Department of Computer Science
Aalborg University
http://www.aau.dk

AALBORG UNIVERSITY

STUDENT REPORT

Title:
MAD-Traj: Multi-modal Attention-based Diffusion Model for Pedestrian Trajectory Prediction

Theme:
Master's Thesis

Project Period:
Spring Semester 2023

Project Group:
CS-23-MI-10-17

Participant(s):
Christian Blæsbjerg
Emil August Hvid Christensen
Nikolai Ajstrup Justesen

Supervisor(s):
Jilin Hu
Haomin Yu

Page Numbers: 25

Date of Completion:
June 16, 2023

Abstract:

The safety of many systems, such as self-driving vehicles, social robots, and delivery drones, depends on predictive models that allow the systems to reason about the movement of nearby agents. In many applications, such systems operate around humans, which are notoriously unpredictable. Generative models allow such systems to make multiple predictions to account for the many future outcomes that can emerge from human unpredictability. Diffusion models are promising, since they avoid the limitations of previous generative models like GAN and VAE. However, diffusion models have their limitations, most notably their slow inference time. In this paper, we propose MAD-Traj; a multi-modal attention-based diffusion model that uses environmental cues from a semantic map and leverages a transformer-based architecture for powerful and efficient spatiotemporal modelling. The model generates trajectories through a non-Markovian diffusion process, accelerating the inference with the DDIM sampler. In a benchmark comparison with 13 state-of-the-art models, MAD-Traj consistently achieves either best or second-best scores for both ADE and FDE, when trained and evaluated on the SDD and ETH/UCY datasets. Using the DDIM sampler, MAD-Traj can reduce its inference time by 87% while only sacrificing 1% of its accuracy in terms of ADE.

The content of this report is freely available, but publication (with reference) may only be pursued due to agreement with the author.

Summary

This master's thesis was developed during the spring semester of 2023 by the three software students Christian, Emil, and Nikolai at Aalborg University. The overarching field of research in this thesis is machine intelligence, with a specific focus on the topic of pedestrian trajectory prediction.

Trajectory prediction is an important problem that is applicable in fields and disciplines, including, but not limited to, robotics, logistics, security, and transportation. It is a vital component in autonomous vehicle systems since planning a safe route in non-static traffic environments requires such systems to account for future positions of nearby traffic agents. The unpredictable nature of human behaviour complicates the problem and motivates research into generative models, that are capable of probabilistically generating multiple predictions, which enables autonomous vehicles to consider multiple future scenarios.

In this paper, we explore the use of a recently proposed class of generative models in the context of trajectory prediction. This class of model, known as Diffusion Models, have quickly become popular in many fields and proven their ability to outperform previous types of generative models, such as Generative Adversarial Networks and Variational Auto Encoders. The primary concern about diffusion models is their computational requirements, which impose limitations on their applicability in real-world systems, such as autonomous vehicles, where fast inference time is a necessity. Accelerating diffusion models is a challenging problem because the underlying issue stems from the mathematical formulation of a diffusion model as an iterative denoising process.

During our pre-specialisation, we explored the predictive capabilities of image-based diffusion models in the context of trajectory prediction. While underpinning the potential of diffusion models as generative trajectory predictors, we concluded an image-based approach was in fact too resource intensive to reasonably consider for real-world applications. In this paper, we propose an alternative model, MAD-Traj, that replaces the image-based trajectory representation with a more compact vector representation. Our model leverages a transformer-based architecture for its excellent ability to efficiently model complex relationships in sequential data and applies this to both temporal and spatial modelling of trajectories and surrounding environments.

We compare our model to 13 state-of-the-art approaches in a benchmark comparison on the Stanford Drone Dataset and the ETH/UCY datasets, which are some of the most widely adopted benchmark datasets in the field. Our model is highly competitive and consistently achieves the best or second-best rank in all of the aforementioned datasets. To speed up the model even further, we experiment with different techniques for accelerating the sampling process of the diffusion model. Utilising a non-Markovian sampling procedure allows us to skip individual steps in the diffusion process. As a result, we can reduce the inference time of our model by 87%, while only sacrificing 1% of its predictive accuracy.

MAD-Traj: Multi-modal Attention-based Diffusion Model for Pedestrian Trajectory Prediction

Christian Blæsbjerg Emil August Hvid Christensen Nikolai Ajstrup Justesen

Aalborg University, Denmark
{cblasb18, echri16, naju18}@student.aau.dk

***Abstract*—The safety of many systems, such as self-driving vehicles, social robots, and delivery drones, depends on predictive models that allow the systems to reason about the movement of nearby agents. In many applications, such systems operate around humans, which are notoriously unpredictable. Generative models allow such systems to make multiple predictions to account for the many future outcomes that can emerge from human unpredictability. Diffusion models are promising, since they avoid the limitations of previous generative models like GAN and VAE. However, diffusion models have their limitations, most notably their slow inference time. In this paper, we propose MAD-Traj; a multi-modal attention-based diffusion model that uses environmental cues from a semantic map and leverages a transformer-based architecture for powerful and efficient spatiotemporal modelling. The model generates trajectories through a non-Markovian diffusion process, accelerating the inference with the DDIM sampler. In a benchmark comparison with 13 state-of-the-art models, MAD-Traj consistently achieves either best or second-best scores for both ADE and FDE, when trained and evaluated on the SDD and ETH/UCY datasets. Using the DDIM sampler, MAD-Traj can reduce its inference time by 87% while only sacrificing 1% of its accuracy in terms of ADE.**

***Index Terms*—Trajectory Prediction, Diffusion Models, Transformer, Knowledge Distillation**

1. Introduction

Sequence modelling and time series forecasting are core problems in computer science and play a significant role in many areas, such as medicine [1], finance [2], meteorology [3], energy [4], robotics [5], urban transportation [6], and computer vision [7]. In recent years, the topic of trajectory prediction has received a great deal of attention in academic literature—undoubtedly motivated by the influx of smart devices and the prospect of autonomous robots and vehicles. Trajectory prediction is an essential part of automated logistic systems to control warehouse robots and delivery drones, and it is a critical constituent of enabling safe autonomous vehicles. At its core, trajectory prediction is the problem of estimating a sequence of future positions for one or more agents, e.g., pedestrians, cyclists, and cars.

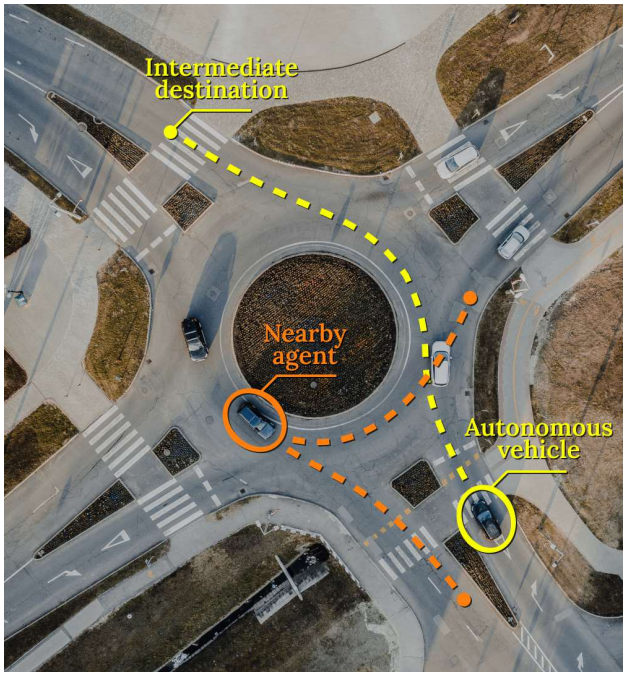


Figure 1: A real-world traffic scenario that highlights the importance of multi-future trajectory prediction. In this scenario, the autonomous vehicle (yellow) needs to navigate to its destination on the other side of the roundabout. To do this safely and avoid potential collisions, it must acknowledge the fact that the nearby agent (orange) might not take the adjacent exit and instead stay in the roundabout until the next exit.

Trajectory prediction is challenging due to the inherent uncertainty associated with human behaviour and the many factors that influence it—factors such as historical movement patterns, environmental cues, social dynamics, goals, aggressiveness, etc. In the context of autonomous vehicles, this poses a significant challenge, since the future behaviour of surrounding agents influences the distribution of collision-avoiding trajectories for an autonomous vehicle. A scenario that highlights this is illustrated in Fig. 1. In this scenario, the autonomous vehicle (highlighted in yellow) must

decide whether to slow down and yield, or accelerate and enter the roundabout. The outcome of this decision is greatly impacted by the behaviour of the nearby agent (highlighted in orange). If the nearby agent decides to exit the roundabout, then the autonomous vehicle can safely enter the roundabout and proceed to its intermediate destination. If, on the other hand, the nearby agent continues on a trajectory within the roundabout, the only way to avoid a collision is for the autonomous vehicle to slow down and yield to the nearby agent. Since human behaviour is inherently uncertain and the nearby agent can change its mind at any moment, there is no way to know, with absolute certainty, which trajectory the nearby agent will take. Consequently, the safest option for the autonomous vehicle is to slow down and yield to avoid a potential collision. To arrive at this conclusion, and make similar well-informed decisions about navigating in traffic, the ability to consider multiple future outcomes is crucial.

Due to the safety-critical demands imposed by autonomous vehicles, there has been a substantial increase in academic interest and research [8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18] into machine learning models that are capable of predicting multiple future trajectories[1]. Generative Adversarial Networks [32] (GANs) and Variational Auto Encoders [33] (VAEs) are common choices for multi-future models, due to their generative nature, which allows them to produce multiple different outputs through a process involving stochastic sampling. However, both of these training frameworks have some inherent issues that impose limitations on the capabilities of the resulting models [34, 35, 36, 37, 38]. Diffusion models are a recently proposed alternative, that generates its output through a discrete denoising process that avoids the commonly cited issues of GANs and VAEs [39, 40].

During the first part of our thesis, we developed MEAT-DDPM [41], an environmentally aware model whose architecture comprised both a VAE and a diffusion model. We showed that by encoding trajectories in images, we could leverage the image processing capabilities of a U-Net [42] based diffusion model to generate trajectories and increase their diversity by conditioning them on diversely sampled goals from the VAE. The trajectories and goals in MEAT-DDPM are encoded as on-scene-heatmaps to preserve spatial alignment in the latent embeddings. This allows the model to spatially correlate trajectory positions and environmental features to produce realistic trajectories that are sensible in the context of the surrounding environment. Despite this, the approach is limited in accuracy due to the nature of the image-encoded trajectories. The time and memory intensiveness of diffusion models renders them infeasible for high-resolution images, and consequently, the common practice, which we also adopted for MEAT-DDPM, is to train and execute the diffusion model in a smaller pixel space and subsequently upscale the generated images. This, however, introduces a large margin of error in the predicted trajectories, as the pixels that encode the coordinates are scaled up, effectively increasing the size of positions, so that they are no longer captured in points but rather in larger areas. Despite generating the images in lower resolution, MEAT-DDPM still suffers long inference times, which renders it infeasible for real-world applications such as autonomous vehicles.

The goal of this paper is to reduce the gap between diffusion and real-world feasible prediction models that are capable of producing accurate predictions in a timely manner. To that end, we propose a model that leverages a transformer-based architecture for powerful spatiotemporal modelling, and applies diffusion to generate trajectory coordinates directly, as opposed to the image-encoded coordinates generated by MEAT-DDPM. We also experiment with ways of accelerating the inference time through model distillation and non-Markovian diffusion processes.

In summary, our contributions are:

- We expand the sparse body of academic literature surrounding trajectory prediction with diffusion.
- We propose a novel transformer-based diffusion model, MAD-Traj, that employs environmental attention and achieves highly competitive results, using as low as 4 sampling steps.
- We show the efficacy of our model through experiments and comparisons.
- We provide the results of our experiments on inference acceleration.

2. Related Work

Trajectory prediction is broadly divided into three sub-problems:

1) **Sequence modelling**, which focuses on ways to model the temporal dynamics of trajectories,

[1]We note that the term *multi-modality* is used interchangeably in literature to refer to models that output multiple predictions [5, 14, 19, 20, 21, 22, 23, 24], and models that use multiple sources and representations of data [24, 25, 26, 27, 28, 29, 30]. In light of the lack of academic consensus [31], we use the term, in this paper, to explicitly refer to models that take input with different representations from multiple sources. We shall conversely refer to models that output multiple predictions as *multi-future* models.

and is generally concerned with individual agent trajectories.

2) **Multi-modal modelling**, which focuses on ways of enriching a model's perception of the world by including additional information from sensors, such as RGB camera, B/W camera, radar LiDAR and sound, or by estimating latent variables that model real-world phenomena such as social dynamics and the intentions of an agent.
3) **Multi-future modelling**, which focuses on ways to produce multiple feasible predictions and increase the diversity of the model output.

In the following sections, we explore recent works related to each category and identify some of the current limitations in the state-of-the-art.

1. Sequence Modelling

Trajectory data is inherently sequential in nature, and therefore, it is unsurprising that a large body of work has focused on the sequential modelling capabilities of neural networks. Recurrent Neural Networks (RNNs), and their derivatives Long Short Term Memory (LSTM) and Gated Recurrent Units (GRU) are popular model foundations since their recurrent connections allow them to remember previous states and model temporal dependencies more effectively than feed-forward networks [43]. Trajectron++ [14], SoPhie [8], SocialGAN [19] Social LSTM [44], PCC-SNET [21], CGNS [45], and Expert [46] all use recurrent networks in the form of LSTMs, GRUs or a combination. While these types of RNN are superior to feed-forward approaches when it comes to temporal modelling, they still have some major limitations. Because the hidden state has a fixed size and is updated over time to encode information about previous input, there is a practical limit on the sequence length, beyond which the models will suffer from memory loss. The sequential nature of such models also makes them notoriously slow to train and the recurrent connections can induce vanishing and exploding gradients which lead to unstable training [43].

