MG-GAN: Supplementary Material

Patrick Dendorfer* Sven Elflein* Laura Leal-Taixé
Technical University Munich
{patrick.dendorfer,sven.elflein,leal.taixe}@tum.de

Abstract

The supplementary material complements our work with the implementation details of MG-GAN in Appendix A. Furthermore, we provide details on the training procedure and additional experiments on the hyperparameters of MG-GAN in Appendix B. In Appendix C, we explain how we compute Precision and Recall and describe the synthetic dataset in Appendix D. We discuss the performance of MG-GAN on real-world datasets in Appendix E. Lastly, we investigate the benefits of multi-generator models in learning distinct modes on a toy dataset in Appendix F and add visualizations of predicted trajectories on the considered datasets in Appendix G.

A. Architecture

The main contribution of our method is the use of multiple generators and the proposed Path Mode Network (PM-Net). The architecture of the individual generators is a standard Long Short-Term Memory (LSTM) encoder-decoder model [1] with social and physical attention as proposed in [11, 2]. The entire model is trained in a GAN framework and uses a discriminator and additional classifier that encourages the generators to specialize to a specific mode.

In the following paragraphs, we describe the architecture of all our components to generate a set of K future trajectories $\{\hat{Y}_i^k\}_{k=1,\dots,K}$ with $t \in [t_{obs}+1, t_{pred}]$ given the input trajectory X_i with $t \in [t_1, t_{obs}]$ for each pedestrian i. The source code of our model is provided with the supplementary material.

A.1. Encoding

To extract dynamic features from the past trajectory of a pedestrian i in a scene, we use an LSTM [6] to encode the relative displacements ΔX_i into a high dimensional feature representation d_i

$$d_i = \text{LSTM}_{en}\left(\Delta X_i^t, h_i^t\right),$$

where h_i^t is the hidden state of the LSTM. We compute visual features f_i with a CNN for an image patch I_i cut around the last observed position of pedestrian i

$$f_i = \text{CNN}\left(I_i\right).$$

The image patch I_i is a 32×32 pixel patch with a resolution of $0.7^m/_{pixel}$. The CNN has 2 layers with 16 filters, kernel size 3, max-pooling and ReLU activations and is trained from scratch.

The scene layout as well as other interacting pedestrians affect the path of a pedestrians and have to be considered by the model. For our method, we model physical (agent-scene) and social (agent-agent) interaction with corresponding soft-attention modules [13], following [11, 2].

Social Attention [2]: To account for social interaction, we apply soft-attention on the hidden states $\{d_j\}_{j \in J}$ of the other pedestrians in the scene where we compute the attention score a_{ij} based on the distance and the bearing angle between the agent i and a neighbouring agent j. The social information for pedestrian i is then defined as

$$s_i = \sum_{j \in J} a_{ij} d_j \tag{1}$$

Physical Attention [11]: The interaction with the scene around a pedestrian is also modeled with a soft-attention network [13] ATT applied on the CNN features f_i based on the motion encoding d_i. Thus, the physical features v_i are given by

$$v_i = \text{ATT}\left(f_i, d_i\right) \tag{2}$$

Final Encoding. Finally, the dynamic features d_i, social features s_i, and physical features v_i are concatenated to form the conditional encoding c_i for pedestrian i which is used for generating predictions in the following. All vectors d_i, v_i, and s_i have length 32.

*Equal contribution.

A.2. Generator

For MG-GAN, we propose n_G individual generators g. Each generator consists of an LSTM decoder, initialized with the encoded features c and combined with a random noise vector $z \sim \mathcal{N}(0,\, 1)$ as the initial hidden state h^0. The final trajectory $\hat{Y}$ is predicted recurrently:

$$\Delta\hat{Y}^t = \text{LSTM}_g\left(\Delta\hat{Y}^{t-1}, h^{t-1}\right), \tag{3}$$

where $\Delta\hat{Y}^{t_{obs}}$ is initialised with the last displacement of the observation $\Delta X^{t_{obs}}$.

