new information. Usually, in the context of DGMs, Bayes' Rule is frequently used to derive an approximated loss function. The equation for Bayes' Rule is:

$$p(A|B) = \frac{p(B|A)p(A)}{p(B)}, \tag{A.1}$$

  where $p(A|B)$ is the posterior probability of the hypothesis A given the evidence B, $p(B|A)$ is the likelihood of observing the evidence B given that A is true, $p(A)$ is the prior probability of the hypothesis A before observing the evidence, and $p(B)$ is the probability of observing the evidence B, also known as the marginal likelihood.

- **Data Distribution:** This refers to the way data is spread or distributed in a particular space. Generative models aim to learn this distribution so they can generate new samples that resemble the original data.

- **Likelihood:** Likelihood measures the plausibility of a model parameter value given specific observed data. In the context of generative models, it refers to how likely the observed data is under the model's assumed data distribution.

- **KL divergence (Kullback–Leibler divergence; KLD):** This is a measure of how one probability distribution is different from a second, reference probability distribution. It provides a way to quantify the difference between two probability distributions in bits. Mathematically, given two probability distributions $p(x)$ and $q(x)$ over the same random variable x, the KL divergence of q from p is defined as:

$$D_{KL}\left(p(\mathbf{x})||q(\mathbf{x})\right) = \mathbb{E}_{\mathbf{x}\sim p(\mathbf{x})}\left[\log\left(\frac{p(\mathbf{x})}{q(\mathbf{x})}\right)\right] = \int_{\mathbf{x}\in\mathcal{X}} p(\mathbf{x})\log\left(\frac{p(\mathbf{x})}{q(\mathbf{x})}\right)dx, \tag{A.2}$$

  where $p(\mathbf{x})$ and $q(\mathbf{x})$ are probability density functions (PDFs) over a continuous variable $\mathbf{x}$. The integral is taken over the entire range of $\mathbf{x}$ where the distributions are defined. It is worth noting that KL divergence is non-negative and is not symmetric, meaning $D_{KL}(p(\mathbf{x})||q(\mathbf{x})) \neq D_{KL}(q(\mathbf{x})||p(\mathbf{x}))$.

- **Latent Space:** This is a conceptual space where the high-dimensional data is compressed into lower dimensions. The data in this latent space is usually the input to the generative part of the model that produces the output samples.

A.1. Commonly used performance metrics

In what follows, let $\{(y_i, \hat{y}_i)\}_{i=1}^n$ denote the set of true values y_i and corresponding predictions (estimation) $\hat{y}_i$. We use $\bar{y}$ to denote the sample mean of the true values and $\bar{\hat{y}}$ to denote the sample mean of the predictions.

A.1.1 Mean Absolute Error (MAE)

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}\left|y_i - \hat{y}_i\right|. \tag{A.3}$$

A.1.2 Mean Relative Error (MRE) or Mean Absolute Percentage Error (MAPE)

Sometimes referred to interchangeably (with slight differences in definition), a common version of MAPE is given by

$$\text{MAPE} = \frac{100\%}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right|. \tag{A.4}$$

For MRE, it is given by:

$$\text{MRE} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right|. \tag{A.5}$$

A.1.3 Symmetric Mean Absolute Percentage Error (SMAPE)

$$\text{SMAPE} = \frac{100\%}{n} \sum_{i=1}^{n} \frac{|y_i - \hat{y}_i|}{\left(|y_i| + |\hat{y}_i|\right)/2}, \tag{A.6}$$

A.1.4 Root Mean Squared Error (RMSE)

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2}. \tag{A.7}$$

A.1.5 Normalized Root Mean Squared Error (NRMSE)

The NRMSE is then given by normalizing the RMSE, commonly using the range of the observed data:

$$\text{NRMSE} = \frac{\text{RMSE}}{y_{\max} - y_{\min}}, \tag{A.8}$$

where $y_{\max}$ and $y_{\min}$ are the maximum and minimum true values, respectively.

A.1.6 Standardized Root Mean Squared Error (SRMSE)

A common normalization for the RMSE is to divide by a characteristic scale of the data (e.g., the mean of the true values):

$$\text{SRMSE} = \frac{\text{RMSE}}{\bar{y}} = \frac{\sqrt{\frac{1}{n}\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}}{\frac{1}{n}\sum_{i=1}^{n} y_i}. \tag{A.9}$$

A.1.7 Pearson's Correlation Coefficient

$$\rho = \frac{\sum_{i=1}^{n} \left(y_i - \bar{y}\right)\left(\hat{y}_i - \bar{\hat{y}}\right)}{\sqrt{\sum_{i=1}^{n} (y_i - \bar{y})^2 \ \sum_{i=1}^{n} (\hat{y}_i - \bar{\hat{y}})^2}}, \tag{A.10}$$

A.1.8 Continuous Ranked Probability Score (CRPS)

For a probabilistic forecast F_i (cumulative distribution function) and an observation y_i, the CRPS is defined as:

$$\text{CRPS}(F_i, y_i) = \int_{-\infty}^{\infty} \Big[F_i(t) - \mathbf{1}\{t \geq y_i\} \Big]^2 \, dt, \tag{A.11}$$

where $\mathbf{1}\{\cdot\}$ denotes the indicator function. The average CRPS over n samples is then given by:

$$\text{CRPS}_{sum} = \frac{1}{n} \sum_{i=1}^{n} \text{CRPS}(F_i, y_i). \tag{A.12}$$

A.1.9 Coefficient of Determination (R^2)

$$R^2 = 1 - \frac{\sum_{i=1}^{n} \left(y_i - \hat{y}_i\right)^2}{\sum_{i=1}^{n} \left(y_i - \bar{y}\right)^2}. \tag{A.13}$$

A.1.10 Relative Squared Error (RSE)

In a regression context, it is defined as:

$$\text{RSE} = \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}. \tag{A.14}$$

In a reconstruction context, for an original matrix or tensor X and its reconstruction $\hat{X}$, the RSE is given by:

$$\text{RSE} = \frac{\|\hat{X} - X\|_F}{\|X\|_F}, \tag{A.15}$$

where the Frobenius norm is defined as $\|A\|_F = \sqrt{\sum_{i,j} A_{ij}^2}$.

A.1.11 Common Part of Commuting (CPC)

The Common Part of Commuting (CPC) is a metric that quantifies the overlap between two flow distributions. Let $X_{i,j}$ represent the flow from location i to location j in the first distribution, and $Y_{i,j}$ the flow for the same OD pair in the second distribution. The CPC is defined as:

$$\text{CPC} = \frac{2\sum_{i,j}\min\left(X_{i,j}, Y_{i,j}\right)}{\sum_{i,j} X_{i,j} + \sum_{i,j} Y_{i,j}}. \tag{A.16}$$

A.1.12 Performance Metrics (for classification)

A.1.13 Classification Metrics

Let TP = True Positives, FP = False Positives, FN = False Negatives, and TN = True Negatives. The following metrics are defined as:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \tag{A.17}$$

$$\text{Recall (Sensitivity)} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \tag{A.18}$$

$$\text{Specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}}, \tag{A.19}$$

$$\text{False Positive Rate (FPR)} = \frac{\text{FP}}{\text{FP} + \text{TN}}, \tag{A.20}$$

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}, \tag{A.21}$$

$$\text{Balanced Accuracy (BA)} = \frac{1}{2}\left(\frac{\text{TP}}{\text{TP} + \text{FN}} + \frac{\text{TN}}{\text{TN} + \text{FP}}\right), \tag{A.22}$$

$$\text{G-mean} = \sqrt{\frac{\text{TP}}{\text{TP} + \text{FN}} \cdot \frac{\text{TN}}{\text{TN} + \text{FP}}}, \tag{A.23}$$

$$\text{F1-Score} = 2 \cdot \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}, \tag{A.24}$$

$$\text{AUC} = \text{Area Under the ROC Curve}. \tag{A.25}$$

A.1.14 mean Average Precision (mAP)

For each class, the Average Precision (AP) is calculated as the area under the Precision-Recall curve. The mAP is the average of these AP values across all classes:

$$\text{mAP} = \frac{1}{C}\sum_{c=1}^{C} \text{AP}_c, \tag{A.26}$$

where C is the total number of classes, and AP_c is the Average Precision for class c.

A.1.15 Cosine Similarity

For two vectors $\mathbf{u}$ and $\mathbf{v}$ in $\mathbb{R}^d$,

$$\text{CosineSim}(\mathbf{u}, \mathbf{v}) = \frac{\mathbf{u} \cdot \mathbf{v}}{\|\mathbf{u}\|\|\mathbf{v}\|}. \tag{A.27}$$

A.1.16 BLEU Score

The BLEU score is computed as

$$\text{BLEU Score} = BP \cdot \exp\left(\sum_{n=1}^{N} w_n \log p_n\right), \tag{A.28}$$

where p_n is the modified precision for n-grams, w_n is the weight for each n-gram (typically $\frac{1}{N}$), and the brevity penalty BP is defined as

$$BP = \begin{cases} 1, & \text{if } c > r, \\ \exp\left(1 - \frac{r}{c}\right), & \text{if } c \leq r, \end{cases} \tag{A.29}$$

with c representing the length of the candidate translation and r the effective reference length.

A.1.17 METEOR Score

The METEOR score incorporates precision, recall, and a penalty for fragmented matches. It is defined as:

$$\text{METEOR} = (1 - Penalty) \cdot F_{mean}, \tag{A.30}$$

where the harmonic mean F_{mean} is computed as:

$$F_{mean} = \frac{10PR}{9P + R}, \tag{A.31}$$

with P (precision) and R (recall) based on unigram matches, and the penalty is given by:

$$Penalty = \gamma \left(\frac{ch}{m}\right)^{\beta}. \tag{A.32}$$

Here, ch denotes the number of chunks, m is the number of mapped unigrams, and γ and β are parameters typically set by empirical tuning.

A.1.18 Jensen-Shannon Divegence (JSD)

The Jensen-Shannon distance measures the similarity between two probability distributions P and Q. It is defined as:

$$JSD(P \parallel Q) = \sqrt{\frac{1}{2} D_{KL}(P \parallel M) + \frac{1}{2} D_{KL}(Q \parallel M)}, \tag{A.33}$$

where

$$M = \frac{1}{2}(P + Q), \tag{A.34}$$

and D_{KL} represents the Kullback-Leibler divergence.

A.1.19 Wasserstein Distance (WD)

The Wasserstein Distance (WD), also known as the Earth Mover's Distance (EMD), measures the distance between two probability distributions over a space. For distributions μ and ν defined on a space $(\mathcal{X}, d)$, the p-Wasserstein Distance is given by,

$$W_p(\mu, \nu) = \left(\inf_{\gamma \in \Gamma(\mu,\nu)} \int_{\mathcal{X}\times\mathcal{X}} d(x, y)^p \, d\gamma(x, y)\right)^{\frac{1}{p}}, \tag{A.35}$$

where $d(x, y)$ is the distance between points x and y, and $\Gamma(\mu, \nu)$ is the set of all ways to match the distributions μ and ν.

