Representation Learning of Multivariate Time Series using Adversarial Training

Leon Scharwächter[1]
Commissioned by Dr. Sebastian Otte[2]

[1] *M.Sc. Cognitive Science Student at University of Tübingen*
[2] *Neuro-Cognitive Modeling Group, Department of Computer Science, University of Tübingen*

Start date: 01.06.2022, End date: 14.03.2023

Abstract— A critical factor in trustworthy machine learning is to learn an appropriate representation of the training data. Only under this guarantee methods are legitimate to artificially generate data, for example, to counteract imbalanced datasets or provide counterfactual explanations for blackbox decision-making systems. In recent years, Generative Adversarial Networks (GANs) have shown considerable results in forming adequate representations and generating realistic data. While many applications focus on generating image data, less effort has been made in generating time series data, especially multivariate signals. In this work, a transformer-based autoencoder is proposed that is regularized using an adversarial training scheme to generate artificial multivariate time series signals. The representation is evaluated using t-SNE visualizations, Dynamic Time Warping (DTW) and Entropy scores. Results show that the generated signals show a higher similarity to an exemplary dataset than using a convolutional network approach.

Keywords— deep learning, multivariate time series, generative adversarial networks, transformer, convolutional neural networks, autoencoder, unsupervised learning, self-supervised learning

I. Introduction

Background

Learning patterns in multivariate time series signals has been gaining popularity due an increasing use of real-time sensors and recent advances in deep learning architectures [1]. Electrocardiogram records, climate measurements or motion sensors are some examples of multivariate time series, which comprise more than one time-dependent variable, and where dependencies do not solely exist on past values, but also between the variables. The analysis of such inherent patterns using machine learning led to a wide range of applications like early detection of heart diseases, seismic activity forecasting or gesture recognition in human-machine-interaction. However, deep learning models require large amounts of data to train successfully and available datasets are often scarce or imbalanced. Training deep learning models on small datasets consequently results in overfitting and low generalization capabilities. For time series, the possibilities to use data augmentation tricks to artificially expand the datasets are limited by the need to preserve the inherent structural properties of the examined signals. Representation learning refers to a set of techniques in machine learning to automatically discover the relevant features from the input data, where typically a lower dimensional latent encoding is learned. For this purpose, autoencoders often serve as a basic framework. These are neural network architectures that consist of an encoder and a decoder part, both containing trainable parameters. The encoder transforms the input into a latent encoding, while the decoder creates a reconstruction of the true data given the latent encoding. For example for univariate time series signals, in [2] the authors used a convolutional autoencoder to compress univariate time series signals into a low-dimensional encoding. Afterwards, a local neighborhood is sampled around the latent code of a certain input query. These samples are then decoded to create counterfactual explanations. This is a common procedure in the field of explainable machine learning [3]. Although using a compressed representation increases the control over the latent features, vanilla autoencoders do not guarantee any structure or organisation of the latent space and thereby lack of plausibility of the generated samples.

Contact: L. Scharwächter, leon.scharwaechter@student.uni-tuebingen.com

Generative Adversarial Networks

A Generative Adversarial Network (GAN) is a model that is suitable to shape the latent space by enforcing a structure that follows the distribution of the given data [4]. There have been recent successes using GANs in the domains of computer vision or natural language processing, such as image generation [5] or text-to-image synthesis [6]. The utilization of GANs in the field of time series for artificial signal generation and signal forecasting has been gaining traction as

well [7]. A GAN consist of a generator- and a discriminator part, typically initialized as neural networks. The generator $G(z)$ samples a random latent point z from a prior distribution p_{noise} and transforms it into an output of the same dimension of the true data, i.e. generates a fake sample. The discriminator $D(x)$ is a binary classifier that learns to distinguish the fake samples from the true data, i.e. calculates the probability that a data point x is a sample from the data distribution p_{data} rather than a sample from the generative process. $G(z)$ is however optimized to confuse $D(x)$ into believing that the fake samples come from the data distribution. Both networks are optimized simultaneously until they reach an equilibrium, where the solution of this adversarial scheme can be expressed as follows [4]:

$$\min_G \max_D \mathbb{E}_{x\sim p_{data}} logD(x) + \mathbb{E}_{z\sim p_{noise}} log(1-D(G(z))) \quad (1)$$

The authors furthermore provide a theoretical proof that given enough model capacity and training time, the generator shapes the distribution of the latent space by a mapping $G(z)$ such that $p_{noise} = p_{data}$. For real-world applications it is important to mention that even after this convergence implausible data points can be sampled from the latent space if the dataset consists of noisy or incomplete data.