A.3. Path Mode Network (PM-Net)

The Path Mode Network $\Pi(c_i)$ outputs a probability $\boldsymbol{\pi}$ over the generators conditioned on the encoded features c_i for a pedestrian i. The network consists of a 3-layer Multi-Layer Perceptron (MLP) with ReLU activations and the hidden dimension of size 48 and computes the final distribution over generators with a Softmax layer.

A.4. Discriminator and Classifier

Our model is trained in a GAN framework using a discriminator D and classifier C. Both networks use shared weights to encode the scene and trajectory mirroring the generator architecture described in Appendix A.1 to obtain the encoding c for a pedestrian. Additionally, we encode either the predicted or ground-truth trajectory, Y or $\hat{Y}$ respectively, with a two-layer MLP and concatenate with c to obtain the input for the two separate branches of the discriminator D and classifier C. Both use a two-layer MLP and produce the probability of the trajectory being real through a Sigmoid activation (Discriminator) or a distribution from which generator the trajectory was sampled from a Softmax activation (Classifier). Unless otherwise specified, we use LeakyReLU activations with a slope of 0.2 across the module.

B. Training MG-GAN

To train MG-GAN, we present an alternating training scheme as is explained in Algorithm 1. The proposed training scheme optimizes the generators and PM-Net in the model. During the training, we first optimize PM-Net based on the approximated likelihood of the generated trajectories by evaluating l samples each. In the second step, we sample q trajectories from the generator and apply the adversarial loss, best-of-many loss, and classification loss to the trajectories. In this Section, we provide additional information on the derivation of the PM-Net training objective (Appendix B.1) and discuss the effect of hyperparameters for the training (Appendix B.2).

B.1. PM-Net Objective

To estimate probabilities of trajectories, we assume normal distributed errors of the ground-truth $p_Y = Y + N(0, \sigma I) = N(Y, \sigma I)$. Thus, we can define the probability of a prediction $\hat{Y}$ as $p(\hat{Y}|c, z, g) = p_Y(\hat{Y})$ where $z \sim N(0, I)$ is the GAN noise distribution, c the encoded, conditional scene information and g the generator index. Using symmetry of the Normal distribution and marginalizing z through l Monte Carlo samples $\{z_{(i)}\}_{i=1}^{l}$, the likelihood of the ground-truth trajectory Y can be written as

$$\begin{aligned} p(Y|g, c) &= \int p(Y|g, z, c)dz \\ &\approx \frac{1}{l}\sum_{i=1}^{l} p(Y|c, z_{(i)}, g) \\ &\approx \frac{1}{l}\sum_{i=1}^{l} \mathcal{N}(Y; \hat{Y}_{c,z_{(i)},g}, \sigma I) \\ &\propto \frac{1}{l}\sum_{i=1}^{l} \exp\left(\frac{-\left\|\hat{Y}_{g,c,z_i} - Y\right\|_2^2}{2\sigma}\right). \end{aligned} \tag{4}$$

By applying Bayes' rule, one obtains the posterior distribution over generators

$$\begin{aligned} p(g|Y, c) &= \frac{p(Y|c, g)p(g|c)}{p(Y|c)} \\ &= \frac{p(Y|c, g)p(g|c)}{\sum_g p(Y|g, c)p(g|c)}. \end{aligned}$$

We use a non-informative, uniform prior distribution over generators $p(g|c) = 1/n_G$ as we do not have knowledge which generator is relevant for a given scene context c at the start of training. Overall, we obtain

$$p(g|Y, c) = \frac{p(Y|c, g)}{\sum_h p(Y|h, c)} \tag{5}$$

which can be computed using the approximation of Equation (4) concluding our derivation.

As described in the main paper, we train PM-Net by minimizing the Cross-Entropy between the approximated distribution over generators in Equation (5) which we derived in this section and the output distribution $\Pi(c)$ produced by PM-Net.