A.1.20 Fréchet Inception Distance (FID)

The Fréchet Inception Distance (FID) is a metric used to assess the performance of generative models by comparing the distributions of feature representations extracted from real and generated images. Typically, these features are obtained using a pretrained Inception network. Assuming that the features are well-approximated by a multivariate Gaussian distribution, let μ_r and Σ_r denote the mean and covariance of the features for real images, and μ_g and Σ_g for generated images. The Fréchet Feature Distance (FFD) is then defined as:

$$\text{FID} = \|\mu_r - \mu_g\|_2^2 + \text{Tr}\Big(\Sigma_r + \Sigma_g - 2\Big(\Sigma_r\Sigma_g\Big)^{\frac{1}{2}}\Big), \tag{A.36}$$

where $\|\mu_r - \mu_g\|_2^2$ represents the squared Euclidean Distance between the mean feature vectors, and the trace term $\text{Tr}(\cdot)$ measures the difference between the covariance matrices Σ_r and Σ_g. Although this metric is commonly called the Fréchet Inception Distance when applied to images, it can be generalized to other domains—as the Fréchet Feature Distance—by employing an appropriate feature extractor.

A.1.21 Segment-Path Distance (SPD) and Symmetrized Segment-Path Distance (SSPD)

Let trajectory T^1 consist of n_1 points $\{p_i^1\}_{i=1}^{n_1}$. For each point p_i^1 in T^1, define $D_{\text{pt}}(p_i^1, T^2)$ as the minimal distance from p_i^1 to any points of trajectory T^2. The SPD from T^1 to T^2 is defined as:

$$D_{\text{SPD}}(T^1, T^2) = \frac{1}{n_1}\sum_{i=1}^{n_1} D_{\text{pt}}(p_i^1, T^2). \tag{A.37}$$

Since $D_{\text{SPD}}(T^1, T^2)$ may differ from $D_{\text{SPD}}(T^2, T^1)$, the Symmetrized Segment-Path Distance (SSPD) is defined as:

$$D_{\text{SSPD}}(T^1, T^2) = \frac{D_{\text{SPD}}(T^1, T^2) + D_{\text{SPD}}(T^2, T^1)}{2}. \tag{A.38}$$

A.1.22 Dynamic Time Warping (DTW)

Let $X = (x_1, x_2, \dots, x_N)$ and $Y = (y_1, y_2, \dots, y_M)$ be two sequences, and let $d(x_i, y_j)$ denote a local distance measure between elements x_i and y_j. The DTW distance is defined recursively by:

$$\text{DTW}(i, j) = d(x_i, y_j) + \min\Big\{\text{DTW}(i-1, j),\ \text{DTW}(i, j-1),\ \text{DTW}(i-1, j-1)\Big\}, \tag{A.39}$$

with the boundary conditions:

$$\begin{aligned} &\text{DTW}(0, 0) = 0, \\ &\text{DTW}(i, 0) = \infty \quad \text{for } i > 0, \\ &\text{DTW}(0, j) = \infty \quad \text{for } j > 0. \end{aligned} \tag{A.40}$$

The DTW distance between the full sequences X and Y is given by $\text{DTW}(N, M)$.

A.1.23 Maximum Final Distance (MFD)

Let N be the number of agents, and for each agent i, let the model generate K predicted trajectories. Denote the final position of the k-th predicted trajectory for agent i by $(x_T^{i,k}, y_T^{i,k})$. The indices k and l range over the set of predicted trajectories $\{1, 2, \dots, K\}$. For each agent i, the MFD is calculated by taking the maximum Euclidean distance between the final positions of any pair of trajectories indexed by k and l. Formally, the MFD over all agents is given by:

$$\text{MFD}_K = \frac{1}{N}\sum_{i=1}^{N} \max_{k,l} \sqrt{\left(x_T^{i,k} - x_T^{i,l}\right)^2 + \left(y_T^{i,k} - y_T^{i,l}\right)^2}. \tag{A.41}$$

A.1.24 Average Displacement Error (ADE) and minADE$_k$

Given the ground truth trajectory $\{\mathbf{q}_1, \dots, \mathbf{q}_T\}$ and a predicted trajectory $\{\hat{\mathbf{q}}_1, \dots, \hat{\mathbf{q}}_T\}$, the ADE is defined as:

$$\text{ADE} = \frac{1}{T}\sum_{t=1}^{T} \|\hat{\mathbf{q}}_t - \mathbf{q}_t\|_2. \tag{A.42}$$

When k different predictions $\{\hat{\mathbf{q}}_1^{(i)}, \dots, \hat{\mathbf{q}}_T^{(i)}\}$ are available, we define

$$\text{minADE}_k = \min_{i\in\{1,\dots,k\}} \frac{1}{T}\sum_{t=1}^{T} \|\hat{\mathbf{q}}_t^{(i)} - \mathbf{q}_t\|_2. \tag{A.43}$$

A.1.25 Final Displacement Error (FDE) and minFDE$_k$

For the final time step, the FDE is given by:

$$\text{FDE} = \|\hat{\mathbf{q}}_T - \mathbf{q}_T\|_2. \tag{A.44}$$

Similarly, for k predictions,

$$\text{minFDE}_k = \min_{i\in\{1,\dots,k\}} \|\hat{\mathbf{q}}_T^{(i)} - \mathbf{q}_T\|_2. \tag{A.45}$$

A.1.26 Kernel Density Estimate-based Negative Log Likelihood (KDENLL)

Let $\{\hat{\mathbf{q}}_t^{(i)}\}_{i=1}^k$ denote the k predicted positions at time step t, and let $\mathbf{q}_t$ be the corresponding ground truth position. Here, T denotes the total number of time steps in the trajectory. The Kernel Density Estimate (KDE) at $\mathbf{q}_t$ is computed as:

$$\hat{p}(\mathbf{q}_t) = \frac{1}{kh^d}\sum_{i=1}^{k} K\left(\frac{\mathbf{q}_t - \hat{\mathbf{q}}_t^{(i)}}{h}\right), \tag{A.46}$$

where $K(\cdot)$ is a kernel function (commonly a Gaussian), h is the bandwidth, and d is the dimensionality of $\mathbf{q}_t$.

The KDENLL for the trajectory is then defined as:

$$\text{KDENLL} = -\frac{1}{T}\sum_{t=1}^{T} \log \hat{p}(\mathbf{q}_t). \tag{A.47}$$

When k distinct sets of predictions are available, one can compute a KDENLL for each set. Let $\hat{p}^{(i)}(\mathbf{q}_t)$ be the KDE computed using the i-th prediction set (for $i = 1, \dots, k$). The best (lowest) KDENLL among these is then given by

$$\text{KDENLL}_k = \min_{i\in\{1,\dots,k\}} \left[-\frac{1}{T}\sum_{t=1}^{T} \log \hat{p}^{(i)}(\mathbf{q}_t)\right]. \tag{A.48}$$

A.1.27 Maximum Mean Discrepancy (MMD)

The Maximum Mean Discrepancy (MMD) is a kernel-based measure for comparing two distributions using sample data. Let $\{x_i\}_{i=1}^n$ and $\{y_j\}_{j=1}^m$ be samples from distributions P and Q, respectively, and let $k(\cdot,\cdot)$ be a positive-definite kernel (e.g., the RBF kernel). The squared MMD is given by:

$$\text{MMD}(P,Q) = \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n} k(x_i, x_j) + \frac{1}{m^2}\sum_{i=1}^{m}\sum_{j=1}^{m} k(y_i, y_j) - \frac{2}{nm}\sum_{i=1}^{n}\sum_{j=1}^{m} k(x_i, y_j). \tag{A.49}$$

B. Derivation for Evidence Lower Bound for Variational Autoencoder

The probability $p_\theta(\mathbf{x})$ with the latent vector $\mathbf{z}$ can be formulated as:

$$p(\mathbf{x}) = \int p_\theta(\mathbf{x}|\mathbf{z})p(\mathbf{z})d\mathbf{z}. \tag{B.1}$$

However, Equation (B.1) is intractable due to the integral over the $\mathbf{z}$ space. Even though Monte Carlo methods can be used to approximate the integral and apply maximum likelihood estimation, they may result in suboptimal generation due to poor-quality samples being assigned higher likelihoods. To address this issue, VAEs incorporate the encoder $q_\phi(\mathbf{z}|\mathbf{x})$ into their objective function, reformulated as:

$$\begin{aligned}
\log p(\mathbf{x}) &= \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p(\mathbf{x})\right] \\
&= \mathbb{E}_{\mathbf{z}}\left[\log \frac{p_\theta(\mathbf{x},\mathbf{z})}{p_\theta(\mathbf{z}|\mathbf{x})}\right] && \Rightarrow \textit{Bayes' Rule} \\
&= \mathbb{E}_{\mathbf{z}}\left[\log \frac{p_\theta(\mathbf{x}|\mathbf{z})p(\mathbf{z})}{p_\theta(\mathbf{z}|\mathbf{x})}\right] \\
&= \mathbb{E}_{\mathbf{z}}\left[\log \frac{p_\theta(\mathbf{x}|\mathbf{z})p(\mathbf{z})}{p_\theta(\mathbf{z}|\mathbf{x})}\frac{q_\phi(\mathbf{z}|\mathbf{x})}{q_\phi(\mathbf{z}|\mathbf{x})}\right] && \Rightarrow \textit{Multiply by } \frac{q_\phi(\mathbf{z}|\mathbf{x})}{q_\phi(\mathbf{z}|\mathbf{x})} = 1 \\
&= \mathbb{E}_{\mathbf{z}}\left[\log p_\theta(\mathbf{x}|\mathbf{z})\right] - \mathbb{E}_{\mathbf{z}}\left[\log \frac{q_\phi(\mathbf{z}|\mathbf{x})}{p(\mathbf{z})}\right] + \mathbb{E}_{\mathbf{z}}\left[\log \frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{z}|\mathbf{x})}\right] && \Rightarrow \textit{Logarithms} \\
&= \mathbb{E}_{\mathbf{z}}\left[\log p_\theta(\mathbf{x}|\mathbf{z})\right] - D_{KL}\left(q_\phi\left(\mathbf{z}|\mathbf{x}\right)||p(\mathbf{z})\right) + D_{KL}\left(q_\phi\left(\mathbf{z}|\mathbf{x}\right)||p_\theta(\mathbf{z}|\mathbf{x})\right).
\end{aligned} \tag{B.2}$$

The first term in Equation (B.2), $\mathbb{E}_z\left[\log p_\theta(\mathbf{x}|\mathbf{z})\right]$, is the likelihood of $\mathbf{x}$ generated from the decoder based on the sampled $\mathbf{z}$ from the posterior distribution. This term can be estimated through sampling using the reparameterization trick, which will be discussed later. The second term is the KL divergence between the approximate posterior distribution $q_\phi(\mathbf{z}|\mathbf{x})$ and the prior distribution $p(\mathbf{z})$. Assuming both distributions follow a tractable distribution (typically Gaussian), this term can be computed in closed form. The third term is intractable because the true posterior distribution $p_\theta(\mathbf{z}|\mathbf{x})$ involves evaluation of intractable Equation (B.2) $\left(p_\theta(\mathbf{z}|\mathbf{x}) = \frac{p_\theta(\mathbf{x}|\mathbf{z})p(\mathbf{z})}{p_\theta(\mathbf{x})}\right)$. However, the third term is always non-negative due to the properties of KL divergence.