The Transformer [47] was introduced in 2017 to solve the problems RNNs, and thanks to its liberal use of attention mechanisms, its capabilities for complex sequence modelling are outstanding in many fields [48, 49, 50, 51]. It has also become widely adopted in the field of trajectory prediction, and many works use it as a foundation for their models [15, 52, 53, 54, 55, 56]. DTO [53] employs a spatiotemporal transformer that encodes a trajectory by self-attending to the trajectory to model its temporal dynamics and subsequently attending to the temporal embeddings of neighbouring agents at a fixed time step.

AgentFormer[15] also leverages a transformer-based architecture, but rather than attending separately to the temporal and social dimensions separately, it flattens multiple trajectories in a scene across time and agents, and uses the flattened sequence as input to the transformer. This allows the model to capture not only temporal dependencies between time locations of the target agent, but also attend to the locations of the other agents at different points in time. MID [52] adopts a transformer as the noise predictor in a Denoising Diffusion Probabilistic Model [40] (DDPM) to model the temporal dynamics of the noisy trajectory and improve the predictions of noise to remove during the sampling process.

In this paper, we also base the noise predictor of our diffusion model on the transformer architecture. We use two encoders to separately encode the observed trajectory and semantic map of the environment, and use cross-attention in the decoder to leverage information from these embeddings for noise prediction during sampling.

2. Multi-modal Modelling

Since human trajectories are influenced by multiple factors, many prior works seek to improve the predictive capabilities by either enriching the models with additional sensory input or by modelling unobservable factors as latent variables. Trajectron++ [14], SocialGAN [19], Social LSTM [44], DTO [53], LB-EBM [22], CGNS [45], and MID [52] all model social dynamics through various techniques. Trajectron++ [14] employs an explicit representation in the form of a graph where a directed edge between two agent nodes indicates the influence of one agent on the other. This is encoded using an LSTM and processed by an attention mechanism to model the importance of different influences at each time step of the trajectory. Social LSTM [44] uses a separate LSTM encoder-decoder for each agent in the scene and applies social pooling to the hidden states to model interactions of nearby agents. Social ODE [57] uses a neural network to estimate the directed pairwise intensity of agent interactions, and model how the dynamics of one agent affect another agent. They also estimate the aggressiveness of individual agents, which controls the degree to which an agent adheres to social etiquette or ignores it. SCM [24] uses a combination of LiDAR and RGB camera data to improve their estimation of a latent distribution, from which variables are sampled to condition the prediction of future trajectories.

GoalGAN [9], SGNET [30], MUSE-VAE [16], GTP [58], Y-Net [59], and our own previous work MEAT-DDPM [41] all estimate one or more goals of the agent to condition the prediction of the future trajectory. These methods also incorporate environmental information in the form of a semantic map, which is a segmentation of the birds-eye-view image of the scene, often generated using a Convolutional Neural Network (CNN). As the name suggests, a semantic map assigns a semantic label to each segment of image, indicating what type of area the segment covers. The semantic labels used by Y-Net [59] are road, pavement, structure, tree and terrain. Y-Net [59], MUSE-VAE [16], and MEAT-DDPM [41] all employ a trajectory-on-scene heatmap representation to spatially align trajectories with the environment in the latent space. While effective, this approach is computationally demanding due to the large amounts of data contained in such image representations—this is especially true in the case of MEAT-DDPM, because of the repeated computations during the sampling process of the diffusion model.

In this paper, we follow previous works [16, 30, 41, 58, 59] in the adoption of a semantic map. We encode information from the map, using an approach inspired by Vision Tranformer [51], where we split the map into a sequence of patches that are flattened and used as input tokens to a transformer encoder.

3. Multi-future Modelling

When multiple predictions are desired, it does not suffice to train a discriminative model to map observations to predicted outcomes, since such models are deterministic and unable to produce more than a single unique prediction for a given input. To circumvent this, generative models are commonly trained to approximate the prior distribution of data, given an estimation of a tractable posterior that is commonly chosen to be Gaussian. Given such a model, one can sample multiple times from the posterior distribution and use the model to obtain a different sample from the approximated prior, given each of the posterior samples. The most popular type of generative models are Variational Auto-Encoders (VAE) [14, 15, 16, 17, 18, 33, 41, 60] and Generative Adversarial Networks (GAN) [8, 9, 10, 11, 12, 13, 19, 32, 61]. [19], for instance, uses an LSTM encoder-decoder network as a generator, which takes noise as input and outputs a trajectory. To condition the model on the observed trajectory, the initial hidden LSTM state is initialised as a MLP encoding of the previous coordinates. A second LSTM decoder is used as a discriminator to classify real data trajectories and generated trajectories, and its classifications are used in the training objective to increase the quality of generated trajectories. Trajectron++ [14] encodes the observed trajectory using an LSTM. Latent variables produced by a conditional VAE are then used to condition the decoder, allowing it to produce multiple trajectories.

While different in architecture and mathematical foundation, the two classes of models are both effective at producing multi-future predictions. The primary drawbacks of GAN models are the fact that their training procedure is defined as a game between the generator and discriminator networks. This both leads to unstable training and causes a problem known as *mode collapse*, where the networks converge on a Nash equilibrium, resulting in samples that lack diversity and resemble an average of the training data [34, 35, 36, 37]. Because VAE models are not subject to mode collapse, they are able to produce highly diverse samples. However, their output generally tends to be of lower quality and include undesirable and unrealistic samples [52, 62, 63], due to issues like over-regularisation caused by the training objective and *posterior collapse* where the learned latent representations become uninformative [38].

Inspired by thermodynamics, the recently proposed class of Diffusion Models [39] avoids the aforementioned issues by generating samples through a sequential denoising process, which results in stable training and diverse samples of high quality. Diffusion Models were originally proposed for image generation, a field in which they are now deemed superior and have received widespread adoption [64, 65, 66, 67, 68, 69], but they have also been applied for video generation [70, 71], text generation [72, 73, 74], audio generation [75, 76], computer vision [77, 78, 79, 80], and time series forecasting [81]. While research on the use of diffusion models for trajectory prediction is sparse, the results of prior works have thus far been promising[41, 52, 82].

In our previous work, we proposed MEAT-DDPM [41], which is a multi-modal generative model based on diffusion. It is based on the two-stage architecture of [16], and consists of a *macro-stage* with a U-Net-based VAE to sample realistic long-goals. The long-term goals are mapped to a sequence of short-term goals using a second U-Net, after which the macro-stage generates multiple trajectories for each sequence of goals, using a DDPM. While this model excels in generating realistic and diverse trajectories, its predictive capability is highly limited to the error margin induced by the image-encoded trajectories, and its U-Net-based DDPM architecture is computationally demanding and leads to excessive inference time.

MID [52] proposes a transformer-based DDPM that

performs diffusion directly in the trajectory coordinates, instead of encoding the trajectory in an image. For multi-modal learning, they adopt the encoder from Trajectron++ [14], which allows them to model social interactions among multiple agents, but we hypothesise that the results could be further improved by the addition of environmental information. Avoiding image computation in the diffusion process eliminates much of the computational footprint of the model, but while the model is faster than Meat-DDPM, the authors of MID still acknowledge its infeasibility in real-world applications due to the inference time.

A recent proposal by Liu et al. [82] involves explicitly parameterising the distribution of future trajectories and generating the distribution parameters using a deterministic diffusion process. This, in theory, allows their model to sample multiple trajectories despite only running a single diffusion process. However, this approach requires careful selection of sampling parameters due to their hybrid approach which poses a direct trade-off between precision and speed. This is exemplified by their experiments which show that this model is only able to achieve competitive results when executing a diffusion process with 100 steps 10 times and then sampling 10 trajectories from the best predicted distribution.

Inspired by Song, Meng, and Ermon [83], our model uses a non-Markovian generalisation of the traditional diffusion process, which allows us to control the stochasticity of the process. In the extreme case, the sampling process is deterministic and can be accelerated by skipping steps. The deterministic sampling process also paves the way for accelerated sampling through *progressive distillation* [84], where a diffusion model is progressively taught to imitate itself with fewer and fewer steps in the sampling process.

3. Preliminaries

1. Problem Formulation

Trajectory prediction is the task of estimating the future positions of an agent, given an observation of their previous positions. Positional information is commonly obtained from video data, and like most previous works, we assume that the input videos are preprocessed with object detection and tracking algorithms, such that spatiotemporal information is available for each agent. Formally, given a sequence of observed positions $X = \{x_1, x_2, \ldots, x_p\}$ where $x_t \in \mathbb{R}^2$ are the Euclidean coordinates of an agent at the observed time step t and p denotes the number of observed time steps, the goal is to estimate a sequence of future positions $Y = \{y_{p+1}, y_{p+2}, \ldots, y_{p+f}\}$ where $y_t \in \mathbb{R}^2$ are the Euclidean coordinates of an agent at the future time step t, and f denotes the number of future time steps.

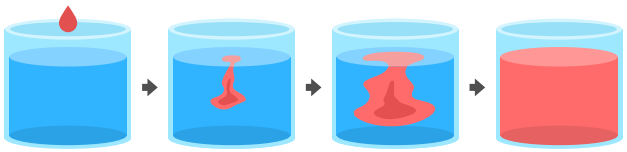


Figure 2: Adding a droplet of dye into a glass water initially results in a non-uniform distribution of water particles and dye particles. The second law of thermodynamics asserts that such heterogeneous systems undergo a natural, irreversible process of diffusion that destroys order and results in a homogeneous system [85].

2. Diffusion Models

Motivated by the trade-off between tractability and flexibility in probabilistic models, Sohl-Dickstein et al. [39] proposed the concept of diffusion models in 2015 as a way to define a probabilistic model that is highly flexible in model structure and distribution coverage while simultaneously being computationally tractable. This class of generative model is inspired by thermodynamics and mimics the way a non-uniform distribution of particles gradually diffuses over time until it becomes a uniform distribution (see Fig. 2). The intuition is to define a diffusion process to gradually destroy the structure in the training data, and then train a neural network to reverse this process. Defining the diffusion process, such that its outcome resembles a simple and tractable distribution, makes it straightforward to generate new data by sampling random noise from the distribution and executing the reversed diffusion process. The most widely employed diffusion model is the Denoising Diffusion Probabilistic Model (DDPM) proposed by Ho, Jain, and Abbeel [40], which makes a few simplifications to the original proposal in order to improve sample quality and simplify training. The authors defined the diffusion process as a Markov process with T transitions (see Fig. 3a) with a transition kernel corresponding to a Gaussian perturbation:

$$q(\mathbf{x}_1, \ldots, \mathbf{x}_T \mid \mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t \mid \mathbf{x}_{t-1}). \quad (1)$$

$$q(\mathbf{x}_t \mid \mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t;\ \sqrt{1-\beta_t}\mathbf{x}_{t-1},\ \beta_t \mathbf{I}). \quad (2)$$

Here, $\beta_t \in [0, 1]$ is a hyperparameter that determines the amount of noise added at each step. The value of β_t is controlled by a predefined variance schedule that

assigns increasing values such that data retains more fidelity early in the process. This ensures the reverse process is able to recover structure early, which it can then refine in smaller increments.

Because the product of multiple Gaussians is itself Gaussian, the use of Gaussian perturbation as the transition kernel enables a clever exploitation that simplifies the training process substantially. The presence of the Markov property means that intermediate diffusion steps can be factored using the chain rule, which allows for closed form computation of samples at arbitrary points in the diffusion process:

$$q(\mathbf{x}_t \mid \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t;\ \sqrt{\bar{\alpha}_t}\mathbf{x}_0,\ (1-\bar{\alpha}_t)\mathbf{I}). \quad (3)$$

Here, $\alpha_t = 1 - \beta_t$ is the inverse of β_t and $\bar{\alpha}_t = \prod_{s=0}^{t} \alpha_s$ is the cumulative product of α. Reparameterising the expression with $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, a neural network, ϵ_θ, can be trained to predict the noise to remove at each step in the reverse diffusion process by sampling random time steps t, computing

$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, \quad (4)$$

and updating the network parameters based on a simple mean squared error (MSE) loss:

$$L_t = \mathbb{E}_{t\sim[1,T],\mathbf{x}_0,\epsilon_t}\left[\|\epsilon_t - \epsilon_\theta(\mathbf{x}_t,\ t))\|^2\right]. \quad (5)$$

Using the neural network, ϵ_θ, a new sample can then be generated by sampling $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and successively computing each step of the denoising process:

$$\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(\mathbf{x}_t, t)) + \sigma_t\mathbf{z}, \quad (6)$$

where $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ is randomly sampled Gaussian noise and $\sigma^2 = \beta_t$ is variable determined by the variance schedule, which schedules how much noise is added to the sample between each reverse diffusion step.

