A Datasets

We use 2 simulated (Sines, MuJoCo) and 2 real-world (Stocks, Energy) datasets. Table. 7 shows the statistics of the datasets. All datasets are available online via the link. We note that in some of our datasets, the time series length N can be varied from one time series sample to another. However, our framework has no problems in dealing with those varying lengths.

Table 7: Dataset Statistics

Dataset	# of Samples	$\dim(\mathbf{x})$	Average of N	Link	License
Sines	10,000	5	24 time-points	-	-
Stocks	3,773	6	24 days	Link	-
Energy	19,711	28	24 hours	Link	CC BY 4.0
MuJoCo	4,620	14	24 time-points	Link	Apache License 2.0

B ODE/CDE functions in GT-GAN

B.1 Encoder

Our encoder based on NCDEs has the following CDE function f.[2]

Table 8: The architecture of the network f in the encoder

Layer	Design	Input Size	Output Size
1	ReLU(Linear)	$N \times \dim(\mathbf{x})$	$N \times 4\dim(\mathbf{x})$
2	ReLU(Linear)	$N \times 4\dim(\mathbf{x})$	$N \times 4\dim(\mathbf{x})$
3	ReLU(Linear)	$N \times 4\dim(\mathbf{x})$	$N \times 4\dim(\mathbf{x})$
4	Tanh(Linear)	$N \times 4\dim(\mathbf{x})$	$N \times \dim(\mathbf{x})$

B.2 Decoder, Discriminator

Our decoder and discriminator based on GRU-ODEs have the following ODE functions.[3] They have the same architecture but their parameters are separated.

Table 9: The architecture of the network g in the decoder

Layer	Design	Input Size	Output Size
1	r_t =Sigmoid(Linear)	$N \times \dim(\mathbf{h})$	$N \times \dim(\mathbf{h})$
	z_t =Sigmoid(Linear)	$N \times \dim(\mathbf{h})$	$N \times \dim(\mathbf{h})$
	u_t =Tanh(Linear)	$N \times \dim(\mathbf{h})$	$N \times \dim(\mathbf{h})$
	$dh = (1 - z_t) * (u_t - h_t)$	$N \times \dim(\mathbf{h})$	$N \times \dim(\mathbf{h})$

Table 10: The architecture of the network q in the discriminator

Layer	Design	Input Size	Output Size
1	r_t =Sigmoid(Linear)	$N \times \dim(\mathbf{x})$	$N \times \dim(\mathbf{x})$
	z_t =Sigmoid(Linear)	$N \times \dim(\mathbf{x})$	$N \times \dim(\mathbf{x})$
	u_t =Tanh(Linear)	$N \times \dim(\mathbf{x})$	$N \times \dim(\mathbf{x})$
	$dh = (1 - z_t) * (u_t - h_t)$	$N \times \dim(\mathbf{x})$	$N \times \dim(\mathbf{x})$

[2]CDE: `https://github.com/patrick-kidger/NeuralCDE` (Apache-2.0 license)
[3]ODE: `https://github.com/rtqichen/torchdiffeq` (MIT license)

B.3 Generator

Our generator has the following ODE function f in Table 11.

Table 11: The architecture of the network r in the generator

Layer	Design	Input Size	Output Size
1	Softplus(Linear)	$N \times \dim(\mathbf{h}+1)$	$N \times \dim(\mathbf{h})$
2	Softplus(Linear)	$N \times \dim(\mathbf{h}+1)$	$N \times \dim(\mathbf{h})$
3	Softplus(Linear)	$N \times \dim(\mathbf{h}+1)$	$N \times \dim(\mathbf{h})$

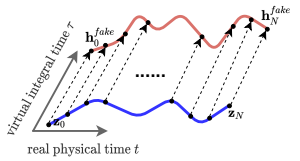


Figure 7: An example of our generation

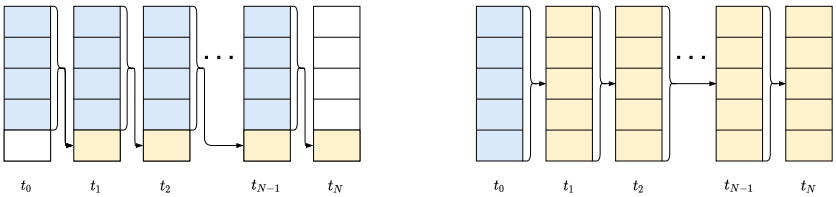
C Baselines

For the regular time series baseline models, i.e., TimeGAN, RCGAN, C-RNN-GAN, T-forcing, and P-forcing, we use the 3-layer GRU-based neural network architecture with a hidden size that is 4 times larger than the input size. We use or modify the following accessible source codes to run.

- TimeGAN : https://github.com/jsyoon0823/TimeGAN
- RCGAN : https://github.com/3778/Ward2ICU
- C-RNN-GAN : https://github.com/olofmogren/c-rnn-gan
- T-forcing, P-forcing : https://github.com/mojesty/professor_forcing
- GRU-D : https://github.com/zhiyongc/GRU-D

Because ordinary GRUs can not be applied to irregular time series, we replace the first layer GRU to GRU-$\triangle t$ and GRU-D in all those baselines so that the redesigned baseline models, i.e., TimeGAN-$\triangle t$, RCGAN-$\triangle t$, C-RNN-GAN-$\triangle t$, T-forcing-$\triangle t$, P-forcing-$\triangle t$, TimeGAN-D, RCGAN-D, C-RNN-GAN-D, T-forcing-D and P-forcing-D, can process irregular time series data.

D Evaluation metrics



(a) How to calculate the predictive score for the regular time series synthesis in TimeGAN

(b) How to calculate the predictive score for the irregular time series in this paper

Figure 8: Predictive task according to the data type

For fair comparison, we reuse the experimental environments of TimeGAN for the discriminative score. However, we found that TimeGAN's predictive task is rather straightforward as shown in Fig. 8 (a). It predicts only one element in yellow from other four past elements in blue. Since only one element is used for evaluation, we found that the original predictive score of TimeGAN can be biased. Instead, our predictive task predicts the entire vector, as shown in Fig. 8 (b), and therefore, our predictive score is measured under much more challenging environments. We use this more challenging predictive score definition for our irregular time series synthesis. We stick to the TimeGAN's definition for the regular time series experiment for fair comparison but use our challenging predictive score metric for all other experiments.

E Additional ablation studies

In Tables 12 to 17, we report the missing ablation study tables in the main paper.