C. Discussion on implicit likelihood maximization of GANs

Consider p_{model} as the learned model distribution imposed by the $\mathbf{z} \sim p(\mathbf{z})$ and the θ-parameterized mapping function G_θ. If G_θ is fixed, then D would converge to $D^*_\phi(\mathbf{x}) = \frac{p_{\text{model}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x})+p_{\text{model}}(\mathbf{x})}$, since

$$\begin{aligned}
&\max_\phi \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}(\mathbf{x})}[\log D_\phi(\mathbf{x})] + \mathbb{E}_{\mathbf{x}\sim p_{\text{model}}(\mathbf{x})}[\log(1 - D_\phi(\mathbf{x}))] \\
=&\max_\phi \int_{\mathbf{x}} \left(p_{\text{data}}(\mathbf{x})\log D_\phi(\mathbf{x}) + p_{\text{model}}(\mathbf{x})\log\left(1 - D_\phi(\mathbf{x})\right)\right) d\mathbf{x},
\end{aligned} \tag{C.1}$$

which is optimal when $D^*_\phi(\mathbf{x}) = \frac{p_{\text{model}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x})+p_{\text{model}}(\mathbf{x})}$.

For the optimal discriminator, the objective of the generator can be expressed as

$$
\begin{aligned}
&\min_G V(D_\phi^*, G_\theta) \\
&= \min_\theta \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}(\mathbf{x})}[\log D_\phi^*(\mathbf{x})] + \mathbb{E}_{\mathbf{x}\sim p_{\text{model}}(\mathbf{x})}[\log(1 - D_\phi^*(\mathbf{x}))] \\
&= \min_\theta \int_{\mathbf{x}} \left(p_{\text{data}}(\mathbf{x}) \log D_\phi^*(\mathbf{x}) + p_{\text{model}}(\mathbf{x}) \log \left(1 - D_\phi^*(\mathbf{x})\right)\right) d\mathbf{x} \\
&= \min_\theta \int_{\mathbf{x}} \left(p_{\text{data}}(\mathbf{x}) \log \frac{p_{\text{model}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_{\text{model}}(\mathbf{x})} + p_{\text{model}}(\mathbf{x}) \log \left(1 - \frac{p_{\text{model}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_{\text{model}}(\mathbf{x})}\right)\right) d\mathbf{x} \\
&= \min_\theta \int_{\mathbf{x}} \left(p_{\text{data}}(\mathbf{x}) \log \frac{p_{\text{model}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_{\text{model}}(\mathbf{x})} + p_{\text{model}}(\mathbf{x}) \log \frac{p_{\text{data}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_{\text{model}}(\mathbf{x})}\right) d\mathbf{x} \\
&= \min_\theta \int_{\mathbf{x}} \left(p_{\text{data}}(\mathbf{x}) \log \frac{p_{\text{model}}(\mathbf{x})}{(p_{\text{data}}(\mathbf{x}) + p_{\text{model}}(\mathbf{x}))/2} - p_{\text{data}}(\mathbf{x}) \log 2 + p_{\text{model}}(\mathbf{x}) \log \frac{p_{\text{data}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_{\text{model}}(\mathbf{x})/2} - p_{\text{model}}(\mathbf{x}) \log 2\right) d\mathbf{x} \\
&= \min_\theta D_{KL}\left(p_{\text{model}}(\mathbf{x}) || \frac{(p_{\text{data}}(\mathbf{x}) + p_{\text{model}}(\mathbf{x}))}{2}\right) + D_{KL}\left(p_{\text{data}}(\mathbf{x}) || \frac{(p_{\text{data}}(\mathbf{x}) + p_{\text{model}}(\mathbf{x}))}{2}\right) - 2\log 2 \\
&= \min_\theta 2 D_{JS}\left(p_{\text{model}}(\mathbf{x}) || p_{\text{data}}(\mathbf{x})\right) - 2\log 2,
\end{aligned}
\tag{C.2}
$$

where D_{JS} is the Jensen-Shannon divergence. The minimization is achieved when $D_{JS}\left(p_{\text{model}}(\mathbf{x}) || p_{\text{data}}(\mathbf{x})\right)$ is minimized, i.e., the model distribution p_{model} is close to the data distribution p_{data}.

The training process continues until a balance between the Generator and the Discriminator is achieved, where the Generator can produce realistic data and the Discriminator cannot differentiate between real and generated data, i.e., $D^*(\mathbf{x}) = \frac{p_{\text{model}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_{\text{model}}(\mathbf{x})} = \frac{p_{\text{data}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_{\text{data}}(\mathbf{x})} = 0.5$, for all $\mathbf{x}$.

However, in practice, achieving equilibrium is challenging due to issues like mode collapse, vanishing gradients, and non-convergence. Various modifications and extensions of the original GAN framework have been proposed to mitigate these issues, such as Wasserstein GANs (Arjovsky et al., 2017), Least-Squares GANs (Mao et al., 2017), and Conditional GANs (Shahbazi et al., 2022), each offering unique perspectives and solutions. Despite these challenges, the ability to generate highly realistic and diverse data has made GAN-based models essential tools in many domains of machine learning and data science.

D. Derivation for Diffusion Model

Similar to VAE and Normalizing Flows (models using explicit densities for training), the training objective is to maximize the likelihood, or minimize the negative log-likelihood as follows:

$$
\theta^* = \arg\min_\theta \mathbb{E}\left[-\log p_\theta\left(\mathbf{x}_0\right)\right] \tag{D.1}
$$

Similar to VAE, instead of directly training the model with negative log-likelihood, we can consider q as the approximate posterior and use the variational bound on negative log-likelihood to train the model as follows:

$$
\begin{aligned}
&\mathbb{E}\left[-\log p_\theta\left(\mathbf{x}_0\right)\right], \\
&= \mathbb{E}\left[-\log \frac{p_\theta\left(\mathbf{x}_0, \mathbf{x}_1, \cdots, \mathbf{x}_T\right)}{p_\theta\left(\mathbf{x}_1, \cdots, \mathbf{x}_T | \mathbf{x}_0\right)}\right], && \Rightarrow \text{Baye's Rule} \\
&= \mathbb{E}\left[-\log \frac{p_\theta\left(\mathbf{x}_{0:T}\right)}{p_\theta\left(\mathbf{x}_{1:T} | \mathbf{x}_0\right)} \cdot \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\right], \\
&= \mathbb{E}\left[-\log \frac{p_\theta\left(\mathbf{x}_{0:T}\right)}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \cdot \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta\left(\mathbf{x}_{1:T} | \mathbf{x}_0\right)}\right] = \mathbb{E}\left[-\log \frac{p_\theta\left(\mathbf{x}_{0:T}\right)}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\right] - \mathbb{E}\left[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta\left(\mathbf{x}_{1:T} | \mathbf{x}_0\right)}\right], \\
&\leq \mathbb{E}\left[-\log \frac{p_\theta\left(\mathbf{x}_{0:T}\right)}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\right], && \Rightarrow \text{KL divergence } \geq 0
\end{aligned}
$$

(D.2)

Using Equation (2.27) and Equation (2.28), we can further derive:

$$\begin{aligned}
&\mathbb{E}\left[-\log \frac{p_\theta\left(\mathbf{x}_{0:T}\right)}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\right], \\
&= \mathbb{E}\left[-\log \frac{p\left(\mathbf{x}_T\right)\prod_{t=1}^{T} p_\theta\left(\mathbf{x}_{t-1}|\mathbf{x}_t\right)}{\prod_{t=1}^{T} q\left(\mathbf{x}_t|\mathbf{x}_{t-1}\right)}\right], \\
&= \mathbb{E}\left[-\log p\left(\mathbf{x}_T\right) - \log \frac{\prod_{t=1}^{T} p_\theta\left(\mathbf{x}_{t-1}|\mathbf{x}_t\right)}{\prod_{t=1}^{T} q\left(\mathbf{x}_t|\mathbf{x}_{t-1}\right)}\right], \\
&= \mathbb{E}\left[-\log p\left(\mathbf{x}_T\right) - \sum_{t=1}^{T} \log \frac{p_\theta\left(\mathbf{x}_{t-1}|\mathbf{x}_t\right)}{q\left(\mathbf{x}_t|\mathbf{x}_{t-1}\right)}\right],
\end{aligned} \tag{D.3}$$

Appendix A in Ho et al. (2020) further provides a detailed derivation of the reduced variance variational bound for diffusion models as follows:

$$\begin{aligned}
&\mathbb{E}\left[-\log p\left(\mathbf{x}_T\right) - \sum_{t=1}^{T} \log \frac{p_\theta\left(\mathbf{x}_{t-1}|\mathbf{x}_t\right)}{q\left(\mathbf{x}_t|\mathbf{x}_{t-1}\right)}\right], \\
&= \mathbb{E}\left[-\log p\left(\mathbf{x}_T\right) - \sum_{t=2}^{T} \log \frac{p_\theta\left(\mathbf{x}_{t-1}|\mathbf{x}_t\right)}{q\left(\mathbf{x}_t|\mathbf{x}_{t-1}\right)} - \log \frac{p_\theta\left(\mathbf{x}_0|\mathbf{x}_1\right)}{q\left(\mathbf{x}_1|\mathbf{x}_0\right)}\right], \\
&= \mathbb{E}\left[-\log p\left(\mathbf{x}_T\right) - \sum_{t=2}^{T} \log \frac{p_\theta\left(\mathbf{x}_{t-1}|\mathbf{x}_t\right)}{q\left(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0\right)} \cdot \frac{q\left(\mathbf{x}_{t-1}|\mathbf{x}_0\right)}{q\left(\mathbf{x}_t|\mathbf{x}_0\right)} - \log \frac{p_\theta\left(\mathbf{x}_0|\mathbf{x}_1\right)}{q\left(\mathbf{x}_1|\mathbf{x}_0\right)}\right],
\end{aligned} \tag{D.4}$$

since

$$\begin{aligned}
q\left(\mathbf{x}_t|\mathbf{x}_{t-1}\right) &= q\left(\mathbf{x}_t|\mathbf{x}_{t-1},\mathbf{x}_0\right), && \Rightarrow \text{Markov chain property} \\
&= \frac{q\left(\mathbf{x}_t,\mathbf{x}_{t-1},\mathbf{x}_0\right)}{q\left(\mathbf{x}_{t-1},\mathbf{x}_0\right)} = \frac{q\left(\mathbf{x}_t,\mathbf{x}_{t-1},\mathbf{x}_0\right)}{q\left(\mathbf{x}_{t-1},\mathbf{x}_0\right)} \cdot \frac{q\left(\mathbf{x}_t,\mathbf{x}_0\right)}{q\left(\mathbf{x}_t,\mathbf{x}_0\right)}, \\
&= \frac{q\left(\mathbf{x}_t,\mathbf{x}_{t-1},\mathbf{x}_0\right)}{q\left(\mathbf{x}_t,\mathbf{x}_0\right)} \cdot \frac{q\left(\mathbf{x}_t,\mathbf{x}_0\right)}{q\left(\mathbf{x}_{t-1},\mathbf{x}_0\right)}, \\
&= q\left(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0\right) \cdot \frac{q\left(\mathbf{x}_t,\mathbf{x}_0\right)}{q\left(\mathbf{x}_{t-1},\mathbf{x}_0\right)}.
\end{aligned} \tag{D.5}$$

