Supplementary Material for Masked Autoencoder with Time Series Generation

Anonymous Author(s)
Affiliation
Address
email

The source code is under the folder *MAI*. The reader may also download it from Github. You can check the results in jupyter notes *ExtraMAE.ipynb*, *MaskRatio.ipynb*, and *Imputation.ipynb* to save your time. The jupyter notes only show final results, and readers have to download the full datasets and model files to run the jupyter notes. Since the full datasets and model files are too large, the supporting codes and files for *ExtraMAE.ipynb*, *MaskRatio.ipynb*, and *Imputation.ipynb* are available upon request.

1 Visualization

We provide t-SNE [1] plots for all candidate models (ExtraMAE in row1, TimeGAN [2] in row2, RC-GAN [3] in row3, and C-RNN-GAN [4] in row4). We randomly select six features to plot. Each column visualizes one feature. Features from left to right are Close, Volume, Sine1, Sine4, Application, and T1.

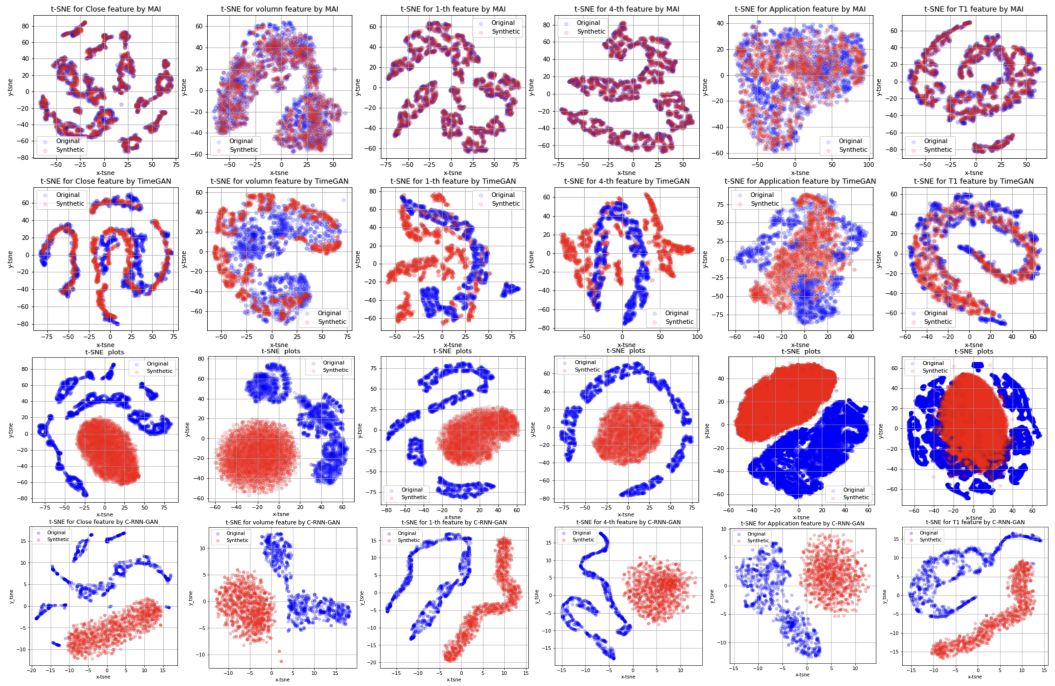


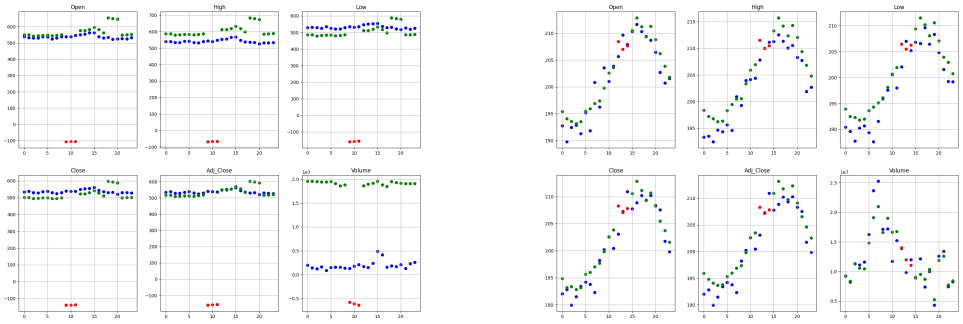
Figure 1: Visualization by t-SNE.