Table 12: Ablation study for training options with the irregular time series (30% dropped)

Metric	Method	Sines	Stocks	Energy	MuJoCo
Discriminative	GT-GAN	**.363**	**.251**	**.333**	.249
Score	w/o Eq. (8)	.498	.266	.392	.303
(Lower the Better)	w/o pre-training	.499	.305	.345	**.241**
Predictive	GT-GAN	**.099**	.021	.066	**.048**
Score	w/o Eq. (8)	.241	**.015**	.064	.061
(Lower the Better)	w/o pre-training	.273	.022	**.061**	.049

Table 13: Ablation study for training options with the irregular time series (50% dropped)

Metric	Method	Sines	Stocks	Energy	MuJoCo
Discriminative	GT-GAN	**.372**	.265	**.317**	**.270**
Score	w/o Eq. (8)	.500	.323	.381	.274
(Lower the Better)	w/o pre-training	.500	**.209**	.325	**.270**
Predictive	GT-GAN	**.101**	.018	.064	.056
Score	w/o Eq. (8)	.277	.018	**.063**	**.051**
(Lower the Better)	w/o pre-training	.103	**.017**	.071	**.051**

Table 14: Ablation study for training options with the irregular time series (70% dropped)

Metric	Method	Sines	Stocks	Energy	MuJoCo
Discriminative	GT-GAN	**.278**	**.230**	**.325**	.275
Score	w/o Eq. (8)	.319	.274	.382	.290
(Lower the Better)	w/o pre-training	.408	.311	.345	**.249**
Predictive	GT-GAN	.088	**.020**	.076	.052
Score	w/o Eq. (8)	**.082**	.025	**.066**	.051
(Lower the Better)	w/o pre-training	.104	**.020**	.085	**.049**

Table 15: Ablation study for model architecture in Sines

Sines	GT-GAN (w.o. AE)		GT-GAN (Flow only)		GT-GAN (AE only)		GT-GAN (Full model)	
Metric	Disc.	Pred.	Disc.	Pred.	Disc.	Pred.	Disc.	Pred.
30% dropped	.472	.262	.500	.513	.477	.270	**.363**	**.099**
50% dropped	.480	.254	.500	.610	.475	.253	**.372**	**.101**
70% dropped	.481	.248	.499	.614	.477	.266	**.278**	**.088**

Table 16: Ablation study for model architecture in Stocks

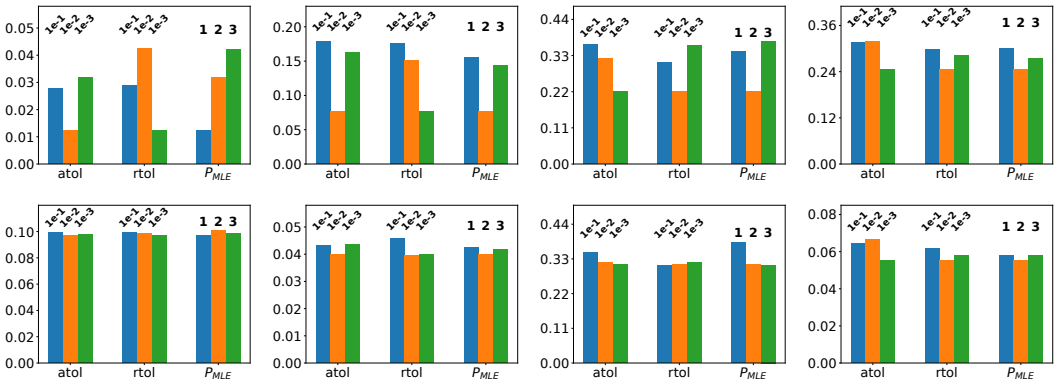
Stocks	GT-GAN (w.o. AE)		GT-GAN (Flow only)		GT-GAN (AE only)		GT-GAN (Full model)	
Metric	Disc.	Pred.	Disc.	Pred.	Disc.	Pred.	Disc.	Pred.
30% dropped	.500	.088	.492	.140	.486	.088	**.251**	**.021**
50% dropped	.500	.088	.491	.128	.492	.125	**.265**	**.018**
70% dropped	.500	.088	.490	.128	.492	.122	**.230**	**.020**

Table 17: Ablation study for model architecture in Energy

Energy	GT-GAN (w.o. AE)		GT-GAN (Flow only)		GT-GAN (AE only)		GT-GAN (Full model)	
Metric	Disc.	Pred.	Disc.	Pred.	Disc.	Pred.	Disc.	Pred.
30% dropped	.500	.305	.498	.252	.495	.162	**.333**	**.066**
50% dropped	.500	.365	.499	.160	.499	.135	**.317**	**.064**
70% dropped	.499	.376	.499	.184	.499	.131	**.325**	**.076**

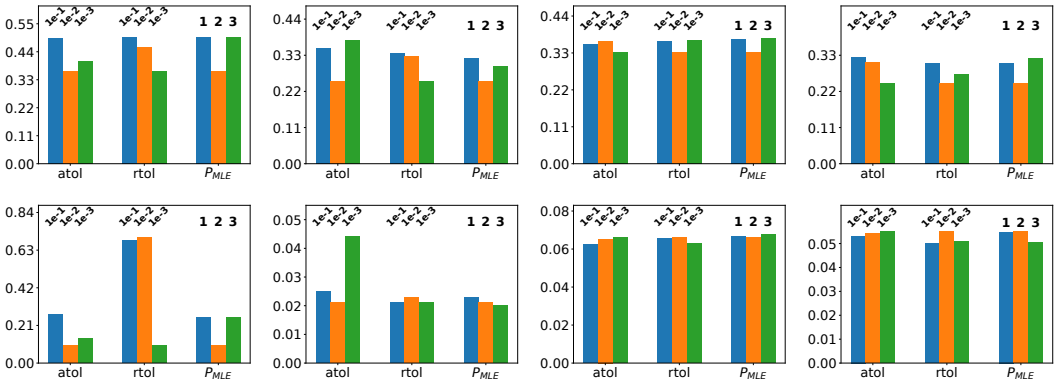
F Sensitivity analyses

We provide performance (discriminative score and predictive score) depending on hyperparameters (i.e. atol (absolute tolerance)= $\{1e-1, 1e-2, 1e-3\}$, rtol (relative tolerance)= $\{1e-1, 1e-2, 1e-3\}$ and $P_{MLE} = \{1, 2, 3\}$) for each different datasets.



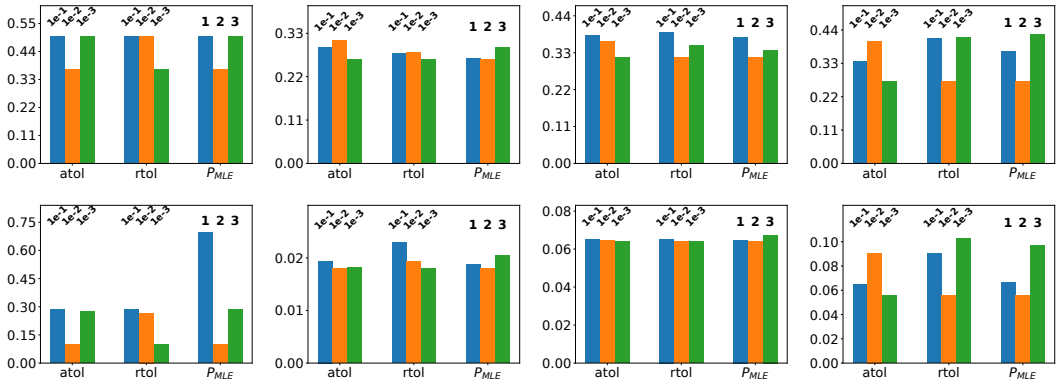
(a) Sines (b) Stocks (c) Energy (d) MuJoCo

Figure 9: The sensitivity of the discriminative score (the 1st row) and predictive score (the 2nd row) w.r.t. some key hyperparameters for regular data



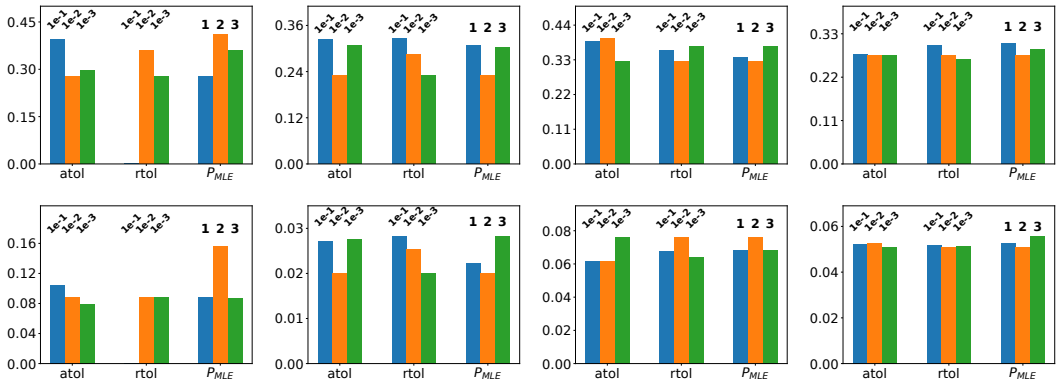
(a) Sines (b) Stocks (c) Energy (d) MuJoCo