Song et al. [83] recently proposed a generalisation of the DDPM, redefining the diffusion process to a non-Markovian process, where each state depends not only on the previous state but also on the initial state:

$$q(\mathbf{x}_1, \ldots, \mathbf{x}_T \mid \mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{x}_0). \quad (7)$$

$$q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mu_\theta(\mathbf{x}_t, \mathbf{x}_0), \sigma_t^2\mathbf{I}). \quad (8)$$

$$\begin{aligned}\mu_\theta(\mathbf{x}_t, \mathbf{x}_0) &= \sqrt{\bar{\alpha}_{t-1}}\mathbf{x}_0 \\ &+ \sqrt{1-\bar{\alpha}_{t-1}-\sigma_t^2} \\ &\cdot \frac{\mathbf{x}_t - \sqrt{\bar{\alpha}_t}\mathbf{x}_0}{\sqrt{1-\bar{\alpha}_t}}.\end{aligned} \quad (9)$$

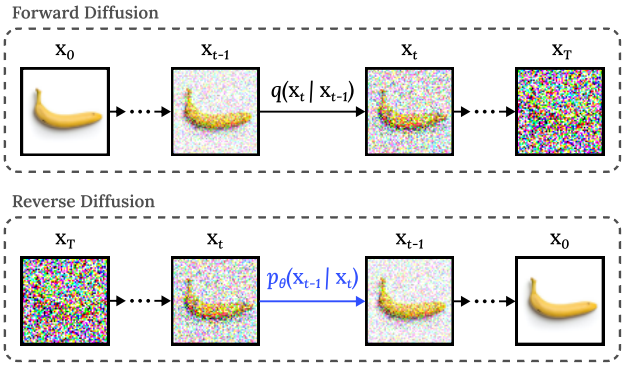
At each step t of the reversed diffusion process, one can use the predicted noise $\epsilon_\theta(\mathbf{x}_t, t)$ to estimate the initial state $\mathbf{x}_0$ given the current state $\mathbf{x}_t$, which in turn can be used to obtain $\mathbf{x}_{t-1}$:

$$\begin{aligned}\mathbf{x}_{t-1} = \sqrt{\bar{\alpha}_{t-1}} \cdot \underbrace{\left(\frac{x_t - \sqrt{1-\bar{\alpha}_t} \cdot \epsilon_\theta(x_t, t)}{\sqrt{\bar{\alpha}_t}}\right)}_{\text{estimated } \mathbf{x}_0} \\ + \underbrace{\sqrt{1-\bar{\alpha}_{t-1}-\sigma_t^2} \cdot \epsilon_\theta(x_t, t)}_{\text{direction pointing to } \mathbf{x}_t} + \underbrace{\sigma_t\mathbf{z}}_{\text{random noise}}\end{aligned} \quad (10)$$

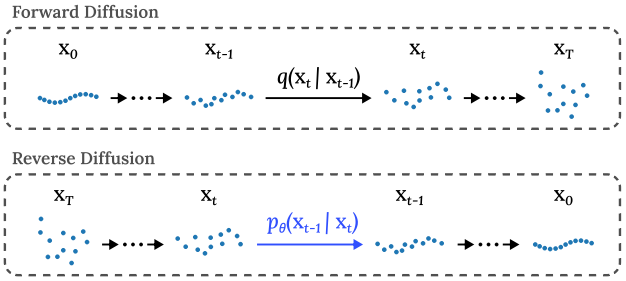
If $\sigma_t = \sqrt{\frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}} \cdot \sqrt{\frac{1-\bar{\alpha}_t}{\bar{\alpha}_{t-1}}}$ is chosen for all t, the diffusion process becomes Markovian and identical to that of DDPM. However, when $\sigma_t = 0$ for all t, the diffusion process becomes deterministic and the resulting model is a Denoising Diffusion Implicit Model (DDIM). Due to the non-Markovian nature of the forward process, the DDIM sampling procedure can be accelerated by using only a subset of the time steps used for training, resulting in a much smaller trade-off between speed and sample quality compared to DDPMs. DDIMs are trained using the same denoising loss as DDPMs (Eq. 5), which depends only on the marginal probabilities of the diffusion process and not the joint probability of each diffusion step. Consequently, noise-predicting neural networks trained under either framework are interchangeable, which allows the use of the DDIM sampling procedure to accelerate existing DDPMs models without retraining.

3. Progressive Distillation

To combat the challenge regarding the slow inference speed of diffusion models, Salimans and Ho [84] suggest applying *Progressive Distillation*. The goal is to reduce the number of diffusion steps required during inference to produce a sample of sufficient quality. To reduce the number of diffusion steps, the progressive distillation algorithm iteratively trains a series of diffusion models and halves the number of required sampling steps with each model. Progressive distillation employs a student-teacher paradigm, and starts off with a trained model referred to as the *teacher*, and a copy of said teacher referred to as the *student*. The differentiating factor is that the student is



(a) Diffusion process for image generation.



(b) Diffusion process for trajectory prediction.

Figure 3: Illustration of the forward and reverse diffusion process of **(a)** a model for image generation and **(b)** a model for trajectory prediction.

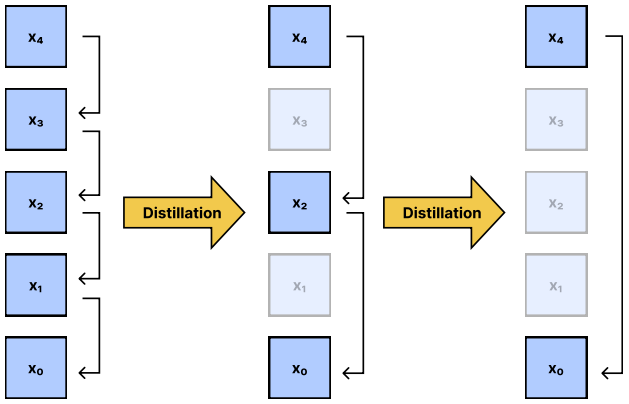


Figure 4: Illustration of progressive distillation, where the initial model (teacher) starts at $\mathbf{x}_4$ and takes one denoising step at a time until it arrives at the clean sample $\mathbf{x}_0$. The teacher is distilled into a new model (student), which skips the steps $\mathbf{x}_3$ and $\mathbf{x}_2$, requiring only two steps to produce the clean sample $\mathbf{x}_0$. The student is then distilled into a new student that skips steps $\mathbf{x}_3$, $\mathbf{x}_2$, and $\mathbf{x}_4$, requiring only a single step to produce $\mathbf{x}_0$.

only allowed half the number of diffusion steps compared to the teacher. The training proceeds by having the teacher predict two successive reverse diffusion steps, which the student then aims the predict in a single step, as shown in Fig. 4. The training objective aims to guide the prediction of the student towards the prediction of the teacher using mean squared error as loss, ideally training the student to produce the same prediction in half the steps. Once the loss of the student has converged, the student becomes the teacher for the next iteration of the distillation process, and the cycle repeats until the student is no longer able to approximate the results of the teacher accurately. At this point, through the guidance of its preceding teachers, the distilled model has been reduced to the smallest number of diffusion steps with which the results of the original model can be sufficiently approximated.

4. Transformers

To combat the limitations of RNNs, Vaswani et al. [47] proposed the Transformer architecture, which is a variant of the common encoder-decoder architecture. The encoder encodes a full sequence, e.g., a sentence in a source language. Given an intermediate translated target sentence and the encoded source sentence, the decoder predicts the next token in the target sequence. Each token in a sequence is shaped into vectors of size d_{model}, and the standard size, proposed by Vaswani et al. [47] is $d_{\text{model}} = 512$. Both the encoder and decoder consist of N stacked identical layers. Each encoder layer has two sub-layers; a multi-headed self-attention layer and a feed-forward layer, both with residual connections around them. The decoder contains a multi-head attention layer, followed by a multi-head cross-attention layer, allowing it to apply attention to important relationships between the encoded source sequence, and the target sequence that is being translated. The last layers of the decoder are a feed-forward layer and a linear layer, which maps the output of the decoder into the desired dimensions, e.g., the shape of the available token list where each cell describes the likelihood of said token being the next. For both the input to the encoder and decoder, an added sinusoidal embedding allows the model to understand the sequential nature of the input sequence.

Attention is the foundation of the Transformer architecture and is a mechanism inspired by retrieval systems, where a mapping is made from key-value pairs and a query to the output. The general idea is to learn the dependencies and importance between tokens internally in a single sequence or externally between sequences. They call their specific version *Scaled Dot-Product Attention*, which is formulated as shown in Eq. 11. Here the query Q, key K, and value V are learned matrix representations of sequences

with each column being the representation of a token.

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}}) \quad (11)$$

As an example, to perform self-attention, one would parse the matrix representation of the sequence itself as both Q, K, and V, whereas to perform cross attention between two sequences e.g., for translation, the target sequence would be the query Q, and the key-value pairs K and V would be the source sequences. [47] extend the Scaled Dot-Product Attention into *Multi-Head Attention*, where instead of conducting attention on the d_{model} sized keys, values, and queries, they split the original matrix into h multiple smaller matrices, which are then concatenated, as shown in Eq. 13, where projection matrices W_i^Q, W_i^K, W_i^V, and W_O are parameter matrices.

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O, \quad (12)$$

where

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \quad (13)$$

This allows for each attention head to learn a different relation between various tokens, giving the model the ability to understand complex relations between tokens in a sequence.

4. Methodology

This section introduces our proposed model, MAD-Traj; a generative multi-modal trajectory predictor that leverages a transformer-based architecture and reverse diffusion process for spatiotemporal and generative modelling. Fig. 5 shows an overview of our proposal. Our model consists of a history encoder, an environment encoder, and a denoising decoder that is used in the reverse diffusion process. Given an observed trajectory and a semantic map, describing the environment, our model generates a trajectory over a number of diffusion steps. To accelerate the sampling process, we experiment with DDIM sampling [83] and progressive distillation [84], both of which are techniques for decreasing the number of diffusion steps required to generate a sample.

Fig. 6 shows a detailed overview of our model architecture and its components. The history encoder (Sec. 4.1) leverages the power of self-attention [47] to encode temporal dependencies between individual trajectory time steps. The environment encoder operates on a sequence of image patches and uses self-attention to encode spatial relationships between different areas of the map. Both of these encodings are used to condition by the denoising decoder (Sec. 4.3), which utilises cross-attention to model the temporal and spatial relationships between the observed trajectory, the environment, and the noisy future trajectory. This allows the decoder to leverage the knowledge of these relationships and learn which historical time steps and areas of the environment require the most attention during the reverse diffusion process.

1. History Encoder

The history encoder (see Fig. 6) takes the coordinates of the observed trajectory as input, from which the first and second derivatives are computed, yielding the velocities and accelerations in both the x and y dimensions, respectively. This results in a six-dimensional vector for each point, each of which is upscaled to the desired dimensions of the transformer model with a 2-layer feed-forward MLP. To preserve the notion of sequential order in the trajectory when it is processed by the transformer encoder, each point is encoded using a learned sinusoidal embedding. The upscaled and positionally embedded vector is fed into a multilayer transformer encoder, where each layer consists of a self-attention mechanism, and a feed-forward MLP, both of which have residual connections around them, feeding into addition and normalisation layers. The output of the history encoder of this is used as memory input to the Denoising Decoder.

2. Environment Encoder

The map encoder (see Fig. 6) encodes a semantic map, which describes the navigability of the various areas surrounding the agent. The semantics maps are generated from RGB images of the scenes in the dataset, by segmenting the images and assigning a class label to each pixel, denoting the type of segment that pixel belongs to. Given a semantic map of the entire scene, we cut a local map around the last known position of the agent, with the size of the local map determined by statistics of the dataset, to ensure map is able to sufficiently contain the entire trajectory. For each trajectory, we compute the distance from the last known position to all other positions in the same trajectory and set the width and height of all local maps to 3σ, where σ is the standard deviation of the computed distances over the entire dataset.

The Environment Encoder is inspired by Vision Transformers [51] (ViT) which utilises a transformer architecture for image classification by constructing a sequence of patches from the input image and flattening each patch into a vector. By using the transformer architecture, the ViT model learns to attend to the most important parts of an image through the power of self-attention, and we argue that this is a desirable

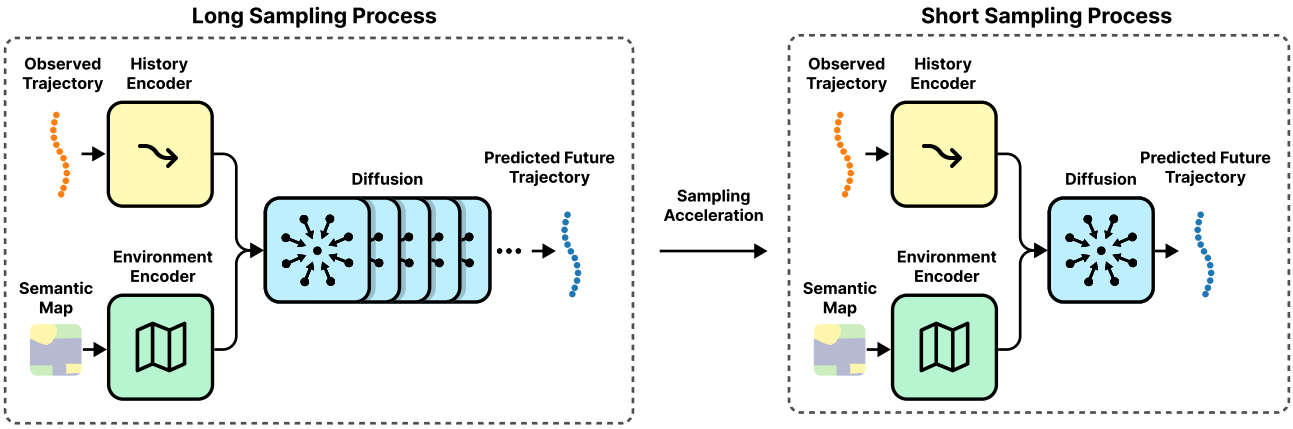


Figure 5: High-level overview of our proposal. Our model consists of three main components: A history encoder and an environment encoder to encode the observed trajectory and environment information, respectively, and a diffusion process to generate the predicted future trajectory. The diffusion process is a bottleneck in terms of inference time and to combat this, we apply acceleration techniques in the form of DDIM sampling and progressive distillation to decrease the inference time of our model.