As a result,

$$\begin{aligned}
&\mathbb{E}\left[-\log p\left(\mathbf{x}_T\right) - \sum_{t=2}^{T} \log \frac{p_\theta\left(\mathbf{x}_{t-1}|\mathbf{x}_t\right)}{q\left(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0\right)} \cdot \frac{q\left(\mathbf{x}_{t-1}|\mathbf{x}_0\right)}{q\left(\mathbf{x}_t|\mathbf{x}_0\right)} - \log \frac{p_\theta\left(\mathbf{x}_0|\mathbf{x}_1\right)}{q\left(\mathbf{x}_1|\mathbf{x}_0\right)}\right], \\
&= \mathbb{E}\left[-\log p\left(\mathbf{x}_T\right) - \sum_{t=2}^{T} \log \frac{p_\theta\left(\mathbf{x}_{t-1}|\mathbf{x}_t\right)}{q\left(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0\right)} - \sum_{t=2}^{T} \log \frac{q\left(\mathbf{x}_{t-1}|\mathbf{x}_0\right)}{q\left(\mathbf{x}_t|\mathbf{x}_0\right)} - \log \frac{p_\theta\left(\mathbf{x}_0|\mathbf{x}_1\right)}{q\left(\mathbf{x}_1|\mathbf{x}_0\right)}\right], \\
&= \mathbb{E}\left[-\log p\left(\mathbf{x}_T\right) - \sum_{t=2}^{T} \log \frac{p_\theta\left(\mathbf{x}_{t-1}|\mathbf{x}_t\right)}{q\left(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0\right)} + \log q\left(\mathbf{x}_T|\mathbf{x}_0\right) - \log p_\theta\left(\mathbf{x}_0|\mathbf{x}_1\right)\right],
\end{aligned} \tag{D.6}$$

since

$$\begin{aligned}
&-\sum_{t=2}^{T}\log\frac{q\left(\mathbf{x}_{t-1}|\mathbf{x}_0\right)}{q\left(\mathbf{x}_t|\mathbf{x}_0\right)}-\log\frac{p_\theta\left(\mathbf{x}_0|\mathbf{x}_1\right)}{q\left(\mathbf{x}_1|\mathbf{x}_0\right)},\\
&=-\log\frac{q\left(\mathbf{x}_1|\mathbf{x}_0\right)}{q\left(\mathbf{x}_2|\mathbf{x}_0\right)}\cdot\frac{q\left(\mathbf{x}_2|\mathbf{x}_0\right)}{q\left(\mathbf{x}_3|\mathbf{x}_0\right)}\cdot\frac{q\left(\mathbf{x}_3|\mathbf{x}_0\right)}{q\left(\mathbf{x}_4|\mathbf{x}_0\right)}\cdots\frac{q\left(\mathbf{x}_{T-1}|\mathbf{x}_0\right)}{q\left(\mathbf{x}_T|\mathbf{x}_0\right)}\cdot\frac{p_\theta\left(\mathbf{x}_0|\mathbf{x}_1\right)}{q\left(\mathbf{x}_1|\mathbf{x}_0\right)},\\
&=\log q\left(\mathbf{x}_T|\mathbf{x}_0\right)-\log p_\theta\left(\mathbf{x}_0|\mathbf{x}_1\right).
\end{aligned}\tag{D.7}$$

Therefore,

$$\begin{aligned}
&\mathbb{E}\left[-\log p\left(\mathbf{x}_T\right)-\sum_{t=2}^{T}\log\frac{p_\theta\left(\mathbf{x}_{t-1}|\mathbf{x}_t\right)}{q\left(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0\right)}+\log q\left(\mathbf{x}_T|\mathbf{x}_0\right)-\log p_\theta\left(\mathbf{x}_0|\mathbf{x}_1\right)\right],\\
&=\mathbb{E}\left[\log\frac{q\left(\mathbf{x}_T|\mathbf{x}_0\right)}{p\left(\mathbf{x}_T\right)}-\sum_{t=2}^{T}\log\frac{p_\theta\left(\mathbf{x}_{t-1}|\mathbf{x}_t\right)}{q\left(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0\right)}-\log p_\theta\left(\mathbf{x}_0|\mathbf{x}_1\right)\right],\\
&=\mathbb{E}\left[\underbrace{D_{KL}\left(q\left(\mathbf{x}_T|\mathbf{x}_0\right)||p\left(\mathbf{x}_T\right)\right)}_{L_T}+\sum_{t=2}^{T}\underbrace{D_{KL}\left(q\left(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0\right)||p_\theta\left(\mathbf{x}_{t-1}|\mathbf{x}_t\right)\right)}_{L_{t-1}}\underbrace{-\log p_\theta\left(\mathbf{x}_0|\mathbf{x}_1\right)}_{L_0}\right].
\end{aligned}\tag{D.8}$$

Therefore, the overall loss function of minimizing the negative log-likelihood in Equation (D.1) is decomposed into several losses, L_T, L_{t-1}, and L_0. Here, L_T is constant since both $q\left(\mathbf{x}_T|\mathbf{x}_0\right)$ and $p\left(\mathbf{x}_T\right)$ are fixed, and therefore, we can ignore this term. Also, in Ho and Ermon (2016), L_0 is explicitly defined by using the characteristics of the image generation problem, and as a result, L_0 can be interpreted as a reconstruction loss of a problem-specific decoder. As a result, the actual learning process of the diffusion model is related to L_{t-1}.

L_{t-1} measures the KL-divergence of $q\left(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0\right)$ from $p_\theta\left(\mathbf{x}_{t-1}|\mathbf{x}_t\right)$. The diffusion process, q, represents the process of adding small noise to the data; i.e., given a less noisy data $\mathbf{x}_{t-1}$, the distribution of a more noisy data $\mathbf{x}_t$. The first term, $q\left(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0\right)$, represents the true denoising process which is derived from the definition of q given the true data without noise, $\mathbf{x}_0$. What the diffusion models try to learn is the denoising process $p_\theta\left(\mathbf{x}_{t-1}|\mathbf{x}_t\right)$; i.e., given a more noisy data $\mathbf{x}_t$, the distribution of a less noisy data $\mathbf{x}_{t-1}$. As a result, L_{t-1} captures the distributional difference between the true denoising process (given the true data) and the approximated denoising process (without the true data).

Since the diffusion process follows Gaussian distribution, the true reverse process, $q\left(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0\right)$, can be assumed to follow a Gaussian distribution if T is sufficiently large, or $T\to\infty$. Let $q\left(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0\right)=\mathcal{N}(\mathbf{x}_{t-1};\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t,\mathbf{x}_0),\tilde{\beta}_t\mathbb{I})$. To derive explicit form of $\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t,\mathbf{x}_0)$ and $\tilde{\beta}_t$, first, we should derive a closed-form equation for sampling $\mathbf{x}_t$ at an arbitrary timestep t from Equation (2.25) as follows:

$$\begin{aligned}
\mathbf{x}_t&=\sqrt{1-\beta_t}\mathbf{x}_{t-1}+\sqrt{\beta_t}\boldsymbol{\epsilon}_t,\\
&=\sqrt{\alpha_t}\mathbf{x}_{t-1}+\sqrt{1-\alpha_t}\boldsymbol{\epsilon}_t, && \Rightarrow \text{Let } \alpha_t=1-\beta_t\\
&=\sqrt{\alpha_t}\left(\sqrt{\alpha_{t-1}}\mathbf{x}_{t-2}+\sqrt{1-\alpha_{t-1}}\boldsymbol{\epsilon}_{t-2}\right)+\sqrt{1-\alpha_t}\boldsymbol{\epsilon}_t,\\
&=\sqrt{\alpha_t\alpha_{t-1}}\mathbf{x}_{t-2}+\sqrt{1-\alpha_t\alpha_{t-1}}\bar{\boldsymbol{\epsilon}}_t, && \Rightarrow \bar{\boldsymbol{\epsilon}}_t \text{ merges two Gaussians } (\boldsymbol{\epsilon}_t \text{ and } \boldsymbol{\epsilon}_{t-1})\\
&=\cdots\\
&=\sqrt{\alpha_t\alpha_{t-1}\cdots\alpha_1}\mathbf{x}_0+\sqrt{1-\alpha_t\alpha_{t-1}\cdots\alpha_1}\boldsymbol{\epsilon},\\
&=\sqrt{\bar{\alpha}_t}\mathbf{x}_0+\sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, && \Rightarrow \text{reparameterize } \bar{\alpha}_t=\prod_{i=1}^{t}\alpha_t
\end{aligned}\tag{D.9}$$

therefore,

$$q(\mathbf{x}_t|\mathbf{x}_0)=\mathcal{N}(\mathbf{x}_t;\sqrt{\bar{\alpha}}_t\mathbf{x}_0,(1-\bar{\alpha}_t)\mathbb{I}).\tag{D.10}$$

Then,

$$
\begin{aligned}
q\left(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0\right) &= q\left(\mathbf{x}_t|\mathbf{x}_{t-1},\mathbf{x}_0\right)\frac{q\left(\mathbf{x}_{t-1}|\mathbf{x}_0\right)}{q\left(\mathbf{x}_t|\mathbf{x}_0\right)},\\
&\propto \exp\left(-\frac{1}{2}\left(\frac{(\mathbf{x}_t-\sqrt{\alpha_t}\mathbf{x}_{t-1})^2}{\beta_t}+\frac{(\mathbf{x}_{t-1}-\sqrt{\bar{\alpha}_{t-1}}\mathbf{x}_0)^2}{1-\bar{\alpha}_{t-1}}-\frac{(\mathbf{x}_t-\sqrt{\bar{\alpha}_t}\mathbf{x}_0)^2}{1-\bar{\alpha}_t}\right)\right),\\
&= \exp\left(-\frac{1}{2}\left(\left(\frac{\alpha_t}{\beta_t}+\frac{1}{1-\bar{\alpha}_{t-1}}\right)\mathbf{x}_{t-1}^2-\left(\frac{2\sqrt{\alpha_t}\mathbf{x}_t}{\beta_t}+\frac{2\sqrt{\bar{\alpha}_{t-1}\mathbf{x}_0}}{1-\bar{\alpha}_{t-1}}\right)\mathbf{x}_{t-1}+\cdots\right)\right),
\end{aligned}
\tag{D.11}
$$

where $\cdots$ include the terms irrelevant to $\mathbf{x}_{t-1}$. Since $q\left(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0\right)$ is a Gaussian distribution, we can find the mean and the variance from Equation (D.11):

$$
\tilde{\beta}_t = \frac{1}{\frac{\alpha_t}{\beta_t}+\frac{1}{1-\bar{\alpha}_{t-1}}} = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\cdot\beta_t,
\tag{D.12}
$$

and

$$
\begin{aligned}
\tilde{\boldsymbol{\mu}}_t &= \left(\frac{\sqrt{\alpha_t}\mathbf{x}_t}{\beta_t}+\frac{\sqrt{\bar{\alpha}_{t-1}}\mathbf{x}_0}{1-\bar{\alpha}_{t-1}}\right)/\left(\frac{\alpha_t}{\beta_t}+\frac{1}{1-\bar{\alpha}_{t-1}}\right),\\
&= \left(\frac{\sqrt{\alpha_t}\mathbf{x}_t}{\beta_t}+\frac{\sqrt{\bar{\alpha}_{t-1}}\mathbf{x}_0}{1-\bar{\alpha}_{t-1}}\right)\cdot\left(\frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\cdot\beta_t\right),\\
&= \frac{(1-\bar{\alpha}_{t-1})\sqrt{\alpha_t}}{1-\bar{\alpha}_t}\mathbf{x}_t+\frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathbf{x}_0,\\
&= \frac{(1-\bar{\alpha}_{t-1})\sqrt{\alpha_t}}{1-\bar{\alpha}_t}\mathbf{x}_t+\frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\left(\frac{1}{\sqrt{\bar{\alpha}_t}}(\mathbf{x}_t-\sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_t)\right),\\
&= \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t-\frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_t\right).
\end{aligned}
\tag{D.13}
$$