B.2. Hyperparameters

For optimization, we use the Adam [7] optimizer with learning rate 0.001, $\beta_1 = 0.5$ and $\beta_2 = 0.999$. We set the number of G training samples $q = 20$ and PM-Net training samples $l = 1$. Further, we set the weighting coefficients $\lambda_{Traj}, \lambda_{Cl}$, and the standard deviation σ to 1. We study different settings of the hyperparameters specific to our model on the synthetic dataset in the following.

Table 1: Results of MG-GAN ($n_G = 5$) trained with different values for λ_{Traj} and λ_{Cl}.

λ_{Traj}	λ_{Cl}	ADE	FDE	Precision	Recall
0.1	0.1	0.37	0.46	0.60	0.80
0.1	1.0	0.36	0.44	0.64	0.80
0.1	5.0	0.34	0.44	0.68	0.82
1.0	0.1	0.35	0.46	0.73	0.86
1.0	1.0	0.33	0.44	0.71	0.90
1.0	5.0	0.32	0.45	0.76	0.94
5.0	0.1	0.32	0.46	0.80	0.92
5.0	1.0	0.30	0.44	0.81	0.97
5.0	5.0	0.32	0.45	0.79	0.93

Effect of Loss Weighting. Results for different settings of λ_{Traj}, weighting the $L2$ best-of-many loss [3, 4] loss term L_{Traj}, and λ_{Cl}, weighting the classifier loss term L_{Cl}, can be found in Table 1. We observe that higher settings of λ_{Traj} help slightly to improve ADE, Precision, and Recall. A higher λ_{Traj} enforces that at least one generated sample is close to the ground-truth trajectory reducing ADE and increasing Recall. Additionally, it enforces further specialization of the generators as the L_{Traj} loss only applies to the closest sample that always comes from the same generator when they already cover distinct modes. As a result, we find that λ_{Cl} does not affect the results significantly since it encourages the generators to be distinguishable. If the classifier drives the generators to cover distinct modes already in the early stages of training, the loss is small and has little influence during the remaining training independent of the weighting.

Effect of σ for PM-Net Training. The parameter σ represents the standard deviation of the normally distributed error $\mathcal{N}(0, \sigma I)$ in meters around the ground-truth trajectory Y and shapes the likelihood approximation in Equation (4). For large σ the probability over generators $p(g|Y, c)$ becomes uniform in Equation (5) while the probability converges to a one-hot vector for the generator producing the closest samples to the ground-truth as $\sigma \to 0$. We find in Table 2 that the results of MG-GAN are stable w.r.t. reasonable choices of σ indicating that no hyperparameter tuning on σ is necessary in order for MG-GAN to converge to a correct solution producing high precision.

Effect of the Number of Training Samples. For the training of MG-GAN we can choose the number of Monte Carlo samples l used for the likelihood estimation in Equation (4) and the number of generator training samples q for computing the $L2$ best-of-many loss [3, 4] $\mathcal{L}_{Traj}$. In Table 3, we find that a single sample $q = 1$ is not enough to train a multimodal model, because it still results in a model predicting linear straight motion. By increasing q we find that this model can predict more multimodal trajectories since the loss is only applied on the sample closest to the ground truth and the other $q - 1$ predictions in potentially other directions are not punished. Increasing the number of samples l has a positive effect on the performance because the likelihood estimation becomes more accurate and the generators can develop the specialization to a specific mode even further.

Table 2: Results of MG-GAN ($n_G = 5$) trained with different values for σ.

σ	ADE	FDE	Precision	Recall
0.1	0.33	0.49	0.77	0.95
0.5	0.38	0.53	0.70	0.89
1.0	0.32	0.44	0.77	0.95
2.5	0.32	0.48	0.80	0.95
5.0	0.34	0.47	0.72	0.90

Table 3: Results of MG-GAN ($n_G = 5$) trained with different number of PM-Net training samples l and generator samples q.