Figure 10: The sensitivity of the discriminative score (the 1st row) and predictive score (the 2nd row) w.r.t. some key hyperparameters for irregular data (dropped 30%)



(a) Sines (b) Stocks (c) Energy (d) MuJoCo

Figure 11: The sensitivity of the discriminative score (the 1st row) and predictive score (the 2nd row) w.r.t. some key hyperparameters for irregular data (dropped 50%)



(a) Sines (b) Stocks (c) Energy (d) MuJoCo

Figure 12: The sensitivity of the discriminative score (the 1st row) and predictive score (the 2nd row) w.r.t. some key hyperparameters for irregular data (dropped 70%)

G The best hyperparamter set for GT-GAN

- 'atol' means absolute tolerance for the generator.
- 'rtol' means relative tolerance for the generator.
- 'P_{MLE}'means the period of the negative log-density training for the generator.
- 'K_{AE}' means the autoencoder's pre-training iteration numbers.
- 'd-layer' means the number of discriminator's GRU layers.
- 'r-acti' means the last activation function of the decoder.
- 'reg-recon' means the reconstruction regularization for the generator.
- 'reg-kinetic' means the kinetic-energy regularization for the generator.
- 'reg-jacobian' means the Jacobian-norm2 regularization for the generator.
- 'reg-direct' means the directional-penalty regularization for the generator.

Table 18: The best hyperparameters

method	Data	atol	rtol	P_{MLE}	K_{AE}	d-layer	r-acti	reg-recon	reg-kinetic	reg-jacobian	reg-direct
GT-GAN (Regular)	Sines	1e-2	1e-3	1	5000	1	softplus	0.01	0.05	0.1	0.1
	Stocks	1e-2	1e-3	2	10000	1	softplue	0.01	0.01	0.05	0.01
	Energy	1e-3	1e-2	2	5000	2	sigmoid	0.01	0.5	0.1	0.01
	MuJoCo	1e-3	1e-2	2	5000	2	sigmoid	0.01	0.05	0.01	0.01
GT-GAN (Dropped 30%)	Sines	1e-2	1e-3	2	5000	1	softplus	0.01	0.05	0.01	0.01
	Stocks	1e-2	1e-3	2	10000	1	softplue	0.01	None	None	0.05
	Energy	1e-3	1e-2	2	5000	2	sigmoid	0.01	0.5	0.1	0.01
	MuJoCo	1e-3	1e-2	2	2500	2	sigmoid	0.01	0.5	0.1	0.01
GT-GAN (Dropped 50%)	Sines	1e-2	1e-3	2	5000	2	softplus	0.01	0.05	0.01	0.01
	Stocks	1e-3	1e-3	2	10000	1	softplue	None	0.05	0.01	0.05
	Energy	1e-3	1e-2	2	5000	2	sigmoid	0.01	0.5	0.1	0.01
	MuJoCo	1e-3	1e-2	2	1500	2	sigmoid	0.1	0.1	0.01	0.01
GT-GAN (Dropped 70%)	Sines	1e-2	1e-3	2	5000	1	softplus	0.01	0.05	0.01	0.01
	Stocks	1e-2	1e-3	1	10000	1	softplue	None	0.05	0.01	0.05
	Energy	1e-3	1e-2	2	2500	2	sigmoid	0.01	0.5	0.1	0.01
	MuJoCo	1e-3	1e-2	2	2500	2	sigmoid	0.01	0.5	0.1	0.01

H Model size & training time comparison

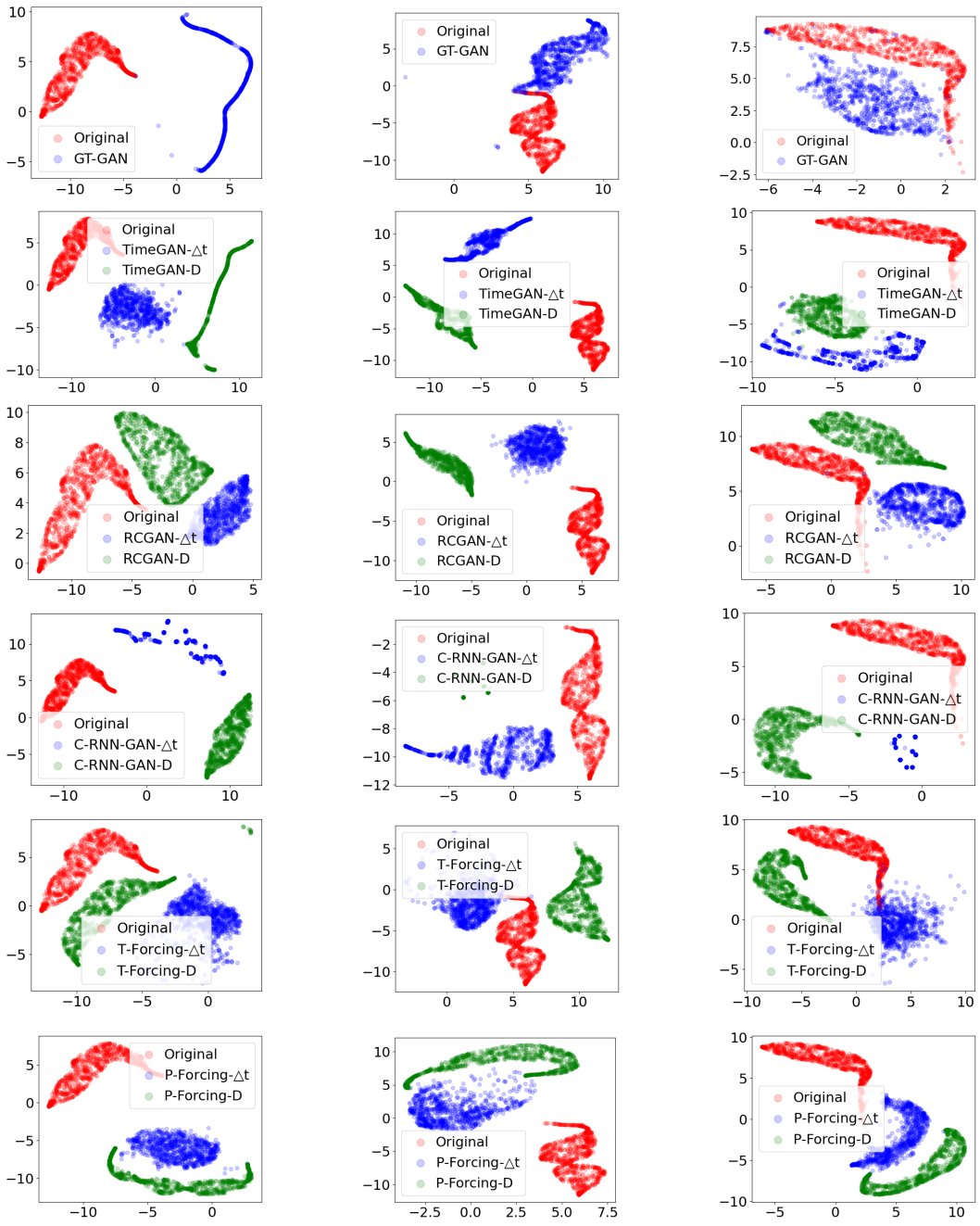
In Table 19, we report the model size and training time of our method and TimeGAN, one of the best performing baseline. As shown, our model has much smaller numbers of parameters than TimeGAN. However, it take much longer time to train our model than TimeGAN. This is mainly because we need to solve various differential equations, which is not needed for TimeGAN. The memory requirements are more or less the same in both models. Therefore, these exist pros and cons for our method in comparison with the state-of-the-art baseline.