Therefore, we can rewrite the L_{t-1} term in Equation (D.8) using the KL-divergence between two Gaussian distributions as:

$$
L_{t-1} = D_{KL}\left(q\left(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0\right)||p_\theta\left(\mathbf{x}_{t-1}|\mathbf{x}_t\right)\right) = \mathbb{E}_q\left[\frac{1}{2\sigma_t^2}\|\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t,\mathbf{x}_0)-\boldsymbol{\mu}_\theta(\mathbf{x}_t,t)\|^2\right]+C.
\tag{D.14}
$$

Essentially, $\mu_\theta(\mathbf{x}_t,t)$ is a θ-parameterized function that predicts the distribution mean given $\mathbf{x}_t$ and t. We can reparameterize it as:

$$
\mu_\theta(\mathbf{x}_t,t) = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t-\frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t,t)\right),
\tag{D.15}
$$

which then

$$
\begin{aligned}
L_{t-1} &= \mathbb{E}_q\left[\frac{1}{2\sigma_t^2}\left\|\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t,\mathbf{x}_0)-\boldsymbol{\mu}_\theta(\mathbf{x}_t,t)\right\|^2\right]+C,\\
&= \mathbb{E}_q\left[\frac{(1-\alpha_t)^2}{2\alpha_t(1-\alpha_t)\sigma_t^2}\left\|\boldsymbol{\epsilon}_t-\boldsymbol{\epsilon}_\theta(\mathbf{x}_t,t)\right\|^2\right]+C,\\
&= \mathbb{E}_{t,\mathbf{x}_0,\boldsymbol{\epsilon}}\left[\frac{(1-\alpha_t)^2}{2\alpha_t(1-\alpha_t)\sigma_t^2}\left\|\boldsymbol{\epsilon}-\boldsymbol{\epsilon}_\theta\left(\sqrt{\bar{\alpha}_t}\mathbf{x}_0+\sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon},t\right)\right\|^2\right]+C.
\end{aligned}
\tag{D.16}
$$

E. Derivation for Score-based Generative Model

Once the score network is trained, data generation can be accomplished through an iterative method known as *Langevin dynamics*, or Langevin Monte Carlo (Parisi, 1981; Grenander and Miller, 1994). Langevin dynamics is originally formulated to describe the behavior of molecular systems, such as Brownian motion. The fundamental principle of Langevin dynamics in the context of Brownian motion is captured by the Langevin Equation, which is expressed as follows:

$$m\ddot{x} = -\lambda\dot{x} + \eta, \tag{E.1}$$

where m is the mass of the particle, λ is the damping coefficient, and $\eta \sim \mathcal{N}(0, 2\sigma^2)$ is a Gaussian noise.

The dynamics of Brownian motion as described in Equation (E.1) can be further derived by considering the potential energy in the system. Specifically, the force acting on a particle can be expressed in terms of the gradient of potential energy; i.e., $\frac{\partial V(x)}{\partial x} = \nabla V(x)$. This leads to a reformulation of the equation as:

$$\nabla V(x) = -\lambda\dot{x} + \eta, \tag{E.2}$$

or equivalently,

$$\lambda\dot{x} = -\nabla V(x) + \eta. \tag{E.3}$$

This relationship allows us to interpret the motion of the particle in terms of potential energy changes, where $-\nabla V(x)$ represents the deterministic force derived from the potential energy landscape and η accounts for stochastic thermal fluctuations. To capture these dynamics in a more mathematically rigorous framework suitable for simulation and analysis, we can transform this physical concept into a stochastic differential equation (SDE):

$$dx = -\nabla V(x)dt + \sqrt{2}\sigma dW, \tag{E.4}$$

where the term $-\nabla V(x)$ is the drift term, which guides the particle towards lower potential energy states, thus simulating deterministic motion under the influence of forces. The term $\sqrt{2}\sigma dW$ represents the stochastic component of the motion, with σ being the volatility (akin to the temperature in physical systems) and dW denoting the derivative of a standard Wiener process, which models the random thermal fluctuations.

In the context of probability distributions, the stochastic process described in Equation (E.4) corresponds to a Fokker-Planck equation (Fokker, 1914; Planck, 1917; Kadanoff, 2000) for the probability density function $p(x, t)$ of finding the system in state x at time t:

$$\frac{\partial}{\partial t}p(x,t) = \nabla \cdot \left[\nabla V(x)p(x,t) + \sigma^2\nabla p(x,t)\right]. \tag{E.5}$$

At steady state, the time derivative of the probability density function becomes zero, implying:

$$0 = \nabla \cdot \left[\nabla V(x)p(x,t) + \sigma^2\nabla p(x,t)\right], \tag{E.6}$$

which simplifies to:

$$\nabla V(x)p(x) = -\sigma^2\nabla p(x,t). \tag{E.7}$$

Dividing both sides by $p(x)$ and rearranging gives:

$$\nabla V(x) \propto -\frac{\nabla p(x,t)}{p(x)} = -\nabla \log p(x), \tag{E.8}$$

indicating that, at equilibrium, the gradient of the potential energy is proportional to the gradient of the log of the probability distribution. **This equilibrium condition underlies the principle that in a score-based generative model, the score function $\nabla_{\mathbf{x}} \log p(\mathbf{x})$ can be interpreted as akin to a force derived from a potential energy landscape,** guiding the generation process towards high-probability regions of the data distribution.

Therefore, Equation (E.4) can be re-written as:

$$dx = \nabla \log p(x)dt + \sqrt{2}\sigma dW, \tag{E.9}$$

To implement this in a discrete setting for numerical simulation or data generation, the SDE can be approximated as follows:

$$\mathbf{x}_{i+1} \leftarrow \mathbf{x}_i + \nabla_x \log p(\mathbf{x})\epsilon + \sqrt{2}\sigma\Delta W, \tag{E.10}$$

where ϵ is a small timestep, and $\Delta W \sim \mathcal{N}(0, \epsilon)$ represents a discrete approximation of the Wiener process increment. Thus, it can be further simplified as:

$$\mathbf{x}_{i+1} \leftarrow \mathbf{x}_i + \nabla_x \log p(\mathbf{x}) \cdot \epsilon + \sqrt{2\epsilon} \cdot \mathbf{z}_i, \tag{E.11}$$

where $\mathbf{z}_i \sim \mathcal{N}(0, I)$.

F. Tutorials for Generating Household Travel Survey Data

F.1. Variational Autoencoder

The VAE loss function consists of two primary components as shown in Equation (2.6): the *reconstruction loss* and the *regularization loss*. The reconstruction loss measures how well the model can reproduce the original data. Since the encoder and decoder are both deterministic functions when input is given, the reconstruction loss term, $\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x}|\mathbf{z})\right]$, can be re-written as $\mathbb{E}_{\mathbf{x}}\left[\log p_\theta(\hat{\mathbf{x}}|\mathbf{x})\right]$. If we assume that this distribution has a Gaussian form:

$$\log p_\theta(\hat{\mathbf{x}}|\mathbf{x}) \propto \log e^{-|\mathbf{x}-\hat{\mathbf{x}}|^2} = -|\mathbf{x}-\hat{\mathbf{x}}|^2, \tag{F.1}$$

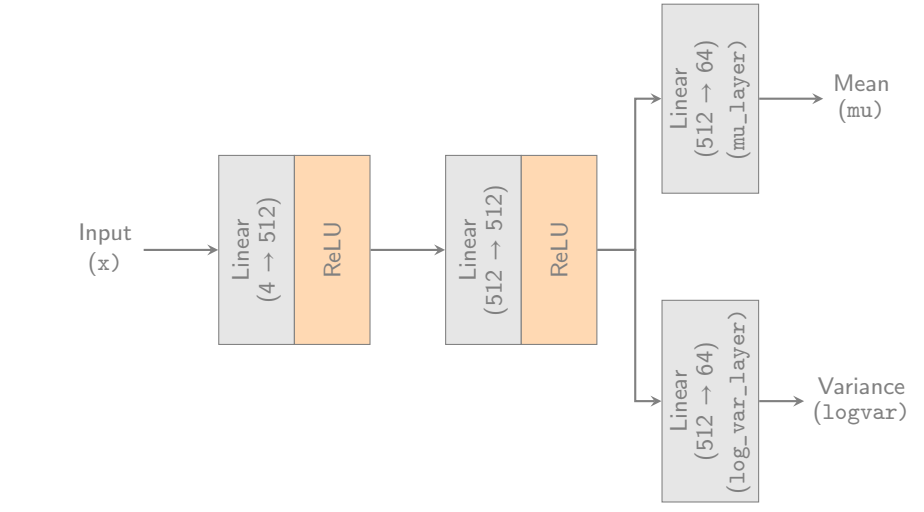
then maximizing $\mathbb{E}_{\mathbf{x}}\left[\log p_\theta(\hat{\mathbf{x}}|\mathbf{x})\right]$ corresponds to minimizing the square loss between $\mathbf{x}$ and $\hat{\mathbf{x}}$. The regularization loss, on the other hand, ensures that the induced latent space from the encoder (i.e., the posterior distribution) follows the same distribution as the prior distribution. However, in practical implementations, the regularization term often requires simplifying assumptions. In this case, we assume that both the posterior distribution $q_\phi(\mathbf{z}|\mathbf{x})$ and the prior distribution $p(\mathbf{z})$ are Gaussian distribution. Under these assumptions, the Kullback-Leibler (KL) divergence between the two Gaussian distributions, considering D-dimensional vector $\mathbf{z}$ can be simplified into a more computationally manageable form as:

$$\begin{aligned} D_{KL}\left(q_\phi(\mathbf{z}|\mathbf{x})\,||p(\mathbf{z})\right) &= D_{KL}\left(\mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)||\mathcal{N}(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)\right) \\ &= \frac{1}{2}\left(\mathrm{Tr}(\boldsymbol{\Sigma}_2^{-1}\boldsymbol{\Sigma}_1) + (\boldsymbol{\mu}_2-\boldsymbol{\mu}_1)^\top\boldsymbol{\Sigma}_2^{-1}(\boldsymbol{\mu}_2-\boldsymbol{\mu}_1) - D + \log\frac{\det\boldsymbol{\Sigma}_2}{\det\boldsymbol{\Sigma}_1}\right). \end{aligned} \tag{F.2}$$

