Group GAN

Anonymous Author(s)
Affiliation
Address
email

Abstract

Generating multivariate time series is a promising approach for sharing sensitive data in many medical, financial, and IoT applications. A common type of multivariate time series originates from a single source such as the biometric measurements from a medical patient. This leads to complex dynamical patterns between individual time series that are hard to learn by typical generation models such as GANs. There is valuable information in those patterns that machine learning models can use to better classify, predict or perform other downstream tasks. We propose a novel framework that takes time series' common origin into account and favors inter-channel relationships preservation. The two key points of our method are: 1) the individual time series are generated from a common point in latent space and 2) a central discriminator favors the preservation of inter-channel dynamics. We demonstrate empirically that our method helps preserve channel correlations and that our synthetic data performs very well downstream tasks with medical and financial data.

1 Introduction

Multivariate Time Series (MTS) are composed of individual time series (TS) sharing the same time reference. In some cases, the individual time series further share a common source or cause such as the biometric values from a medical patient, the stock prices from economic events or geographically separated seismic measurements from a single earthquake. This leads to specific correlation patterns and time dynamics across the time series. Such complex patterns can be crucial when a model is trained on MTS and might need a huge amount of training samples to be captured by a machine learning algorithm. For many applications, however, there may not be enough high-quality training data. For example, for many biomedical and health care applications, data scarcity is common and data sharing to build a larger training set is challenging due to regulatory requirements or ethical concerns [1], [2]). Such concerns are justified; it is well known that sharing data associated with single individuals, even anonymized, can lead to unexpected privacy breaches ([3], [4]). Synthetic MTS could be an attractive alternative to share the patterns and statistical information of an MTS dataset. If done properly, synthetic MTS do not have a one-to-one mapping to the original data, although it comes with its own privacy and quality challenges (see [5] for example).

Beyond data sharing, synthetic MTS can be used for augmenting a dataset to improve the performance of a trained model [6] or increase the contribution of underrepresented sub-populations [7]. For example, in the health domain, a patient can generate multiple TS from biometric measurements, wearable and IoT sensors, but the collection of such data may not have a representative coverage of a full population. Synthetic MTS can help to augment datasets for improving downstream analysis, such as for forecasting and classification tasks. In this work, we are interested in generating synthetic MTS that both preserve utility and statistical properties of the original data. By utility we mean the ability to support a specific downstream task such as the performance of a classifier when trained on synthetic data. Preserving statistical properties such as inter-channel correlation increase the potential

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

benefits to an unforeseen downstream task, exploratory data analysis or educational purposes when sharing data is not possible.

In this paper, we propose a novel method to generate MTS with a common source (or cause) by defining an architecture that explicitly takes inter-channel correlation into account. While MTS can be created with a typical deep learning architecture (such as VAE or GAN) with multiple channels output, we show that assembling individual TS generators with input from a common noise preserves the quality of each individual TS. Additionally, our method addresses the relationships between time series with a central discriminator receiving all the individual TS as a single input.

We perform extensive empirical evaluation on MTS datasets where individual TS originate from a single source. We evaluate the resulting synthetic MTS by comparing their statistical features with the real MTS and their utility on downstream classification task. We pick classification to measure the utility of the synthetic data because classification is one of the most popular analytic tasks in machine learning. We also compare the real and synthetic MTS visually in embedding spaces, and evaluate our method against state-of-the-art baseline methods. Our contributions can be summarized as follows:

- To our knowledge, this is the first study to analyse how to generate multivariate time series with individual channel generation originating from a common noise while inter-channel correlation preservation is forced with a central discriminator.
- Demonstrating that GroupGAN compares favourably with state-of-the-art algorithms in downstream tasks on an Electroencephalography (EEG) eye state time series dataset.
- Demonstrating that GroupGAN results compares favourable with state-of-the-art algorithms in preservation statistical properties of on the EEG dataset.
- Open sourcing the implementation of our methods and experiments[1].

The rest of the paper is arranged as follows: Section 2 discusses related work. Section 3 formalizes he problem description. Section 4 presents our GroupGAN model architecture and gives implementation details. Section 5 presents an extensive empirical evaluation. Section 6 discusses the benefits, limitations and potential negative societal impacts.

2 Related Work

Synthetic data is often used to provide or to increase privacy protection [8]. The only privacy protection framework with predictable effect on privacy is to incorporate *differential privacy (DP)* into the learning algorithm ([9]) by adding calibrated noise to the parameter updates. For every non-DP generation method, the privacy must be assessed empirically with privacy attacks ([10]).