Table 19: Comparison of model size and training time

	Sines		Stocks		Energy		MuJoCo	
Model	GT-GAN	TimeGAN	GT-GAN	TimeGAN	GT-GAN	TimeGAN	GT-GAN	TimeGAN
Parameter	41,913	34,026	41,776	48,775	57,104	1,043,179	47,346	264,447
Memory (MB)	1,675	1,419	1,653	1,423	1,839	1,611	1,655	1,546
Training Time (HH:MM)	10:12	2:56	12:20	2:59	10:39	3:37	13:12	3:10

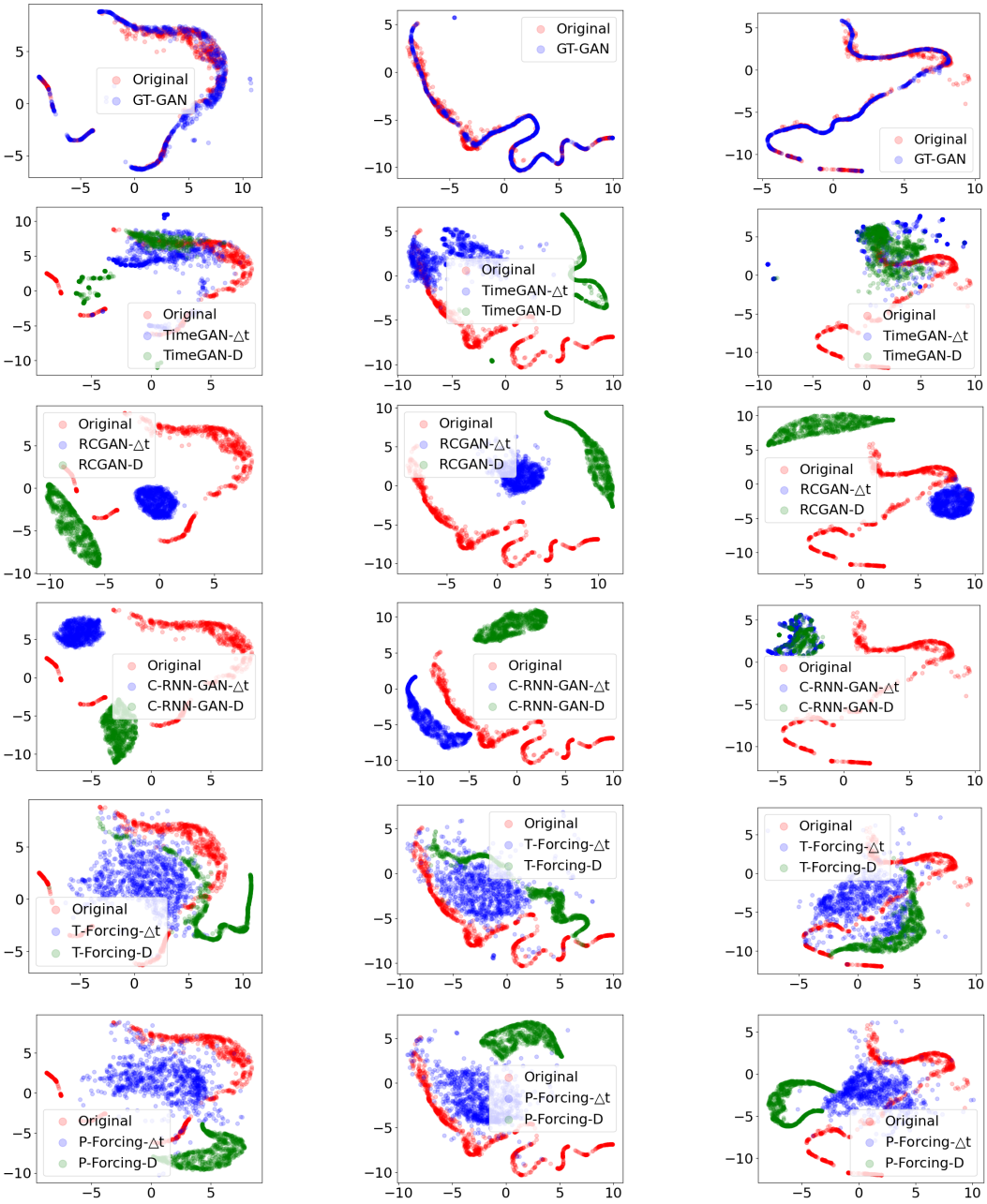
I Visualizations with t-SNE and data distribution

We introduce additional visualization outcomes in Figs. 13 to 20.