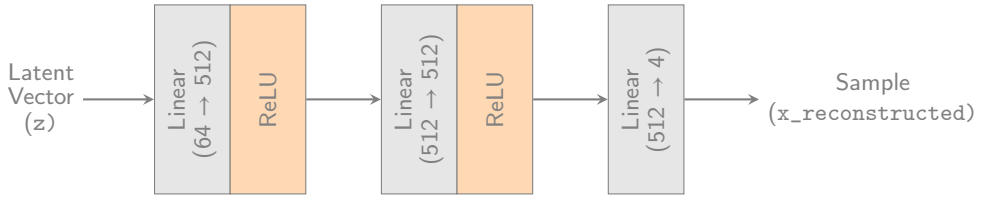
where $(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$, and $(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$ are the mean vectors and covariance matrices for $q_\phi(\mathbf{z}|\mathbf{x})$ and $p(\mathbf{z})$, respectively. We can further simplify the equations by setting the prior distribution, $p(\mathbf{x})$, to follow standard Gaussian distribution, i.e., $\boldsymbol{\mu}_2 = 0$, $\boldsymbol{\Sigma}_2 = \mathbf{I}$, and the posterior distribution, $q_\phi(\mathbf{z}|\mathbf{x})$, follows zero-mean Gaussian distribution with diagonal covariance matrix, i.e., $\boldsymbol{\mu}_1 = 0$, and $\boldsymbol{\Sigma}_1 = \boldsymbol{\sigma}_1^2\mathbf{I}$:

$$\begin{aligned} D_{KL}\left(q_\phi(\mathbf{z}|\mathbf{x})\,||p(\mathbf{z})\right) &= \frac{1}{2}\left(\mathrm{Tr}(\boldsymbol{\Sigma}_1) + \boldsymbol{\mu}_1^\top\boldsymbol{\mu}_1 - D - \log\det\boldsymbol{\Sigma}_1\right) \\ &= \frac{1}{2}\sum_{i=1}^{D}\left(\sigma_{1,i}^2 + \mu_{1,i}^2 - 1 - \log\sigma_{1,i}^2\right), \end{aligned} \tag{F.3}$$

where $\mu_{1,i}$ is i-th element in $\boldsymbol{\mu}_1$ and $\sigma_{1,i}$ is i-th element in $\boldsymbol{\sigma}_1$. The corresponding code implementation of the VAE loss is stated below. We use $\log\boldsymbol{\sigma}_1$ instead of $\boldsymbol{\sigma}_1$ for numerical stability, and the variable `logvar` corresponds to $2\log\boldsymbol{\sigma}_1$. The variable `mu` corresponds to $\boldsymbol{\mu}_1$. Figure 13 illustrates the structure of the neural network for this section. Figure 13 (a) shows the structure of the encoder which encodes the inputs to D-dimensional vectors for both Mean and Variance and we used $D = 64$, and Figure 13 (b) shows the structure of the decoder which generates the sample from the latent vector. The Pytorch code implementation is as follows:



(a) Network structure of encoder



(b) Network structure of decoder

Figure 13: Neural network structures of VAE in HTS data

```python
class VAE(nn.Module):
    def __init__(self, input_dim, hidden_dim, z_dim):
        super(VAE, self).__init__()
        # Encoder
        self.encoder = nn.Sequential(
            nn.Linear(input_dim, hidden_dim),
            nn.ReLU(),
            nn.Linear(hidden_dim, hidden_dim),
            nn.ReLU()
        )
        self.mu_layer = nn.Linear(hidden_dim, z_dim)
        self.log_var_layer = nn.Linear(hidden_dim, z_dim)

        # Decoder
        self.decoder = nn.Sequential(
            nn.Linear(z_dim, hidden_dim),
            nn.ReLU(),
            nn.Linear(hidden_dim, hidden_dim),
            nn.ReLU(),
            nn.Linear(hidden_dim, input_dim)
        )

    def encode(self, x):
        h = self.encoder(x)
        mu = self.mu_layer(h)
        log_var = self.log_var_layer(h)
        return mu, log_var

    def reparameterize(self, mu, log_var):
        std = torch.exp(0.5 * log_var)
        eps = torch.randn_like(std)
```

```
            return mu + eps * std

        def decode(self, z):
            return self.decoder(z)

        def forward(self, x):
            mu, log_var = self.encode(x)
            z = self.reparameterize(mu, log_var)
            x_reconstructed = self.decode(z)
            return x_reconstructed, mu, log_var

    def compute_loss(x, x_reconstructed, mu, log_var):
        """
        MSE based reconstruction loss and KL divergence loss for VAE.
        """
        # Reconstruction loss
        recon_loss = nn.functional.mse_loss(x_reconstructed, x, reduction='sum')
        # KL Divergence
        kl_loss = -0.5 * torch.sum(1 + log_var - mu.pow(2) - log_var.exp())
        # total loss
        loss = recon_loss + kl_loss
        return loss
```

F.2. Generative Adversarial Networks

Unlike VAEs, GANs do not require complex derivations of the loss function. The code implementation of the Equations (2.8) and (2.9) can be written intuitively. In practice, this involves using binary cross-entropy loss. The discriminator tries to identify if the given input is from the real data set or the generated data set. On the contrary, the generator tries to overcome the identification from the discriminator. Similar to the VAE, we constructed the 3-layer neural network for (a) generator and (b) discriminator as shown in Figure 14. The noise vector (latent vector) is D-dimensional vector where $D = 64$. The Pytorch code implementation is as follows:

```
    class Generator(nn.Module):
        def __init__(self, input_dim, output_dim, hidden_dim):
            super(Generator, self).__init__()
            self.net = nn.Sequential(
                nn.Linear(input_dim, hidden_dim),
                nn.ReLU(),
                nn.Linear(hidden_dim, hidden_dim),
                nn.ReLU(),
                nn.Linear(hidden_dim, output_dim)
            )

        def forward(self, z):
            return self.net(z)

    class Discriminator(nn.Module):
        def __init__(self, input_dim, hidden_dim):
            super(Discriminator, self).__init__()
            self.net = nn.Sequential(
                nn.Linear(input_dim, hidden_dim),
                nn.ReLU(),
                nn.Linear(hidden_dim, hidden_dim),
                nn.ReLU(),
                nn.Linear(hidden_dim, 1),
                nn.Sigmoid()
            )

        def forward(self, x):
            return self.net(x)

    def compute_discriminator_loss(discriminator, real_data, fake_data, criterion,
    real_label, fake_label):
        batch_size = real_data.size(0)
        # loss for real data
        real_targets = torch.full((batch_size, 1), real_label, device=real_data.device)
```

```
        d_real = discriminator(real_data)
        d_loss_real = criterion(d_real, real_targets)

        # loss for generated data
        fake_targets = torch.full((batch_size, 1), fake_label, device=real_data.device)
        d_fake = discriminator(fake_data)
        d_loss_fake = criterion(d_fake, fake_targets)

        d_loss = d_loss_real + d_loss_fake
        return d_loss

    def compute_generator_loss(discriminator, fake_data, criterion, real_label):
        batch_size = fake_data.size(0)
        real_targets = torch.full((batch_size, 1), real_label, device=fake_data.device)
        d_fake = discriminator(fake_data)
        g_loss = criterion(d_fake, real_targets)
        return g_loss
```

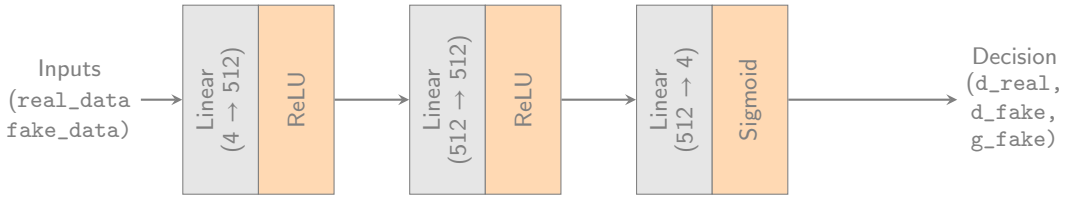
F.3. Normalizing Flows (Flow-based Generative Models)

We use a flow-based generative model inspired by RealNVP (Dinh et al., 2016), which leverages affine coupling layers and alternating masking strategies to transform complex data distributions into a standard normal distribution. For simplicity, and since the data dimensionality is not too large, the model in this tutorial does not include permutation layers between coupling layers and batch normalization that were proposed in the original RealNVP paper. A more detailed implementation of RealNVP will be discussed in Section F.3. Despite these simplifications, our model maintains the core idea of using invertible affine transformations to compute the log-likelihood and optimize the data distribution. The training objective is to maximize the likelihood of the observed data under this transformation, which can be expressed as minimizing the negative log-likelihood. The original loss function for the normalizing flow is the same as stated in Equation (2.14). For the affine coupling layers, we use the same equations as described from Equation (2.16) to Equation (2.19). We use much simpler networks for scaling and translation network as shown in Figure 15. The code Pytorch implementation is as follows:

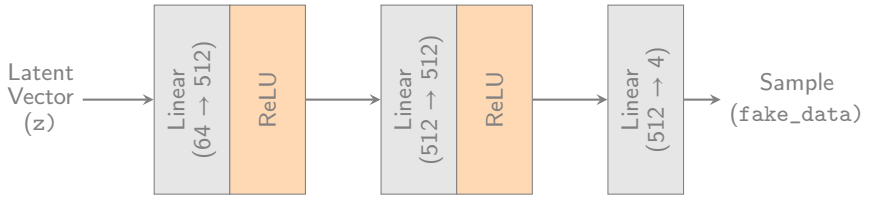
```
    class AffineCoupling(nn.Module):
        [...]
        # provide calculation of log_det_jacobian using scale and translation functions
        def forward(self, x):
            x_masked = x * self.mask
            s = self.scale_net(x_masked) * (1 - self.mask)
            t = self.translate_net(x_masked) * (1 - self.mask)
            y = x_masked + (1 - self.mask) * (x * torch.exp(s) + t)
            log_det_jacobian = torch.sum(s, dim=1)
            return y, log_det_jacobian
            [...]

    class NormalizingFlow(nn.Module):
        def __init__(self, input_dim, hidden_dim, num_layers):
            super(NormalizingFlow, self).__init__()
            self.layers = nn.ModuleList()
            mask = self.create_mask(input_dim, even=True)
            for i in range(num_layers):
                self.layers.append(AffineCoupling(input_dim, hidden_dim, mask))
                # Alternate the mask for the next layer
                mask = 1 - mask

        def forward(self, x):
            """
            Forward pass through the flow: transform x to z and compute log_prob.
            """
            log_det_jacobian = torch.zeros(x.size(0))
            for layer in self.layers:
                x, ldj = layer(x)
                log_det_jacobian += ldj
            # Compute log_prob under base distribution
            log_prob_z = self.base_log_prob(x)
```

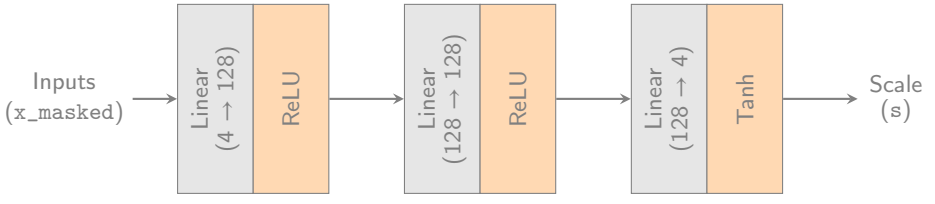


(a) Network structure of discriminator

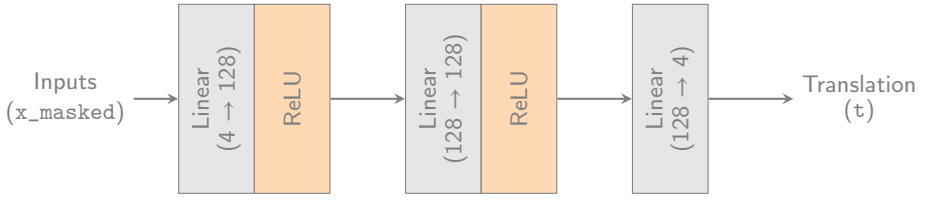


(b) Network structure of generator

Figure 14: Neural network structures of GAN in HTS data



(a) Network structure of scale net



(b) Network structure of translation net

Figure 15: Neural network structures of normalizing flow in HTS data

```
            log_prob = log_prob_z + log_det_jacobian
            return log_prob

    def compute_nf_loss(model, x):
        log_prob = model(x)
        loss = -torch.mean(log_prob)
        return loss
```