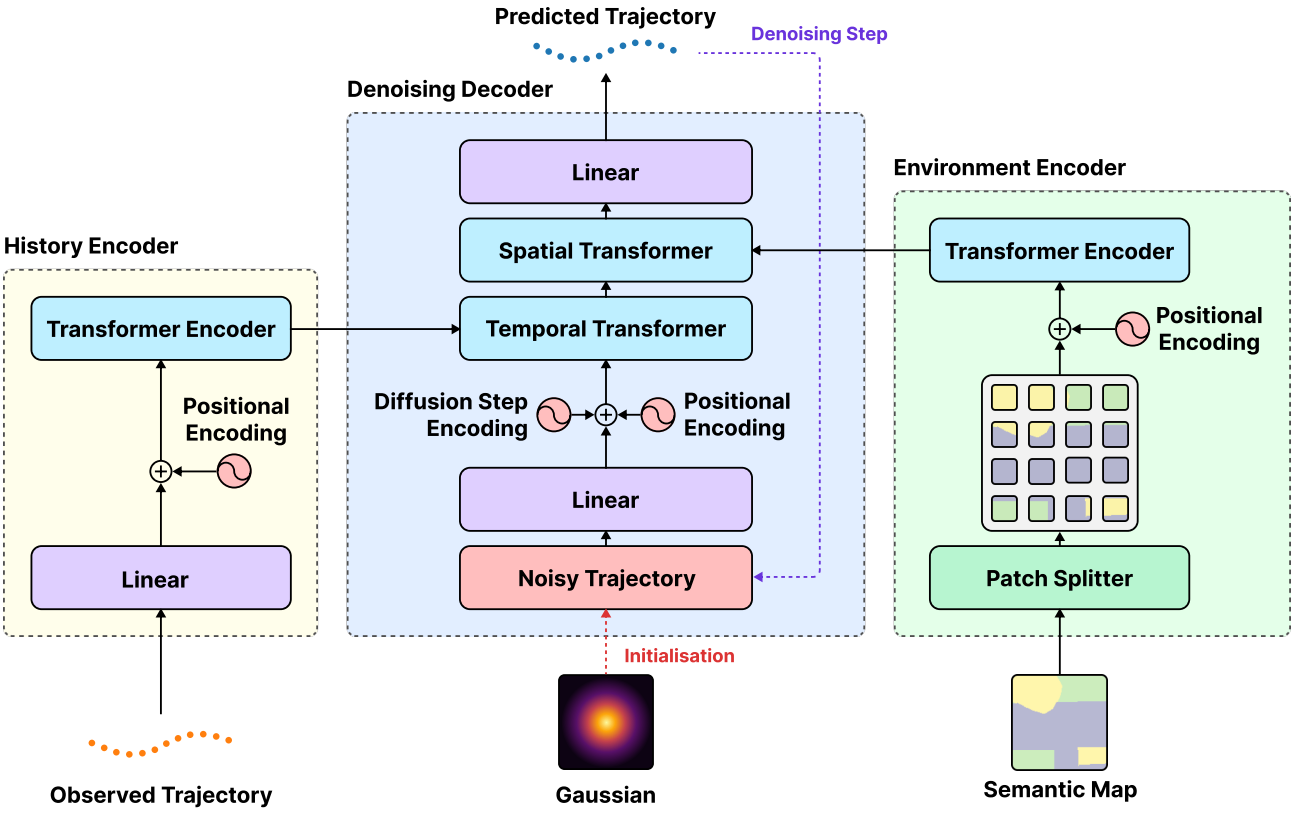


Figure 6: Our proposed framework consists of three modules. The History Encoder takes the past points, and makes an encoding which can be fed into the Denoising Decoder using an attention mechanism. The Environment Encoder encodes the semantic map into a sequential format, which also is parsed into the Denoising Deocder using an attention mechanism. Lastly, the Denoising Decoder runs the reverse diffusion process, where the first step is initialised with Gaussian Noise (red dashed line), and for each following reverse diffusion step, is parsed the output of the previous iteration (purple dashed line).

property for environment modelling in the context of trajectory prediction. We adapt this approach by dividing the local map surrounding the agent into 64 patches, flattening each patch, and adding a learned sinusoidal embedding to allow the model to understand the positional relation between the various patches. We use a single-dimensional embedding as described by Dosovitskiy et al. [51], which found that there is no performance difference between one or two-dimensional embeddings. These patches are fed into a Transformer Encoded similar to the one described in Sec. 4.1. The output of the environment encoder is used for environmental attention in the Denoising Decoder.

3. Denoising Decoder

During the sampling process (reverse diffusion), the denoising decoder acts as the noise predictor that is responsible for determining the noise to remove at each step of the process. We model each point of the future trajectory as a 2-dimensional vector, containing only the x and y velocities, unlike the 6-dimensional vector in the History Encoder. Given the predicted velocities, we determine the trajectory by integrating the velocities with respect to the trajectory time steps. We found this to result in more realistic accurate trajectories, compared to predicting the coordinates directly. For each iteration in the diffusion process, a 2-layer Feed-Forward MLP to upscale the vectors to the internal dimensions of the transformer. To model both the sequential nature of the trajectory and the diffusion process, despite the non-sequential processing of the transformer, each trajectory encodes both its own sequence position as well as the current diffusion step using learned sinusoidal embeddings.

The decoder consists of two main modules, the first of which is a temporal transformer that follows the standard transformer decoder architecture. First, it applies self-attention to the noisy input trajectory and then proceeds to use the output of the history encoder for cross-attention with the noisy input trajectory. The second module is a spatial transformer that cross-attends to the output of the temporal transformer and the environment encoder. The output of the denoising decoder is a noise tensor with equal dimensions to the trajectory, which can subsequently be subtracted from the trajectory to produce a less noisy trajectory for the next iteration of the sampling process.

4. Training and Inference

Our model is trained end-to-end as shown in Alg. 1, by retrieving a historical and future trajectory $X, Y_0 \sim \mathcal{T}$ together with its corresponding semantic map $M \sim \mathcal{M}$. We then sample a random time step $t \sim Uniform(\{1, \dots, T\}$ and add noise $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ to the future trajectory Y_0, according to the variance schedule, which yields the noisy trajectory Y_t. We then predict the noise added to Y_t using the Denoising Decoder $\mathcal{D}_\theta$, which takes a Y_t, t, the history encoding $z_{hist} = \mathcal{E}^{hist}_\phi(X)$ and environment encoding $z_{env} = \mathcal{E}^{env}_\psi(M)$ as input. Finally, the gradient descent step is taken using MSE loss between the noise ϵ and the predicted noise $\hat{\epsilon}$. We repeat this process until the model is converged.

Algorithm 1 Training

Require: Trajectory dataset $\mathcal{T}$
Require: Map dataset $\mathcal{M}$
Require: History encoder $\mathcal{E}^{hist}_\phi$
Require: Environment encoder $\mathcal{E}^{env}_\psi$
Require: Denoising decoder $\mathcal{D}_\theta$

repeat
  $X, Y_0 \sim \mathcal{T}$
  $M \sim \mathcal{M}$
  $t \sim$ Uniform({1, …, T})
  $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  $Y_t \leftarrow \sqrt{\alpha_t} \cdot Y_0 + \sqrt{1-\alpha} \cdot \epsilon$ ▷ Noisy future
  $z_{hist} \leftarrow \mathcal{E}^{hist}_\phi(X)$
  $z_{env} \leftarrow \mathcal{E}^{env}_\psi(M)$
  $\hat{\epsilon} \leftarrow \mathcal{D}_\theta(Y_t, z_{hist}, z_{env}, t)$ ▷ Predicted noise
  Take gradient descent step on
    $\nabla_{\theta,\phi,\psi} \| \epsilon$ - $\hat{\epsilon} \|_2^2$
until converged

During inference, we use the DDIM update rule described by Song, Meng, and Ermon [83], as shown in Alg. 2. This algorithm describes the generation of K trajectory predictions, while only encoding the historical trajectory $z_{hist} = \mathcal{E}^{hist}_\phi(X)$ and $z_{env} = \mathcal{E}^{env}_\psi(M)$ once since the observed trajectory and semantic map remain constant for the entire sampling process. We accumulate the predicted trajectories in the set $\tilde{Y}$ until it reaches a size of K. For each desired prediction in $[1, \dots, K]$, we sample random noise from a standard normal distribution and run the reverse diffusion process for each t in T. We predict the noise ϵ_t to remove at each reverse diffusion step, and using the DDIM[83] update rule, we compute the denoised trajectory $\hat{Y}_{t-1}$ given $\hat{Y}_t$ and $\hat{\epsilon}$. When $t = 0$, the reverse diffusion process terminates and $\hat{Y}_t$ becomes the final prediction of Y_0. We then add $\hat{Y}_t$ to the collection of predictions $\tilde{Y}$ and run the same process for the next sample in K.

5. Experiments

To assess the efficacy of our diffusion model, we conduct extensive experiments and present the results

Algorithm 2 Multi-sampling (inference)

Require: Trajectory dataset $\mathcal{T}$
Require: Map dataset $\mathcal{M}$
Require: Trained history encoder $\mathcal{E}_\phi^{hist}$
Require: Trained environment encoder $\mathcal{E}_\psi^{env}$
Require: Trained denoising decoder $\mathcal{D}_\theta$

$X \sim \mathcal{T}$
$M \sim \mathcal{M}$
$z_{hist} \leftarrow \mathcal{E}_\phi^{hist}(X)$
$z_{env} \leftarrow \mathcal{E}_\psi^{env}(M)$
$\tilde{Y} \leftarrow \emptyset$ ▷ Initialise list of trajectories
for $1, \ldots, K$ **do**
  $\hat{Y}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ ▷ Initialise noisy trajectory
  for $t \leftarrow T, \ldots, 1$ **do**
    $\hat{\epsilon}_t \leftarrow \mathcal{D}_\theta(Y_t, z_{hist}, z_{env}, t)$
    $\hat{Y}_{t-1} \leftarrow \sqrt{\alpha_{t-1}}\left(\frac{\hat{Y}_t - \sqrt{1-\alpha_t}\cdot\hat{\epsilon}_t}{\sqrt{\alpha}}\right) + \sqrt{1-\alpha_{t-1}}\cdot\hat{\epsilon}_t$
    $\hat{Y}_t \leftarrow \hat{Y}_{t-1}$
  end for
  $\tilde{Y} \leftarrow \tilde{Y} \cup \hat{Y}_t$ ▷ Include clean trajectory
end for

in the following section. Sec. 5.1 and Sec. 5.4 respectively outline the dataset and define the evaluation metrics used in our experiments. Sec. 5.5 provides an overview of the baselines we use for benchmark comparison, and Sec. 5.6 presents the quantitative experiment results. Sec. 5.7 showcases examples of trajectories generated by our model, and Sec. 5.8 provides an ablation study to assess the impact of the environment encoder on the performance of the model. Finally, to accelerate the inference time of our model, we conduct an experiment with multiple variants of our model using progressive distillation and DDIM sampling, and present the results in Sec. 5.9 and Sec. 5.10.

1. Datasets

The **Stanford Drone Dataset (SDD)** [86] is a real-world dataset, which aims to provide a base for the modelling of human etiquette and interactions, i.e., yielding, social etiquette, personal distance, and how agents interact with the environment. The dataset is used extensively in literature to benchmark and compare models for trajectory prediction. The annotated data of SDD consist of timestamped positions of agents, obtained by tracking birds-eye drone footage from the Stanford University campus. It is comprised of more than 100 different static scenes and contains a total of almost 20,000 targets including ∼11,200 pedestrians, ∼6,400 cyclists, and ∼1,300 cars.

The **ETH/UCY** [87, 88] datasets consist of manually annotated real-world trajectory data obtained from birds-eye video of various locations around the campuses of The University of Cyprus and ETH Zürich. They collectively contain 5 sets of data from 4 unique scenes with ∼1500 unique pedestrians. Like SDD, ETH/UCY is one of the most commonly used benchmarks in the field.

2. Implementation Details

For SDD, we utilise the preprocessed data and semantic maps made available by Mangalam et al. [59], where 30 scenes are used for training, and 17 are used for testing. Coordinates are sampled at 2.5Hz and we use 8 time steps for observation and 12 time steps for prediction. We filter outliers in the training data by removing samples with a higher velocity than 5 times the standard deviation, due to unrealistic movement or camera issues. The remaining samples are augmented by rotating each sample trajectory 90 degrees three times to produce four times the data. The testing data is unfiltered and unmodified. The SDD semantic maps contain 5 different classes: pavement, road, structure, terrain, and tree. We pad the map with a sixth value, due to some samples being close to the edges, and we want to be able to create local maps for these samples. The maps are then scaled to fit in the range [-1, 1].

We also adopt the preprocessed ETH/UCY data, semantic maps and related homography matrices from Mangalam et al. [59]. Similar to SDD, the data samples of the preprocessed ETH/UCY datasets have a sample frequency 2.5Hz and contain 8 historical trajectory positions and 12 future trajectory positions. Following prior works [52, 59], we adopt a leave-one-out approach, where an entire scene is used exclusively for testing, while the remaining scenes are used for training. We also augment the training data for ETH/UCY, but do not filter outliers in the training data. The semantic maps for ETH/UCY are binary, with the only class labels being road and non-road due to the simplicity of the scenes.

3. Hyperparameters

The transformers modules of our model use three layers with four attention heads and an internal dimension of 256. Of the four configurations we tested (Tab. 1, we found this combination produced the best performance. The model is trained with 64 diffusion steps, as we found no noticeable increase in accuracy when increasing the number of steps above this. Despite the common practice of using 1,000 steps for image generation, our observation in this regard is in line with MID [52], who are also able to use a relatively short diffusion process of 100 steps. We hypothesise that this is possible due to the smaller size

Table 1: Results on SDD of four unique combinations of Transformer hyperparameters, labeled L, M, S and XS. The error metrics are measured in pixels. Time (s) is measured using a Nvidia 3080 on a single batch of size 512 samples. * The L model could likely have been optimised further, but due to inference time, we did not find this model to viable for further work.

Hyperparameters									
Model	**Dimension**	**Layers**	**Attention Heads**	**ADE (p)**	**FDE (p)**	**ADE (%)**	**FDE (%)**	**Time (s)**	**Time (%)**
*L	512	6	8	7.08	12.45	101.2	98.65	91.5	502.7
M	256	3	4	6.96	12.62	100.0	100.0	18.2	100.0
S	128	2	4	7.43	12.30	106.8	95.5	6.6	36.3
XS	64	1	2	9.61	14.66	138.1	116.2	3.6	19.9

and lower complexity of trajectory data compared to image data. The local semantic maps are downscaled to 128×128, and split into 64 patches of shape 16×16, which, when flattened, results in vectors of size 256. We use a linear schedule, having found that other schedules such as cosine and quadratic result in inferior performance. We also tried training our model using the SNR loss and truncated SNR loss proposed by Salimans and Ho [84] but ultimately found the MSE loss proposed by Ho, Jain, and Abbeel [40] to yield models capable of producing more realistic trajectories. Lastly, we also tried predicting x_0 and $v = \alpha_t \epsilon - \sigma_t \mathbf{x}$ as proposed by Salimans and Ho [84], but found that predicting ϵ, as proposed by Ho, Jain, and Abbeel [40], yielded more accurate models. We acknowledge the modesty of this experiment and defer a more extensive hyperparameter optimisation for future work.