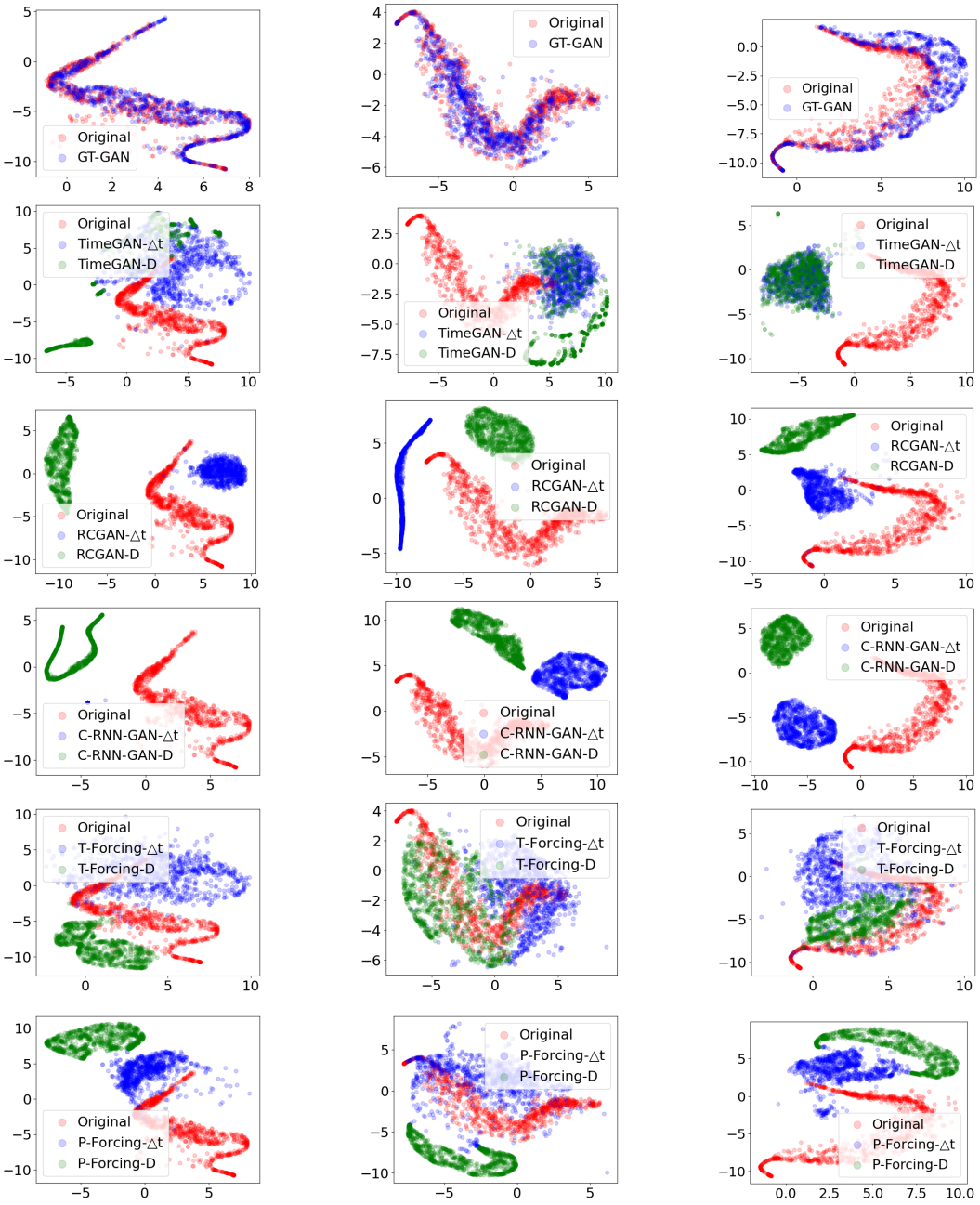
(a) 30% (b) 50% (c) 70%

Figure 13: t-SNE visualization of recovered irregular Sines data (the 1st column is for a dropping rate of 30%, the 2nd column for a rate of 50%, and the 3rd column for a rate of 70%)



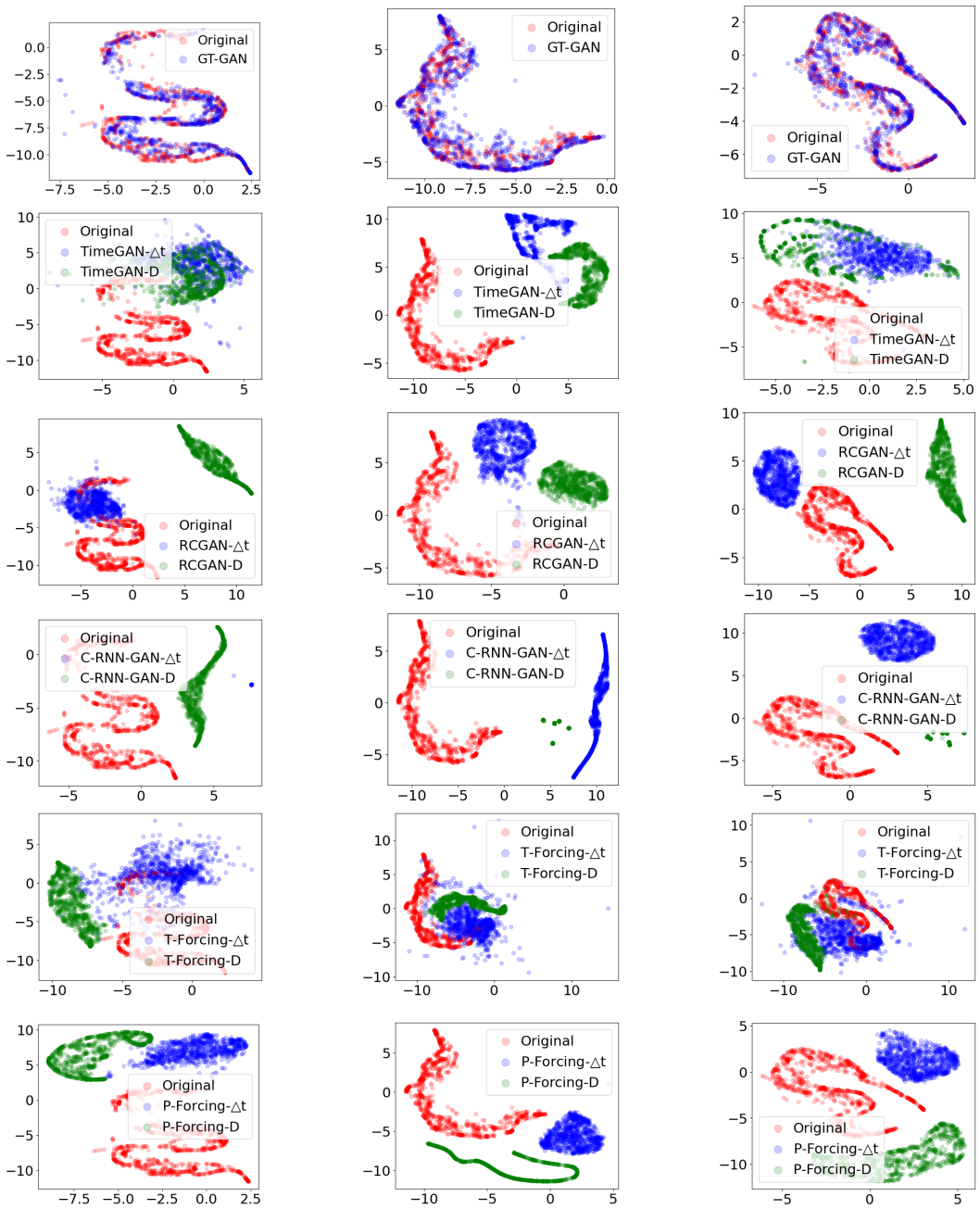
(a) 30% (b) 50% (c) 70%

Figure 14: t-SNE visualization of recovered irregular Stocks data (the 1st column is for a dropping rate of 30%, the 2nd column for a rate of 50%, and the 3rd column for a rate of 70%)