F.4. Score-based Generative Model

The original objective function of the Score-based Model is defined as Equation (2.21). Here we show how the transformation to Equation (2.22) proceeds when we are using the normal distribution for the noise $q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})$. In this case, the equation can be simplified as shown follows:

$$\left|\left|\nabla_{\mathbf{x}} \log q_\sigma(\tilde{\mathbf{x}}|\mathbf{x}) - s_\theta(\tilde{\mathbf{x}}, \sigma)\right|\right|_2^2 = \left|\left|\frac{(\tilde{\mathbf{x}} - \mathbf{x})}{\sigma^2} + s_\theta(\tilde{\mathbf{x}}, \sigma)\right|\right|_2^2, \tag{F.4}$$

since

$$
\begin{aligned}
q_\sigma(\tilde{\mathbf{x}}|\mathbf{x}) &= \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(\tilde{\mathbf{x}}-\mathbf{x})^2}{2\sigma^2}\right), \\
\log q_\sigma(\tilde{\mathbf{x}}|\mathbf{x}) &= -\frac{(\tilde{\mathbf{x}}-\mathbf{x})^2}{2\sigma^2} - \log\left(\sqrt{2\pi\sigma^2}\right), \\
\nabla_{\mathbf{x}} \log q_\sigma(\tilde{\mathbf{x}}|\mathbf{x}) &= -\frac{(\tilde{\mathbf{x}}-\mathbf{x})}{\sigma^2}.
\end{aligned}
\tag{F.5}
$$

where the term $\tilde{\mathbf{x}} - \mathbf{x}$ corresponds to the variable `noise`. We use a neural network with three linear layers and ReLU activations to obtain the estimated score as shown in Figure 16. In this tutorial, we train the NCSN using sequential sigma noise levels rather than randomly mixing them. This approach diverges from the typical method of random sampling of sigma values but was chosen to provide a clearer illustration of the overall training process. Given the relatively low dimension of the dataset, the results show that the sequential method is also feasible. The Pytorch code implementation is as follows:

```
    class ScoreNetwork(nn.Module):
        def __init__(self, input_dim, hidden_dim, sigma_min, sigma_max, num_sigma):
            super(ScoreNetwork, self).__init__()
            [...]
            self.sigmas = torch.exp(torch.linspace(
                np.log(self.sigma_max), np.log(self.sigma_min), self.num_sigma))

            # Neural network architecture
            self.net = nn.Sequential(
                nn.Linear(input_dim + 1, hidden_dim),
                nn.ReLU(),
                nn.Linear(hidden_dim, hidden_dim),
                nn.ReLU(),
                nn.Linear(hidden_dim, input_dim)
            )

        def forward(self, x, sigma):
            # Concatenate sigma as an additional feature
            sigma_feature = sigma.view(-1, 1)
            x_sigma = torch.cat([x, sigma_feature], dim=1)
            score = self.net(x_sigma)
            return score

        def loss_fn(self, x):
            total_loss = 0.0
            for sigma in self.sigmas:
                sigma = sigma.to(device)
                sigma = sigma.expand(x.size(0), 1)
                noise = torch.randn_like(x) * sigma
                x_noisy = x + noise
                score = self.forward(x_noisy, sigma)
                loss = ((score + noise / (sigma ** 2)) ** 2).mean()
                total_loss += loss
            return total_loss

    [...]
    score_model = ScoreNetwork(input_dim, hidden_dim, sigma_min, sigma_max, num_sigma)

    [...] # in training loop
    loss = score_model.loss_fn(x) # Equation (4.6)
```

F.5. Diffusion Models

In the tutorial code, we used the Denoising Diffusion Probabilistic Model (DDPM) as the reference for generating data (Ho et al., 2020). Instead of using the traditional U-Net architecture for noise prediction, we implemented a simpler neural network structure, as shown in Figure 17. The U-Net architecture is commonly used in diffusion models due to its powerful ability to capture multi-scale features in image data. However, in this tutorial, we implemented a simpler

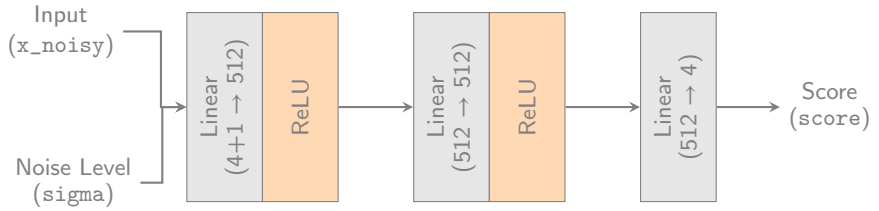


Figure 16: Neural network structure of NCSN in HTS data

neural network to reduce computational complexity and facilitate a focus on the fundamental concepts of diffusion models, such as stepwise noise estimation and progressive denoising. The loss function for training is calculated based on Equation (2.32) presented above, which measures the model's ability to predict noise accurately at each step of the diffusion process. The Pytorch code implementation is as follows:

```
    class DiffusionModel(nn.Module):
        [...]

        def forward_diffusion_sample(x0, t):
            noise = torch.randn_like(x0)
            sqrt_alpha_prod = sqrt_alphas_cumprod[t].view(-1, 1)
            sqrt_one_minus_alpha_prod = sqrt_one_minus_alphas_cumprod[t].view(-1, 1)
            x_t = sqrt_alpha_prod * x0 + sqrt_one_minus_alpha_prod * noise
            return x_t, noise
        [...]

    def compute_diffusion_loss(model, x0, t, sqrt_alphas_cumprod,
    sqrt_one_minus_alphas_cumprod):
        """
        Do a forward diffusion with the given x0 and timestep t,
        compute the MSE between the noise predicted by the model and the actual noise.
        """
        x_t, noise = forward_diffusion_sample(x0, t, sqrt_alphas_cumprod,
    sqrt_one_minus_alphas_cumprod)
        noise_pred = model(x_t, t)
        loss = nn.functional.mse_loss(noise_pred, noise)
        return loss
```

G. Tutorials for Generating Highway Traffic Speed Contour

G.1. Variational Autoencoder

We utilize the result of Equation (F.3) to compute the loss of the VAE. The code structure and implementation of the loss function are consistent with the approach presented in Section F.1. Figure 18 illustrates the structure of the neural network for traffic speed contour generation. Figure 18 (a) illustrates the structure of the encoder. The encoder encodes the data to Mean and Variance with a D-dimensional vector. In this section, $D = 64$ is applied. Figure 18 (a) is the structure of the decoder that generates the image from the latent vector.

G.2. Generative Adversarial Networks

The loss function and model architecture of the GAN for traffic speed contour estimation follows the implementation outlined in Section F.2. We employ binary cross-entropy loss for both the discriminator and the generator.

Figure 19 presents the detailed neural network architecture of the GAN, which closely resembles the VAE structure depicted in Figure 18. The discriminator shown in Figure 19 (a) outputs the decision for the input if it is real or fake. The generator, shown in Figure 19 (b) outputs the image from the D-dimensional noise vector. Here, $D = 64$ is applied. This similarity of the neural network structure between GAN and VAE enables us to qualitatively compare the performance of these two models. The models discussed in subsequent sections, such as Normalizing Flows, Score-based Generative Models, and Diffusion Models, possess more intricate and distinctive structures. Consequently,

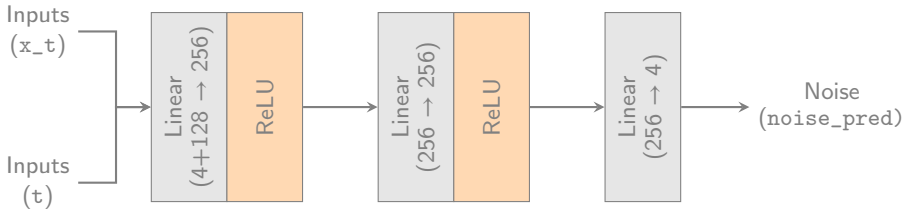


Figure 17: Neural network structure of DDPM in HTS data

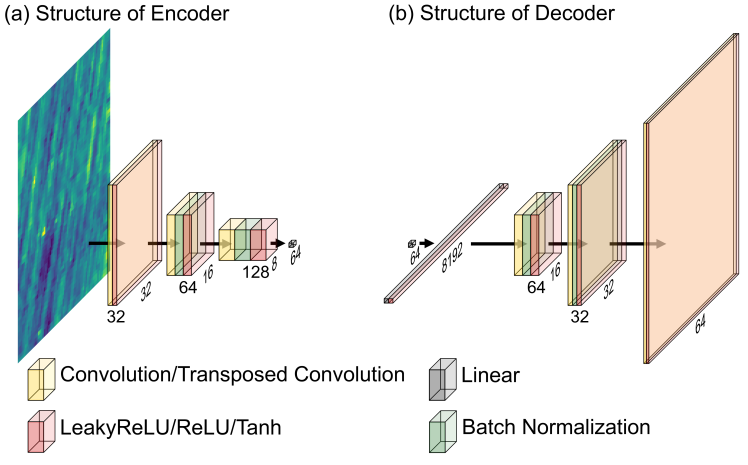


Figure 18: Neural network structure of VAE in highway traffic speed contour

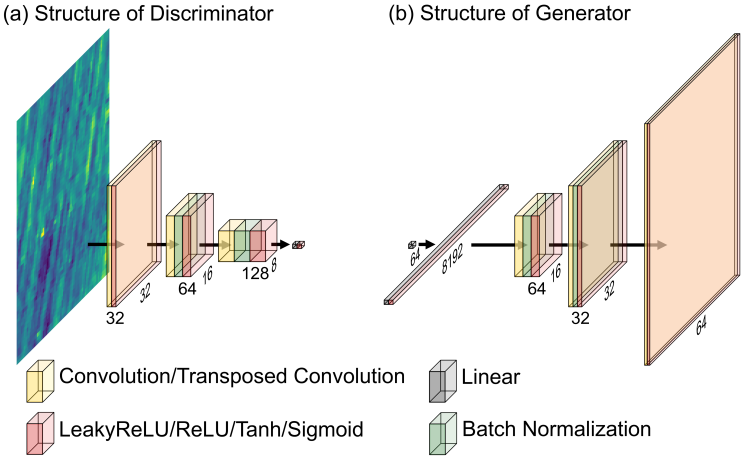


Figure 19: Neural network structure of GAN in highway traffic speed contour

evaluating these models within the same neural network framework is impractical. However, the architectures of the VAE and GAN can be designed to be similar, which fulfills our purpose of comparison.

G.3. Normalizing Flows (Flow-based Generative Models)

As we discussed in Section 2.5, among various Flow-based Generative Models such as Dinh et al. (2014); Rezende and Mohamed (2015); Kingma and Dhariwal (2018); Durkan et al. (2019), we implemented RealNVP (Dinh et al., 2016) because of its ability to generate high-dimensional distribution due to its flexibility. As we can see in Equa-

tion (2.14), we need to calculate the $\log p_0(z_0)$ in Equation (2.13) to get the likelihood of the data. In the implementation of RealNVP, we can simplify the log-likelihood $\log(p(\mathbf{z}))$ by assuming z is sampled from the standard normal distribution. The other term in Equation (2.13), the summation of the determinant of the Jacobian, is simplified by applying a coupling layer as described through Equation (2.15) to Equation (2.19). Thus, the summation of the determinant of the Jacobian can be simplified as $\sum_j s(x_{1:d})_j$. In our code application, the regularization loss for batch normalization and scale parameters are applied following the contents in RealNVP (Dinh et al., 2016).

Dinh et al. (2016) multiplied 5×10^{-5} for the regularization term of scale network. However, since the coefficients must be re-calibrated based on the data, we fine-tuned each coefficient for the loss term in our implementation. A general guideline is to prioritize log-likelihood over the sum of the determinant of the Jacobian. Additionally, the sum of the batch normalization also plays a critical role in the qualitative performance, which has been incorporated into the provided code.