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

We note that RC-GAN and C-RNN-GAN have terrible performance. The manifolds of synthetic data (red dots) are entirely different from the manifold of original data (blue dots) on row 3 (RCGAN) and row 4 (C-RNN-GAN). However, synthetic data (red dots in the 1st row) generated by ExtraMAE is in perfect sync with the original data (blue dots).

2 Avoid mask tokens

Figure 2: Illustration for ExtraMAE with mask tokens *v.s* ExtraMAE without mask tokens. We randomly pick up a time series from Stock. Since each time series in Stock has six features, we plot each feature as an independent univariate time series. We first visualize the original time series in blue dots. Then, we mask three consecutive positions in the original time series. ExtraMAE reconstructs the original signal from the partial observation. Then, we plot the reconstructed time series in two colors. For positions visible to ExtraMAE, the reconstructed time series are green dots. For masked positions, we plot the reconstructed time series with red dots. ExtraMAE with mask tokens generates highly discrete reconstruction, but ExtraMAE without mask tokens reconstructs the original signal perfectly.



(a) With mask tokens

(b) Without mask tokens

We compare ExtraMAE with mask tokens $v.s$ ExtraMAE without mask tokens in Figure 2. ExtraMAE with mask tokens (figure on the left) generates highly discontinuous reconstruction. Note that the time series takes a sudden jump in the missing positions (red dots). We guess the reason for such discontinuity is mask tokens. Therefore, we introduce an extrapolator in our paper to avoid mask tokens. The ExtraMAE with extrapolator avoids mask tokens (figure on the right) and reconstructs the original signal perfectly.

3 Benchmarks

We implements all benchmarks in experiments from their original source codes.

1. TimeGAN https://github.com/jsyoon0823/TimeGAN
2. RCGAN https://github. com/ratschlab/RGAN
3. C-RNN-GAN https://github. com/olofmogren/c-rnn-gan
4. Mean, Median, Soft, and KNN https://github.com/iskandr/fancyimpute
5. BRITS https://github.com/caow13/BRITS

4 Consumption

Simple algorithms that perform well lie at the core of deep learning. We prune stacked Transformer blocks which are pretty popular in previous masking-based models. ExtraMAE replaces the heavy Transformer blocks with lightweight RNNs [5]. The pruning makes ExtraMAE the fastest model for time series generation.

Table 1 lists the number of parameters in each model and the time consumption for 50000 epochs on Stock. We run all experiments on a GeForce RTX 2080 GPU. Table 1 shows that C-RNN-GAN is heavy and slow. RC-GAN is the smallest model, but its performance is terrible. TimeGAN achieves better fidelity and practicality than RC-GAN, but it is much bigger than RC-GAN. TimeGAN also takes a longer time to train. ExtraMAE is the fastest model with only 2718 trainable parameters. ExtraMAE only needs 0.24 hours to finish 50000 epochs on Stock. The time ExtraMAE consumed is 4.1%, 9.6%, and 0.3% of the time required by TimeGAN, RC-GAN, and C-RNN-GAN, respectively.