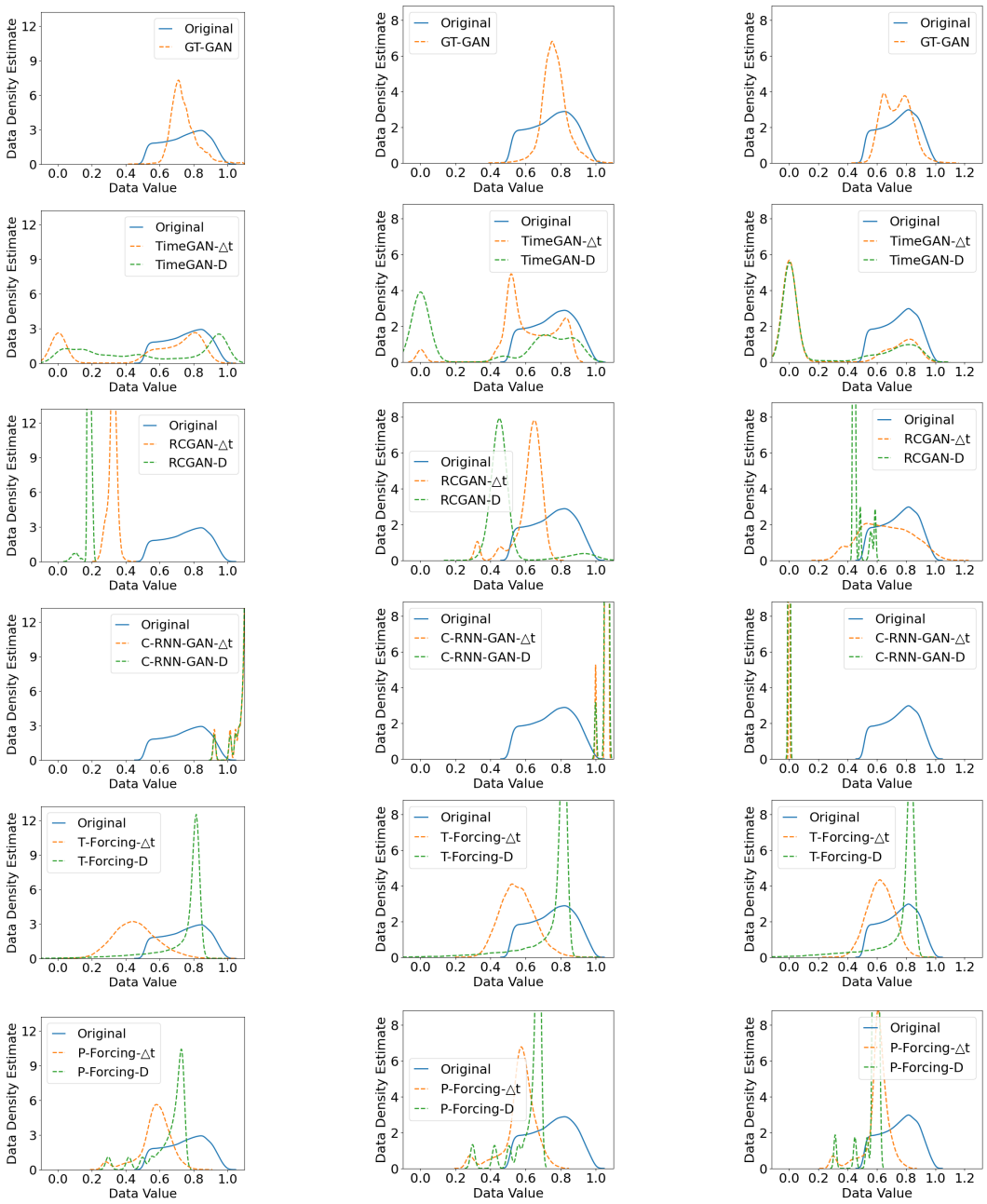
(a) 30% (b) 50% (c) 70%

Figure 15: t-SNE visualization of recovered irregular Energy data (the 1st column is for a dropping rate of 30%, the 2nd column for a rate of 50%, and the 3rd column for a rate of 70%)



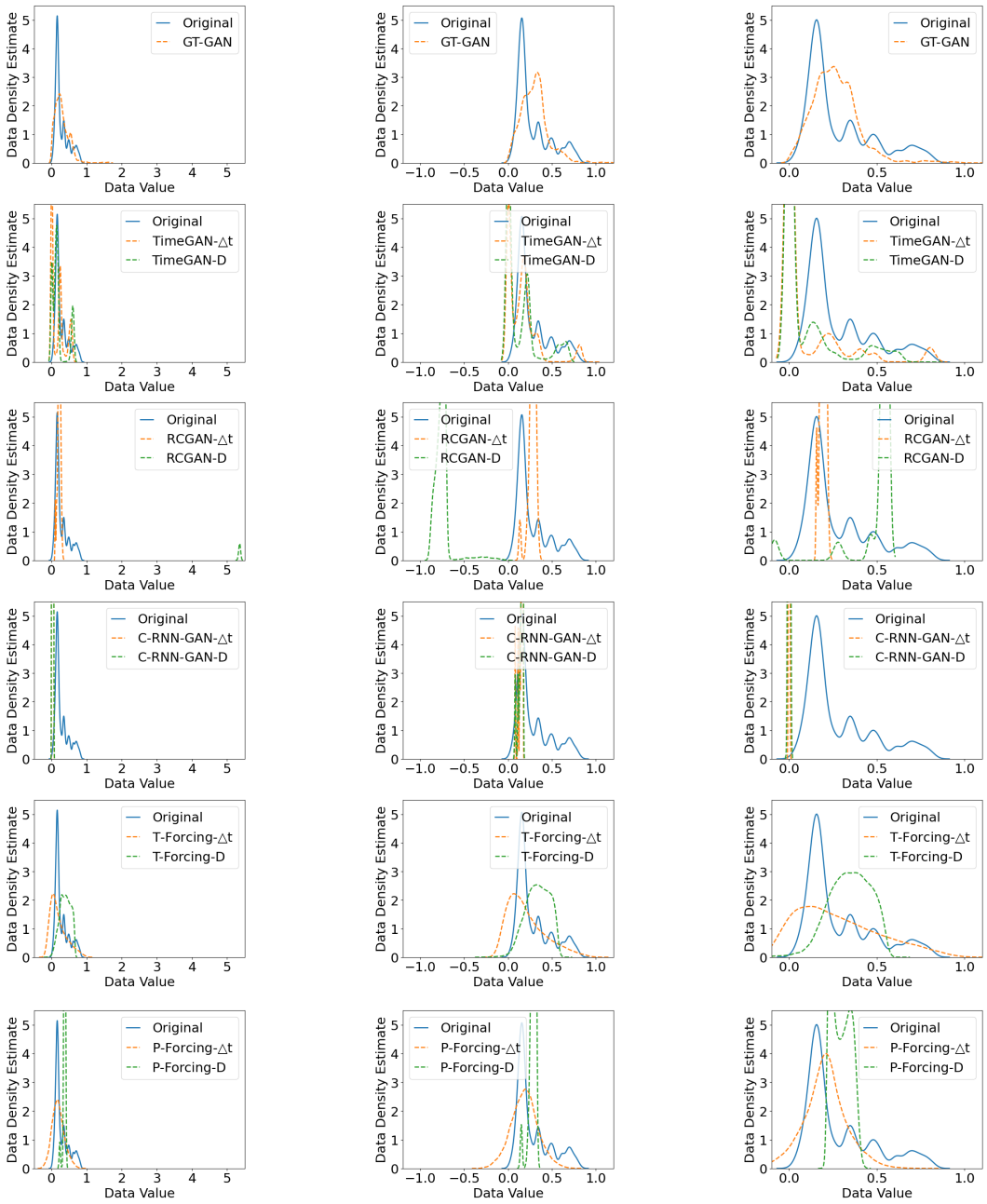
(a) 30% (b) 50% (c) 70%

Figure 16: t-SNE visualization of recovered irregular MuJoCo data (the 1st column is for a dropping rate of 30%, the 2nd column for a rate of 50%, and the 3rd column for a rate of 70%)