Π-Net Samples l	G Training Samples q	ADE	FDE	Precision	Recall 5
1	1	2.18	4.57	0.31	0.30
1	10	0.46	0.52	0.55	0.66
1	20	0.45	0.57	0.57	0.82
5	1	2.20	4.56	0.33	0.29
5	10	0.46	0.49	0.56	0.68
5	20	0.31	0.45	0.79	0.95

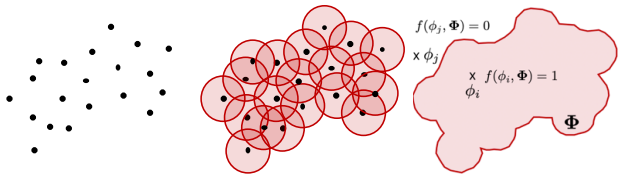
C. Definition of Precision and Recall

To measure the performance of models in preventing out-of-distribution (OOD) samples while covering the entire support of the distribution, we follow the GAN literature [12, 8] and estimate the manifolds of predicted trajectories and ground-truth samples to compute Precision and Recall.

For a set of future trajectories $\Phi = \{\phi_k\}$, we estimate the corresponding manifold in the output space by considering the points of all trajectories $\{\phi_k^t\}$ at time t (Figure 1a), constructing a disc with radius R^t around each point ϕ_k^t (Figure 1b). The R^t represents the maximum distance error we can accept for the predictions. The union of all disc areas serves as an estimate of the true manifold (Figure 1c). We do this for every time $t \in \{1, ..., T\}$ and define

$$R^t = \frac{R_{max} \cdot t}{T}$$

where we set $R_{max} = 2m$. To determine if a given sample ϕ lies inside this manifold, we define a binary score func-



(a) (b) (c)

Figure 1: (a) Trajectory endpoints, (b) estimating manifold through disc around the points with radius R, and (c) testing if samples ϕ are in estimated manifold.

tion:

$$score(\phi, \mathbf{\Phi}) = \begin{cases} 1, \text{ if } \forall t \exists \phi' \in \mathbf{\Phi} \text{ with } \|\phi_t - \phi'_t\|_2 \leq R^t \\ 0, \text{ otherwise.} \end{cases} \tag{6}$$

Following the above definition, we construct the manifold $\mathbf{\Phi}_G$ based on the set of model predictions ϕ_G. Similarly, we estimate the ground-truth manifold $\mathbf{\Phi}_R$ using the set of ground-truth trajectories ϕ_R as provided in the FPD [9] and the synthetic dataset (Appendix D).

Precision and Recall are then defined using Equation (6) as

$$\text{Precision} = \frac{1}{|\phi_G|} \sum_{\phi \in \phi_G} score(\phi, \mathbf{\Phi}_R) \text{ and} \tag{7}$$

$$\text{Recall} = \frac{1}{|\phi_R|} \sum_{\phi \in \phi_R} score(\phi, \mathbf{\Phi}_G). \tag{8}$$

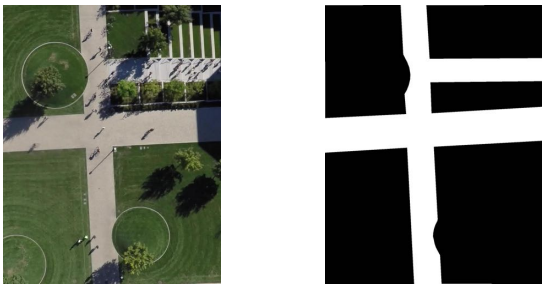
Intuitively, Precision measures the realism of a trajectory because it queries if the prediction falls inside the ground-truth manifold. Symmetrically, Recall measures if real samples lie within the manifold generated by predictions and thus measures if all modes present in the ground truth are covered.