4. Evaluation Metrics

We evaluate the predictive capabilities of our model by comparing the generated predictions with ground-truth trajectories from the dataset. This is done using the Average Displacement Error (ADE) and Final Displacement Error (FDE) as error metrics. These metrics are widely used in the field [14, 15, 16, 19, 44, 52, 59, 89, 90, 91], and allow for easy benchmark comparisons with other previous and future works. As our model is generative and capable of producing multiple predictions for the same observation, we follow the widely adopted approach of sampling k predictions per observation and reporting the best-of-K error metrics averaged over the entire test set. [8, 14, 16, 19, 41, 46, 52, 54, 59, 91] We denote this as minADE and minFDE, respectively and follow the definitions given in [41].

Minimum Average Displacement Error (minADE) is the smallest ADE over k predictions. Formally, it is defined as

$$minADE(\tilde{Y}, Y) = \min_{i=1}^{K} ADE(\hat{Y}_i, Y) \tag{14}$$

where $\tilde{Y} = [\hat{Y}_1, \hat{Y}_2 \ldots, \hat{Y}_K]$ is a set of predicted future trajectories and Y is the ground truth future trajectory. The ADE of a predicted future trajectory (see Fig. 7) is the average Euclidean (L2) distance between its coordinates and the coordinates of the ground truth future trajectory. Formally, this is defined as

$$ADE(\hat{Y}, Y) = \frac{1}{f} \sum_{t=t_{p+1}}^{t_{p+f}} \sqrt{(\hat{y}_{t,0} - y_{t,0})^2 + (\hat{y}_{t,1} - y_{t,1})^2}, \tag{15}$$

where $\hat{Y}$ is the predicted future trajectory, Y is the ground truth future trajectory, and the number of past and future trajectory time steps are denoted by p and f, respectively. The x and y coordinates of trajectory y at time step t are denoted $y_{t,0}$ and $y_{t,1}$, respectively.

Minimum Final Displacement Error (minFDE) is the smallest FDE over K predictions. Formally,

$$minFDE(\tilde{Y}, Y) = \min_{i=1}^{K} FDE(\hat{Y}_i, Y), \tag{16}$$

where $\tilde{Y} = [\hat{Y}_1, \hat{Y}_2 \ldots, \hat{Y}_K]$ is a set of predicted future trajectories, Y is the ground truth future trajectory. The FDE of a predicted future trajectory (see Fig. 7) is the Euclidean (L2) distance between its coordinates at the final time step and the coordinates of the ground truth future trajectory at the final time step. This is defined formally as

$$FDE(\hat{Y}, Y) = \sqrt{(\hat{y}_{T,0} - y_{T,0})^2 + (\hat{y}_{T,1} - y_{T,1})^2}, \tag{17}$$

where $\hat{Y}$ is the predicted future trajectory and Y is the ground truth future trajectory. The x and y coordinates of trajectory y at the final time step T

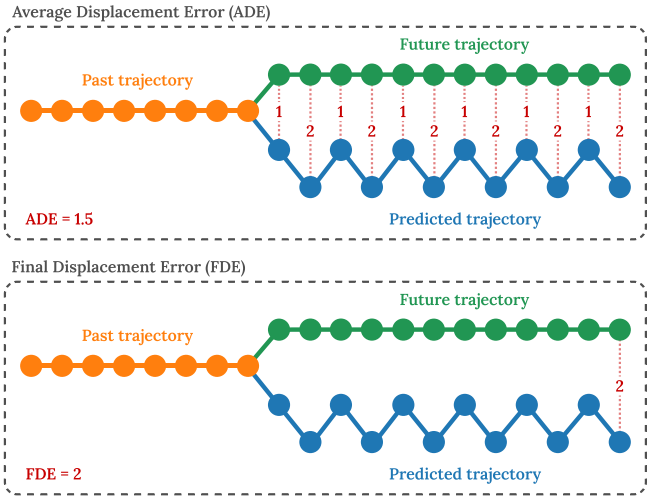


Figure 7: Illustrative example of ADE and FDE metrics. FDE measures the distance between the final pair of points in the two trajectories, which, in this example, is 2. ADE measures the average distance between each pair of points in the two trajectories, which, in this example, amounts to $\frac{6\times1+6\times2}{12} = 1.5$.

are respectively denoted $y_{T,0}$ and $y_{T,1}$.

5. Baseline Models

We compare our method to the current state-of-the-art, using MID [52], Social-GAN [19], PECNet [20], Trajectron++ [14], LB-EBM [22], PCCSNET [21], Expert [46], Y-NET [59], CGNS [45], SimAug [92], STAR [54], Social-BiGAT [93], and MG-GAN [94] as baselines. This section provides a brief overview of the different approaches.

MID [52] uses the encoder from Trajectron++ [14] to represent past agent trajectories and social dynamics. Similar to our model, MID also employs a transformer decoder in a denoising diffusion process to generate trajectories directly from Gaussian noise. However, their approach differs from ours in that it is based on the older DDPM model and does not utilise environmental information.

Social-GAN [19] is a multi-agent prediction model, using the GAN framework. The generator consists of stacked LSTM encoder-decoders with a pooling module to summarise, for each person, the social dynamics across the encoder stack, which is then used as contextual information for the decoder stack. The discriminator is a stack of separate LSTM encoders, followed by an MLP to map the hidden state to probability score.

PECNet [20] is a generative multi-agent model based on the VAE framework. The first module in the two-step approach encodes multiple past trajectories, which, together with a sampled Gaussian vector, are used by the VAE decoder to produce end goals for each agent. Conditioned on the encoded end goals, a Social Pooling module is then applied to each of the past trajectories, and the pooled latent vectors are finally decoded into future trajectories.

Trajectron++ [14] is a graph-structured recurrent model where scenes are represented as spatiotemporal graphs that model the past trajectories and social dynamics of agents. It uses LSTMs to encode the historical graph states in a hidden state, which is then decoded using a gated recurrent unit (GRU) to produce the predicted agent trajectory.

LB-EBM [22] learns an Energy Based Model (EBM), from which it samples latent variables conditioned on a social pooled encoding of the observed trajectory. The sampled latent is used to condition a planning module that predicts a sequence of intermediate steps, which in turn condition the prediction module to produce the predicted trajectory. The model has no recurrent parts, and aside from the attention-based pooling mechanism, all modules consist of MLPs.

PCCSNET [21] encourages diversity in its predictions by explicitly modelling the modality of its output. It does this by clustering trajectories in the training data that are similar in terms of positions and directions. The clusters are used as pseudo-labels to train a modality classifier that takes an LSTM encoding of the observed trajectory as input. During evaluation, the k modalities with the highest probability assignment are used to condition an LSTM decoder that synthesises a future trajectory for each modality.

Expert [46] leverages a repository of previously seen samples, from which the trajectory most similar to the one observed is retrieved. This is used to produce a goal candidate based on the coordinates at the time step matching the final time step of the trajectory to be predicted. The goal and observed trajectory are encoded together using an LSTM, after which attention is applied between this hidden state and the hidden state of nearby agents. Another LSTM then decodes this to produce a bi-variate Gaussian distribution for each future time step, and the future coordinates are sampled from these distributions.

Y-NET [59] uses a semantic map of the scene, similar to our approach, and encodes the past trajectory as a heatmap on top of the semantic map to retain spatial alignment between the signals in the trajectory and the environment. It uses a U-Net [42] encoder to produce a latent representation, which is then decoded through two separate U-Net decoders. Similar to PECNet, the first decoder produces a goal distribution, from which

long-term and intermediate goals are sampled and used as contextual information in the second decoder.

CGNS [45] uses a deep feature extractor, that encodes environmental features with CNNs and social dynamics through a CNN with pooling and attention mechanisms. Together with the observed trajectories, these features are encoded through a GRU, and used as input to a VAE encoder, to produce a latent distribution that approximates the posterior. CGNS employs adversarial learning, similar to the GAN framework, but instead of noise from a standard normal distribution, however, the GRU based generator network takes a latent variable, sampled from the learned distribution.

SimAug [92] uses a 3D simulator to generate training data with multiple camera angles (views) for each sample. During training, the hardest view is selected for each sample based on classification loss, and its features are mixed with those of the original view. Predictions are made with the Multiverse [95] model, which consists of a ConvRNN encoder followed by two ConvRNN decoders for coarse and fine predictions.

STAR [54] encodes observed trajectories of multiple agents using two stacked encoder blocks in succession. The first block consists of two parallel transformers to model the temporal and spatial features independently. The second block pipes the transformers, to model the relationship between the temporal and spatial features. To capture information about surrounding agents, the stacked encoders use a novel attention-based graph convolution for message passing between the spatial transformers. To make multiple predictions, STAR adds Gaussian noise to the embedding before decoding them with an MLP and repeats this for each desired prediction.

Social-BiGAT [93] is based on BicycleGAN [96], which is a framework for training a generative model using a combination of GAN, VAE, and a latent regression. It follows an encoder-decoder structure, where environment information is encoded with a CNN and trajectories are encoded with a stack of LSTMs, the hidden states of which a combined using Graph Attention [97]. LSTM decoders are conditioned on a latent variable sampled from a Gaussian, whose parameters are estimated by a VAE encoder.

MG-GAN [94] encodes an image of the environment with a CNN and future processed with an attention block. The observed trajectories are encoded with LSTMs and jointly processed by an attention block to model social dynamics. A series of GAN generators are trained to model different trajectory distributions given the observed trajectory. During inference, the output of the encoders is mapped to a probability distribution over generator indices, which is used to choose the appropriate generator to produce the trajectory.

Table 2: Quantitative evaluation results for the Stanford Drone Dataset (SDD), where sampling is performed $K = 20$ times. *H* means that a model only utilises information from the historic trajectory, while *H+I* indicates that a model is utilising both the historical trajectory and image data from the surrounding environment. The errors are measured in pixels, and we report the best-of-k metrics (minADE and minFDE). The best results are **bold** and the second-best results are underlined.

Results: Stanford Drone Dataset				
Model	**Input**	***K***	**ADE**	**FDE**
MID [52]	H	20	7.61	14.30
Social-GAN [19]			27.23	41.44
PECNet [20]			9.96	15.88
Trajectron++ [14]			8.98	19.02
LB-EBM [22]			8.87	15.61
PCCSNET [21]			8.62	16.16
Expert [46]			10.67	14.38
Expert + GMM [46]	H	20×20	7.65	14.38
Y-Net + TTST [59]	H+I	10,000	7.85	**11.85**
Y-Net [59]	H+I	20	8.97	14.61
CGNS [45]			15.6	28.20
SimAug [92]			10.27	19.71
Ours	H+I	20	**6.96**	12.62

6. Quantitative Results

We present a comparison between the baselines and our model on the Stanford Drone Dataset in Tab. 2. We categorise the models based on the type of input they utilise, with *H* denoting that a model only observes the historic trajectory, and *H+I* denoting that a model also utilises image data from the environment. The baseline results are the ones presented in the original papers, with the exception of Trajectron++ [14], Y-Net [59], and Expert [46], the results of which are provided by MID [52]. who retrained the models, because the Trajactron++ [14] paper does not include experiments on SDD, and the Y-Net [59] and Expert [46] papers obtain improved results using test-time sampling tricks. Specifically, Y-Net [59] uses a test time sampling trick, where 10,000 points are sampled, which is then clustered using k-means clustering algorithm, making their model generate diverse samples. The results of this approach are presented as Y-Net + TTST. Expert [46] uses a test procedure where 20 goals are sampled, and the one closest to the ground truth is used to condition its LSTM decoder. The output of the decoder is used as the parameters of a bivariate Gaussian for each time

Table 3: Quantitative evaluation results for the ETH/UCY Datasets, where sampling is performed $K = 20$ times. *H* means that a model only utilises information from the observed trajectory, while *H* means that a model only utilises information from the historic trajectory, while *H+I* indicates that a model is utilising both the historical trajectory and image data from the surrounding environment. The errors are measured in meters, and we report the best-of-k metrics (minADE and minFDE). The best results are **bold** and the second-best results are underlined.

Results: ETH/UCY Datasets

Model	Input	*K*	ETH		HOTEL		UNIV		ZARA1		ZARA2		AVG	
			ADE	FDE	ADE	FDE	ADE	FDE	ADE	FDE	ADE	FDE	ADE	FDE
MID [52]	H	20	0.39	0.66	0.13	0.22	0.22	0.45	0.17	0.30	0.13	0.27	0.21	0.38
Social-GAN [19]			0.81	1.52	0.72	1.61	0.60	1.26	0.34	0.69	0.42	0.84	0.58	1.18
PECNet [20]			0.54	0.87	0.18	0.24	0.35	0.60	0.22	0.39	0.17	0.30	0.29	0.48
STAR [54]			0.36	0.65	0.17	0.36	0.31	0.62	0.26	0.55	0.22	0.46	0.26	0.53
Trajectron++ [14]			0.39	0.83	0.12	0.21	0.20	0.44	0.15	0.33	0.11	0.25	0.19	0.41
LB-EBM [22]			0.30	0.52	0.13	0.20	0.27	0.52	0.20	0.37	0.15	0.29	0.21	0.38
PCCSNET [21]			0.28	0.54	0.11	0.19	0.29	0.60	0.21	0.44	0.15	0.34	0.21	0.42
Expert [46]			0.37	0.65	0.11	0.15	0.20	0.44	0.15	0.31	0.12	0.26	0.19	0.36
Expert + GMM [46]	H	20×20	0.29	0.65	**0.08**	0.15	**0.15**	0.44	**0.11**	0.31	**0.09**	0.26	**0.14**	0.36
Y-Net + TTST [59]	H+I	10,000	0.28	**0.33**	0.10	**0.14**	0.24	**0.41**	0.17	**0.27**	0.13	**0.22**	0.18	**0.27**
SoPhie [8]	H+I	20	0.70	1.43	0.76	1.67	0.54	1.24	0.30	0.63	0.38	0.78	0.54	1.15
CGNS [45]			0.62	1.40	0.70	0.93	0.48	1.22	0.32	0.59	0.35	0.71	0.49	0.97
Social-BiGAT [93]			0.69	1.29	0.49	1.01	0.55	1.32	0.30	0.62	0.36	0.75	0.48	1.00
MG-GAN [94]			0.47	0.91	0.14	0.24	0.54	1.07	0.36	0.73	0.29	0.60	0.36	0.71
Ours	H+I	20	**0.21**	0.36	0.09	**0.14**	0.21	0.44	0.15	0.28	0.11	0.21	0.15	0.29

step, and the 20 trajectories, used to compute the error metrics, are then obtained by sampling 20 coordinates from each Gaussian. The results of this approach are presented as Expert + GMM.