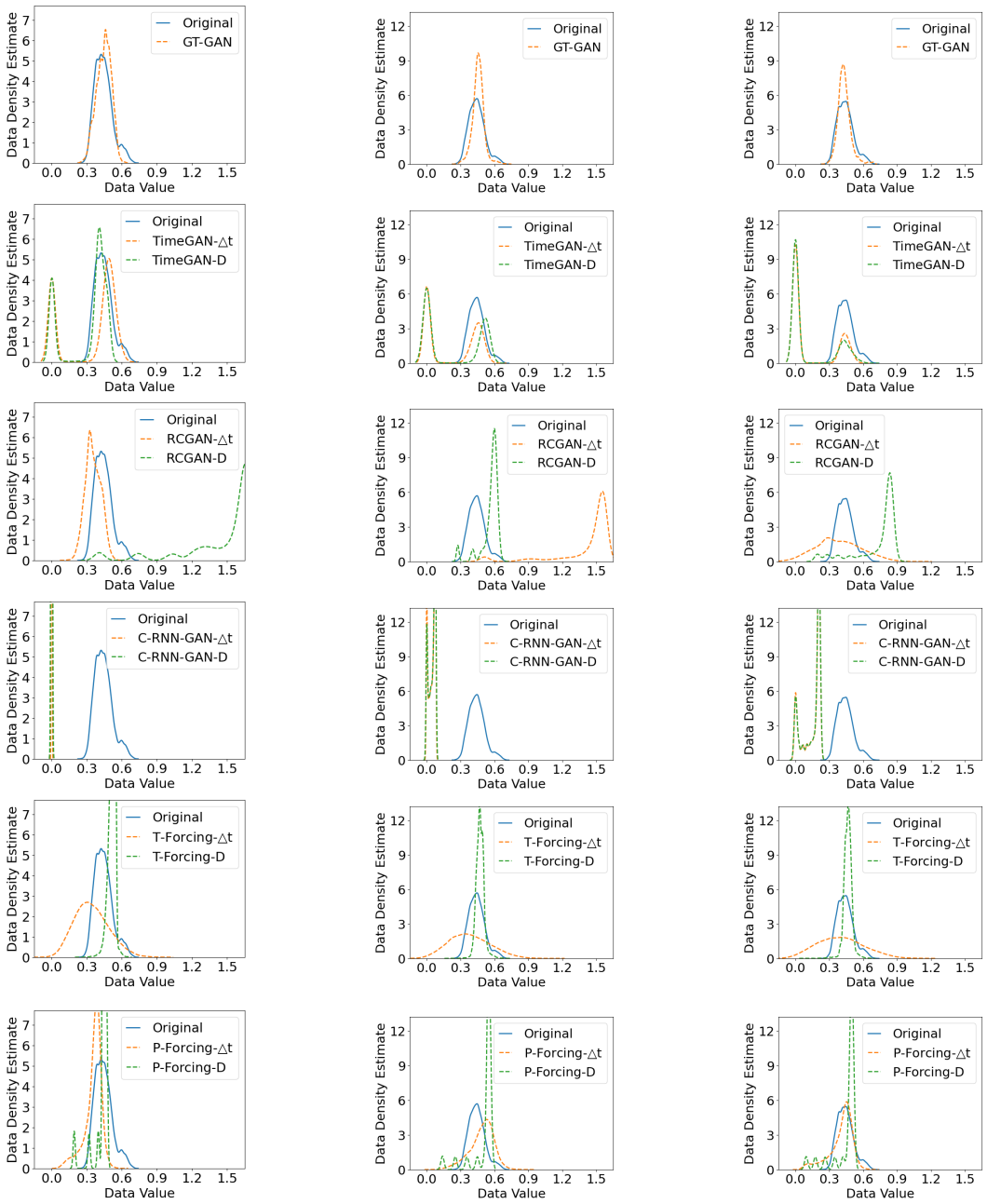
(a) 30% (b) 50% (c) 70%

Figure 17: Distributions of the Sines data (the 1st column is for a dropping rate of 30%, the 2nd column for a rate of 50%, and the 3rd column for a rate of 70%)



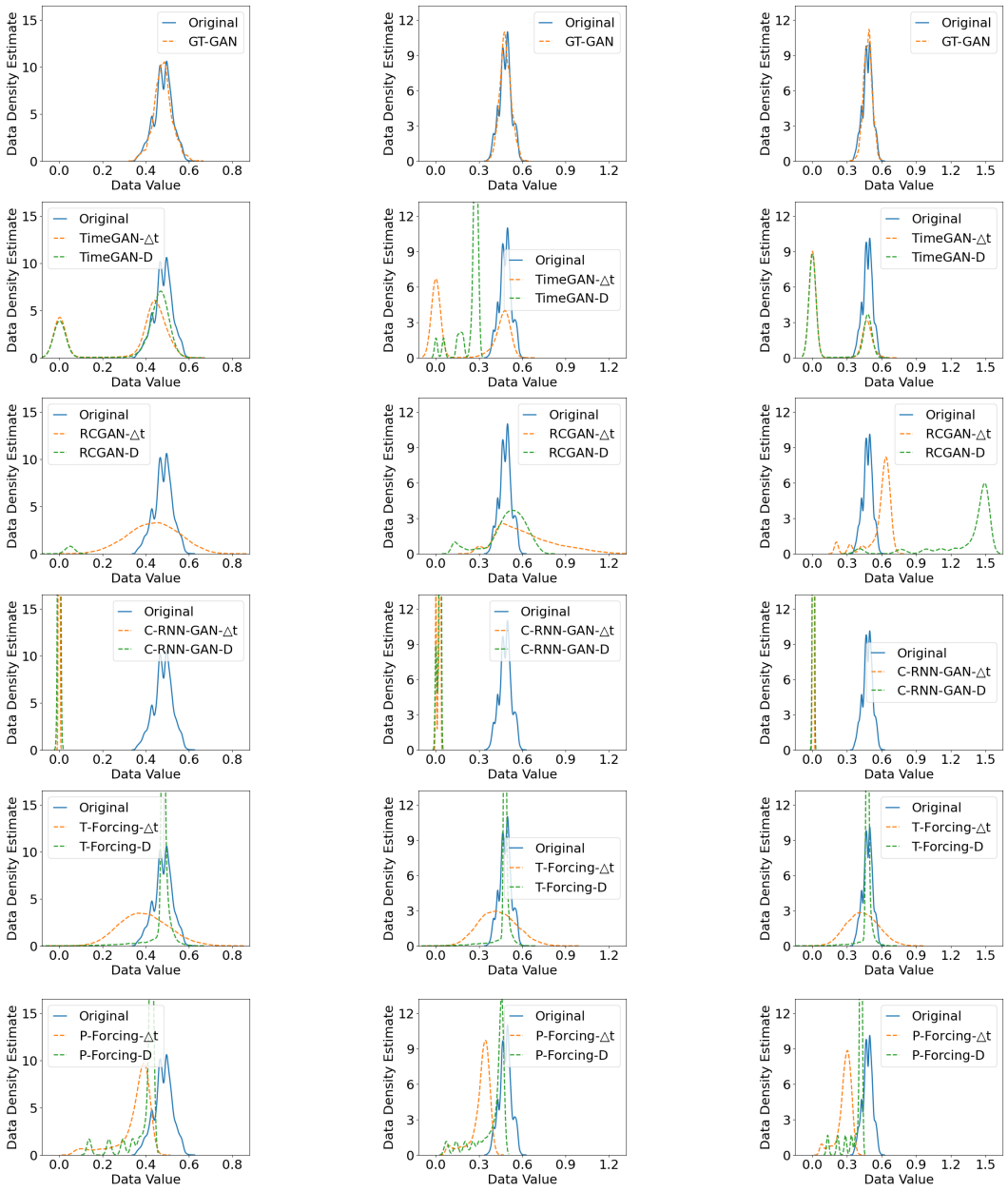
(a) 30% (b) 50% (c) 70%

Figure 18: Distributions of the Stocks data (the 1st column is for a dropping rate of 30%, the 2nd column for a rate of 50%, and the 3rd column for a rate of 70%)



(a) 30% (b) 50% (c) 70%

Figure 19: Distributions of the Energy data (the 1st column is for a dropping rate of 30%, the 2nd column for a rate of 50%, and the 3rd column for a rate of 70%)



(a) 30% (b) 50% (c) 70%

Figure 20: Distributions of the MuJoCo data (the 1st column is for a dropping rate of 30%, the 2nd column for a rate of 50%, and the 3rd column for a rate of 70%)