D. Synthetic Dataset

In the main paper, we present a synthetic dataset to study the generated multimodality of the models. We generate the dataset on the Hyang-4 scene of the SDD [10], as shown in Figure 2a. This scene is well suited because it provides separated spatial modes with an upper and lower junction with two and three modes respectively. We simulate the dynamics of $\approx 80,000$ pedestrians using the Social Force Model [5]. In order to control and limit the modes of future trajectories, we use an occupancy map shown in Figure 2b restricting the area the pedestrians can walk on. In the dataset, we primarily focus on spatial multimodality and limit the number of pedestrians to a maximum of two.

E. Multimodality of Real Datasets

In this section, we try to measure the overall multimodality of or public benchmarks. As we do not have multiple



(a) (b)

Figure 2: Scene image (a) and occupation map (b) of the synthetic dataset.

Table 4: ADE and FDE results for MG-GAN with different number of generators on the SDD dataset.

# Generators	2	3	4	5
ADE	13.7	14.6	13.6	14.5
FDE	26.1	27.6	25.8	27.6

ground-truth trajectories on these datasets, we consider similar trajectories which are (i) in close proximity (closer than $2m$), (ii) walk-in similar directions ($\pm 45°$), and (iii) walk with similar speed ($\pm 0.5 {}^{m}/_{s}$). We then filter trajectories if they collide with other pedestrians in the scene (distance $\leq 0.5m$).

Finally, we use the procedure described in Appendix C to estimate the manifold based on the collected set of trajectories and count the number of disconnected components across timesteps.

We conclude from Figure 3 that SDD is less multimodal (Avg. # of modes: 1.15) than the other datasets, i.e., FPD (Avg. # of modes: 1.36) and UCY/ETH (Avg. # of modes: 1.34). This observation is somewhat congruent with the performance of MG-GAN on the public benchmarks. While our method achieves state-of-the-art performance on ETH and UCY which is more multimodal than SDD that naturally makes it hard for our method to show benefits over existing methods.

We elaborate this further and train the single generator baseline *GAN+L2* with the same backbone on SDD. We obtain similar results compared to MG-GAN with ADE of 14.6 and FDE 27.5 as shown in Table 4. These results further indicate that SDD does not contain sufficient multi-modality for our method achieving better results than single-generator methods.

F. Toy Experiment

In addition to the experiments in the main paper, we study multimodality on a toy dataset introduced in [2]. The data consists of six starting positions equidistantly dis-

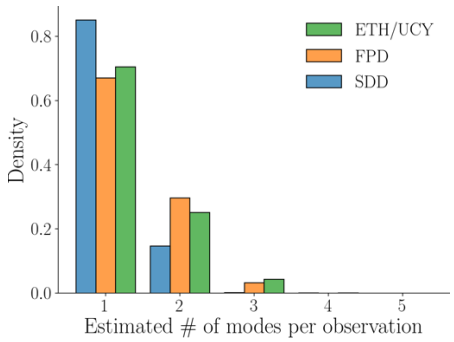


Figure 3: Histogram of the estimated, relative number of modes for the real datasets.

tributed on a circle where we generate three paths with uniform probability for each starting position, as shown in Figure 4a. This experiment demonstrates how different models represent the multiple modes inside a lower-dimensional latent space. This experiment should give us a deeper understanding of the architectural requirements for modeling distinct modes. For this experiment, all methods use a single encoder to encode the observation, then an MLP is used to transform the encoding with the GAN noise z to the three-dimensional latent space, and a decoder decodes the latent space to predictions. For MG-GAN, we use separate MLPs mapping to the latent space resembling the different generators in our main model.

Results. In Figure 4, we visualize the predictions (left) and corresponding latent space vectors (right) for the considered models. The simple GAN baseline fails to recover all three modes, while training with an additional $L2$ loss leads to unrealistic out-of-distribution samples since the loss encourages the samples to spread over the entire output space which is also reflected in the latent space. InfoGAN does not learn to encode different modes inside its categorical values. Ultimately, our multi-generator model covers all disconnected modes without producing out-of-distribution samples in between. The different sub-networks learn to map the different modes in separated areas in the latent space and hence introduce the required disconnectedness that allows the prediction of disjoint manifolds. This further demonstrates the efficacy of multi-generator models compared to single generator models in preventing OOD samples while covering the entire distribution.