As evidenced by the results in Tab. 2 and Tab. 3, our method is highly competitive in the current state of the art. When evaluated on SDD, our model achieves the best ADE of the 13 compared models and achieves the second-best FDE—only surpassed by the version of Y-Net [59] that includes its test time sampling trick, which earns it a 19% decrease in FDE by constructing its final predictions from a pool of 10,000 predictions. A similar pattern emerges when our model is evaluated on the ETH/UCY datasets. In the ETH scene, our model achieves the best ADE and second-best FDE. In the HOTEL scene, our model has the second-best ADE, and shares the best FDE with Y-Net + TTST [59]. In all other scenes, our model achieves the second-best results for both ADE and FDE, with the only exception being ADE in the UNIV scene, where our model takes third place.

7. Qualitative Results

The numbers we present in Tab. 2 and Tab. 3 are supported by our observations when visualising and examining the predictions made by our model. Fig. 8 and Fig. 9 provide visualisation of randomly selected predictions made by our model for SDD and ETH/UCY, respectively. The visualisations and error metrics are indicative of the general ability of our model to generate trajectories that are close to the ground truth. We also observe that generated predictions, in some instances, exhibit a low degree of diversity, and based on our previous work [41], we attribute this to the absence of diversely sampled long-term goals to guide the predictions.

8. Ablation Study

To assess the effect of our environment encoder, we conduct an experiment, in which we train and evaluate a variation of our model where the environment encoder and spatial transformer have been removed, and compare it to the full model. The models we compare are trained and evaluated on SDD, as we argue that this dataset features a richer and more complex environment than the ETH/UCY datasets. Tab. 4 shows the error metrics resulting from the evaluation of the two models. We observe a 6.5% decrease in ADE and a slight decrease of 0.3% in FDE as a result including the environment encoder and spatial transformer, which is also in line with observations made in our previous work [41], and observations made by Salzmann et al. [14].

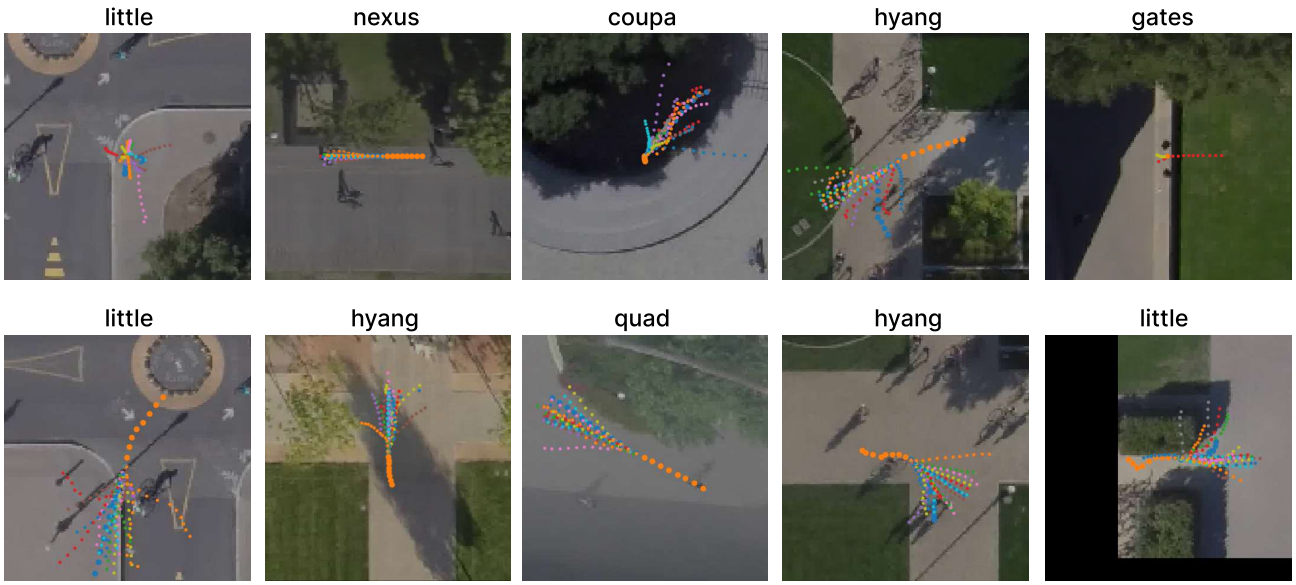


Figure 8: Compilation of trajectories produced by our model in various scenes of SDD using $K = 20$. The observed trajectory is shown as large orange points, and the ground truth future trajectory is shown as large blue points. The small points denote the predicted trajectories, and the colours of their denote which trajectory they belong to.

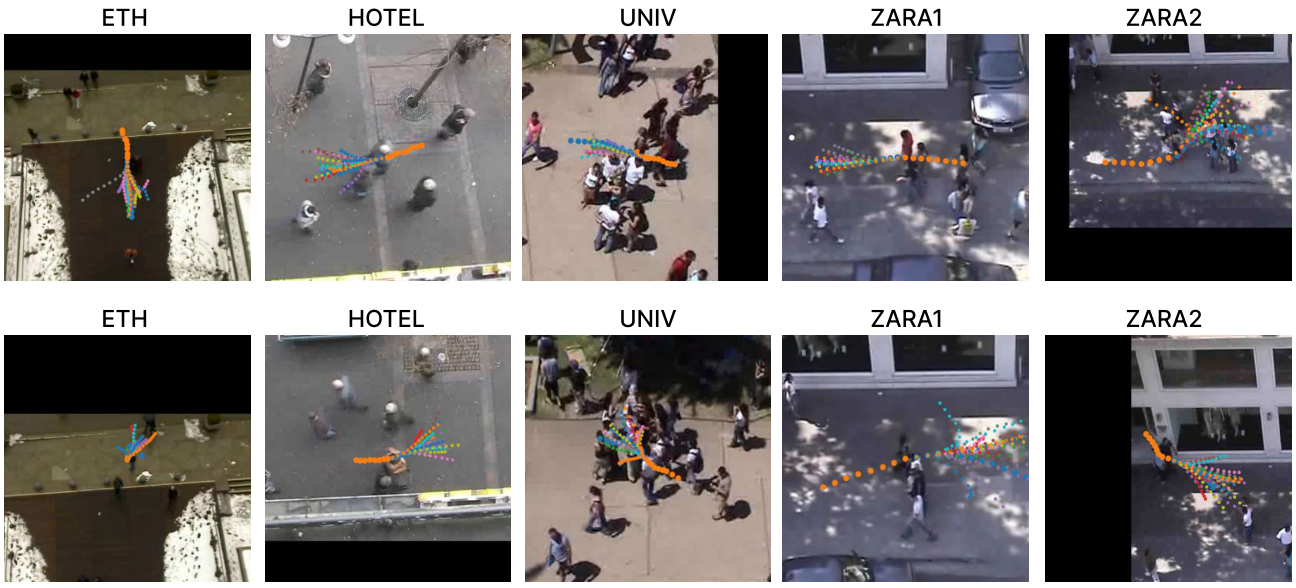


Figure 9: Compilation of trajectories produced by our model in each scene in the ETH/UCY datasets using $K = 20$. The observed trajectory is shown as large orange points, and the ground truth future trajectory is shown as large blue points. The remaining points are the predicted trajectories, and different colours indicate different predictions.

Table 4: The results of the ablation study two variations of the model is trained and evaluated on SDD. The first model is unable to leverage information about the environment, since the environment encoder and spatial transformer module have been removed. The second model is our full model, including the environment encoder and spatial transformer module. The error metrics are measured in pixels.

Ablation Study: Environment Encoder		
	ADE	**FDE**
w/o Encoder	7.44	12.66
w/ Encoder	6.99	12.62

9. DDIM Acceleration

Due to our high inference times, we conducted experiments to gauge the effects of using the DDIM sampler to skip sampling steps, as described by Song, Meng, and Ermon [83]. We train our model on SDD, using a diffusion process of 64 steps. We then evaluate this model using the DDIM sampler with 7 distinct subsets of linearly spaced samplings steps. The results are presented in Tab. 5, sorted in descending order with respect to the number of sampling steps used for the evaluation. The table shows both absolute values for the error metrics and inference time, as well as the relative values with respect to the base results, (64 steps). The experiments are performed on an Nvidia RTX 3080, and sampling times exclude data loader batch collation.

It is apparent that DDIM sampling with step skipping has a profound effect on the inference time. We also observe that the ensuing reduction in predictive accuracy is negligible, when the sampling process has no fewer than 4 steps. Comparing the results of the 8-step sampling process to the original 64-step process, we attain an 87.3% reduction in inference, while only suffering a 1.4% increase in ADE and 7.1% increase in FDE, resulting in a model that is 8 times faster and almost as accurate. Interestingly, an examination of Tab. 2, reveals that our model might be capable of keeping its position in the lead, using only 8 sample steps. To confirm this, however, requires a re-evaluation of this model configuration using the same test split as the SDD benchmark (Sec. 5.7), which we defer to future work.

Fig. 12 provides a qualitative comparison between trajectories produced with 64, 8, and 4 DDIM sampling steps. As the sampling steps are reduced, we observe a reduction in predictive diversity. This results in a better alignment of the predictions and the ground truth, but restricts the ability of the model to account for multiple distinct future outcomes. This could be alleviated by adopting a goal-directed approach, as we illustrated in our previous work [41], but such an approach is outside the scope of this paper. App. 1 provides additional samples of trajectories generated using 4 DDIM sampling steps.

Table 5: ADE, FDE and inference time of the same model on a single batch of size 512 using an Nvidia 3080, evaluated on SDD using DDIM sampling with different sized subsets of linearly spaced diffusion steps. The columns marked with (p) and (s) show absolute values in pixels and seconds, respectively, while columns marked with (%) show percentages that represent the relative differences in values, as compared to the base results (64 steps).

DDIM Acceleration						
Steps	**ADE (p)**	**FDE (p)**	**ADE (%)**	**FDE (%)**	**Time (s)**	**Time (%)**
64	6.96	12.62	100.0	100.0	18.19	100.0
32	6.97	12.73	100.0	100.8	9.14	50.2
16	7.00	13.01	100.1	103.1	4.63	25.6
8	7.06	13.52	100.05	101.4	2.30	12.64
4	7.61	15.01	109.3	118.9	1.18	6.49
2	10.49	20.35	150.7	161.25	0.61	3.5
1	99.38	133.0	1423	1053	0.31	1.8

10. Progressive Distillation Acceleration

Inspired by Salimans and Ho [84], we also experiment with progressive distillation, to assess its effectiveness in accelerating our diffusion model. We train a model on SDD, using a diffusion process with 64 steps. Following the iterative student-teacher transfer learning setup of [84], we use the 64-step model as the initial teacher and copy its model parameters to initialise the first student. [84] propose various parameterisations, and in our experimentation, we have tried predicting ϵ, x_0, x_t, and $v = \alpha_t \epsilon - \sigma_t \mathbf{x}$, and found no notable difference in training stability or model accuracy. We have also tried the proposed SNR loss and truncated SNR loss, and ultimately found MSE loss to result in more stable training with better convergence. Fig. 11 shows an example of the predicted target of the student during training and Fig. 10 shows a handful of randomly selected trajectories produced by the first student, i.e. a model that was distilled from a 64-step diffusion process to a 32-step diffusion process. Despite accurately predicting the denoising target and converging on a low MSE loss of 0.00371, the distilled student model is unable to generalise across individual training steps and produce meaningful trajectories through the entire denoising process. As a result, the predicted trajectories end up resembling random noise,

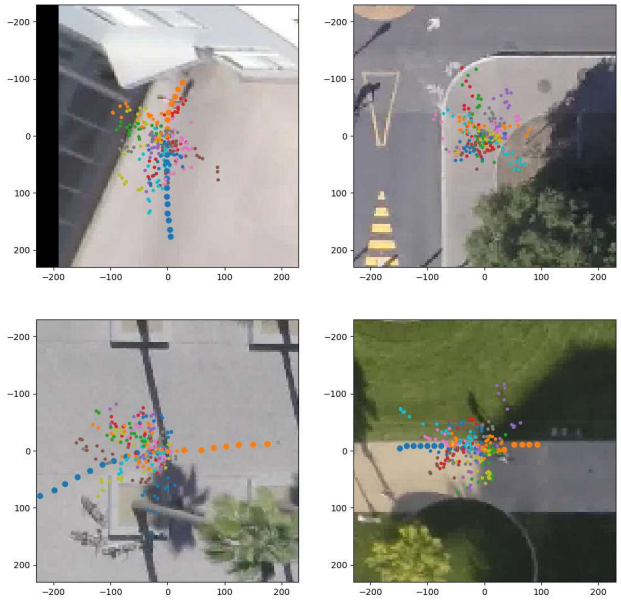


Figure 10: Predictions from the student from our experiment of distilling a 64 step model to 32 steps on the SDD. The large orange and blue points are respectively the history and future points, and various other points of similar colour coding are predictions made by the student model.

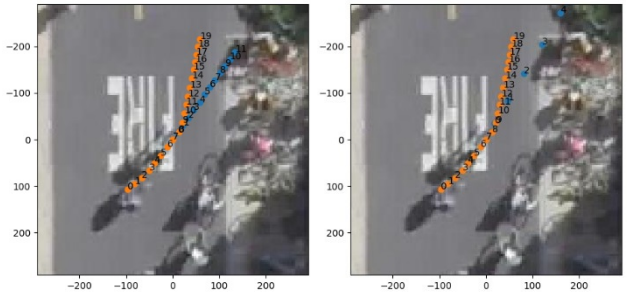


Figure 11: Predictions from the student from our experiment of distilling a 64 step model to 32 steps on the SDD. The left frame is the prediction time step 24 of 32, while the right is diffusion step 23 of 32. The orange points are the ground truth, and the blue points are the predictions made by the student.

which results in unreasonably high values of ADE and FDE. For comparison, Fig. 13 plots the ADE values of the distilled students together with the ADE values obtained by retraining and evaluating different sizes of DDPM, and evaluating the teacher model with the DDIM sampler. As evidenced by the error metrics and visualised trajectories, the predictions made by progressively distilled models are vastly inferior to those made using the DDIM sampler or even a DDPM, trained with a short diffusion process. This is in conflict with the observation made by Salimans and Ho [84], and we hypothesise that the issue lies in the hyperparameters or in our implementation. Hence, we are unable to rule out progressive distillation as a viable acceleration technique in the context of trajectory prediction. Further research into this matter is left for future work.