Table 1: Time consumption and the number of parameters on the stock dataset. The unit h in the column Time is *hours*

Model	Params	Time	Pred	Disc
ExtraMAE	2718	0.24h	.037	.071
TimeGAN	48775	5.80h	.053	.256
RCGAN	1218	2.50h	.375	.488
C-RNN-GAN	2523258	72.0h	.086	.500

Table 1 also provides the prediction scores (Pred) and discrimination scores (Disc) as references for the synthetic time series' practicality and fidelity. Both scores are the lower, the better. We note that ExtraMAE is consistently and significantly better than other benchmarks in practicality and fidelity.

5 TSTR Prediction

5.1 TSTR Framework

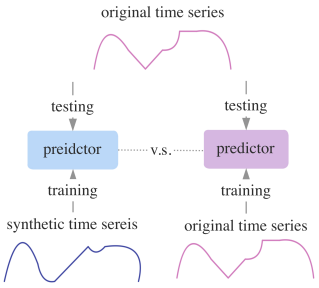


Figure 3: Illustration for TSTR Framework.

We consider TSTR on prediction. Usually, we train a predictor (pink on the right) on the original time series and then test it on the original time series. In TSTR, however, we train the predictor (blue on the left) on synthetic time series but test it on the original time series. Then, we compare the predictor trained by synthetic time series (blue on the left) and the predictor trained by the original time series (pink on the right). The blue predictor is supposed to have a similar performance to the pink one if the synthetic data is a good substitute for the original data.

5.2 Statistics of datasets

We provide additional statistics on Stock, Sine, and Energy in Table 2. The Google Stocks dataset is available online and can be downloaded from LINK. The UCI Appliances Energy Prediction dataset is also available online and can be downloaded from LINK. Both Stock and Energy are public, and Sine is simulated. These three datasets are also available in the source code. The datasets contain no personal information or offensive information.

5.3 Hyperparameters

We train ExtraMAE 50000 epochs on each dataset since 50000 is the minimal number of epochs that gives satisfactory results. We also try different batch sizes (i.e., 128, 256, 512, 1024) and found that the batch size does not significantly influence the model performance. For each dataset, We reserve

Table 2: Statistics of Stock, Sine and Energy. *sequences* is the number of time sereis in the dataset. We also provide the feature correlations and temporal correlations of the three datasets.

dataset	sequences	features	length	feature corr	temporal variance	temporal corr
Stock	3773	6	24	.8596	.0129	.9902
Sine	10000	5	24	.0117	.3167	.2056
Energy	19711	29	24	.2843	.0444	.8506

80% to train ExtraMAE and test it on the left 20%. In Table 3, we list parameters of ExtraMAE when calculating discrimination scores and prediction scores. Note that L is the length of each time series, and k is the number of features in each time series. The dataset determines both L and k.

Table 3: Hyperparameters of ExtraMAE on Stock, Sine and Energy

dataset	epochs	batch size	L	k	h	layers	split ratio
Stock	50000	128	24	6	12	3	0.8
Sine	50000	128	24	5	10	3	0.8
Energy	50000	128	24	28	12	3	0.8

In ExtraMAE, we implement both encoder and decoder as stacked RNNs followed by a fully connected layer. The number of layers stacked in the encoder and decoder is the same. In Table 3, the number of RNN layers stacked is denoted as *layers*. According to Table 3, we implement both encoder and decoder as 3-layer RNNs followed by a fully connected layer. We find that empirically, h (the dimension of hidden vector in RNNs) should be twice the number of features k. However, for time series with many features (e.g., $k = 28$ in Energy), $h = 12$ is enough.

6 TSTR classification

6.1 Datasets

We first describe the three datasets for TSTR classification [3]. Readers may download these three public datasets from LINK. The datasets contain no personal information or offensive content.

Wafer. Wafer [6] dataset relates to semiconductor microelectronics fabrication. This dataset consists of several measurements recorded by sensors in the processing of wafers. The dataset marks time series data of qualified wafers as normal and labels time-series data produced by defective wafers as abnormal. The classification problem distinguishes between normal time series and abnormal time series.