G. Visualizations

We present additional visualizations of generated trajectories of MG-GAN on ETH and UCY in Figure 5, SDD in Figure 6 and the FPD in Figure 7.

In Figure 8, we explore the latent space of the GAN baseline and MG-GAN. While a latent space interpolation results in out-of-distribution samples for the baseline, we show that each generator is limited to the support of a specific mode preventing the generation of OOD samples.

References

[1] Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. Social LSTM: Human trajectory prediction in crowded spaces. In *Conference on Computer Vision and Pattern Recognition*, 2016. 1

[2] Javad Amirian, Jean-Bernard Hayet, and Julien Pettré. Social Ways: Learning multi-modal distributions of pedestrian trajectories with GANs. In *Conference on Computer Vision and Pattern Recognition Workshops*, 2019. 1, 4

[3] Apratim Bhattacharyya, Bernt Schiele, and Mario Fritz. Accurate and diverse sampling of sequences based on a "Best of Many" sample objective. In *Conference on Computer Vision and Pattern Recognition*, 2018. 3

[4] Agrim Gupta, Justin Johnson, Li Fei-Fei, Silvio Savarese, and Alexandre Alahi. Social GAN: Socially acceptable trajectories with generative adversarial networks. In *Conference on Computer Vision and Pattern Recognition*, 2018. 3

[5] Dirk Helbing and Péter Molnár. Social force model for pedestrian dynamics. *Physical Review E*, 51, 1995. 4

[6] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 1997. 1

[7] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *European Conference on Computer Vision*, 2014. 2

[8] Tuomas Kynkäänniemi, Tero Karras, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Improved Precision and Recall Metric for Assessing Generative Models. In *Neural Information Processing Systems*, 2019. 3

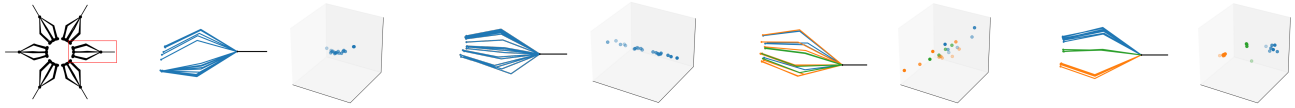
[9] Junwei Liang, Lu Jiang, Kevin Murphy, Ting Yu, and Alexander Hauptmann. The Garden of Forking Paths: Towards Multi-Future Trajectory Prediction. In *Conference on Computer Vision and Pattern Recognition*, 2020. 4

[10] Alexandre Robicquet, Amir Sadeghian, Alexandre Alahi, and Silvio Savarese. Learning social etiquette: Human trajectory understanding in crowded scenes. In *European Conference on Computer Vision*, 2016. 4

[11] Amir Sadeghian, Vineet Kosaraju, Ali Sadeghian, Noriaki Hirose, Hamid Rezatofighi, and Silvio Savarese. Sophie: An attentive GAN for predicting paths compliant to social and physical constraints. In *Conference on Computer Vision and Pattern Recognition*, 2019. 1

[12] Mehdi S. M. Sajjadi, Olivier Bachem, Mario Lucic, Olivier Bousquet, and Sylvain Gelly. Assessing Generative Models via Precision and Recall. In *Neural Information Processing Systems*, 2018. 3

[13] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*, 2015. 1



(a) GT (b) GAN (c) GAN+L2 (d) InfoGAN (e) MG-GAN

Figure 4: (a) shows the entire toy dataset. Plots (b)-(e) demonstrate predictions for different models and the corresponding points in the latent space for the observation in the red box (a). Samples from different discrete latent codes or generators are visualized in different colors.