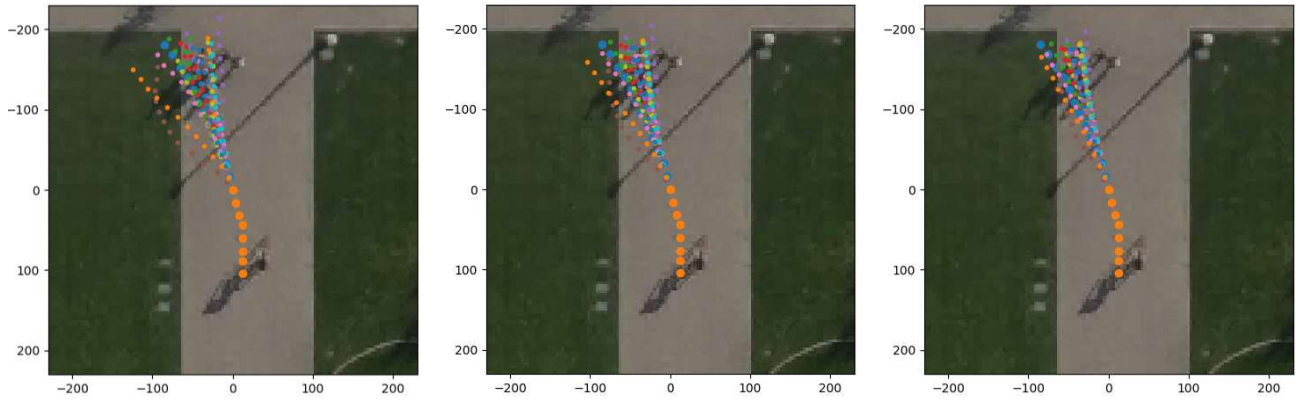
6. Conclusion

In this paper, we set out to reduce the gap between the strong generative capabilities of diffusion models and models that exhibit more desirable characteristics in terms of accuracy and inference time. Our proposed model, MAD-Traj, abandons the popular U-Net-based architecture of classic diffusion models in favour of the attention-based transformer architecture for powerful and efficient sequence modelling. By encoding both the observed trajectory and a semantic map of the environment, our model is able to make use of multi-modal information to increase its predictive accuracy. This is supported by our extensive experiments that underpin the performance of our model and its ability to compete with state-of-the-art methods.

Using a best-of-20 evaluation approach on the Stanford Drone Dataset (SDD), our model achieves the best result in terms of average displacement error (ADE), with a 9% improvement, compared to the second-best result. In terms of final displacement error (FDE), our model takes the second place with a 6% reduction in accuracy, compared to the best results. When trained and evaluated on the ETH/UCY datasets, our model is a top-contender, and consistently achieves either best or second-best results in terms of both ADE and FDE.

A qualitative evaluation of random trajectories, generated by our model, supports the quantitative results and shows the ability of the model to predict trajectories with similarity to the ground truth. It also reveals that the model has a tendency to produce predictions with fair amount of diversity, despite this not being the goal of this paper. In some instances, however, the predictions are extremely similar, despite the environment being suitable for a more diverse range of predictions. In our previous work [41], we have shown that a goal-directed approach can lead to a high degree of predictive diversity, and we hypothesise that our model could benefit from a similar approach.

Our model leverages a more compact vector representation of trajectories compared to our previous work [41], where trajectories are encoded in images. The reduction in complexity of the data representation, combined with the efficiency of transformers, results



(a) 64 sampling steps **(b)** 8 sampling steps **(c)** 4 sampling steps

Figure 12: The output of our original model at various DDIM step sizes given the same initial noise for each sample. The main takeaway from these samples is the reduced diversity of predictions when using DDIM step skipping.

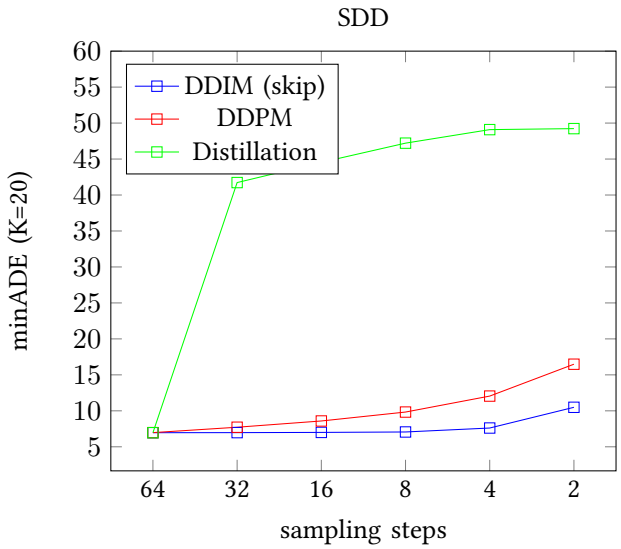


Figure 13: Comparison of minADE performance on the Stanford Drone Dataset using different sampling methods. The DDPM models are re-trained for each specific number of diffusion steps, the DDIM is using DDIM step skipping using a base model trained for 64 steps, and the distillation is the same model used for DDIM progressively distilled down to 2 steps.

in a significant reduction in the number of diffusion steps required to generate samples, allowing us to train our model that achieves highly competitive results using just 64 diffusion steps. As our experiments show, this can be amplified by using the non-Markovian DDIM sampler, which allows the model to skip sampling steps during inference. The accuracy of our model remains competitive with as few as 4 sampling steps, and with 8 sampling steps, our model retains 99% of its accuracy and requires just 13% of the time to generate trajectories.

We have also explored the use of progressive distillation as an alternative means of model acceleration. We have tried multiple model parameterisations and loss functions, but despite converging, our experiments show that the distilled models are unable to produce realistic trajectories and are vastly outperformed by models that are accelerated by the DDPM sampler. We suspect the cause to be related to hyperparameters and model implementation and defer further research into this matter to future work.

7. Acknowledgements

We wish to thank associate professor Jilin Hu for supervising our team on the development of this thesis. We also thank postdoc Haomin Yu and guest PhD student Yan Lin for their assistance and technical expertise. Finally, we wish to thank the team at CLAAUDIA Research Data Services for granting us access to powerful computational resources, which have been invaluable for training and evaluating our models.

References

[1] Jieneng Chen et al. *TransUNet: Transformers Make Strong Encoders for Medical Image Segmentation.* 2021. arXiv: 2102.04306 [cs.CV].

[2] Terence C. Mills and Raphael N. Markellos. *The Econometric Modelling of Financial Time Series.* 3rd ed. Cambridge University Press, 2008. DOI: 10.1017/CBO9780511817380.

[3] Xinbang Zhang et al. “Multi-modal spatio-temporal meteorological forecasting with deep neural network”. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 188 (2022), pp. 380–393. ISSN: 0924-2716. DOI: https://doi.org/10.1016/j.isprsjprs.2022.03.007. URL: https://www.sciencedirect.com/science/article/pii/S0924271622000703.

[4] Ljubisa Sehovac, Cornelius Nesen, and Katarina Grolinger. “Forecasting Building Energy Consumption with Deep Learning: A Sequence to Sequence Approach”. In: *2019 IEEE International Congress on Internet of Things (ICIOT)*. 2019, pp. 108–116. DOI: 10.1109/ICIOT.2019.00029.

[5] Edward Schmerling et al. *Multimodal Probabilistic Model-Based Planning for Human-Robot Interaction*. 2017. DOI: 10.48550/ARXIV.1710.09483. URL: https://arxiv.org/abs/1710.09483.

[6] Toon Bogaerts et al. “A graph CNN-LSTM neural network for short and long-term traffic forecasting based on trajectory data”. In: *Transportation Research Part C: Emerging Technologies* 112 (2020), pp. 62–77. ISSN: 0968-090X. DOI: https://doi.org/10.1016/j.trc.2020.01.010. URL: https://www.sciencedirect.com/science/article/pii/S0968090X19309349.

[7] Subhashini Venugopalan et al. “Sequence to Sequence - Video to Text”. In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2015.

[8] Amir Sadeghian et al. “SoPhie: An Attentive GAN for Predicting Paths Compliant to Social and Physical Constraints”. In: *CoRR* abs/1806.01482 (2018). arXiv: 1806.01482. URL: http://arxiv.org/abs/1806.01482.

[9] Patrick Dendorfer, Aljoša Ošep, and Laura Leal-Taixé. *Goal-GAN: Multimodal Trajectory Prediction Based on Goal Position Estimation*. DOI: 10.48550/ARXIV.2010.01114. URL: https://arxiv.org/abs/2010.01114.

[10] Xin Huang et al. *DiversityGAN: Diversity-Aware Vehicle Motion Prediction via Latent Semantic Sampling*. 2019. DOI: 10.48550/ARXIV.1911.12736. URL: https://arxiv.org/abs/1911.12736.

[11] Fang Fang et al. “Atten-GAN: Pedestrian Trajectory Prediction with GAN Based on Attention Mechanism”. In: *Cognitive Computation* 14 (June 2022), pp. 1–10. DOI: 10.1007/s12559-022-10029-z.

[12] Fang Fang et al. “Atten-GAN: Pedestrian Trajectory Prediction with GAN Based on Attention Mechanism”. In: *Cognitive Computation* 14 (June 2022), pp. 1–10. DOI: 10.1007/s12559-022-10029-z.

[13] Patrick Dendorfer, Sven Elflein, and Laura Leal-Taixé. “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 (ICCV)*. 2021, pp. 13158–13167.

[14] Tim Salzmann et al. *Trajectron++: Dynamically-Feasible Trajectory Forecasting With Heterogeneous Data*. 2020. DOI: 10.48550/ARXIV.2001.03093. URL: https://arxiv.org/abs/2001.03093.

[15] Ye Yuan et al. *AgentFormer: Agent-Aware Transformers for Socio-Temporal Multi-Agent Forecasting*. 2021. DOI: 10.48550/ARXIV.2103.14023. URL: https://arxiv.org/abs/2103.14023.

[16] Mihee Lee et al. “MUSE-VAE: Multi-Scale VAE for Environment-Aware Long Term Trajectory Prediction”. In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2022). DOI: 10.1109/cvpr52688.2022.00226. URL: http://dx.doi.org/10.1109/CVPR52688.2022.00226.

[17] Xinyu Chen et al. “TrajVAE: A Variational AutoEncoder model for trajectory generation”. In: *Neurocomputing* 428 (2021), pp. 332–339. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2020.03.120. URL: https://www.sciencedirect.com/science/article/pii/S0925231220312017.

[18] Pei Xu, Jean-Bernard Hayet, and Ioannis Karamouzas. “SocialVAE: Human Trajectory Prediction Using Timewise Latents”. In: *Lecture Notes in Computer Science*. 2022, pp. 511–528. DOI: 10.1007/978-3-031-19772-7_30. URL: https://doi.org/10.1007\%2F978-3-031-19772-7_30.

[19] Agrim Gupta et al. “Social GAN: Socially Acceptable Trajectories with Generative Adversarial Networks”. In: (2018). DOI: 10.48550/ARXIV.1803.10892. URL: https://arxiv.org/abs/1803.10892.

[20] Karttikeya Mangalam et al. “It Is Not the Journey but the Destination: Endpoint Conditioned Trajectory Prediction”. In: *CoRR* abs/2004.02025 (2020). arXiv: 2004.02025. URL: https://arxiv.org/abs/2004.02025.

[21] Jianhua Sun et al. *Three Steps to Multimodal Trajectory Prediction: Modality Clustering, Classification and Synthesis*. 2021.

[22] Bo Pang et al. *Trajectory Prediction with Latent Belief Energy-Based Model*. 2021.

[23] Henggang Cui et al. *Multimodal Trajectory Predictions for Autonomous Driving using Deep Con-*

volutional Networks. 2019. arXiv: 1809.10732 [cs.RO].

[24] Chiho Choi et al. "Shared Cross-Modal Trajectory Prediction for Autonomous Driving". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, pp. 244–253.

[25] Holger Caesar et al. *nuScenes: A multimodal dataset for autonomous driving*. 2019. DOI: 10.48550/ARXIV.1903.11027. URL: https://arxiv.org/abs/1903.11027.

[26] Jiasen Lu et al. *12-in-1: Multi-Task Vision and Language Representation Learning*. 2020. arXiv: 1912.02315 [cs.CV].

[27] Hao Tan and Mohit Bansal. "LXMERT: Learning Cross-Modality Encoder Representations from Transformers". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 5100–5111. DOI: 10.18653/v1/D19-1514. URL: https://aclanthology.org/D19-1514.

[28] Tadas Baltrušaitis, Chaitanya Ahuja, and Louis-Philippe Morency. "Multimodal Machine Learning: A Survey and Taxonomy". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 41.2 (2019), pp. 423–443. DOI: 10.1109/TPAMI.2018.2798607.

[29] Wenzhong Guo, Jianwen Wang, and Shiping Wang. "Deep Multimodal Representation Learning: A Survey". In: *IEEE Access* 7 (2019), pp. 63373–63394.

[30] Chuhua Wang et al. "Stepwise Goal-Driven Networks for Trajectory Prediction". In: *IEEE Robotics and Automation Letters* 7.2 (Apr. 2022), pp. 2716–2723. DOI: 10.1109/lra.2022.3145090. URL: https://doi.org/10.1109\%2Flra.2022.3145090.

[31] Letitia Parcalabescu, Nils Trost, and Anette Frank. *What is Multimodality?* 2021. arXiv: 2103.06304 [cs.AI].