J Algorithm

Algorithm 1: How to train `GT-GAN`

Input: Pre-train iteration number K_{AE}, Joint-train iteration number K_{JOINT}, MLE train period P_{MLE}, Encoder $\boldsymbol{\theta}_f$, Decoder $\boldsymbol{\theta}_g$, Generator $\boldsymbol{\theta}_r$, and Discriminator $\boldsymbol{\theta}_q$

Initialize $\boldsymbol{\theta}_f$, $\boldsymbol{\theta}_g$, $\boldsymbol{\theta}_r$ and $\boldsymbol{\theta}_q$;
$k \leftarrow 0$;
while $k < K_{AE}$ **do**
  $\mathbf{h}_{real} \leftarrow$ Encoder($\mathbf{x}_{real}$;$\boldsymbol{\theta}_f$);
  $\hat{\mathbf{x}}_{real} \leftarrow$ Decoder($\mathbf{h}_{real}$;$\boldsymbol{\theta}_g$);
  Update $\boldsymbol{\theta}_f$ and $\boldsymbol{\theta}_g$ with $\|\mathbf{x}_{real} - \hat{\mathbf{x}}_{real}\|^2$;
  $k \leftarrow k + 1$;
end
$4\ k \leftarrow 0$;
while $k < K_{JOINT}$ **do**
  $\mathbf{h}_{real} \leftarrow$ Encoder($\mathbf{x}_{real}$;$\boldsymbol{\theta}_f$);
  $\hat{\mathbf{x}}_{real} \leftarrow$ Decoder($\mathbf{h}_{real}$;$\boldsymbol{\theta}_g$);
  Update $\boldsymbol{\theta}_f$ and $\boldsymbol{\theta}_g$ with $\|\mathbf{x}_{real} - \hat{\mathbf{x}}_{real}\|^2$;
  if $k \mod P_{MLE} \equiv 0$ **then**
    $\hat{\mathbf{z}} \leftarrow$ Generator$^{-1}(\mathbf{h}_{real}, \boldsymbol{\theta}_r)$;
    $\hat{\mathbf{h}}_{real} \leftarrow$ Generator($\hat{\mathbf{z}}, \boldsymbol{\theta}_r$) ;
    Update $\boldsymbol{\theta}_r$ with $-\log \Pr(\hat{\mathbf{h}}_{real})$;
  end
  $\mathbf{h}_{fake} \leftarrow$ Generator($\mathbf{z}, \boldsymbol{\theta}_r$);
  $\mathbf{x}_{fake} \leftarrow$ Decoder($\mathbf{h}_{fake}, \boldsymbol{\theta}_g$);
  Update $\boldsymbol{\theta}_r$ and $\boldsymbol{\theta}_q$ with the adversarial loss with Discriminator($\mathbf{x}_{fake}, \mathbf{x}_{real}, \boldsymbol{\theta}_q$);
  $k \leftarrow k + 1$;
end

We describe the training method in Alg. (1). We first pre-train the autoencoder in the first while loop, followed by the second while loop for the main training step. The main training step consists of i) fine-tuning the autoencoder, ii) training the generation with the log-density loss, iii) training the GAN part with the adversarial loss.

K Efficacy of the log-density training

In order to see the efficacy of the log-density training, we conduct two more studies. The first model GT-GAN (w/o Eq. (8)) is a model in which the generator is trained only with adversarial loss. The second model GT-GAN (supervised loss) replacing the log-density loss to a supervised loss. To obtain the supervised loss, like TimeGAN, we added a supervisor network between the encoder and decoder.

Table 20: Ablation study for log-likelihood training

Stocks (Regular)	Discriminative Score	Predictive Score
GT-GAN	.077	.040
GT-GAN (w/o Eq. (8))	.159	.043
GT-GAN (supervised loss)	.124	.037

According to the above results, it was confirmed that even if TimeGAN's supervised loss is used, no better results than those of our original design are obtained (the predictive score is slightly improved though). In other words, this experiment confirms the importance of the log-density path in our model.

Table 21: Regular time series

Stocks (Regular)	Discriminative Score	Predictive Score
GT-GAN	.077	.040
TimeGAN (NCDE)	.183	.036
GT-GAN (GRU-$\triangle t$)	.184	.041

Table 22: Irregular time series (30% dropped)

Stocks (30% dropped)	Discriminative Score	Predictive Score
GT-GAN	.077	.021
TimeGAN (NCDE)	.430	.036
GT-GAN (GRU-$\triangle t$)	.345	.022

L Efficacy of the NCDE-based encoder

We execute two experiments to justify using an NCDE-based encoder. First, we experiment by replacing the encoder of TimeGAN with our NCDE-based encoder. Second, the NCDE-based encoder of GT-GAN is changed to GRU-$\triangle t$. The results are in Tables 21 and 22. Our model shows the best outcomes when we use the NCDE-based encoder.

M Role of each network

Although our model looks complicated, we use an appropriate network for each part to fit its role. The role of each part is as follows:

Encoder The neural CDE-based encoder is able to encode a regular/irregular time series sample into a regular/irregular hidden vector sequence. Neural CDEs are sometimes called continuous RNNs and are specially designed for the representation learning of irregular time series. As reported in our first email, generation quality is severely degraded when this network is substituted with GRU-$\triangle t$.

Decoder The GRU-ODE-based decoder is able to decode a regular/irregular hidden vector sequence into a regular/irregular time series sample. One beauty of this decoder is, as shown in Fig. 2 in our main paper, that the sampling time point and the sample length can be freely determined by users.

Generator The CTFP-based invertible generator was intentionally selected by us since we can perform both the log-likelihood and the adversarial training together. Since this network is a key part of our model, we wanted to use the two different training paradigms. Our ablation studies about the log-likelihood and supervised-learning training in Table. 20 justify our design selection.

Discriminator The GRU-ODE-based discriminator is able to process regular/irregular time series. Unlike the encoding task of the neural CDE-based encoder, we observed faster and better results with the GRU-ODE-based discriminator. Moreover, neural CDEs require interpolation of input as a pre-processing. We can do this for real data before training. However, it is hard to perform dynamically for the fake hidden vector sequence due to its excessive computation amount. In particular, it significantly delays the overall training process if we use a neural CDE-based discriminator.

In general, our key design points lie in utilizing i) the continuous-time method-based autoencoder, and ii) the CTFP-based generator. Therefore, we can stabilize the generating performance for complicated irregular time series as well.

N Discriminative vs. predictive score

In Table 5, GT-GAN without pre-training performs better than GT-GAN in terms of the predictive score. In Fig. 21, however, GT-GAN generates samples a little out of the original data distribution whereas GT-GAN w/o pre-training has a severe mode-collapse problem (i.e., generating in a narrow region). We conjecture that those samples a little outside the original data distribution make the prediction tasks' scores a little low. However, note that GT-GAN can successfully recall almost the entire data region.

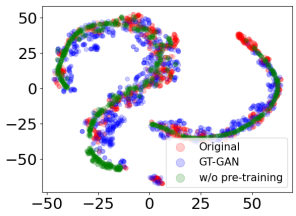


Figure 21: t-SNE visualization of GT-GAN and GT-GAN (w/o pre-training)

O Discussions

Limitations Our model shows the best performance in both regular and irregular time series synthesis. However, since our model has a complicated architecture, many hyperparameters exist. Sometimes it is hard to train such large models, which involves a large scale hyperparameter search.

Societal impacts Time series data is one of the most widely used data in the field of machine learning. In many cases, time series data carries sensitive personal information, in which case one can use our method to synthesize fake time series and protect privacy. Likewise, we believe that our method has much more positive impacts on our society than negative ones.