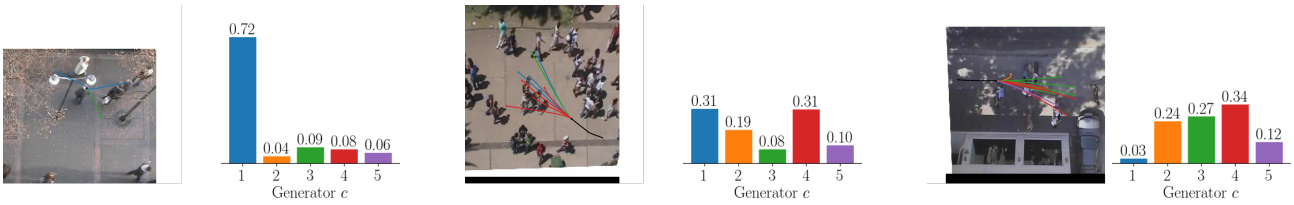


Figure 5: Generated samples of MG-GAN on the ETH and UCY dataset

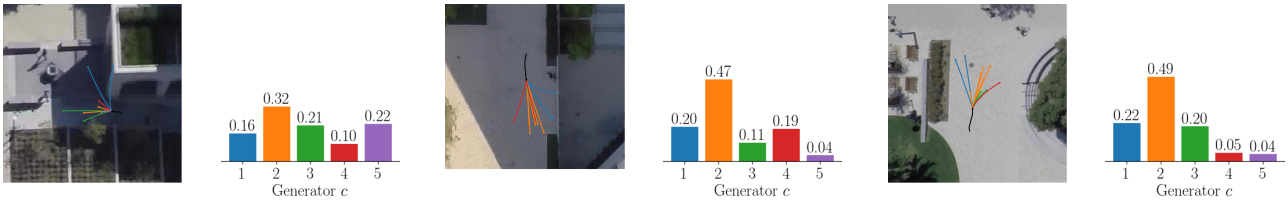


Figure 6: Generated samples of MG-GAN on the Stanford Drone Dataset (SDD)

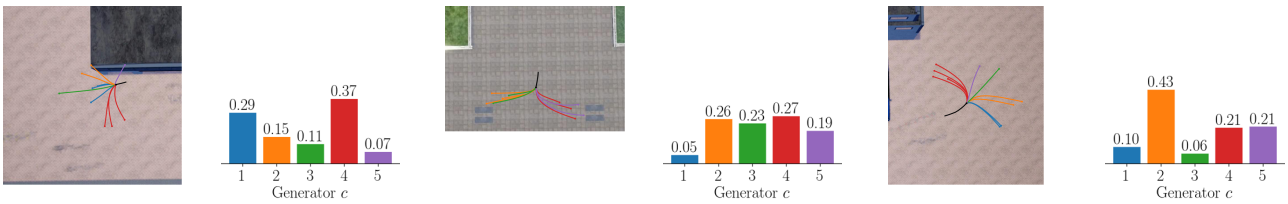
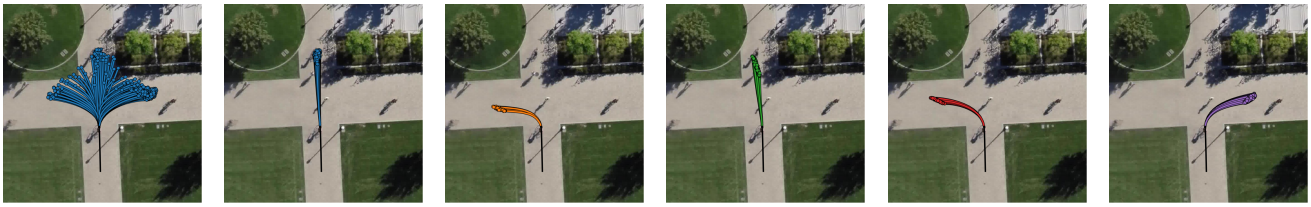


Figure 7: Generated samples of MG-GAN on the Forking Path dataset



(a) GAN+L2 (b) MG-GAN Generator 1 (c) MG-GAN Generator 2 (d) MG-GAN Generator 3 (e) MG-GAN Generator 4 (f) MG-GAN Generator 5

Figure 8: Trajectory samples during latent space walk for the single generator model in Figure 8a and the individual generators of MG-GAN (5) in Figures 8b to 8f.