[32] Ian J. Goodfellow et al. *Generative Adversarial Networks*. 2014. DOI: 10.48550/ARXIV.1406.2661. URL: https://arxiv.org/abs/1406.2661.

[33] Diederik P Kingma and Max Welling. *Auto-Encoding Variational Bayes*. 2013. DOI: 10.48550/ARXIV.1312.6114. URL: https://arxiv.org/abs/1312.6114.

[34] Google Developer. *[GAN] Common Problems*. Accessed 16-12-2022. URL: https://developers.google.com/machine-learning/gan/problems.

[35] David Bau et al. "Seeing What a GAN Cannot Generate". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019.

[36] Bhagyashree, Vandana Kushwaha, and G. C. Nandi. "Study of Prevention of Mode Collapse in Generative Adversarial Network (GAN)". In: *2020 IEEE 4th Conference on Information & Communication Technology (CICT)*. 2020, pp. 1–6. DOI: 10.1109/CICT51604.2020.9312049.

[37] Sanjeev Arora et al. "Generalization and Equilibrium in Generative Adversarial Nets (GANs)". In: *Proceedings of the 34th International Conference on Machine Learning*. Vol. 70. Proceedings of Machine Learning Research. 2017, pp. 224–232. URL: https://proceedings.mlr.press/v70/arora17a.html.

[38] Yaniv Yacoby, Weiwei Pan, and Finale Doshi-Velez. *Failure Modes of Variational Autoencoders and Their Effects on Downstream Tasks*. 2022.

[39] Jascha Sohl-Dickstein et al. *Deep Unsupervised Learning using Nonequilibrium Thermodynamics*. 2015. DOI: 10.48550/ARXIV.1503.03585. URL: https://arxiv.org/abs/1503.03585.

[40] Jonathan Ho, Ajay Jain, and Pieter Abbeel. *Denoising Diffusion Probabilistic Models*. 2020. DOI: 10.48550/ARXIV.2006.11239. URL: https://arxiv.org/abs/2006.11239.

[41] Mads Bach Andersen et al. *MEAT-DDPM: Multi-future Environment Aligned Trajectories using a Denoising Diffusion Probabilistic Model*. Jan. 2023. URL: https://projekter.aau.dk/projekter/files/512499215/P9_10.pdf.

[42] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. 2015. DOI: 10.48550/ARXIV.1505.04597. URL: https://arxiv.org/abs/1505.04597.

[43] Apeksha Shewalkar, Deepika Nyavanandi, and Simone A. Ludwig. "Performance Evaluation of Deep Neural Networks Applied to Speech Recognition: RNN, LSTM and GRU". In: *Journal of Artificial Intelligence and Soft Computing Research* 9.4 (2019), pp. 235–245. DOI: doi:10.2478/jaiscr-2019-0006. URL: https://doi.org/10.2478/jaiscr-2019-0006.

[44] Alexandre Alahi et al. "Social LSTM: Human Trajectory Prediction in Crowded Spaces". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 961–971. DOI: 10.1109/CVPR.2016.110.

[45] Jiachen Li, Hengbo Ma, and Masayoshi Tomizuka. *Conditional Generative Neural System for Probabilistic Trajectory Prediction*. 2019.

[46] He Zhao and Richard Wildes. "Where are you heading? Dynamic Trajectory Prediction with Expert Goal Examples". In: Oct. 2021, pp. 7609–7618. DOI: 10.1109/ICCV48922.2021.00753.

[47] Ashish Vaswani et al. *Attention Is All You Need*. 2017. DOI: 10.48550/ARXIV.1706.03762. URL: https://arxiv.org/abs/1706.03762.

[48] Jacob Devlin et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2019. arXiv: 1810.04805 [cs.CL].

[49] Linhao Dong, Shuang Xu, and Bo Xu. "Speech-Transformer: A No-Recurrence Sequence-to-Sequence Model for Speech Recognition". In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2018, pp. 5884–5888. DOI: 10.1109/ICASSP.2018.8462506.

[50] Iz Beltagy, Matthew E. Peters, and Arman Cohan. *Longformer: The Long-Document Transformer*. 2020. arXiv: 2004.05150 [cs.CL].

[51] Alexey Dosovitskiy et al. "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". In: *CoRR* abs/2010.11929 (2020). arXiv: 2010.11929. URL: https://arxiv.org/abs/2010.11929.

[52] Tianpei Gu et al. *Stochastic Trajectory Prediction via Motion Indeterminacy Diffusion*. 2022. DOI: 10.48550/ARXIV.2203.13777. URL: https://arxiv.org/abs/2203.13777.

[53] Alessio Monti et al. *How many Observations are Enough? Knowledge Distillation for Trajectory Forecasting*. June 2022. DOI: 10.1109/CVPR52688.2022.00644.

[54] Cunjun Yu et al. *Spatio-Temporal Graph Transformer Networks for Pedestrian Trajectory Prediction*. 2020. arXiv: 2005.08514 [cs.CV].

[55] Jiquan Ngiam et al. *Scene Transformer: A unified architecture for predicting multiple agent trajectories*. 2021. DOI: 10.48550/ARXIV.2106.08417. URL: https://arxiv.org/abs/2106.08417.

[56] Lihuan Li, Maurice Pagnucco, and Yang Song. *Graph-based Spatial Transformer with Memory Replay for Multi-future Pedestrian Trajectory Prediction*. 2022. DOI: 10.48550/ARXIV.2206.05712. URL: https://arxiv.org/abs/2206.05712.

[57] Song Wen, Hao Wang, and Dimitris Metaxas. "Social ODE: Multi-agent Trajectory Forecasting with Neural Ordinary Differential Equations". In: *Computer Vision – ECCV 2022*. Ed. by Shai Avidan et al. Cham: Springer Nature Switzerland, 2022, pp. 217–233.

[58] Hung Tran, Vuong Le, and Truyen Tran. "Goal-Driven Long-Term Trajectory Prediction". In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. Jan. 2021, pp. 796–805.

[59] Karttikeya Mangalam et al. "From Goals, Waypoints & Paths To Long Term Human Trajectory Forecasting". In: *CoRR* abs/2012.01526 (2020). arXiv: 2012.01526. URL: https://arxiv.org/abs/2012.01526.

[60] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. "Learning Structured Output Representation using Deep Conditional Generative Models". In: *Advances in Neural Information Processing Systems*. Ed. by C. Cortes et al. Vol. 28. Curran Associates, Inc., 2015. URL: https://proceedings.neurips.cc/paper/2015/file/8d55a249e6baa5c06772297520da2051-Paper.pdf.

[61] Mehdi Mirza and Simon Osindero. "Conditional Generative Adversarial Nets". In: (2014). DOI: 10.48550/ARXIV.1411.1784. URL: https://arxiv.org/abs/1411.1784.

[62] Rongqin Liang et al. *STGlow: A Flow-based Generative Framework with Dual Graphormer for Pedestrian Trajectory Prediction*. 2022. DOI: 10.48550/ARXIV.2211.11220. URL: https://arxiv.org/abs/2211.11220.

[63] Alexey Dosovitskiy and Thomas Brox. *Generating Images with Perceptual Similarity Metrics based on Deep Networks*. 2016.

[64] Prafulla Dhariwal and Alex Nichol. *Diffusion Models Beat GANs on Image Synthesis*. 2021. DOI: 10.48550/ARXIV.2105.05233. URL: https://arxiv.org/abs/2105.05233.

[65] Aditya Ramesh et al. *Hierarchical Text-Conditional Image Generation with CLIP Latents*. 2022. DOI: 10.48550/ARXIV.2204.06125. URL: https://arxiv.org/abs/2204.06125.

[66] Chitwan Saharia et al. *Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding*. 2022. DOI: 10.48550/ARXIV.2205.11487. URL: https://arxiv.org/abs/2205.11487.

[67] Alex Nichol et al. *GLIDE: Towards Photorealistic Image Generation and Editing with Text-Guided Diffusion Models*. 2021. DOI: 10.48550/ARXIV.2112.10741. URL: https://arxiv.org/abs/2112.10741.

[68] Shelly Sheynin et al. *KNN-Diffusion: Image Generation via Large-Scale Retrieval*. 2022. DOI: 10.48550/ARXIV.2204.02849. URL: https://arxiv.org/abs/2204.02849.

[69] Robin Rombach et al. *High-Resolution Image Synthesis with Latent Diffusion Models*. 2021. DOI: 10.48550/ARXIV.2112.10752. URL: https://arxiv.org/abs/2112.10752.

[70] Ruihan Yang, Prakhar Srivastava, and Stephan Mandt. *Diffusion Probabilistic Modeling for Video Generation*. 2022.

[71] Jonathan Ho et al. *Imagen Video: High Definition Video Generation with Diffusion Models*. 2022.

[72] Jacob Austin et al. *Structured Denoising Diffusion Models in Discrete State-Spaces*. 2023.

[73] Peiyu Yu et al. *Latent Diffusion Energy-Based Model for Interpretable Text Modeling*. 2022. DOI: 10.48550/ARXIV.2206.05895. URL: https://arxiv.org/abs/2206.05895.

[74] Nikolay Savinov et al. *Step-unrolled Denoising Autoencoders for Text Generation*. 2021. DOI: 10.48550/ARXIV.2112.06749. URL: https://arxiv.org/abs/2112.06749.

[75] Vadim Popov et al. *Grad-TTS: A Diffusion Probabilistic Model for Text-to-Speech*. 2021. DOI: 10.48550/ARXIV.2105.06337. URL: https://arxiv.org/abs/2105.06337.

[76] Zhifeng Kong et al. "DiffWave: A Versatile Diffusion Model for Audio Synthesis". In: *International Conference on Learning Representations*. 2021. URL: https://openreview.net/forum?id=a-xFK8Ymz5J.

[77] Emiel Hoogeboom et al. *Argmax Flows and Multinomial Diffusion: Learning Categorical Distributions*. 2021. DOI: 10.48550/ARXIV.2102.05379. URL: https://arxiv.org/abs/2102.05379.

[78] Tomer Amit et al. *SegDiff: Image Segmentation with Diffusion Probabilistic Models*. 2021. DOI: 10.48550/ARXIV.2112.00390. URL: https://arxiv.org/abs/2112.00390.

[79] Dmitry Baranchuk et al. *Label-Efficient Semantic Segmentation with Diffusion Models*. 2021. DOI: 10.48550/ARXIV.2112.03126. URL: https://arxiv.org/abs/2112.03126.

[80] Haoying Li et al. *SRDiff: Single Image Super-Resolution with Diffusion Probabilistic Models*. 2021. DOI: 10.48550/ARXIV.2104.14951. URL: https://arxiv.org/abs/2104.14951.

[81] Kashif Rasul et al. "Autoregressive Denoising Diffusion Models for Multivariate Probabilistic Time Series Forecasting". In: (2021). DOI: 10.48550/ARXIV.2101.12072. URL: https://arxiv.org/abs/2101.12072.

[82] Yao Liu et al. *Uncertainty-Aware Pedestrian Trajectory Prediction via Distributional Diffusion*. 2023.

[83] Jiaming Song, Chenlin Meng, and Stefano Ermon. *Denoising Diffusion Implicit Models*. 2022.

[84] Tim Salimans and Jonathan Ho. *Progressive Distillation for Fast Sampling of Diffusion Models*. 2022. arXiv: 2202.00512 [cs.LG].

[85] Donald T. Haynie. "The Second Law of Thermodynamics". In: *Biological Thermodynamics*. 2nd ed. Cambridge University Press, 2008, 58–84. DOI: 10.1017/CBO9780511802690.004.

[86] Alexandre Robicquet et al. "Learning Social Etiquette: Human Trajectory Understanding In Crowded Scenes". In: *European Conference on Computer Vision*. 2016.

[87] Stefano Pellegrini, Andreas Ess, and Luc Van Gool. "You'll Never Walk Alone: Modeling Social Behavior for Multi-target Tracking". In: Sept. 2009, pp. 261–268. DOI: 10.1109/ICCV.2009.5459260.

[88] Alon Lerner, Yiorgos Chrysanthou, and Dani Lischinski. "Crowds by Example". In: *Computer Graphics Forum* 26.3 (2007), pp. 655–664. DOI: https://doi.org/10.1111/j.1467-8659.2007.01089.x. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-8659.2007.01089.x.

[89] Boris Ivanovic and Marco Pavone. *The Trajectron: Probabilistic Multi-Agent Trajectory Modeling With Dynamic Spatiotemporal Graphs*. 2018. DOI: 10.48550/ARXIV.1810.05993. URL: https://arxiv.org/abs/1810.05993.

[90] Bang Cheng et al. "Pedestrian trajectory prediction via the Social-Grid LSTM model". In: *The Journal of Engineering* 2018.16 (2018), pp. 1468–1474. DOI: https://doi.org/10.1049/joe.2018.8316. URL: https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/joe.2018.8316.

[91] Hang Zhao et al. "TNT: Target-driveN Trajectory Prediction". In: (2020). DOI: 10.48550/ARXIV.2008.08294. URL: https://arxiv.org/abs/2008.08294.

[92] Junwei Liang, Lu Jiang, and Alexander Hauptmann. *SimAug: Learning Robust Representations from Simulation for Trajectory Prediction*. 2020.

[93] Vineet Kosaraju et al. *Social-BiGAT: Multimodal Trajectory Forecasting using Bicycle-GAN and Graph Attention Networks*. 2019. arXiv: 1907.03395 [cs.CV].

[94] Patrick Dendorfer, Sven Elflein, and Laura Leal-Taixé. *MG-GAN: A Multi-Generator Model Preventing Out-of-Distribution Samples in Pedestrian Trajectory Prediction*. 2021. arXiv: 2108.09274 [cs.CV].

[95] Junwei Liang et al. *The Garden of Forking Paths: Towards Multi-Future Trajectory Prediction*. 2020.

[96] Jun-Yan Zhu et al. *Toward Multimodal Image-to-Image Translation.* 2018. arXiv: 1711.11586 [cs.CV].

[97] Petar Veličković et al. *Graph Attention Networks.* 2018. arXiv: 1710.10903 [stat.ML].

1. Additional DDIM Trajectory Samples

Figure 14: Compilation of randomly selected trajectories, generated using 8 DDIM sampling steps.