IPD. IPD [7] stands for Italy Power Demand. This dataset records daily electrical power demand in Italy for twelve months. The electricity demand patterns in daylight saving time (from April to September) are quite different from patterns in standard time (from October to March). The classifier distinguishes time series in daylight saving time from standard time.

Berry. Berry [8] records sequential spectrographs for fruits. We classify the fruits into two groups, strawberry and non-strawberry.

6.2 Synthetic labels

We then discuss how to generate labels for the synthetic time series. In this process, there is no modification on ExtraMAE, TimeGAN, RC-GAN, and C-RNN-GAN. We take the berry dataset as an example. Berry has two classes: 0 for non-strawberry and 1 for strawberry. We generate labels in such ways:

1. For ExtraMAE: We copy the label of the original time series to the synthetic time series. For instance, if the label of the original sequence is 1, the label of the synthetic twin will also be 1. Similarly, we label all synthetic twins of non-strawberry (label 0) with the label 0.

2. For unsupervised benchmarks (i.e., TimeGAN, RC-GAN, C-RNN-GAN): We train two independent GANs for sequences of strawberry and sequences belongings to non-strawberry. We take TimeGAN as an example. First, we collect all original data with label 1 and use this subset to train a TimeGAN. We call this model TimeGAN_1. Then, we collect all data with labels 0 and use it to train a new TimeGAN, TimeGAN_0. Finally, we mark synthetic data produced by TimeGAN_1 with the label 1 and synthetic data generated by TimeGAN_0 with 0.

6.3 hyperparameters

Table 4 provides hyperparameters for TSTR classification. Wafer and Berry contain long time series of length 152 and 235, but it is enough to use ExtraMAE with hidden vectors of dimension $h = 12$.

Table 4: Hyperparameters of ExtraMAE on Wafer, IPD and Berry

params	epochs	batch size	L	k	h	layers	split ratio
Wafer	50000	128	152	1	12	3	0.8
IPD	50000	128	24	1	12	3	0.8
Berry	50000	128	235	1	12	3	0.8

7 Imputation

We study how well ExtraMAE imputes missing values in the original time series. For each time series, we randomly mask some patches. We denote the number of mask as n and the mask size as s. For example, s4n1 indicates 1 mask of length 4. We compute the Mean Squared Errors (MSE) and Mean Absolute Errors (MAE) in Table 5 6 7 and 8 9 10 respectively. In Table 11 12 13, we summarize the MSE and MAE for different mask ratios. We repeat one experiment 10 times and calculate the mean and standard deviations for MSE and MAE.

7.1 MSE of imputation

Table 5: MSE for Stock

	mean	median	soft	knn	brits	mai
s1n1	0.053212	0.065393	0.000604	0.000559	0.026711	0.000457
s2n1	0.053295	0.065420	0.000718	0.000694	0.024226	0.000695
s3n1	0.052957	0.064955	0.000679	0.000630	0.023400	0.000692
s4n1	0.053323	0.065357	0.000833	0.000804	0.022601	0.000857
s6n1	0.053216	0.065142	0.000875	0.000889	0.021178	0.000865
s8n1	0.053252	0.065368	0.000986	0.001057	0.018279	0.000911
s12n1	0.052941	0.064985	0.145453	0.145453	0.014931	0.001016

Table 6: MSE for Sine

	mean	median	soft	knn	brits	mai
s1n1	0.053627	0.058219	0.000500	7.933572e-07	0.024364	0.001720
s2n1	0.053921	0.058631	0.000652	8.997953e-07	0.021749	0.003865
s3n1	0.054575	0.059366	0.000962	1.100507e-06	0.018255	0.004988
s4n1	0.053867	0.058276	0.001449	1.494250e-06	0.016074	0.001394
s6n1	0.053877	0.058417	0.004048	3.851234e-06	0.013481	0.015626
s8n1	0.053847	0.058221	0.034252	1.924960e-05	0.011687	0.014909
s12n1	0.054224	0.058897	0.346904	3.469041e-01	0.006979	0.005814

Table 7: MSE for Energy

	mean	median	soft	knn	brits	mai
s1n1	0.034796	0.035353	0.007139	0.008325	0.021030	0.009847
s2n1	0.034788	0.035357	0.007325	0.008478	0.019882	0.010233
s3n1	0.034795	0.035358	0.007590	0.008719	0.018985	0.010166
s4n1	0.034730	0.035293	0.007848	0.009051	0.018166	0.010282
s6n1	0.034811	0.035381	0.008588	0.009910	0.016355	0.010543
s8n1	0.034775	0.035345	0.009833	0.010972	0.014538	0.010858
s12n1	0.034744	0.035307	0.250505	0.250505	0.010821	0.010975

7.2 MAE of imputation

Table 8: MAE for Stock

	mean	median	soft	knn	brits	mai
s1n1	0.184157	0.168686	0.012415	0.007493	0.097953	0.006553
s2n1	0.184718	0.169127	0.012986	0.008162	0.091715	0.009037
s3n1	0.184153	0.168597	0.013111	0.008488	0.089076	0.010114
s4n1	0.184801	0.169203	0.013939	0.009450	0.084816	0.011661
s6n1	0.184504	0.168790	0.014485	0.011569	0.076509	0.012849
s8n1	0.184479	0.168921	0.015776	0.014755	0.068521	0.014008
s12n1	0.184224	0.168702	0.290512	0.290512	0.052654	0.016312

Table 9: MAE for Sine

	mean	median	soft	knn	brits	mai
s1n1	0.184304	0.179196	0.017113	0.000636	0.108003	0.026997
s2n1	0.184737	0.180204	0.018869	0.000676	0.098634	0.041812
s3n1	0.185969	0.180941	0.021697	0.000738	0.086176	0.049650
s4n1	0.184748	0.179598	0.025256	0.000850	0.077266	0.025592
s6n1	0.184748	0.180200	0.038706	0.001331	0.064739	0.075184
s8n1	0.185373	0.180006	0.130514	0.002675	0.057434	0.080307
s12n1	0.185526	0.180587	0.494050	0.494050	0.037486	0.046360

Table 10: MAE for Energy

	mean	median	soft	knn	brits	mai
s1n1	0.144148	0.141668	0.033155	0.024644	0.094619	0.028022
s2n1	0.144102	0.141647	0.034507	0.025826	0.088664	0.030739
s3n1	0.144077	0.141616	0.036125	0.027580	0.084526	0.033818
s4n1	0.143989	0.141547	0.038019	0.030145	0.080053	0.036042
s6n1	0.144140	0.141694	0.042738	0.036618	0.072286	0.040709
s8n1	0.144097	0.141652	0.049980	0.043422	0.064041	0.045425
s12n1	0.144018	0.141571	0.443258	0.443258	0.047903	0.053531

7.3 Summary for imputation

In Table 11, 12, 13, we summary the mean and standard error of MSE and MAE on the three datasets: Stock [9], Sine and Energy [10]. We note that ExtraMAE beats all benchmarks with markedly significant improvements. Comparing with the current state-of-the-art BRITS [11], ExtraMAE decrease 97% mean squared error and 86% mean absolute error on the Stock dataset ($97\% \approx \frac{|0.0007-0.0216|}{0.0216}$, $86\% \approx \frac{|0.0115-0.0802|}{0.0802}$). We also present Table 11, 12, 13 in the paper.

Table 11: Averaged MAE and MAE for Stock

method	mse mean	mse std	mae mean	mae std
mean	0.053171	0.000157	0.184434	0.000265
median	0.065231	0.000200	0.168861	0.000232
soft	0.021450	0.054681	0.053318	0.104599
knn	0.021441	0.054685	0.050061	0.106058
brits	0.021618	0.003936	0.080178	0.015581
mai	0.000785	0.000185	0.011505	0.003260

Table 12: Averaged MAE and MAE for Sine

method	mse mean	mse std	mae mean	mae std
mean	0.053991	0.000312	0.185058	0.000579
median	0.058575	0.000428	0.180105	0.000584
soft	0.055538	0.129064	0.106601	0.175513
knn	0.049562	0.131116	0.071565	0.186300
brits	0.016084	0.005977	0.075677	0.024481
mai	0.006902	0.005937	0.049415	0.021431

8 Mask ratio

8.1 Mask size

We first study prediction and discrimination scores under a single mask of different sizes. The length of the original time series is 24. We denote the number of masked positions as the mask size. Table 14 and Table 15 shows the results on Stock and Sine respectively. We evaluate the practicality by prediction scores. The lower the prediction score is, the more practical the synthetic data is. Generally, the prediction score grows when the size of the mask increases on Stock and Sine. However, the mask size is not always the smaller; the better. The optimal mask ratio is $50.0\% = \frac{12}{24}$ for Energy, $16.7\% \approx \frac{4}{24}$ for Sine, and $4.2\% \approx \frac{1}{24}$ for Stock. Table 16 shows the results for Energy. Surprisingly, the optimal mask size for Energy is 12 (mask ratio $= \frac{12}{24} = 50.0\%$). For most mask sizes, the prediction scores of ExtraMAE are significantly lower than the scores of the runner-up. We only see exceptions under a few mask sizes (i.e., s12n1 on Stock, s6n1, and s8n1 on Sine).

Table 13: Averaged MAE and MAE for Energy

method	mse mean	mse std	mae mean	mae std
mean	0.034777	0.000030	0.144081	0.000059
median	0.035342	0.000031	0.141628	0.000053
soft	0.042690	0.091643	0.096826	0.152870
knn	0.043709	0.091194	0.090213	0.155820
brits	0.017111	0.003522	0.076013	0.016033
mai	0.010415	0.000400	0.038327	0.008912

We evaluate the diversity of synthetic time series by discrimination scores. Low discrimination scores indicate faithful synthetic data, while high discrimination scores indicate diverse synthetic data. Users may manage the diversity of synthetic time series generated by ExtraMAE explicitly by changing the mask ratio. In Table 14 and Table 16, the discrimination scores are consistently lower than the discrimination score of the runner-up. ExtraMAE achieves great fidelity. In Table 15, ExtraMAE generates practical (low prediction score) but diverse (high discrimination score) data. By changing the mask size, we can manage the diversity of synthetic time series directly. Besides, ExtraMAE succeeds in making a trade-off between diversity and practicality under most mask sizes. Table 14, Table 15 and Table 16 show results on Stock, Sine, and Energy, respectively. For simplicity, we denote the mask size as s and the number of masks as n. Therefore, s4n1 stands for an ExtraMAE trained by under 1 mask of size 4. We repeat each experiment for 10 times for more comprehensive results and calculate the mean and standard deviation. For example, we train an ExtraMAE on Stock under 1 mask of size 1 for 10 times and calculate the mean and standard deviation of discrimination and prediction scores. Finally, we get the mean and standard deviation of ExtraMAE under s1n1 is 0.037010 and 0.110437.

Table 14: Results for different mask sizes on Stock.

method	pred mean	pred std	disc mean	disc std
s1n1	0.037010	0.000131	0.110437	0.116452
s2n1	0.037812	0.000141	0.055457	0.037789
s3n1	0.037264	0.000166	0.114870	0.106279
s4n1	0.040024	0.000248	0.145157	0.078061
s6n1	0.039060	0.000235	0.132196	0.100265
s8n1	0.039412	0.000474	0.102319	0.077281
s12n1	0.060575	0.001133	0.198568	0.019001

Table 15: Results for different mask sizes on Sine.

method	pred mean	pred std	disc mean	disc std
s1n1	0.102111	0.000773	0.273150	0.181229
s2n1	0.103612	0.000542	0.402000	0.163587
s3n1	0.105498	0.001218	0.323100	0.176853
s4n1	0.101471	0.000693	0.113475	0.012641
s6n1	0.136297	0.065392	0.248800	0.084467
s8n1	0.162098	0.001763	0.376700	0.024429
s12n1	0.114557	0.004032	0.370875	0.026211

Table 16: Results for different mask sizes on Energy.

method	pred mean	pred std	disc mean	disc std
s1n1	0.305518	0.010932	0.479292	0.019306
s2n1	0.311318	0.001635	0.440883	0.021442
s3n1	0.296835	0.001658	0.428849	0.014600
s4n1	0.285874	0.002197	0.439120	0.011269
s6n1	0.270136	0.002233	0.402080	0.119129
s8n1	0.278986	0.003768	0.376997	0.142961
s12n1	0.256548	0.000569	0.488372	0.014252

8.2 Number of masks

Next, we study the prediction and discrimination scores when simultaneously masking several positions in the time series. The length of the original time series is 24. We try different numbers of masks of length 1 (The number of masks ranges from 1 to 23). Table 17, Table 18, and Table 19 show the results on Stock, Sine, and Energy respectively.

Surprisingly, our ExtraMAE beats the runner up even with an extremely high mask ratio, for example, $91.7\% \approx \frac{22}{24}$ for Energy, $91.7\% \approx \frac{22}{24}$ for Sine, and $50.0\% = \frac{12}{24}$ for Stock.

Table 17: Different number of masks on Stock

	pred_mean	pred_std	disc_mean	disc_std
s1n1	0.037010	0.000131	0.110437	0.116452
s1n2	0.037449	0.000219	0.128718	0.124518
s1n3	0.037860	0.000249	0.064461	0.026741
s1n4	0.038692	0.000218	0.077285	0.050342
s1n5	0.039692	0.000314	0.163915	0.085648
s1n6	0.041617	0.000472	0.127558	0.053436
s1n7	0.042584	0.000345	0.180491	0.073171
s1n8	0.043500	0.000560	0.135880	0.035202
s1n9	0.045731	0.000491	0.167735	0.019849
s1n10	0.046860	0.001090	0.175784	0.067228
s1n11	0.049972	0.000639	0.170532	0.040340
s1n12	0.051676	0.000901	0.201910	0.016770
s1n13	0.060008	0.000741	0.228922	0.017916
s1n14	0.054770	0.000825	0.189359	0.026818
s1n15	0.060909	0.000959	0.242769	0.048269
s1n16	0.068032	0.000947	0.296794	0.017193
s1n17	0.068534	0.001186	0.268213	0.050064
s1n18	0.074174	0.001060	0.273806	0.022039
s1n19	0.083529	0.001307	0.311664	0.017660
s1n20	0.079422	0.001936	0.276330	0.023975
s1n21	0.083176	0.001841	0.260300	0.033009
s1n22	0.123771	0.002201	0.418349	0.015659
s1n23	0.150588	0.001870	0.366780	0.033819

Table 18: Different number of masks on Sine

	pred_mean	pred_std	disc_mean	disc_std
s1n1	0.102111	0.000773	0.273150	0.181229
s1n2	0.103853	0.001202	0.364625	0.187255
s1n3	0.108489	0.002033	0.402650	0.172492
s1n4	0.125481	0.026787	0.408800	0.160596
s1n5	0.120690	0.011490	0.402550	0.128909
s1n6	0.141998	0.021988	0.452200	0.058036
s1n7	0.143282	0.024395	0.403225	0.097286
s1n8	0.141339	0.025425	0.469525	0.043682
s1n9	0.141593	0.030420	0.461425	0.039935
s1n10	0.128350	0.007239	0.489750	0.015517
s1n11	0.120902	0.017048	0.479800	0.030173
s1n12	0.130095	0.005233	0.485075	0.013444
s1n13	0.129454	0.001641	0.494325	0.006016
s1n14	0.119676	0.001610	0.489625	0.009129
s1n15	0.119031	0.000703	0.464150	0.030883
s1n16	0.122080	0.002530	0.472850	0.035636
s1n17	0.115005	0.000779	0.409375	0.029047
s1n18	0.110826	0.000774	0.446025	0.039793
s1n19	0.107586	0.000390	0.402575	0.073884
s1n20	0.112869	0.013346	0.322475	0.094419
s1n21	0.116344	0.014419	0.369475	0.049660
s1n22	0.106240	0.000992	0.388050	0.088368
s1n23	0.138103	0.043557	0.230275	0.142925

Table 19: Different number of masks on Energy

	pred_mean	pred_std	disc_mean	disc_std
s1n1	0.305518	0.010932	0.479292	0.019306
s1n2	0.276237	0.006469	0.485049	0.010093
s1n3	0.281197	0.004245	0.494674	0.002273
s1n4	0.302781	0.009273	0.495295	0.002193
s1n5	0.316248	0.012123	0.497362	0.002736
s1n6	0.327704	0.013487	0.498085	0.002122
s1n7	0.329568	0.014006	0.499341	0.000467
s1n8	0.347378	0.013903	0.498935	0.001107
s1n9	0.346808	0.013220	0.499696	0.000261
s1n10	0.349471	0.005714	0.499658	0.000227
s1n11	0.349237	0.002552	0.499835	0.000413
s1n12	0.364403	0.002055	0.499391	0.000583
s1n13	0.364816	0.001406	0.495891	0.011536
s1n14	0.362502	0.002799	0.499772	0.000487
s1n15	0.366674	0.001830	0.499366	0.001216
s1n16	0.365429	0.001416	0.499645	0.000408
s1n17	0.367440	0.001106	0.499024	0.002069
s1n18	0.358168	0.002255	0.499822	0.000228
s1n19	0.334868	0.002492	0.499543	0.000834
s1n20	0.322398	0.001711	0.499810	0.000190
s1n21	0.310435	0.001696	0.499848	0.000416
s1n22	0.299452	0.003624	0.499911	0.000081
s1n23	0.307476	0.002883	0.499784	0.000278

References

[1] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

[2] Jinsung Yoon, Daniel Jarrett, and Mihaela Van der Schaar. Time-series generative adversarial networks. *Advances in Neural Information Processing Systems*, 32, 2019.

[3] Cristóbal Esteban, Stephanie L Hyland, and Gunnar Rätsch. Real-valued (medical) time series generation with recurrent conditional gans. *arXiv preprint arXiv:1706.02633*, 2017.

[4] Olof Mogren. C-rnn-gan: Continuous recurrent neural networks with adversarial training. *arXiv preprint arXiv:1611.09904*, 2016.

[5] Jeffrey L Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

[6] Wafer dataset. `http://www.timeseriesclassification.com/description.php?Dataset=Wafer`. Donated by: R. Olszewski.

[7] Ipd dataset. `http://www.timeseriesclassification.com/description.php?Dataset=ItalyPowerDemand`. Donated by: E. Keogh, L.Wi.

[8] Berry dataset. `http://www.timeseriesclassification.com/description.php?Dataset=Strawberry`. Donated by K. Kemsley and A. Bagnall.

[9] Stock dataset. `https://github.com/jsyoon0823/TimeGAN/blob/master/data/stock_data.csv`. Date published: 2021-11-01.

[10] Energy dataset. `https://archive.ics.uci.edu/ml/datasets/Appliances+energy+prediction`. Date Donated 2017-02-15.

[11] Wei Cao, Dong Wang, Jian Li, Hao Zhou, Lei Li, and Yitan Li. Brits: Bidirectional recurrent imputation for time series. *Advances in neural information processing systems*, 31, 2018.