Algorithm 1 Proposed algorithm for training MG-GAN.

Precondition: $p(z)$ noise distribution, m batch size, $\{\theta_i\}_{i=1}^{n_G}$ set of generator weights, w weights of discriminator, ζ weights of PM-Net, λ_{Traj} weighting for $L2$ best-of-many loss, λ_{Cl} weighting for generator classification regularization, q number of G training samples, l number of PM-Net samples

repeat

$\{x^i\}_{i=1}^m, \{y^i\}_{i=1}^m \sim p_r(x, y)$ ▷ Batch from real data where x is the input (observed trajectories and image crop) and Y the ground-truth observation

$\{z^i\}_{i=1}^m \sim p(z)$ ▷ Batch from noise distribution

$\{c^i\}_{i=1}^m \sim \Pi(x^i; \zeta)$ ▷ Batch of generator indices samples from PM-Net

$\{\hat{Y}_g^i\}_{i=1}^m \leftarrow G(x^i, z^i; \theta_{c^i})$ ▷ Generate batch using selected generators

$g_w \leftarrow \nabla_w \frac{1}{m} \sum_i \left[\ln D(x^i, y^i; w) + \ln(1 - D(x^i, \hat{Y}_g^i; w))\right]$

$w \leftarrow \text{Adam}(w, g_w)$ ▷ Optimize discriminator D

$g_\gamma \leftarrow \nabla_\gamma \frac{1}{m} \sum_i \left[\ln C(x_g^i, \hat{Y}_g^i; \gamma)\right]_{c^i}$

$\gamma \leftarrow \text{Adam}(g_\gamma, \gamma)$ ▷ Update Classifier C

$\{z^{i,j}\}_{i=1,j=1}^{m,q} \sim p(z)$

$\{c^{i,j}\}_{i=1,j=1}^{m,q} \sim \Pi(x^i; \zeta)$

$\{\hat{Y}_g^{i,j}\}_{i=1,j=1,g=c^{i,j}}^{m,k} \leftarrow G(x^i, z^{i,j}; \theta_{c^{i,j}})$

$\{t^i\}_{i=1}^m \leftarrow \min_{j,g} \left\| y^i - \hat{Y}_g^{i,j} \right\|$

$\{g_{min}^i\}_{i=1}^m \leftarrow \arg\min_g \left\| y^i - \hat{Y}_g^{i,j} \right\|$

for $o \in \{1 \dots n_G\}$ **do**

$g_{\theta_o} \leftarrow \nabla_{\theta_j} \frac{1}{mk} \sum_i \sum_j \left[\ln D(x^i, \hat{Y}_g^{i,j}; w) - \lambda_{Cl} \ln C(x^i, \hat{Y}_g^{i,j}; \gamma)\right] + \delta_{o,g_{min}^i} \lambda_{Traj} t^i$

$\theta_{Cl} \leftarrow \text{Adam}(g_{\theta_o}, \theta_o)$ ▷ Optimize generators

end for

$\{z^{i,j}\}_{i=1,j=1}^{m,l} \sim p(z)$

$\{\hat{Y}_o^{i,j}\}_{i=1,j=1,o=1}^{m,l,n_G} \leftarrow G(x^i, z^{i,j}; \theta_o)$

$\{p^{i,o}\}_{i=1,o=1}^{m,n_G} \leftarrow \mathcal{N}(\hat{Y}_o^{i,j}; y^i, \sigma I)$

$g_\zeta \leftarrow \nabla_\zeta \frac{1}{m} \sum_i H\left(\Pi(x^i; \zeta), p^i\right)$

$\zeta \leftarrow \text{Adam}(g_\zeta, \zeta)$ ▷ Optimize PM-Net

until convergence.