The neural network structure of RealNVP is shown in Figure 20. The neural network is composed of multiple layers of its Backbone Block, which is shown in Figure 20 (a). Each Backbone Block consists of convolution, Instance 2D Normalization, and ReLU layer. Each Backbone Block is the smallest unit for the scale and translation network. In the implementation in the current paper, we stacked 22 layers of scale and translation network. The dimensions and number of channels of the features change at each layer, as detailed in Figure 20 (b). In the generation stage, the computation proceeds in an inverse way.

G.4. Score-based Generative Models

In the code implementation of NCSN in traffic speed contour, we had minor adjustments from the code from Section F.4. By definition, $\tilde{\mathbf{x}}$ is defined as $\tilde{\mathbf{x}} = \mathbf{x} + \sigma \cdot \epsilon$. Then $\nabla_{\mathbf{x}} \log q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})$ in Equation (F.4) is transformed as ϵ/σ. Practically, σ^2 is multiplied by the objective function as the weight of each stage. The detailed implementation can be found in the code.

The neural network of NCSN is formed with a block-wise structure with a Residual Block. Figure 21 (a) and Figure 21 (b) illustrate the structure of the Residual Block and whole neural network for score function, respectively. Skip-connection is applied in each block to maintain the low-level feature information.

G.5. Diffusion Models

The loss function of DDPM is simplified as the difference between the predicted noise and the real noise as stated in Equation (2.32). The criteria of the difference, i.e., the norm of difference, can be varied by the problem. Our code is also designed to select one of three types of loss, which is L_1, MSE loss, and Huber loss. For simplicity, we inserted the code block only including the MSE loss.

The neural network that is used for the DDPM is illustrated in Figure 22. Figure 22 (a), Figure 22 (b), and Figure 22 (c) show the structure of the Residual Block, Backbone Block, and U-Net of DDPM, respectively. The U-Net structure that is applied in the neural network in DDPM enables one to learn the information from high-level features or features from previous layers. The residual block contains the time embedding as an input, which identifies the stage of the current generation process. A similar mechanism can also be found in the Score-based model, as the score-based model creates an image using the scheduled σ. The neural network in DDPM also includes the attention mechanism that captures the relationship between features. The attention mechanism was applied inside the Backbone block, between the Residual block and the group normalization.

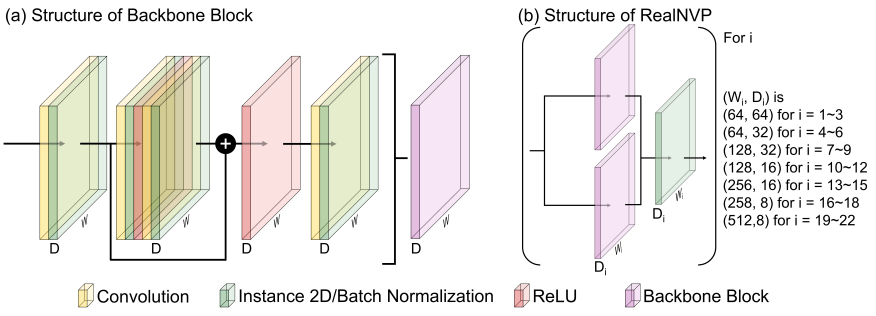


Figure 20: Neural network structure of RealNVP in highway traffic speed contour

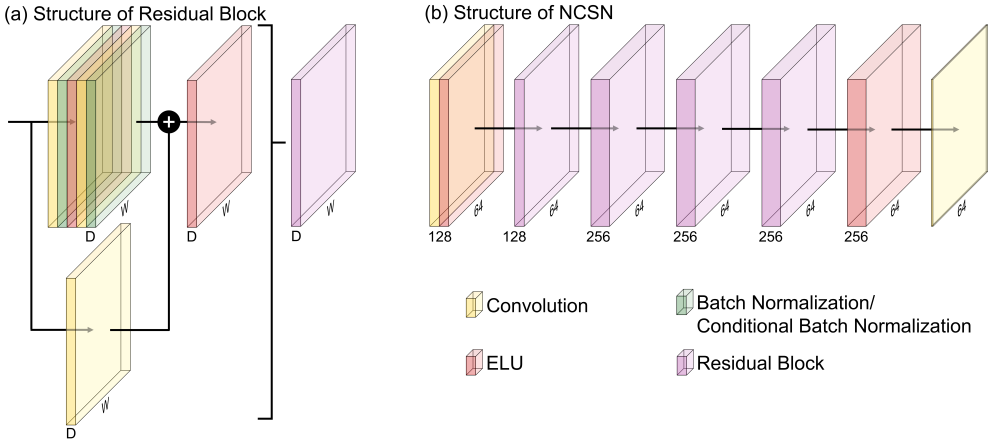


Figure 21: Neural network structure of NCSN in highway traffic speed contour

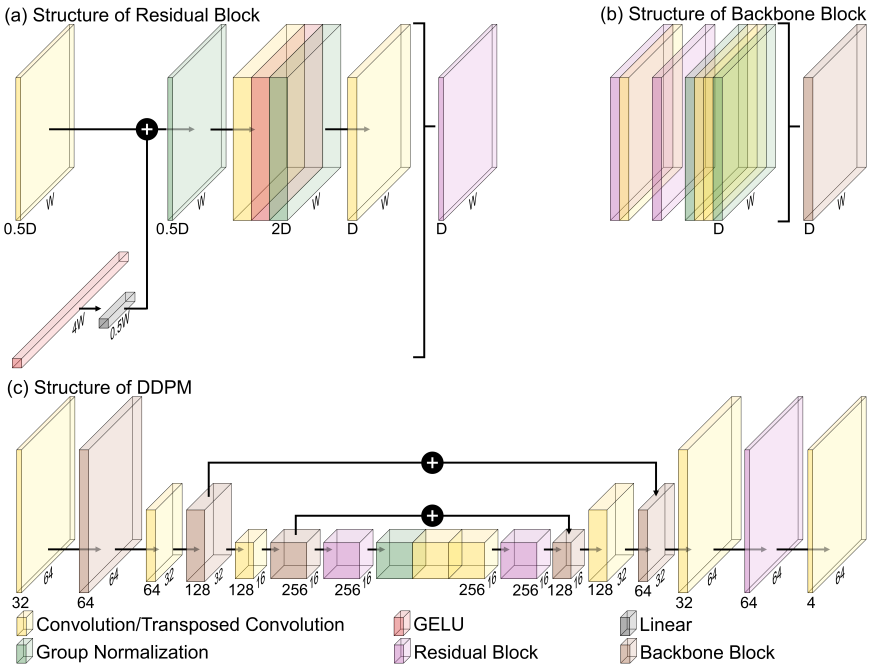


Figure 22: Neural network structure of DDPM in highway traffic speed contour

H. Detailed Description on HTS Data

Table 13 provides a detailed description of the columns contained in the HTS dataset utilized in this study. The complete original dataset can be accessed through the provided Github repository link.

Table 13
Description on columns in HTS data

Field	Description
ID	Household number + Person number
sheet_code	Household number
relation	1: Head of Household, 2: Spouse, 3: Child, 4: Parent, 5: Others
sex	1: Male, 2: Female
age_code	1: Ages 0–11, 2: Ages 12–17, 3: Ages 18–22, 4: Ages 23–27, 5: Ages 28–32, 6: Ages 33–37, 7: Ages 38–42, 8: Ages 43–47, 9: Ages 48–52, 10: Ages 53–57, 11: Ages 58–62, 12: Ages 63–69, 13: Ages 70–
job_type	1: Student, 2: Housewife/Unemployed (includes children not in school), 3: Professionals and Related Workers, 4: Service Workers, 5: Sales Workers, 6: Managers and Office Workers 7: Skilled Workers in Agriculture, Forestry, and Fisheries, 8: Technicians/Machine Operators/Short-Term Laborers, 9: Others
ID_trip_cnt	Trip count for each ID starting from 1
start_type	1: Home, 2: Workplace, 3: School, 4: Others
start_zcode	Refer to the Excel file for details.
start_zcode_num	Numbering for "start_zcode" starting from 0 in alphanumerical order.
start_time	Starting time in hours (e.g., 0.5 represents 30 minutes).
{start; end; trip}_time_{round; num}_{6;12}	Columns for start, end, or trip times with two versions: **round** – time values rounded to the nearest 6 or 12 minutes; **num** – numerical representation based on 6- or 12-minute intervals.
start_time_num_{6;12}	Numerical timestep based on a unit of either 6 or 12 minutes.
prev_act_{num: label}	Columns representing the previous action with both a number and a label. The numbering starts from {home; 0}.
prev_mode_{num: label}	Columns representing the previous mode with both a number and a label. The numbering starts from {stay; 0}.
{start; end}_ {home; travel; work; education; academy; leisure; shopping; escort}_time	For each activity type, records the first or last start/end time of the activity.
first_trip	Indicates the first trip.
last_trip	Indicates the last trip.

I. Smoothing Process of Highway Traffic Speed Contour

The detailed process of interpolation, which is based on Edie's definition of traffic flow dynamics, is outlined in Table 14.

Table 14
Pseudocode for data preprocessing

Algorithm 1: Smoothing the speed using the Edie's definition

Given
$V(x,t)$: Velocity at x,t
W,T: Spatial and temporal width of the data
σ: Spatial range of smoothing
τ: Temporal range of smoothing
c_{free}: Propagation velocity of perturbation in free flow
c_{cong}: Propagation velocity of perturbation in congested flow
V_c: Velocity that transition occurs from free flow to congested flow
ΔV: Width of the transition region
for $x \leftarrow 0$ to W, $t \leftarrow 0$ to T **do**
    Initialize $\phi_{free}(x',t'), \phi_{cong}(x',t')$
    For $x' \leftarrow x - \sigma/2$ to $x + \sigma/2$, $t' \leftarrow t - \tau/2$ to $t + \tau/2$ **do**
        $\Delta x = x' - x$
        $\Delta t_{free} = (t' - t) - (x'-x)/c_{free}$
        $\Delta t_{cong} = (t' - t) - (x'-x)/c_{cong}$
        $\phi_{free}(x',t') = \exp(|\Delta x|/\sigma - |\Delta t_{free}|/\tau)$
        $\phi_{cong}(x',t') = \exp(|\Delta x|/\sigma - |\Delta t_{cong}|/\tau)$
    End for
    Initialize $V_{free}(x,t), V_{free}(x,t)$
    $V_{free}(x,t) = \sum_{x'=x-\sigma/2}^{x+\sigma/2} \Big(\phi_{free}(x',t') \times V(x',t') \Big) \Big/ \sum_{x'=x-\sigma/2}^{x+\sigma/2} \phi_{free}(x',t')$
    $V_{cong}(x,t) = \sum_{x'=x-\sigma/2}^{x+\sigma/2} \Big(\phi_{cong}(x',t') \times V(x',t') \Big) \Big/ \sum_{x'=x-\sigma/2}^{x+\sigma/2} \phi_{cong}(x',t')$
    $V_{min}(x,t) = \min\Big(V_{free}(x,t), V_{cong}(x,t) \Big)$
    $tanh(x,t) = \tanh\Big((V_c - V_{min}(x,t))/\Delta V \Big)$
    $w = 0.5 \times \Big(1 + \tanh(x,t) \Big)$
    $V(x,t) = (1-w) \times V_{free}(x,t) + w \times V_{cong}(x,t)$
End for