Generative Adversarial Networks [11] are a popular method to generate synthetic data. Since their inception, they have expanded to include time-series data production, see [12] for a comprehensive overview. Mogren introduced the C-RNN-GAN approach, which employs RNNs as both the generator and discriminator in order to synthesise time series from a random vector [13]. Esteban et al. later proposed RCGAN to create medical data using a similar architecture [14]. These frameworks have been applied to a wide range of application domains, including biosignals [15], finance [16], sensor [17], text [18], and smart grid data [19]. However, the typical framework and loss function of GANs are insufficient for the production of multivariate time series, especially if we want to prevent the correlation among the channels as we will demonstrate below. Xu et al. developed COT-GAN based on ideas of optimal transport theory [20] and more recently, Li et al. developed TTS-GAN which is capable of generating realistic synthetic time series of any length [21] both work focusing solely on statistical evaluation of the data such as correlations.

Yoon et al. developped TimeGANs based on recurrent conditional GAN for capturing the temporal dynamics of data throughout time [22]. It entails training supervised and unsupervised targets concurrently using a learnt embedding space. It creates time series data by learning an embedding space and optimising it via binary adversarial feedback and stepwise supervised loss. Most recently, Fourier Flows [23] was proposed as a method based on a Fourier transform layer followed by a chain of spectral filters leading to an exact likelihood optimization.

[1] https://github.com/XXX/GroupGAN

3 Problem Formulation

Let X be an MTS dataset of N instances, each composed of C channels X_i, where $i \in 1, \ldots, C$. An instance of X can be described as $x^n = \{(x_1^n, \ldots, x_C^n)\}$. We want to find a distribution $q(X_1, \ldots, X_C)$ that is as close to $p(X_1, \ldots, X_C)$, the real distribution of our dataset, as possible. In the typical GAN framework, it may be difficult to find the best optimization solution for such a complex goal, which depends on the number of channels, duration, and distribution of the data. This is why we use separate generators G_i to learn the marginal distribution of each channels, $p(X_i)$, separately, and then use a central discriminator to force preserving the real correlation between the channels by focusing on the conditional distributions $p(X_i|X_{i\neq j})$, where $X_{i\neq j}$ refers to the joint distribution of all the channels except channel i. Figure 1 depicts this architecture, with more details discussed in the next section. Essentially, we have two objectives: a local and a central one.

Local objective: In the local objective, the goal is to estimate the marginal distribution of each channels, $p(X_i)$; which means for each channel i, we should optimize:

$$\min_q D(p(X_i)||q(X_i)) \tag{1}$$

where D is any suitable measure of the distance between two distributions.

Central objective: In the central objective, the goal is to estimate the conditional distribution of a channel given all the other ones, $p(X_i|X_{i\neq j})$; which means we will have:

$$\min_q D(\prod_{i=1}^{C} p(X_i|X_{i\neq j})||\prod_{i=1}^{C} q(X_i|X_{i\neq j})) \tag{2}$$

Our approach is that each generator G_i share the same initial (noise) source z, such that equation 2 becomes

$$\min_q D(\prod_{i=1}^{C} p(X_i|z)||\prod_{i=1}^{C} q(X_i|z)) \tag{3}$$

We define the global loss to be a linear combination of the loss of local objective and loss of central objective.

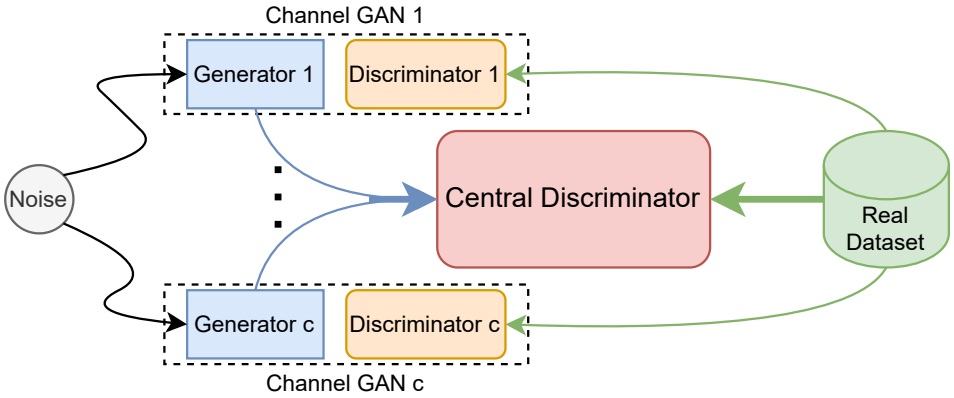


Figure 1: Overall structure of GroupGAN.

4 GroupGAN

As shown in Figure 1, GroupGAN is made up of two main parts: 1) *Channel GANs*, which contain pairs of generator-discriminator dedicated to a single channel (univariate TS), and 2) the *Central*

Discriminator, dedicated to all channels at once. Each of these parts is responsible for a specific task. In channel GANs, the generators are responsible for producing realistic TS and the discriminators are responsible for distinguishing between real and synthetic TS. The central discriminator is responsible for enforcing that all the generated TS of a given instance have the same correlation as those from real MTS. The main reason for having channel GANs as opposed to a single multichannel generator-discriminator pair is to make each generator powerful in its own channel distribution, and by including the central discriminator, we enforce realistic correlations between the channels as much as possible.

4.1 Algorithm

Let our multivariate time series have a dimension of $N * L * C$, where N is the number of instances in the dataset, L is the length of each time series, and C is the number of channels. As shown in Figure 1, there are C pairs of Generator-Discriminator, or channel GANs. All generators are fed a shared noise vector to begin the generation process. Each generator in a channel GAN will synthesize a TS, and both the generated TS and the corresponding channel of real TS will be passed to their paired discriminator, which determines whether the generated TS is from the same distribution as the real ones.

As mentioned earlier, the central discriminator's role is to preserve the inter-channel correlation. The TS synthesized by all channel generators will be concatenated as an MTS and fed to the central discriminator, which aims to determine whether the MTS is real or fake. We hypothesize that this will penalize unrealistic (un)correlation patterns between channels.

4.2 Training

During training, the discriminators in the channel GANs (which we will refer to as channel Discriminators from now on), their paired generators, and the Central Discriminator will engage in a three-player game. The three-player objective of a given channel GAN combined with the central discriminator is:

$$\begin{aligned}\min_{\theta_i}\max_{\phi_i}\max_{\alpha} V(G_{i,\theta_i}, D_{i,\phi_i}, CD_\alpha) = &\mathbb{E}_{x\sim P_{data}}\left[log(D_{i,\phi_i}(x_i)]\right) + \gamma \cdot log(CD_\alpha(x))] \\ &+ \mathbb{E}_{z\sim P_z}[log(1 - D_{i,\phi_i}(G_{i,\theta_i}(z)) \\ &\quad + \gamma \cdot log(1 - CD_\alpha(G_{i,\theta_i}(z), G_{j\neq i}(z))]\end{aligned} \tag{4}$$

where G_{i,θ_i} is the i-th generator with parameters θ_i, D_{i,ϕ_i} is the i-th channel discriminator with parameters ϕ_i, CD_α is the central discriminator with the parameters α, P_{data} is the distribution of the real time series, x_i is the i-th channel of time series x, $G_{j\neq i}$ are all the other generators with fix parameters for the optimization step of G_{i,θ_i}, γ is a hyper-parameter that control the trade-off between well-preserving the correlation among the channels versus generating better quality signal within each channel, and z is the shared noise vector coming from P_z distribution.

For each epoch, we sample a batch of MTS from the dataset, $x^{(1)}, \ldots, x^{(m)} \sim D$ and create as many noise vector $z \sim P_z$ as the batch size. Using generators in each channel, we generate our synthetic signals, and take a gradient ascent step on the discriminators parameters ϕ_is followed by a gradient ascent step on the central discriminator parameters α. Then, we take a gradient descent step on the generators parameters θ_is.

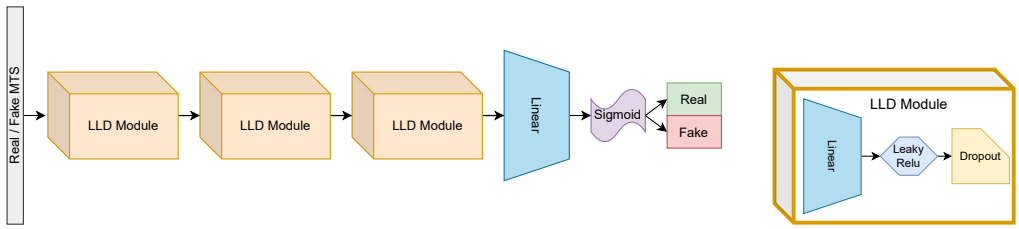
4.3 Key Implementation Details

There are many possible choices of network types for channel GANs. We show in the supplementary materials that networks based on Long short-term memory (LSTM) generate signals of higher quality than the networks based on Multi-layer perceptron (MLP).

We have explored similar options for the central discriminator, and although we initially used an LSTM-based network, the results improved when we switched to an MLP network. We hypothesize that if the central discriminator is too powerful, the results will be of lower quality, as it will strive to make the signals more correlated at the expense of realistic individual TS.

Figure 2a depicts the structure of an MLP-based central discriminator. It consists of three LLD modules, a linear module, and a sigmoid function. An LLD module consists of a linear module

followed by layers of leaky ReLu and Droupouts. Figure 2b demonstrates the structure of an LLD module.



(a) Structure of Central Discriminator. (b) Inside the LLD module.

Figure 2: Structure of an MLP-based Central Discriminator.

5 Empirical Evaluation

5.1 Toy Sine Datasets: Diversity vs Correlation Preservation

To allow us to explore the empirical behaviour of GroupGAN and particularly the impact of the central discriminator (CD), we need to have full control over the nature of the datasets. Thus, we created three "toy medical" datasets with two channels and used them as "real" datasets to generate synthetic data. For all these datasets, we assumed that each instance correspond to a different patient, and each patient produce measurements for two channels (c_1 and c_2). To make the datasets a bit more realistic, we also assumed that there are two types of patients (pt_1 and pt_2), as in "healthy" vs "conditioned". The three datasets are:

- **Simple Sine** is derived from the formula: $x = A\sin(2\pi ft) + \epsilon$. The difference between the signals is that the amplitude (A) for patient type 1 comes from $\mathcal{N}(0.4,\ 0.05)$, whereas the amplitude for patient type 2 comes from $\mathcal{N}(0.6,\ 0.05)$. The other difference is that Channel 1 has a frequency (f) of 0.01, while channel 2 has a frequency of 0.005. In addition, the noise ϵ comes from $\mathcal{N}(0,\ 0.05)$.
- **Sine with frequency changes** contains the same signals as *Simple Sine*, except that the frequency of all sine functions doubles exactly in the middle of the time series. This allows us to examine the situation with varying frequencies.
- **Anomalies** is created by replacing the middle of the time series with Gaussian noise, thus allowing us to examine the impact of anomalies.

We assessed the behaviour of GroupGAN, particularly the central discriminator, by two criteria:

(1) Diversity: requiring that our generators should synthesize from both patients' distributions and there should be no *mode collapse*, which is a common failure of GANs when the generator fails to produce results as diverse as the real data. We measured the diversity by comparing the distribution of patients in the real dataset, which is a bimodal Gaussian distribution, with the distributions of the generated samples using Wasserstein Distance (WD). To aggregate across the channels, we took the average of WDs (AWD). A lower AWD indicates closer similarity to the real distributions. The first column of Table 1 shows the AWD for the various cases.

(2) Correlation Preservation: requiring that the amplitudes of channels for each patient types should be equal to each other as much as possible. We measured the amplitudes of channel 1 and 2 in all signals and verified their similarity. We defined our correlation metric as the average euclidean distance (AED) between the amplitude mapped on a 2D plane (Channel 1 vs Channel 2) and a line with slope 1. The resulting plot is provided in the supplementary materials, and the numeric values are summarized in Table 1. A lower AED indicates stronger preservation of correlation.

As clearly shown in Table 1, there is a trade-off between diversity and correlation preservation. That is, when we increased the strength of the CD (as controlled by the parameter γ in equation (4)), the correlation was better preserved, but at the expense of diversity. Conversely, decreasing the strength of the CD would allow the generated sample distributions to be closer to the real data, but the generated channels were less correlated. As we conducted experiments to tune the hyper-parameter

Table 1: Results of Diversity (AWD) V.S. Correlation Analysis (AED)

Dataset	Method	AWD	AED
Simple Sine	Without CD	**0.0472**	0.1326
	With CD	0.0800	**0.0177**
Freq changes	Without CD	**0.0397**	0.0769
	With CD	0.0679	**0.0242**
Anomalies	Without CD	**0.0540**	0.0766
	With CD	0.0726	**0.0161**

γ, we observed that there is a stable range for γ. We ended up setting γ to 5, which provides stable results for all the experiments presented in this paper, whether the datasets are the toy medical ones or the real ones to be discussed later.

5.2 Toy Sine Datasets - Feature-based Correlation Analysis

The *catch22* feature set has been introduced to capture 22 CAnonical Time series CHaracteristics commonly seen in diverse time series data mining tasks [24]. Using this feature set, we assessed how correlation between any pair of catch 22 features was preserved in synthetic time series data generation. In other words, if a pair is strongly correlated in the real dataset between the two channels, we would like to see that preserved in the synthetic dataset. Similarly, if two features are not correlated in the real dataset, they should remain uncorrelated in the synthetic dataset.

Figure 3 shows three heatmaps for those pairwise correlations between the two channels of simple sine dataset (The same figure for other toy datasets are provided in supplementary materials). To simplify the heatmaps, we removed 7 features that were constant among the real datasets, and kept the remaining 15 features that varied. The left heatmap shows the pairwise correlations of the 15 features for the real dataset. The centre heatmap is the one for the synthetic dataset generated by GroupGAN without a CD, whereas the right heatmap is the one generated by GroupGAN having the CD. Clearly, the right heatmap resembles the left heatmap much more closely. The centre heatmap shows that without the CD, almost all the correlation relationships of the 15 features were destroyed.

While the heatmaps are useful for visualization, we also compared quantitatively the correlation matrices between the two channels using various metrics: (1) Mean Absolute Error (MAE), (2) Frobenius norm, (3) Spearman's ρ, and (4) Kendall's τ. For MAE and the Frobenius norm, a smaller value indicates greater similarity between the correlation matrices of the real and synthetic datasets. For Spearman's coefficient and Kendall's coefficient, the closer the value is to 1, the higher is the similarity. Results shown in Table 2 provide convincing evidence of the effectiveness of the CD in synthetic MTS generation.

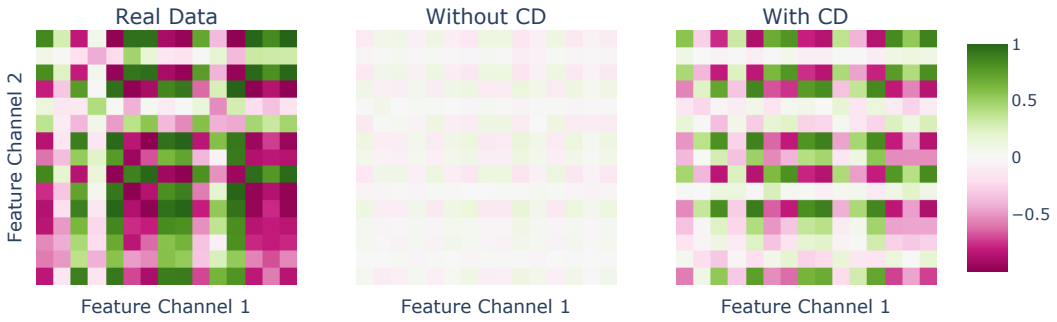


Figure 3: Heatmaps for Catch22 Pairwise Correlations.

5.3 EEG Eye State Dataset and Downstream Classification

We selected a 14-channel EGG eye state dataset to measure the effectiveness of GroupGAN on real signals [25][2]. This dataset contains a label indicating whether the patient's eyes were open or closed (1 indicates closed, and 0 indicates open). Each time series is 117 seconds long, resulting in 14980

[2]https://archive.ics.uci.edu/ml/datasets/EEG+Eye+State with license details therein

Table 2: Similarity between Correlation Matrices

Dataset	Method	MAE	Frobenius norm	Spearman's ρ	Kendall's τ
Simple Sine	Without CD	0.671	11.295	-0.761	-0.569
	With CD	**0.298**	**5.666**	**0.848**	**0.700**
Freq changes	Without CD	0.268	7.889	0.259	0.174
	With CD	**0.131**	**3.413**	**0.834**	**0.661**
Anomalies	Without CD	0.289	8.113	0.428	0.297
	With CD	**0.199**	**5.362**	**0.786**	**0.612**

samples at a sampling rate of 128 per second. To remove outliers in the dataset, we eliminated points with z-scores > 3. The label of the dataset allows us to create a downstream eye blink classification task as follows.

We extracted a window of 800 samples in length containing an eye blink, with a margin of 200 samples at the beginning and end of each eye blink, and labelled those frames with 1. We also extracted 800 samples that did not contain an eye blink, with a margin of 200 samples between the beginning and end of an eye blink, and labelled them with 0. We now have 1024 frames of each label for classification. Then we performed a forward feature selection, and chose top 5 channels regarding the accuracy of our classification task.

To measure the effectiveness of GroupGAN for classification, we used the approach of train-on-real and test-on-fake, and the opposite approach of train-on-fake and test-on-real. Table 3 shows the classification accuracy based on an LSTM-based classifier, which is described in the supplementary materials. We compared the accuracy of the classifier when the synthetic data were generated with and without the CD. Once again, the CD brings significant value to downstream classification tasks.

Table 3: Accuracy in Classification Task

Experiment	GroupGAN with CD	GroupGAN without CD
Train-on-real, Test-on-fake	0.790	0.644
Train-on-fake, Test-on-real	0.634	0.561

Next we compared GroupGAN against a baseline method for generating MTS data. The baseline method is an LSTM-based GAN that generated all of the channels simultaneously. Below we show the results of two experiments: (1) the All-synthetic experiment, and (2) the Augmentation experiment.

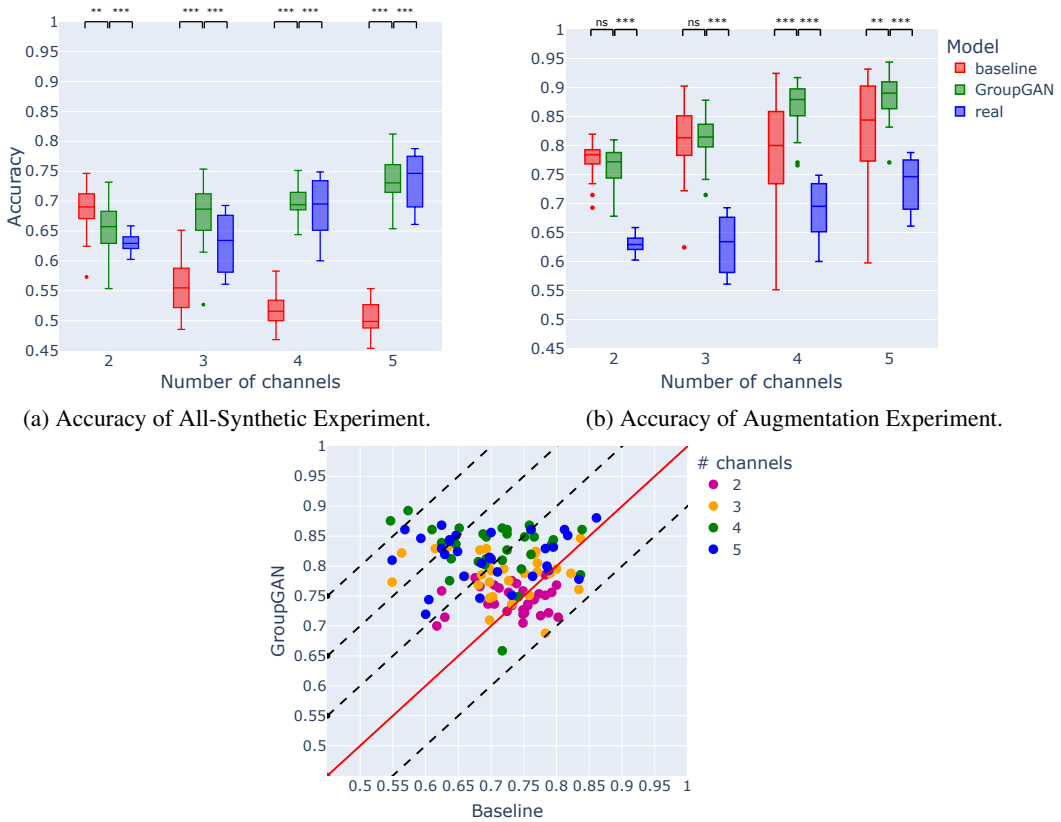
(1) All-synthetic experiment In this experiment, we assessed how well GroupGAN performed in classification task when compared with the baseline method and the actual dataset. We did cross-validation by using 80% of our real dataset for training the GANs. Then only the synthetic data were used to train the classifiers, which were then tested on the unseen 20% of the real dataset. We investigate the utility of the synthetic data for different number of channels in Figure 4a. We repeated each experiment 30 times with different random seeds for each setting, and statistical significance tests were done on the boxplots. A further comparison including GroupGAN without the CD is provided in the supplementary materials.

Figure 4a shows that when the number of channels was increased, the classification accuracy improved for the real dataset. In contrast, the baseline method went the opposite way, showing that MTS generation with all the channels together is too restrictive. Finally, GroupGAN behaved similarly to the real dataset. When the number of channels grew, GroupGAN gave similar average performance as the real dataset.

(2) Augmentation experiment This experiment was set up exactly like the previous one, but instead of using only synthetic time series to train the classifiers, we augmented the real dataset with an equal number of synthetic training samples. In Figure 4b, the boxplots for the real datasets across the different number of channels are exactly the same as those in Figure 4a for the real datasets.

Between Figure 4a and Figure 4b, GroupGAN shows significant improvements in accuracy. The difference was that the synthetic data generated were added to the real data. GroupGAN still outperformed the baseline method both in terms of the median and the variations in accuracy.

Figure 4b was based on an augmentation ratio of 1:1. Figure 4c shows a scatter plot of comparing the accuracy of GroupGAN (y-axis) against the baseline method (x-axis) across six different augmentation



(a) Accuracy of All-Synthetic Experiment.

(b) Accuracy of Augmentation Experiment.

(c) Accuracy of Augmentation Experiment: each point of the same color corresponds to various augmentation ratios, from $1:1$ to $1:10$, see text for details.

Figure 4: Experiment three results. In figure (a) and (b), T-tests were applied to the results in order to show whether there is a statistically significant difference between the distribution of the results. **ns** means no significant difference, ** means a p-value between 0.01 and 0.001, and *** means a p-value less than 0.001.

ratios: $1:1$, $1:2$, $1:4$, $1:6$, $1:8$, $1:10$, i.e. augmenting with up to 10 times more synthetic data than real training data. We repeated the experiment five times with different random seeds for each of these settings; the accuracy of each run is plotted in Figure 4c. Thus, any point above the diagonal line indicates that GroupGAN outperformed the baseline method. The points in the figure are colour-coded based on the number of channels. It is obvious that GroupGAN dominated the baseline method when there is more than two channels. To further quantify the differences in accuracy, we add dotted diagonal lines in 4c, representing a difference in accuracy with increments of 0.1. For instance, the lowest dotted diagonal line represents the cases when the accuracy of GroupGAN is 0.1 below that of the baseline method. Conversely, the other three dotted diagonal lines represent the situations when the accuracy of GroupGAN is 0.1, 0.2 or 0.3 better than that of the baseline. Figure 4c shows that GroupGAN almost always performed better than the baseline method, up to 0.3 higher accuracy, and most exceptions are from the 2 channel dataset where the accuracy is at most 0.1 lower.

5.4 Comparing with State-of-the-art methods on EEG Classification

In the final experiment, we compared GroupGAN with two state-of-the-art (SOTA) methods: TimeGAN[3] [22] and the most recent Fourier Flows [4] [23] discussed previously. In these papers, the downstream task was predicting the next time point of the time series; i.e., forecasting. Because our focus is on classification, we used TimeGAN and Fourier Flows' code to generate the five EEG channels that we chose in the previous experiment and performed the augmentation experiment that

[3]https://github.com/jsyoon0823/TimeGAN with license details therein

[4]https://github.com/ahmedmalaa/Fourier-flows with license details therein

we described in 5.3. As shown in Figure 5 GroupGAN has the best classification accuracy and there is a statistically significant difference between the results of GroupGAN and two other methods.

To compare the statistical properties of GroupGAN with SOTA, we repeated the correlation analysis from 5.2. Figures similar to Figure 3 for TimeGAN and Fourier Flows are provided in the supplementary materials. We computed the MAE between the real correlation matrix and each method's correlation matrix for each pair of channels in the EEG dataset. Table 4 shows that GroupGAN, in addition to giving better classification accuracy, provides the closest similarity in inter-channel correlation to the real dataset.

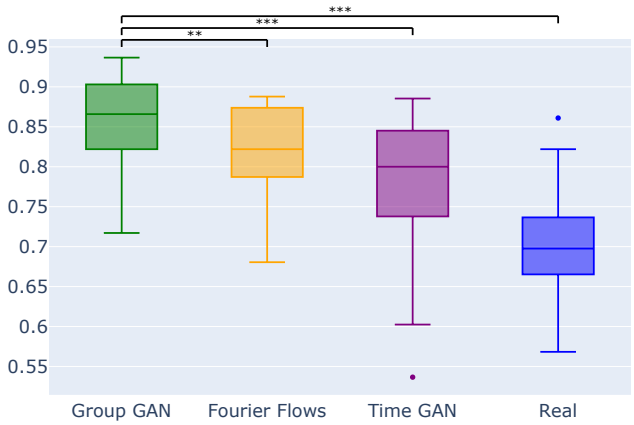


Figure 5: Accuracy for the Augmentation Experiment Comparison with SOTA

Table 4: Similarity between Correlation Matrices in EEG dataset mean $\pm$ standard deviation

Method	MAE
GroupGAN	**0.111** $\pm$ **0.005**
TimeGAN	0.257 $\pm$ 0.008
Fourier Flows	0.146 $\pm$ 0.006

In TimeGAN and Fourier Flows papers, a daily historical Google stocks dataset from 2004 to 2019 was used, and PCA and t-SNE plots were shown to compare their diversity in compare with the real data. We repeated this experiment using data from TimeGAN repository [5], and showed that GroupGAN samples were more diverse and distributed in a way that was more similar to the real dataset distribution. The figures of this experiment are included in the supplementary materials.

6 Discussion

In this paper, we introduced GroupGAN, a novel framework for multivariate time-series generation that delivers more correlated channels. By preserving the correlation between channels, GroupGAN is able to generate time-series that are more similar to the real time-series and achieve better performance in downstream classification tasks than other state of the art methods.

We have shown that our framework is relevant for generating MTS from a common source and we argue that it is particularly suited for human-based biometric measurements. In our experiments, we have never had performance limitations, we foresee however, that GroupGAN will not scale to a very large number of channels as a dedicated GAN for each channel is needed. On the other extreme, we have shown that GroupGAN is not competitive for two channels where it often performed worse than a simple baseline. It is worth re-iterating that synthetic data generation does *not* guarantee privacy and similarly, there is no way to know in advance the performance on a downstream task so both characteristics should be empirically evaluated post-generation. Outliers and minorities are often affected most by privacy leaks as they do not get protected by a large number of similar data samples. We also acknowledge that synthetic data generation can cause harm propagating and even magnifying bias from the data it is based on.

As future work, our method could be extended to more practical use cases where the various channels corresponds to different types of time series, e.g. heartbeats, temperature, respiration and wearable measurements and so on. On the technical side, our framework can be implemented with a wide variety of GANs chosen based on the data type including modern architectures such as transformers.

[5] https://github.com/jsyoon0823/TimeGAN/blob/master/data/stock_data.csv

References

[1] Centers for Medicare & Medicaid Services. The Health Insurance Portability and Accountability Act of 1996 (HIPAA). Online at http://www.cms.hhs.gov/hipaa/, 1996.

[2] 2018 reform of eu data protection rules. URL `https://gdpr-info.eu/`.

[3] Latanya Sweeney. Simple demographics often identify people uniquely. 2000.

[4] Kenneth D. Mandl and Eric D. Perakslis. Hipaa and the leak of "deidentified" ehr data. *The New England journal of medicine*, 384 23:2171–2173, 2021.

[5] Theresa Stadler, Bristena Oprisanu, and Carmela Troncoso. Synthetic data – anonymisation groundhog day. 2020.

[6] Jakub Nalepa, Michal Marcinkiewicz, and Michal Kawulok. Data augmentation for brain-tumor segmentation: A review. *Frontiers in Computational Neuroscience*, 13, 2019.

[7] Jean-François Rajotte, Sumit Mukherjee, Caleb Robinson, Anthony Ortiz, Christopher West, Juan M. Lavista Ferres, and Raymond T. Ng. Reducing bias and increasing utility by federated generative modeling of medical images using a centralized adversary. *Proceedings of the Conference on Information Technology for Social Good*, 2021.

[8] Khaled El Emam, Lucy Mosquera, and Jason Bass. Evaluating identity disclosure risk in fully synthetic health data: Model development and validation. *J Med Internet Res*, 22(11):e23139, Nov 2020. ISSN 1438-8871. doi: 10.2196/23139. URL `https://doi.org/10.2196/23139`.

[9] Martín Abadi, Andy Chu, Ian J. Goodfellow, H. B. McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016.

[10] Nicholas Carlini, Steve Chien, Milad Nasr, Shuang Song, A. Terzis, and Florian Tramèr. Membership inference attacks from first principles. *ArXiv*, abs/2112.03570, 2021.

[11] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, 2014.

[12] Eoin Brophy, Zhengwei Wang, Qi She, and Tomas E. Ward. Generative adversarial networks in time series: A survey and taxonomy. *ArXiv*, abs/2107.11098, 2021.

[13] Olof Mogren. C-rnn-gan: Continuous recurrent neural networks with adversarial training. *ArXiv*, abs/1611.09904, 2016.

[14] Cristóbal Esteban, Stephanie L. Hyland, and Gunnar Rätsch. Real-valued (medical) time series generation with recurrent conditional gans. *ArXiv*, abs/1706.02633, 2017.

[15] Shota Harada, Hideaki Hayashi, and Seiichi Uchida. Biosignal data augmentation based on generative adversarial networks. *2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 368–371, 2018.

[16] Luca Simonetto. Generating spiking time series with generative adversarial networks : an application on banking transactions. 2018.

[17] Moustafa Alzantot, Supriyo Chakraborty, and Mani B. Srivastava. Sensegen: A deep learning architecture for synthetic sensor data generation. *2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, pages 188–193, 2017.

[18] Yizhe Zhang, Zhe Gan, and Lawrence Carin. Generating text via adversarial training. 2016.

[19] Chi Zhang, Sanmukh Rao Kuppannagari, Rajgopal Kannan, and Viktor K. Prasanna. Generative adversarial network for synthetic time series data generation in smart grids. *2018 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*, pages 1–6, 2018.

[20] Tianlin Xu, Li Kevin Wenliang, Michael Munn, and Beatrice Acciaio. Cot-gan: Generating sequential data via causal optimal transport. *ArXiv*, abs/2006.08571, 2020.

[21] Xiaomin Li, Vangelis Metsis, Huangyingrui Wang, and Anne H. H. Ngu. Tts-gan: A transformer-based time-series generative adversarial network. *ArXiv*, abs/2202.02691, 2022.

[22] Jinsung Yoon, Daniel Jarrett, and Mihaela van der Schaar. Time-series generative adversarial networks. In *NeurIPS*, 2019.

[23] Ahmed M. Alaa, Alex J. Chan, and Mihaela van der Schaar. Generative time-series modeling with fourier flows. In *ICLR*, 2021.

[24] Carl Henning Lubba, Sarab S. Sethi, Philip Knaute, Simon R. Schultz, Ben D. Fulcher, and Nick S. Jones. catch22: Canonical time-series characteristics. *Data Mining and Knowledge Discovery*, 33:1821 – 1852, 2019.

[25] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See section 6
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See section 6
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See the supplementary materials.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See the supplementary materials.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See the supplementary materials.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] See the supplementary materials.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [No] The open eeg dataset could potentially be used to identify patients, but it very unlikely and no harmful information is included, only eye state (open or closed).
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]