Transformer

The transformer is a recently developed deep learning model architecture that was initially proposed for natural language translation tasks [8]. Since then, it has become prevalent in many different domains and has surpassed other models such as convolutional or recurrent neural networks. Based on an encoder-decoder scheme, it first encodes the input signal into a latent memory using a multi-headed self-attention mechanism over the whole sequence. This enables the transformer to capture long-term dependencies within the signal, while multiple attention heads can consider different representation structures. The attention mechanism thereby transforms the input into m distinct query, key and value matrices Q, K and V through trainable, linear projections. The decoder then queries the keys of the encoder and uses its values to decode the memory:

$$O_m = softmax(\frac{Q_m K_m^T}{\sqrt{d_k}} V_m), \quad (2)$$

where $\sqrt{d_k}$ is a normalization constant and m is the number of heads. The final output of the multi-head attention layer consists of a linear projection of the concatenation $O_1, ..., O_m$.

Problem formulation

Inspired by the work of [9] and [10] an autoencoder is a suitable candidate architecture for stabilizing GAN training. Not only does it contribute in learning a latent representation of a complex distribution, but also reduces mode collapse[1] and makes it easier to perform complex modifications, e.g. through interpolation in the latent space. Since a transformer comprises an encoder-decoder network, it can easily be adapted into an autoencoder framework that projects the input into a lower dimensionality using a bottleneck. There have been already previous work that utilize GAN training for transformer networks, for example in developing frameworks for univariate [11] or multivariate [9] time series forecasting. In [12] the authors used a GAN scheme solely based on transformer encoders for time series representation learning. The goal of this project is to develop an autoencoder for multivariate time series representation learning that is based on both the encoder and decoder of a transformer network, and organize the latent space by incorporating a GAN training scheme. This way, new plausible time series samples should be generated artificially. The procedure is accompanied by an example dataset. Afterwards the latent space is evaluated using Dynamic Time Warping (DTW), Entropy scores and t-SNE representations [13].

[1] Mode collapse describes a problem in GAN training where the generator only learns to produce a single type of output or a small set of outputs

II. METHODS

Model architectures

The model is based on a transformer network, i.e. an encoder ζ_T and a decoder η_T. While the original paper serves as a reference for a detailed description of each individual component [8], here the changes that turn the model into an adversarial autoencoder for multivariate time series data are presented. The input $X \in \mathbb{R}^{m\times v}$ is a multivariate time series of length m and v numbers of variables and comprises a sequence of m feature vectors $x_t \in \mathbb{R}^v$ for $x = [x_1, x_2, ..., x_m]$. As the model operates on continuous data instead on sequences of discrete word indices, the *Embedding Layer* projects the feature vectors x_t into a d-dimensional vector space: $u_t = W_e x_t + b_e$, where d is the model dimension, $W_e \in \mathbb{R}^{d\times m}$, $b_e \in \mathbb{R}^d$ are learnable parameters and $u_t \in \mathbb{R}^d : U \in \mathbb{R}^{m\times d} = [u_1, u_2, ..., u_m]$ represents the input that corresponds to the word vectors of the original paper. This method is proposed by [14]. Subsequently, since the transformer architecture is insensitive to any sequential ordering of the input, a *Positional Encoding Layer* adds a notion of time dependence. This is done by sinusoidal encodings W_{pos} as proposed in the original paper: $U' = U + W_{pos}$, where $W_{pos} \in \mathbb{R}^{m\times d}$ contains sines and cosines of different frequencies per model dimension d. The transformer encoder ζ_T then computes the latent memory $Z \in \mathbb{R}^{m\times d}$ using the multi-head self-attention mechanism: $Z = \zeta_T(U')$. To enforce a lower representation of the latent memory, two additional *Feed Forward Layers* incorporate a bottleneck mechanism, in which the dimensions of Z are concatenated and then compressed: $Z' \in \mathbb{R}^k = W_{enc} Z + b_{enc}$, where $W_{enc} \in \mathbb{R}^{k\times(md)}$ and $b_{enc} \in \mathbb{R}^k$ are learnable parameters. Considering $k \ll md$, this principle turns the transformer into an autoencoder. A concluding *Tanh Layer* scales the compressed latent memory into the interval $(-1, 1)$, which gives more control over the sampling limits when shaping the distribution of the latent space [15]. For the reconstruction of the time series X, the second *Feed Forward Layer* scales the compressed latent memory Z' back to the initial dimensions $Z'' \in \mathbb{R}^{m\times d}$ with learnable parameters W_{dec} and b_{dec} similar to the preceding layer. Since this model aims to optimize a reconstruction problem rather than a classification or regression problem, the input of the transformer decoder consists of the same in-

put X, shifted to the right by 1 time step and denoted as $\bar{X}$, together with the latent memory Z'': $\hat{Y} = \eta_T(\bar{X}, Z'')$, where $\hat{Y} \in \mathbb{R}^{m \times v}$ is the reconstructed multivariate time series.
The adversarial training process is incorporated into the model: The generator G randomly samples a memory vector Z from a uniform distribution with the interval $[-1,1)^k$, i.e. $Z \overset{iid}{\sim} U[-1,1)^k, Z \in \mathbb{R}^k$ and decodes the sampled memory into an artificial multivariate time series $Y \in \mathbb{R}^{m \times v}$ using the transformer decoder η_T. As the decoder requires a reference time series $\bar{X}$ which does not exist for a sampled memory, the decoding procedure is done in an iterative fashion: Starting with a predefined <*SOS*>-Token, the time series is built up step-wise by appending the current output of the decoder to the time series until the maximum sequence length is reached or a predefined <*EOS*>-Token is obtained. The completed time series then corresponds to the reconstruction of the memory, i.e. a fake time series created by G.
The discriminator D is a separate neural network. In [4] a theoretical proof shows that convergence of the adversarial training can be obtained if both G and D are given enough capacity. To avoid an imbalance of capacity and to guarantee that D is not less complex than G, it consists of a transformer encoder with the same model parameters as ζ_T and latent dimension k. The latent memory of D is then passed through a *Feed Forward Layer* projecting to only one neuron. A subsequent *Sigmoid Layer* completes the architecture to make binary predictions (true/fake).
To compare the performance of this model with an existing approach, a convolutional autoencoder similar to [2] was implemented as well. Here, the encoder ζ_C consists of 8 1-D *Convolutional Layers* and has $u \in \{2,4,8,16,32,64,128,256\}$ kernels of size $s \in \{21,18,15,13,11,8,5,3\}$ per layer. A subsequent *Aggregation Layer* applies a single kernel to all 256 channels, whose output length is then compressed to the latent dimension k using a *Feed Forward Layer* to achieve the same autoencoder bottleneck dimensionality. The decoder η_C has a symmetric structure with the parameters in the reverse order. Since the network operates with 1-D convolutions, all variables of the multivariate input $X \in \mathbb{R}^{m \times v}$ are concatenated into a single dimension $X_{concat} \in \mathbb{R}^{1 \times (mv)}$. Both models are implemented in Python using PyTorch (see Supplementary materials).

Dataset and preprocessing

In this project a subset[2] of the NATOPS dataset [16] is used as an exemplary small dataset, which contains body sensor recordings of gestures used as aircraft handling signals. The data is collected by sensors on the hands, elbows, wrists and thumbs which recorded the x,y,z coordinates relatively to the person. The gestures were originally recorded at 20 FPS (with an average duration of 2.34 seconds). In the given subset, all sequences are normalized to a sequence length of 51 time steps. In total, the training- and validation-set each contain 180 sequences with 24 features (sensor recordings) and 6 classes representing different gesture commands, evenly distributed over both datasets. Figure 1 shows a t-SNE representation of the whole validation set. The plot illustrates that three classes are very well separable as different

[2] timeseriesclassification.com/description.php?Dataset=NATOPS

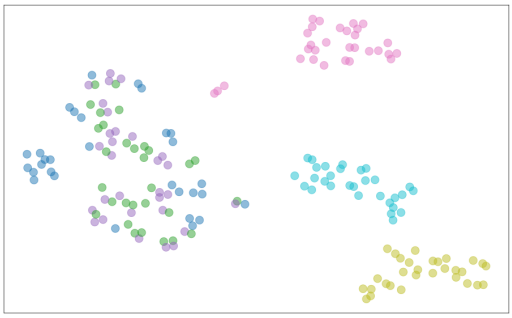

Fig. 1: t-SNE visualization of the NATOPS dataset. The colors represent the different classes (gestures).

modes, while the other three classes have a very similar inherent structure. Figure 3 in the appendix b shows exemplary plots of the dataset. As a preprocessing step, a feature-wise normalization is done in which the values for each feature dimension are transformed into the range $[-1,1]$ using the equation 3.

$$X^v_{std} = \frac{X^v - min(X^v)}{max(X^v) - min(X^v)}; X^v_{scaled} = X^v_{std}(ul - ll) + ll \quad (3)$$

with the limits $ul = 1$ and $ll = -1$, where X^v refers to the values of the dataset X at dimension v. Bringing all dimensions into the same range makes sharing the model parameters more effective. The transformer model requires a Start-Of-Sequence <*SOS*>-Token to decode and generate sequences. This is chosen to be outside of the value range and is set to -3. After the normalization step, the <*SOS*>-Token is prepended to each dimension for all sequences.
It is important to note that time series datasets do not necessarily provide sequences of equal length. If variations in sequence lengths are observed, a maximum sequence length can be set for the transformer autoencoder. Shorter sequences are then padded with arbitrary values and a padding mask adds a large negative value to the attention values, e.g. $-\infty$, before computing the self-attention (equation 2). Furthermore, an <*EOS*>-Token would be needed to represent the End-Of-Sequence.

Training schedule

Before training, the models must be initialized properly. The transformer autoencoder consists of 8 attention heads, 6 encoder layers and 6 decoder layers and has a model dimension $d = 24$ which corresponds to the number of the input variables. The dimension of the hidden feedforward layer is set to 128 and the dimension of the latent memory of the bottleneck is set to $k = 60$. The parameters are initialized using Xavier Initialization [17]. Similar to the approach of [18] this initialization helps to preserve the variance of the gradients across the different layers. This choice coincides with the design of the attention mechanism in the transformer network, which prevents the gradients of growing too large in magnitude with layer depth [8].
As proposed by [10] and [19], both the autoencoder and the adversarial networks are trained jointly on each minibatch in two phases: first, the autoencoder is updated to minimize the reconstruction error of the input. Then, the adversarial networks (i.e. the decoder of the autoencoder and the

discriminator) are updated to regularize the latent space: the discriminator is trained to tell apart the true samples X from the generated samples Y and the generator is trained to fool the discriminator into believing the generated samples Y are real. After successful training, the generator has learned a transformation that maps the imposed prior $p_{noise}(Z)$ to the data distribution $p_{data}(X)$.
The algorithm for the training schedule is given below. Instead of using the objective function of (1), the generator is trained to maximize $log(D(G(Z)))$, which provides stronger gradients and thereby counteracts early vanishing [4]. For simplicity, the transformer autoencoder, which consists of the encoder ζ_T, the decoder η_T, and the architectural adaptions specified in Model architectures, is denoted as Ψ.

Algorithm 1 Simultaneous minibatch stochastic gradient descent training of the autoencoder and the generative adversarial network structure.

for number of epochs **do**
  for number of batches **do**
    - Sample minibatch of b samples $\{X^{(1)},...,X^{(b)}\}$ from data distribution $p_{data}(X)$
    - Update the autoencoder by its stochastic gradient:

$$\nabla_{\theta_\Psi} \frac{1}{b}\sum_{i=1}^{b} \|X^{(i)} - \Psi(X^{(i)})\|_2$$

    - Sample minibatch of b fake samples $\{Y^{(1)},...,Y^{(b)}\}$ using the generator $Y = G(Z)$ and $Z \sim p_{noise}(Z)$
    - Update the discriminator by its stochastic gradient:

$$\nabla_{\theta_D} \frac{1}{b}\sum_{i=1}^{b} [log(D(X^{(i)})) + log(1 - D(Y^{(i)}))]$$

    - Update the generator by its stochastic gradient:

$$\nabla_{\theta_G} \frac{1}{b}\sum_{i=1}^{b} [log(D(Y^{(i)}))]$$

  end for
end for
Adam optimizer [20] is used to incorporate the gradient-based updates into a learning rule.

Given the NATOPS dataset, the transformer autoencoder was trained for 2000 epochs, with a batch size equals to 32 and a learning rate of 0.0001. The training was done at the Training Center for Machine Learning (TCML) Cluster in Tübingen (Grant number 01IS17054). The convolutional autoencoder was trained using the same procedure. However, all weights were instead initialized from a zero-centered Normal distribution with a standard deviation of 0.02 [15] and a learning rate of 0.001 was used.
A variant of the above algorithm is yielded by incorporating the Wasserstein GAN scheme [21] to stabilize GAN training. Here, both the discriminator and the generator are optimized using an adapted loss function (eq. 4), the discriminator uses a linear activation instead of sigmoid, is trained 5 times more than the generator in each iteration and its weights are constrained to a range of $[-0.1, 0.1]$ after each update. Furthermore RMSProp is used as learning rule with a small learning rate of 0.00005 and without momentum.

$$\max \mathbb{E}_{x\sim p_{data}} D(x) - \mathbb{E}_{z\sim p_{noise}} D(G(z)) \tag{4}$$

Evaluation metrics

The characteristic of the latent space and the generation of artificial multivariate time series can be evaluated qualitatively and quantitatively [7]. However, unlike image-based GANs, where the Inception Score [22] or the Fréchet Inception Distance (FID) [23] are established metrics, there are no standards set for time series data, especially not for multivariate time series. [24] gives an overview of different evaluation metrics, which however are mainly focused on image generation and for the most part lack possibilities to adapt to (multivariate) time series data.
In this work, t-SNE visualizations [13] are used to qualitatively compare the distribution of artificially generated signals with the underlying distribution of the dataset. To quantitatively measure the similarity of both distributions Dynamic Time Warping (DTW) is applied, which can be generalized to a dependent, multi-dimensional form proposed by [25]. Furthermore DTW is more robust against time lags than other similarity / distance measures like the correlation coefficient or the euclidean distance. For each generated multivariate time series, DTW is used to determine the shortest distance to a signal from the validation set. The average DTW is then calculated as the mean of all shortest distances. The smaller the distance, the higher the similarity. To additionally quantitatively assess and compare the diversity of the generated time series, the multivariate Entropy is calculated based on [26]. Thereby the total Entropy H is determined by the sum of the Entropy per dimension H_E, normalized by the maximum Entropy H_{max} (see equations 5-7).

$$H_{max}(X) = -\sum_{i=1}^{S} \frac{1}{|S|} log \frac{1}{|S|} \tag{5}$$

$$H_E(X) = -\frac{1}{H_{max}} \sum_{i=1}^{S} p(x_i) log p(x_i) \tag{6}$$

$$H = \frac{1}{k}\sum_{i=1}^{k} H_E(X^i) \tag{7}$$

where k is the number of dimensions and S is a set of probabilities. To define S domain knowledge about the dataset is required: Each dimension is categorized into value ranges and by counting the occurrence of time points within these ranges, the probability for each category is determined. Four categories are determined for the given dataset, where X_t^i refers to all artificial signals at dimension i and time step t: $p_1 : X_t^i \geq 1, p_2 : 1 > X_t^i \geq 0, p_3 : 0 > X_t^i \geq -1, p_4 : X_t^i < -1$. The higher the score, the higher the diversity. For each model 50 artificial time series were generated to calculate the scores of the metrics.

III. RESULTS

Both the transformer autoencoder and the convolutional autoencoder were trained with and without the GAN training scheme. Training without GAN only involved minimizing the reconstruction error without any regularization of the latent space. Additionally, both models were trained using the Wasserstein GAN approach. In this section the models are abbreviated the following: TAE = transformer autoencoder, TAE-GAN = transformer autoencoder with GAN scheme,

Fig. 2: t-SNE representation of all models. The blue dots refer to the true time series from the validation dataset, while the orange dots refer to artificially generated time series from the model.

TAE-WGAN = Transformer autoencoder with Wasserstein GAN scheme. The convolutional autoencoder is abbreviated in the same manner using the basis CAE. Figure 2 shows the t-SNE visualizations of all models. Table 1 contains the results of all models regarding similarity (Avg. DTW), diversity (Entropy) and the reconstruction ability of the original signals (Test Error).

TAE: The model shows a higher diversity (Entropy) compared to the TAE trained with GAN. The t-SNE representation in Figure 2 illustrates this variance, but the generated time series are hardly represented within the modes of the dataset, as the latent space was not shaped any further. The reconstruction error is smaller compared to the TAE trained with GAN. Apparently, the GAN training counteracted decoding accuracy as a regulatory constraint. Although TAE achieved the lowest reconstruction error, it is important to note that the transformer model generally suffers from error accumulation when generating artificial time series signals, because the generation process is done in an interative fashion. Figure 6 in the appendix b shows exemplary generated time series that have similar patterns to those from the dataset, but also sudden disruptions that are not represented in the dataset (Example 1 and 8). In addition, small fluctuations are contained: Straight lines and smooth curves could not be reproduced adequately.

TAE-GAN: Both the diversity (Entropy) and the similarity (DTW) are worse compared to TAE. This is also suggested by the t-SNE representation, since only one mode is learned that does not overlap with the original distribution. This is a typical example of 1. Mode dropping and 2. Mode collapse, where 1. mass is put outside of the support of the underlying distribution of the dataset and 2. only a single type of output is learned and other modes are disregarded [27]. The generated time series in Figure 9 in the appendix b confirm the low diversity, and hardly show any similarity to the dataset. However, the error plots in Figure 8 in the appendix b show stable training and illustrate that although the discriminator became increasingly uncertain after 2000 epochs, overall convergence was not achieved. The generative quality could maybe have been higher if the training had lasted longer.

TAE-WGAN: With a score of 19.9187, the similarity (DTW) to the dataset is the highest compared to all other models. However, the diversity (Entropy) has deteriorated compared to TAE and TAE-GAN. The t-SNE representation in Figure 2 actually suggests a higher score due to the higher dispersion, but it also shows that the three well-separated modes are not sufficiently represented. The reconstruction ability was hardly influenced by the GAN regularization compared to TAE (similar Test Error) and is smaller compared to the convolutional models. The generated time series in Figure 11 in the appendix b have similar patterns to those from the dataset and indeed show some diversity. However, they also contain fluctuations where straight lines or smooth curves were expected. The error plots in Figure 10 in the appendix b illustrate stable training and indicate that convergence was achieved.

CAE: This model shows low similarity (DTW) and low diversity (Entropy) compared to the other models. The low diversity is further illustrated by the distribution in the t-SNE representation in Figure 2. Similar to TAE-GAN, the distribution lies exclusively within a new mode; there is no overlap with the modes of the dataset as the latent space was not regularized any further. The generated time series in Figure 7 in the appendix b confirm the low similarity to the dataset and the low diversity. In contrast to the observations with TAE, the test error is higher than CAE with GAN training.

TABLE 1: AVERAGE DTW, MULTIVARIATE ENTROPY AND RECONSTRUCTION ERROR AFTER TRAINING.

Model Type	Avg. DTW	Entropy	Test Error
TAE	28.2731	0.5074	**0.0183**
TAE-GAN	36.9175	0.4285	0.2828
TAE-WGAN	**19.9187**	0.3503	0.0188
CAE	36.5243	0.2840	0.0817
CAE-GAN	35.1642	0.6086	0.0466
CAE-WGAN	37.9486	**0.8553**	0.0238

CAE-GAN: The Entropy score shows a high diversity compared to the other models, also confirmed by the distribution in the t-SNE plot (Figure 2). Here, the artificial time series lie within the different modes of the dataset, although not sufficiently distributed. The similarity score (DTW) is still low in an overall comparison, however slightly increased compared to CAE. The error plots in Figure 12 in the appendix b show that the regulation by the GAN training was very unstable. There was no steady learning effect of the discriminator; only in isolated epochs it learned to clearly distinguish between true and fake samples, while for the most part it was completely uncertain (outputs near to 0.5). This possibly resulted in hardly useful gradients for the generator. The generated time series in Figure 13 in the appendix b affirms the low similarity to the dataset.

CAE-WGAN: With an Entropy of 0.8553 this model shows the highest diversity, which is also confirmed by the distribution in the t-SNE plot (Figure 2). The generated time series are well balanced among the different modes of the dataset. Nevertheless, the similarity (DTW) remained low and even worsened compared to CAE and CAE-GAN. The GAN training has also been very unstable (Figure 14 in the appendix b); However, a continuous learning effect occurs in the last 250 epochs. Therefore, a longer training might have led to convergence. Like CAE-GAN, the generated time series in Figure 15 in the appendix b do not show similar patterns to the dataset.

IV. Conclusion

In this work, a transformer-based model is proposed to generate multivariate time series signals within the support of the underlying distribution of an existing dataset. The transformer architecture was augmented with a bottleneck to act as an autoencoder and was optimized using a conventional GAN and the Wasserstein GAN approach. Additionally, the model was compared to an existing approach that uses a convolutional autoencoder for local neighborhood generation of time series signals. The results show that the transformer model was more suitable for stable GAN training. Using the Wasserstein GAN approach, the model outperformed the other models in terms of similarity to the dataset, however not all modes were sufficiently represented. The diversity is estimated to be rather low based on the multivariate Entropy, while the t-SNE representation suggests a higher value. This discrepancy underscores the need for profound research on appropriate GAN evaluation metrics for multivariate time series. It is however important to note that distances in t-SNE representations are not necessarily comparable due to nonlinear transformations. Using the conventional GAN training scheme, longer training could have resulted in convergence, however, harbors the risk of overfitting to the training data and thereby leading to phenomena like mode dropping. