PSA-GAN: Progressive Self Attention GANs for Synthetic Time Series

Paul Jeha [1] **Michael Bohlke-Schneider** [1] **Pedro Mercado** [1] **Rajbir Singh Nirwan** [1] **Shubham Kapoor** [1]
Valentin Flunkert [1] **Jan Gasthaus** [1] **Tim Januschowski** [1]

Abstract

Realistic synthetic time series data of sufficient length enables practical applications in time series modeling tasks, such as forecasting, but remains a challenge. In this paper we present PSA-GAN, a generative adversarial network (GAN) that generates long time series samples of high quality using progressive growing of GANs and self-attention. We show that PSA-GAN can be used to reduce the error in two downstream forecasting tasks over baselines that only use real data. We also introduce a Frechet-Inception Distance-like score, Context-FID, assessing the quality of synthetic time series samples. In our downstream tasks, we find that the lowest scoring models correspond to the best-performing ones. Therefore, Context-FID could be a useful tool to develop time series GAN models.

1. Introduction

In the past years, methods such as [29, 9, 15, 5, 23, 25, 4, 34] have consistently showcased their effectiveness of deep learning in time series analysis tasks. Although these deep learning based methods are effective when sufficient clean data is available, this assumption is not always met in practice. For example, sensor outages can cause gaps in IoT data, which might render the data unusable for machine learning applications [39]. An additional problem is that time series panels often have insufficient size for forecasting tasks, leading to research in meta-learning for forecasting [24]. Thus, designing flexible and task-independent models that generate synthetic, but realistic time series for arbitrary tasks poses an important challenge. Generative adversarial network (GAN) is a flexible model family that has had success in other domains. However, for their success to carry over to time series, synthetic time series data must be of realistic length, which current state-of-the-art synthetic time series models struggle to generate because they often rely on recurrent networks to capture temporal dynamics [8, 37].

In this work, we make three contributions: i) We propose `PSA-GAN`, a progressively growing, convolutional time series GAN model augmented with self-attention [13, 33]. `PSA-GAN` can be scaled to long time series because the progressive growing architecture starts modeling the coarse-grained time series features and moves towards modeling fine-grained details during training. The self-attention mechanism captures long-range dependencies in the data [38]. ii) We show empirically that `PSA-GAN` samples are of sufficient quality and length to boost downstream forecasting tasks via data augmentation and data imputation in far-forecasting experiments. iii) Finally, we propose a Frechet Inception distance (FID)-like score [27], Context-FID, leveraging unsupervised time series embeddings [9]. We show that the lowest scoring models correspond to the best-performing models in our downstream tasks and therefore could be a useful general-purpose tool to select GAN models for downstream applications.

Related Work. GANs [11] are an active area of research [14, 37, 7, 18, 8] that have recently been applied to the time series domain [8, 37] to synthesize data [31, 8], and to forecasting tasks [35]. Many time series GAN architectures use recurrent networks to model temporal dynamics [22, 8, 37]. Modeling long-range dependencies and scaling recurrent networks to higher lengths is inherently difficult and limits the application of time series GANs to short sequence lengths (less than 100 time steps) [37, 8]. One way to achieve longer realistic synthetic time series is by employing convolutional [32, 3, 10] and self-attention architectures [33]. Convolutional architectures are able to learn relevant features from the raw time series data [32, 3, 10], but are ultimately limited to local receptive fields and can only capture long-range dependencies via many stacks of convolutional layers. Self-attention can bridge this gap and allow for modeling long-range decencies from convolutional feature maps, which has been a successful approach in the image [38] and time series forecasting domain [17]. Another technique to achieve long sample sizes is progressive growing, which successively increases the resolution by adding layers to generator and discriminator during training [13]. Our proposal, `PSA-GAN`, synthesizes progressive

[1]AWS AI Labs, Germany. Correspondence to: Paul Jeha <pauljeha@amazon.de>, Michael Bohlke-Schneider <bohlkem@amazon.de>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

growing with convolutions and self-attention into a novel architecture particularly geared towards time series.

Another challenge is the evaluation of synthetic data. While the computer vision domain uses standard scores like the Inception Score and the Frechet Inception Distance (FID) [27, 12], such universally accepted scores do not exist in the time series field. Thus, researchers rely on a *Train on Synthetic–Test on Real* setup and assess the quality of the synthetic time series in a downstream classification and/or prediction task [8, 37]. In this work, we build on this idea and assess the GAN models through downstream forecasting tasks. Additionally, we suggest a Frechet Inception Distance-like score that is based on unsupervised time series embeddings [10]. Critically, we want to be able to score the fit of our fixed length synthetic samples into their context of (often much longer) true time series, which is taken into account by the contrastive training procedure in [10]. As we will later show, the lowest scoring models correspond to the best performing models in downstream tasks.

2. Model

Problem formulation We denote the values of a time series dataset by $z_{i,t} \in \mathbb{R}$ (or $\mathbb{N}$), where $i \in \{1, 2, \dots, N\}$ is the index of the individual time series and $t \in \{1, 2, \dots, T\}$ is the time index. Additionally, we consider an associated matrix of time feature vectors $\mathbf{X}_{1:T} = (\mathbf{x}_1, \dots, \mathbf{x}_T)$ in $\mathbb{R}^{D \times T}$, where D is the number of time features. Our goal is to model a time series of fixed length τ, $\hat{Z}_{i,t,\tau} = (\hat{z}_{i,t}, \dots, \hat{z}_{i,t+\tau})$, from this dataset using a conditional generator function G and a fixed time point t. Thus, we aim to model $\hat{Z}_{i,t,\tau} = G(\mathbf{n}, \phi(i), \mathbf{X}_{t:t+\tau})$, where $\mathbf{n}$ is a noise vector drawn from a standard Gaussian and ϕ is an embedding function that maps the index of a time series to a vector representation.

Spectral Normalised Residual Self Attention with Convolution The generator and discriminator use a main function m that is a composition of convolution, self attention and spectral normalisation.

$$\begin{aligned} m : \mathbb{R}^{n_f \times l} &\to \mathbb{R}^{n'_f \times l} \\ x &\mapsto \gamma \, \mathrm{SA}(f(x)) + f(x) \end{aligned} \tag{1}$$

$$f(x) = \mathrm{SN}(\mathrm{LR}(c(x))) \tag{2}$$

where c is the one dimensional convolution operator, LR the LeakyReLU operator [36], SN the Spectral Normalisation operator [21] and SA the Self Attention module. The variable n_f is the number of in-features, n'_f is the number of out-features and l is the length of the sequence. Following the work of [38], the parameter γ is learnable. This is initialized to zero to allow the network to learn local features directly from the building block f, and is later enriched with distant features as the absolute value of gamma increases, hereby more heavily factoring the self-attention term $SA \circ f$.

Downscaling and Upscaling The following sections mention upscaling (UP) and downscaling (DOWN) operators that double and halve, respectively, the length of the time series. In this work, the upscaling operator is a linear interpolation and the downscaling operator is the average pooling.

PSA-GAN `PSA-GAN` is a progressively growing GAN[13]; thus, trainable modules are added during training. Hereby, we model the generator and discriminator as a composition of functions: $G = g_{L+1} \circ ... \circ g_1$ and $D = d_1 \circ ... \circ d_{L+1}$ where each function g_i and d_i for $i \in [1, L+1]$ corresponds to a module of the generator and discriminator.

Generator As a preprocessing step, we first map the input features from a sequence of length τ to a sequence of length 8, denoted by $\tilde{Z}_0$, using average pooling. Then, the first layer of the generator g_1 applies the main function m:

$$\begin{aligned} g_1 : \mathbb{R}^{n'_f \times 2^3} &\to \mathbb{R}^{n_f \times 2^3} \\ \tilde{Z}_0 &\mapsto \tilde{Z}_1 = m(\tilde{Z}_0) \end{aligned} \tag{3}$$

For $i \in [2, L]$, g_i maps an input sequence $\tilde{Z}_{i-1}$ to an output sequence $\tilde{Z}_i$ by applying an upscaling of the input sequence and the function m:

$$\begin{aligned} g_i : \mathbb{R}^{n_f \times 2^{i+2}} &\to \mathbb{R}^{n_f \times 2^{i+3}} \\ \tilde{Z}_{i-1} &\mapsto \tilde{Z}_i = m(\mathrm{UP}(\tilde{Z}_{i-1})) \end{aligned} \tag{4}$$

Lastly the final layer of the generator g_{L+1} reshapes the multivariate sequence $\tilde{Z}_L$ to a univariate time series $\hat{Z}_{i,t,\tau}$ of length $\tau = 2^{L+3}$ using a one dimensional convolution.

Discriminator The architecture of the discriminator mirrors the architecture of the generator. It maps the generator's output $\hat{Z}_{i,t,\tau}$ and the time features $\mathbf{X}_{t:t+\tau}$ to a score d. The first module of the discriminator d_{L+1} uses a one dimensional convolution c_1 and a LeakyReLU activation function:

$$\begin{aligned} d_{L+1} : \mathbb{R}^{1+D,\tau} &\to \mathbb{R}^{n_f,\tau} \\ (\tilde{Z}_{L+1}, \mathbf{X}_{t:t+\tau}) &\mapsto \tilde{Z}_L = \mathrm{LR}(c_1(\tilde{Z}_{L+1}, \mathbf{X}_{t:t+\tau}) \end{aligned} \tag{5}$$

For $i \in [L, 2]$, the module d_i applies a downscale operator and the main function m:

$$\begin{aligned} d_i : \mathbb{R}^{n_f \times 2^{i+3}} &\to \mathbb{R}^{n_f \times 2^{i+2}} \\ Y_i &\mapsto Y_{i-1} = \mathrm{DOWN}(m(Y_i)) \end{aligned} \tag{6}$$

The last module d_1 turns its input sequence into a score:

$$\begin{aligned} d_1 : \mathbb{R}^{n_f \times 2^3} &\to \mathbb{R} \\ Y_1 &\mapsto Y_0 = \mathrm{SN}(\mathrm{FC}(\mathrm{LR}(\mathrm{SN}(c(Y_1))))) \end{aligned} \tag{7}$$

where FC is a fully connected layer.

2.1. Loss functions

PSA-GAN can be trained both via a Wasserstein loss [2] and via least squares [20]. We use an auxiliary moment loss to the generator to match the first and second order moment between a batch of synthetic samples and real samples.

2.2. Training procedures

GANs are notoriously difficult to training, have hard-to-interpret learning curves, and are susceptible to mode collapse. PSA-GAN uses spectral normalization and progressive fade-in of new layers to stabilize training. We provide implementation details in the Supplement.

3. Experiments

3.1. Context-FID Score on Time Series Embeddings

The Frechet Inception Distance [12] assesses the performance of a GAN by measuring the similarity between the synthetic samples and the real samples. Here, we replace InceptionV3 [30] with the encoder E of Franceschi et al. [9], trained separately for each dataset.
Our proposed Context-FID score is computed as follows: we first select a time range $[t, t+\tau]$. We then sample a batch of synthetic time series $\hat{\mathbf{Z}}_{t,\tau} = [\hat{Z}_{1,t,\tau}, ..., \hat{Z}_{N,t,\tau}]$ and a batch of real time series $\mathbf{Z}_{t,\tau} = [Z_{1,t,\tau}, ..., Z_{N,t,\tau}]$ that we encode with E into $\hat{\mathbf{Z}}^e_{t,\tau}$ and $\mathbf{Z^e}_{t,\tau}$ respectively. Finally, we compute the FID score of the embeddings.

3.2. Datasets and Baselines

We use the following public, standard benchmark datasets in the time series domain: M4, hourly time series competition data (414 time series) [19]; Solar, hourly solar energy collection data in Alabama State (137 stations) [16]; Electricity, hourly electricity consumption data (370 customers) [6]; Traffic: hourly occupancy rate of lanes in San Francisco (963 lanes) [6].

We split all data into a training/test set with a fixed date and use all data before that date for training. For testing, we use a rolling window evaluation with a window size of 32 and seven windows. We minmax scale each dataset to be within $[0, 1]$ for all experiments in this paper (we scale the data back before evaluating the forecasting experiments). We compare PSA-GAN with different GAN models from the literature (TIMEGAN [37] and EBGAN [40]). In what follows PSA-GAN and PSA-GAN$_W$ denote our proposed PSA-GAN with least squares and Wasserstein loss, respectively. In the data augmentation and far-forecasting experiments, we use the GluonTS [1] implementation of DeepAR which is a well-performing forecasting model and established baseline [28].

3.3. Evaluation Results

Direct Evaluation with Context-FID Scores: Table 1 shows the Context-FID scores for PSA-GAN, baselines, and ablations. For all sequence length, we find that PSA-GAN consistently produces the lowest Context-FID scores. PSA-GAN also scores lower Context-FID scores than its ablation models without moment loss and self-attention (see Table 1, last two columns) For a sequence length of 256 time steps, TIMEGAN is the second best performing model for all datasets, except for Electricity. We show example plots of the samples generated by all models in the Supplement. We find that low Context-FID scores correlate well with samples that qualitatively look realistic. As we show later, low Context-FID score models also correspond to the best-performing models in our downstream tasks. In the next two paragraphs, we will show the performance of PSA-GAN on downstream tasks: data augmentation and far-forecasting.

Data Augmentation: In this experiment, we train DeepAR by randomly sampling time windows of length 256 from the dataset. We average the real data and GAN samples to augment the data during training. [1] During inference, DeepAR is conditioned on real data only to generate forecasts. We evaluate the forecasts with the *normalized root mean squared error* (NRMSE), normalized by the absolute mean of the target in the forecasting window. We give more details on the forecasting setup in the Supplement. Using PSA-GAN for data augmentation results into the lowest NRMSE for three out of four datasets (M4, Solar, Traffic) (see Table 2). Although using synthetic samples from PSA-GAN improves over using real data only, we notice that our improvements are small and more research is required to explore the use of synthetic data for forecasting via data augmentation.

Far-forecasting: In this experiment, we forecast far into the future by assuming that the data points between the training end time and the rolling evaluation window are not observed (in other words, we are increasing the lead time). For example, the last evaluation window would have $32 * 6$ unobserved values between the training end time and the forecast start date. Because DeepAR conditions on this data during inference, we need to impute these missing observations. We use the GAN models during inference to fill the missing observations with synthetic data. As a baseline, we use DeepAR and impute the missing observations with a moving average during inference. Here, we find that using the synthetic data from GAN models drastically

[1] We also tried using only the GAN generated data for training and experimented with ratios of mixing synthetic and real samples, similar to the work in [26], but we were not able to obtain consistent improvements in this way.

		Models				PSA-GAN Ablation study	
Dataset	**Length**	EBGAN	TIMEGAN	PSA-GAN$_W$	PSA-GAN (*ours*)	without SA without ML	with SA without ML
Electricity	64	2.07 ± 0.30	0.77 ± 0.12	0.95 ± 0.07	$\mathbf{0.03 \pm 0.01}$	0.11 ± 0.02	0.24 ± 0.03
	128	1.93 ± 0.28	1.03 ± 0.13	4.08 ± 0.24	$\mathbf{0.04 \pm 0.01}$	1.65 ± 0.12	0.16 ± 0.03
	256	3.02 ± 0.25	1.58 ± 0.08	1.10 ± 0.10	$\mathbf{0.08 \pm 0.03}$	0.44 ± 0.05	$\mathbf{0.08 \pm 0.03}$
M4	64	0.93 ± 0.15	0.93 ± 0.10	0.80 ± 0.08	$\mathbf{0.14 \pm 0.05}$	0.14 ± 0.05	$\mathbf{0.09 \pm 0.03}$
	128	2.53 ± 0.27	0.60 ± 0.07	1.15 ± 0.10	$\mathbf{0.45 \pm 0.09}$	0.59 ± 0.19	$\mathbf{0.28 \pm 0.09}$
	256	2.08 ± 0.13	0.89 ± 0.13	1.97 ± 0.64	$\mathbf{0.52 \pm 0.07}$	1.46 ± 0.22	1.12 ± 0.13
Solar-Energy	64	0.48 ± 0.11	0.13 ± 0.02	1.89 ± 0.40	$\mathbf{0.03 \pm 0.01}$	0.17 ± 0.05	0.17 ± 0.04
	128	1.87 ± 0.43	0.22 ± 0.05	0.15 ± 0.04	$\mathbf{0.02 \pm 0.01}$	0.05 ± 0.01	0.60 ± 0.08
	256	2.38 ± 0.43	0.09 ± 0.02	2.62 ± 0.24	$\mathbf{0.06 \pm 0.01}$	1.46 ± 0.22	1.12 ± 0.13
Traffic	64	2.65 ± 0.15	0.59 ± 0.06	0.92 ± 0.08	$\mathbf{0.23 \pm 0.04}$	$\mathbf{0.16 \pm 0.03}$	0.38 ± 0.04
	128	2.09 ± 0.26	2.13 ± 0.12	2.08 ± 0.10	$\mathbf{0.42 \pm 0.08}$	0.55 ± 0.06	0.51 ± 0.04
	256	1.86 ± 0.14	1.82 ± 0.12	2.76 ± 0.20	$\mathbf{0.56 \pm 0.10}$	1.74 ± 0.06	1.21 ± 0.08

Table 1. Context FID-scores (lower is better) of PSA-GAN, baselines, and ablations. We score 1600 randomly selected windows and report the mean and standard deviation. The first four result columns correspond to the different models that we tested. The last two columns are ablations of PSA-GAN without self-attention and moment loss.

	NRMSE			
Dataset Model	Electricity	M4	Solar	Traffic
DeepAR	$\mathbf{0.49}_{\pm 0.03}$	$0.26_{\pm 0.02}$	$1.07_{\pm 0.01}$	$0.45_{\pm 0.01}$
EBGAN	$0.64_{\pm 0.05}$	$0.32_{\pm 0.09}$	$1.18_{\pm 0.03}$	$0.50_{\pm 0.03}$
TIMEGAN	$0.56_{\pm 0.05}$	$0.28_{\pm 0.02}$	$1.06_{\pm 0.02}$	$0.47_{\pm 0.004}$
PSA-GAN$_W$	$0.50_{\pm 0.01}$	$0.26_{\pm 0.07}$	$1.08_{\pm 0.03}$	$0.64_{\pm 0.13}$
PSA-GAN	$0.53_{\pm 0.07}$	$\mathbf{0.23}_{\pm 0.03}$	$\mathbf{1.04}_{\pm 0.01}$	$\mathbf{0.43}_{\pm 0.003}$

Table 2. NRMSE accuracy comparison of data augmentation experiments (lower is better, best method in bold). Mean and 95% confidence intervals are obtained by re-running each method five times. DeepAR is only run on the real time series data. The other models correspond to DeepAR and one of the GAN models for data augmentation.

	NRMSE			
Dataset Model	Electricity	M4	Solar	Traffic
DeepAR	$4.01_{\pm 0.18}$	$1.0_{\pm 0.14}$	$1.64_{\pm 0.01}$	$0.95_{\pm 0.03}$
EBGAN	$3.3_{\pm 0.43}$	$1.44_{\pm 0.12}$	$2.5_{\pm 0.05}$	$0.88_{\pm 0.03}$
TIMEGAN	$2.1_{\pm 0.18}$	$1.67_{\pm 0.03}$	$\mathbf{1.15}_{\pm 0.02}$	$0.85_{\pm 0.04}$
PSA-GAN$_W$	$2.17_{\pm 0.05}$	$1.13_{\pm 0.05}$	$1.49_{\pm 0.04}$	$1.0_{\pm 0.07}$
PSA-GAN	$\mathbf{1.72}_{\pm 0.05}$	$\mathbf{0.47}_{\pm 0.03}$	$1.19_{\pm 0.02}$	$\mathbf{0.56}_{\pm 0.00}$

Table 3. NRMSE accuracy comparison of far-forecasting experiments (lower is better, best method in bold). Mean and 95% confidence intervals are obtained by re-running each method five times. DeepAR is only run on the real time series data. The other models correspond to DeepAR and one of the GAN models for filling missing observations.

improve over the DeepAR baseline and using samples from PSA-GAN results into the lowest NRMSE for three out of four datasets (see Table 3).

Low Context-FID Score Models Correspond to Best-performing Forecasting Models: One other observation is that the lowest Context-FID score models correspond to the best models in the data augmentation and far-forecasting experiments. For each experiment, the model with lowest Context-FID is also the model that produces the lowest NRMSE in three out of four cases, for both the data augmentation and far-forecasting experiment. At least one of the models with the lowest two Context-FID scores performs best in the forecasting experiments. This suggests that the Context-FID score could be a useful score to develop time series GAN models for downstream use.

4. Conclusion

In this paper, we have presented PSA-GAN, a progressive growing time series GAN, augmented with self-attention, that produces long realistic time series and improves downstream forecasting tasks. Furthermore, we introduced the Context-FID score to assess the quality of synthetic time series samples produced by GAN models. We found that the lowest Context-FID scoring models correspond to the best-performing models in downstream tasks. We believe that time series GANs that scale to long sequences combined with a reliable metric to assess their performance might lead to their routine use in time series modeling.

References

[1] Alexandrov, A., Benidis, K., Bohlke-Schneider, M., Flunkert, V., Gasthaus, J., Januschowski, T., Maddix, D. C., Rangapuram, S., Salinas, D., Schulz, J., Stella, L., Türkmen, A. C., and Wang, Y. GluonTS: Probabilistic time series models in python. *arXiv preprint arXiv:1906.05264*, 2019.

[2] Arjovsky, M., Chintala, S., and Bottou, L. Wasserstein gan, 2017.

[3] Bai, S., Kolter, J. Z., and Koltun, V. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling, 2018.

[4] Cui, Z., Chen, W., and Chen, Y. Multi-scale convolutional neural networks for time series classification, 2016.

[5] de Bézenac, E., Rangapuram, S. S., Benidis, K., Bohlke-Schneider, M., Kurle, R., Stella, L., Hasson, H., Gallinari, P., and Januschowski, T. Normalizing kalman filters for multivariate time series analysis. *Advances in Neural Information Processing Systems*, 33, 2020.

[6] Dheeru, D. and Karra Taniskidou, E. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

[7] Engel, J., Agrawal, K. K., Chen, S., Gulrajani, I., Donahue, C., and Roberts, A. GANSynth: Adversarial neural audio synthesis. In *International Conference on Learning Representations*, 2019.

[8] Esteban, C., Hyland, S. L., and Rätsch, G. Real-valued (Medical) Time Series Generation with Recurrent Conditional GANs. *arXiv:1706.02633 [cs, stat]*, December 2017. arXiv: 1706.02633.

[9] Franceschi, J.-Y., Dieuleveut, A., and Jaggi, M. Unsupervised scalable representation learning for multivariate time series. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[10] Franceschi, J.-Y., Dieuleveut, A., and Jaggi, M. Unsupervised scalable representation learning for multivariate time series, 2020.

[11] Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial networks, 2014.

[12] Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. Gans trained by a two time-scale update rule converge to a local nash equilibrium, 2018.

[13] Karras, T., Aila, T., Laine, S., and Lehtinen, J. Progressive growing of gans for improved quality, stability, and variation, 2018.

[14] Karras, T., Laine, S., and Aila, T. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[15] Kurle, R., Rangapuram, S. S., de Bézenac, E., Günnemann, S., and Gasthaus, J. Deep rao-blackwellised particle filters for time series forecasting. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 15371–15382. Curran Associates, Inc., 2020.

[16] Lai, G., Chang, W., Yang, Y., and Liu, H. Modeling long- and short-term temporal patterns with deep neural networks. *CoRR*, abs/1703.07015, 2017.

[17] Li, S., Jin, X., Xuan, Y., Zhou, X., Chen, W., Wang, Y.-X., and Yan, X. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[18] Lin, K., Li, D., He, X., Zhang, Z., and Sun, M.-t. Adversarial ranking for language generation. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[19] Makridakis, S., Spiliotis, E., and Assimakopoulos, V. The M4 Competition: 100,000 time series and 61 forecasting methods. *International Journal of Forecasting*, 36(1):54–74, 2020. ISSN 0169-2070.

[20] Mao, X., Li, Q., Xie, H., Lau, R. Y. K., Wang, Z., and Smolley, S. P. Least squares generative adversarial networks, 2017.

[21] Miyato, T., Kataoka, T., Koyama, M., and Yoshida, Y. Spectral normalization for generative adversarial networks, 2018.

[22] Mogren, O. C-RNN-GAN: Continuous recurrent neural networks with adversarial training. *arXiv:1611.09904 [cs]*, November 2016. arXiv: 1611.09904.

[23] Oreshkin, B. N., Carpov, D., Chapados, N., and Bengio, Y. N-beats: Neural basis expansion analysis for interpretable time series forecasting. *arXiv preprint arXiv:1905.10437*, 2019.

[24] Oreshkin, B. N., Carpov, D., Chapados, N., and Bengio, Y. Meta-learning framework with applications to zero-shot time-series forecasting. *arXiv preprint arXiv:2002.02887*, 2020.

[25] Rasul, K., Seward, C., Schuster, I., and Vollgraf, R. Autoregressive denoising diffusion models for multivariate probabilistic time series forecasting, 2021.

[26] Ravuri, S. and Vinyals, O. Seeing is Not Necessarily Believing: Limitations of BigGANs for Data Augmentation. March 2019.

[27] Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., and Chen, X. Improved Techniques for Training GANs. *arXiv:1606.03498 [cs]*, June 2016. arXiv: 1606.03498.

[28] Salinas, D., Flunkert, V., Gasthaus, J., and Januschowski, T. DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 2019.

[29] Salinas, D., Flunkert, V., Gasthaus, J., and Januschowski, T. Deepar: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 36(3):1181–1191, 2020.

[30] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. Rethinking the inception architecture for computer vision. *CoRR*, abs/1512.00567, 2015.

[31] Takahashi, S., Chen, Y., and Tanaka-Ishii, K. Modeling financial time-series with generative adversarial networks. *Physica A Statistical Mechanics and its Applications*, 527:121261, August 2019. doi: 10.1016/j.physa.2019.121261.

[32] van den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., and Kavukcuoglu, K. Wavenet: A generative model for raw audio, 2016.

[33] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need, 2017.

[34] Wang, Z., Yan, W., and Oates, T. Time series classification from scratch with deep neural networks: A strong baseline, 2016.

[35] Wu, S., Xiao, X., Ding, Q., Zhao, P., Wei, Y., and Huang, J. Adversarial sparse transformer for time series forecasting. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 17105–17115. Curran Associates, Inc., 2020.

[36] Xu, B., Wang, N., Chen, T., and Li, M. Empirical evaluation of rectified activations in convolutional network, 2015.

[37] Yoon, J., Jarrett, D., and van der Schaar, M. Time-series generative adversarial networks. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[38] Zhang, H., Goodfellow, I., Metaxas, D., and Odena, A. Self-attention generative adversarial networks, 2019.

[39] Zhang, Y.-F., Thorburn, P. J., Xiang, W., and Fitch, P. Ssim—a deep learning approach for recovering missing time series sensor data. *IEEE Internet of Things Journal*, 6(4):6618–6628, 2019. doi: 10.1109/JIOT.2019.2909038.

[40] Zhao, J. J., Mathieu, M., and LeCun, Y. Energy-based generative adversarial networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

Supplementary Material

Paul Jeha [1] **Michael Bohlke-Schneider** [1] **Pedro Mercado** [1] **Rajbir Singh Nirwan** [1] **Shubham Kapoor** [1]
Valentin Flunkert [1] **Jan Gasthaus** [1] **Tim Januschowski** [1]

A. Time Series Features

We use time series features to represent the time dimension for the GAN models and also for DeepAR. For our hourly datasets, we use `HourOfDay`, `DayOfWeek`, `DayOfMonth` and `DayOfYear` features. Each feature is encoded using a single index number (for example, between $[0, 365[$ for `DayOfYear`) and normalized to be within $[-0.5, 0.5]$. We also use an age feature to represent the age of the time series to be $log(2.0 + t)$ where t is the time index.

B. Compute details

PSA-GAN (and its variants) has been trained on ml.p2.xlarge Amazon instances. The table 1 summarizes the training time for each variant and each length of time series:

	PSA-GAN(-W)	PSA-GAN(-W) w/o self attention
16	2h	2h
32	4h	3h
64	5h	3h
128	7h	4h
256	9h	6h

Table 1. Training time for PSA-GAN and PSA-GAN-W with and without self attention

C. Model Details

C.1. Loss Function

PSA-GAN can be trained both via a Wasserstein loss [2] and via least squares [5].

We also introduce a auxiliary moment loss to the generator to match the first and second order moment between a batch of synthetic samples and a batch of real samples denoted $\hat{\mathbf{Z}}_\tau$ and $\mathbf{Z}_\tau$ respectively:

$$\mathrm{ML}(\hat{\mathbf{Z}}_\tau, \mathbf{Z}_\tau) = |\sigma(\hat{\mathbf{Z}}_\tau) - \sigma(\mathbf{Z}_\tau)| + |\mu(\hat{\mathbf{Z}}_\tau) - \mu(, \mathbf{Z}_\tau)|$$

where μ is the mean and σ the standard deviation operator.

C.2. Training Procedures

GANs' training is notoriously difficult. They are prone to unstable training, do not necessarily have meaningful learning curves and can exibit mode collapse. PSA-GAN addresses these issues as follows:

Spectral Normalisation Both the generator and the discriminator benefit from using spectral normalisation layers. In the discriminator, Spectral Normalisation stabilizes training [6]. It also constrains the Lipschitz constant of the discriminator to be bounded by one, thus respecting the condition of the Wasserstein GAN optimisation problem. In the generator, according to [8] the spectral normalisation stabilises the training and avoids escalation of the gradient magnitude.

Progressive fade in of new layers As training progresses, new layers are added to double the length of time series. However, simply adding new untrained layers drastically changes the number of parameters and the loss landscape, which destabilizes training. To mitigate this effect, we updated the model to progressively fade in new layers g_i for $i \in [2, L]$ [3], as follows:

$$\begin{aligned} g_i :& \mathbb{R}^{n_f \times 2^{i+2}} \to \mathbb{R}^{n_f \times 2^{i+3}} \\ & \tilde{Z}_{i-1} \mapsto \tilde{Z}_i = \alpha m(\mathrm{UP}(\tilde{Z}_{i-1})) + (1-\alpha)\,\mathrm{UP}(\tilde{Z}_{i-1}) \end{aligned} \tag{1}$$

where α is a scalar initialized to be zero and grows linearly to one over the given number of epochs.

The same procedure is also applied to d_i for $i \in [2, L]$:

[1]AWS AI Labs, Germany. Correspondence to: Paul Jeha <pauljeha@amazon.de>, Michael Bohlke-Schneider <bohlkem@amazon.de>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

$$d_i : \mathbb{R}^{n_f \times 2^{i+3}} \to \mathbb{R}^{n_f \times 2^{i+2}}$$
$$Y_i \mapsto Y_{i-1} = \alpha\,\mathrm{DOWN}(m(Y_i)) + (1-\alpha)\,\mathrm{DOWN}(Y_i) \quad (2)$$

D. Training details

Both the generator and the discriminator has been trained using the ADAM optimizer [4] with a learning of 0.0005 and betas of $(0.9, 0.999)$.
A new module is added to the discriminator and generator for each 1000 epochs passed. Each module is faded in over 200 epochs.

	Epoch number	Batch size	Number of batches per epoch
16	2500	128	100
32	3500	128	100
64	4500	128	100
128	5500	128	100
256	6500	128	100

Table 2. Number of epochs used to train PSA-GAN(-W) and batch size.

E. Architecture details

The number of features n_f in the generator and discriminator equals 32. The kernel size of the convolutional layer c equals 3 and the kernel size of the convolutional layer c_1 equals 1.

F. Forecasting Experiment Details

We use the `DeepAR` [7] implementation in GluonTS [1] for the forecasting experiments. We used the default hyperparameters of `DeepAR` with the following exceptions: `epochs=100`, `num_batches_per_epoch=100`, `dropout_rate=0.01`, `scaling=False`, `prediction_length=32`, `context_length=64`, `use_feat_static_cat=True`. We use the `Adam` optimizer with a learning rate of $1e-3$ and weight decay of $1e-8$. Additionally, we clip the gradient to 10.0 and reduce the learning rate by a factor of 2 for every 10 consecutive updates without improvement, up to a minimum learning rate of $5e-5$. `DeepAR` also uses lagged values from previous time steps that are used as input to the LSTM at each time step t. We set the time series window size that `DeepAR` samples to 256 and truncate the default hourly lags to fit the context window, the prediction window, and the longest lag into the window size 256. We use the following lags: [1, 2, 3, 4, 5, 6, 7, 23, 24, 25, 47, 48, 49, 71, 72, 73, 95, 96, 97, 119, 120, 121, 143, 144, 145].

For the far-forecasting experiment, we additionally impute the missing observations during inference with a moving average. For the moving average, we tried short and long window sizes (3 and 24) but they essentially resulted into the same performance.

G. Plots of time series

Below we plot time series generated by all the models presented in the paper.

References

[1] Alexandrov, A., Benidis, K., Bohlke-Schneider, M., Flunkert, V., Gasthaus, J., Januschowski, T., Maddix, D. C., Rangapuram, S., Salinas, D., Schulz, J., Stella, L., Türkmen, A. C., and Wang, Y. GluonTS: Probabilistic time series models in python. *arXiv preprint arXiv:1906.05264*, 2019.

[2] Arjovsky, M., Chintala, S., and Bottou, L. Wasserstein gan, 2017.

[3] Karras, T., Aila, T., Laine, S., and Lehtinen, J. Progressive growing of gans for improved quality, stability, and variation, 2018.

[4] Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization, 2017.

[5] Mao, X., Li, Q., Xie, H., Lau, R. Y. K., Wang, Z., and Smolley, S. P. Least squares generative adversarial networks, 2017.

[6] Miyato, T., Kataoka, T., Koyama, M., and Yoshida, Y. Spectral normalization for generative adversarial networks, 2018.

[7] Salinas, D., Flunkert, V., Gasthaus, J., and Januschowski, T. Deepar: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 36(3):1181–1191, 2020.

[8] Zhang, H., Goodfellow, I., Metaxas, D., and Odena, A. Self-attention generative adversarial networks, 2019.

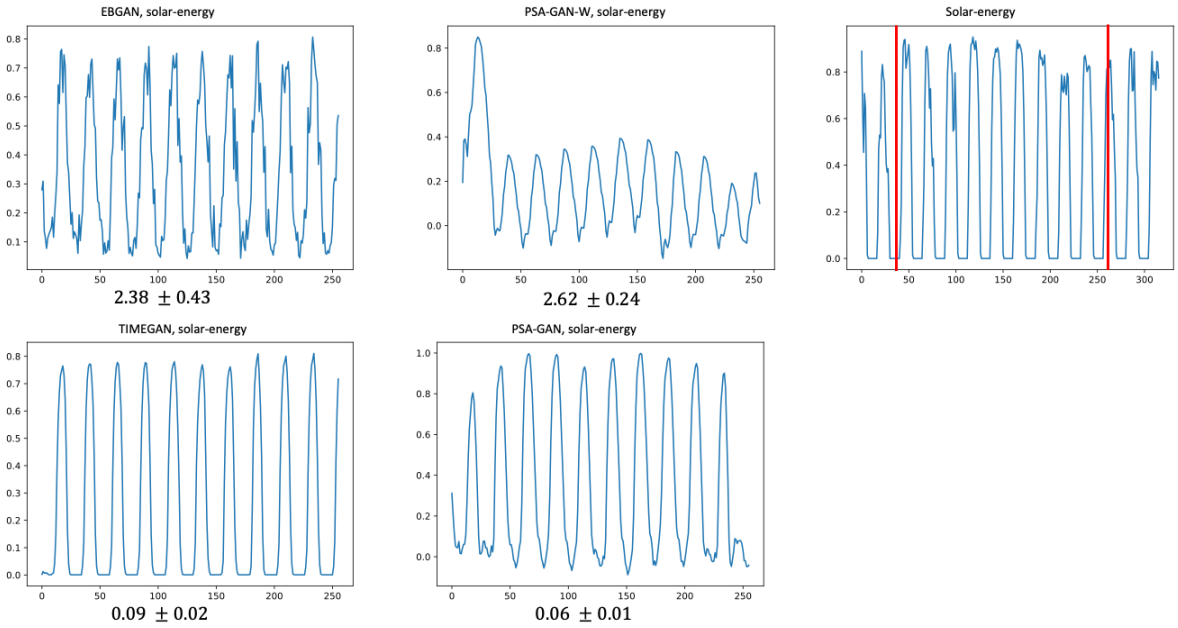

Figure 1. Time Series - Solar-Energy: On the left we plot four synthetic time series at the same time range. On the right we plot between two vertical red line a real time series at the same time range. On the left and right of the red lines is the context of the real time series, 30 time points before and after. We aslo show the FID score for each model.

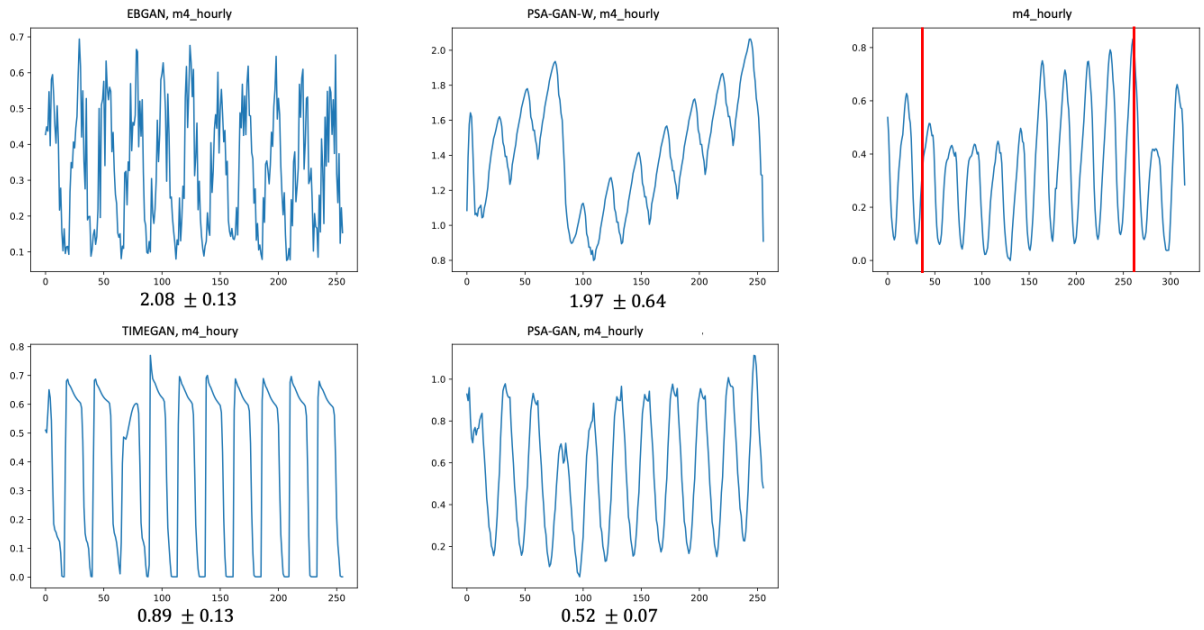

Figure 2. Time Series - M4 Hourly: On the left we plot four synthetic time series at the same time range. On the right we plot between two vertical red line a real time series at the same time range. On the left and right of the red lines is the context of the real time series, 30 time points before and after. We aslo show the FID score for each model.

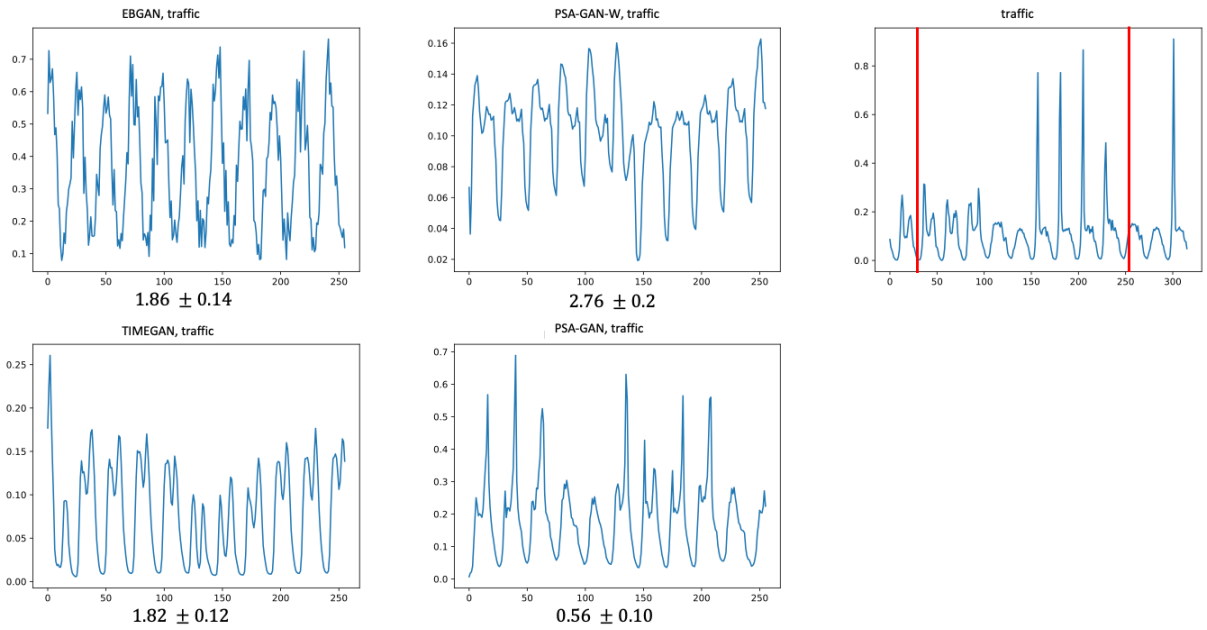

Figure 3. Time Series - Traffic: On the left we plot four synthetic time series at the same time range. On the right we plot between two vertical red line a real time series at the same time range. On the left and right of the red lines is the context of the real time series, 30 time points before and after. We aslo show the FID score for each model.

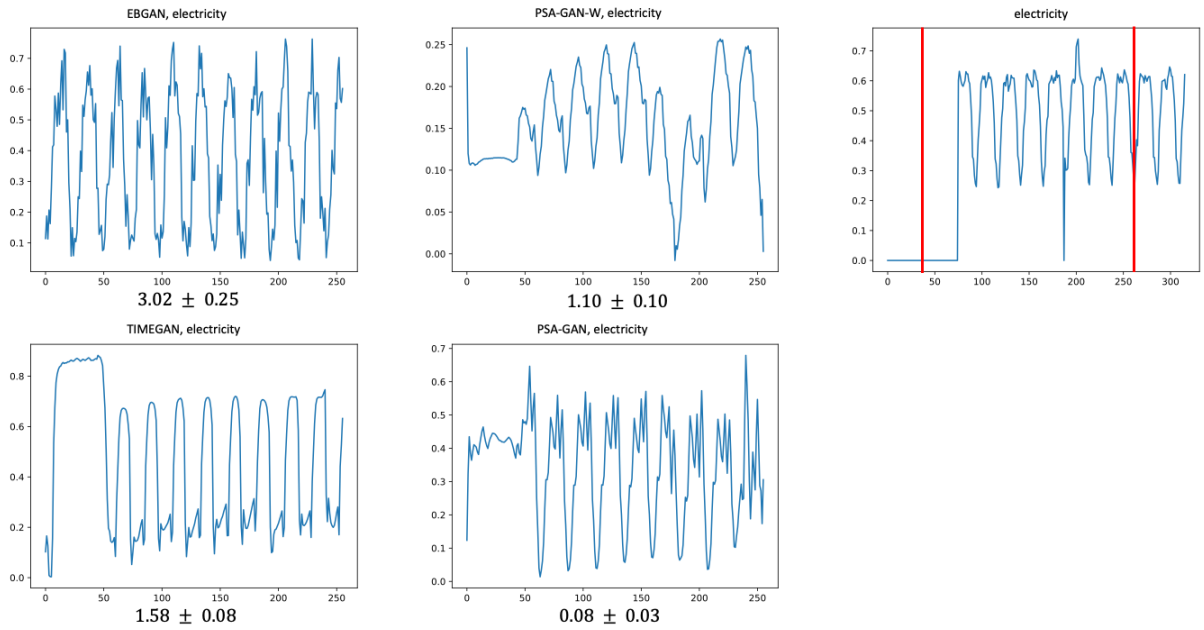

Figure 4. Time Series - Electricity: On the left we plot four synthetic time series at the same time range. On the right we plot between two vertical red line a real time series at the same time range. On the left and right of the red lines is the context of the real time series, 30 time points before and after. We aslo show the FID score for each model.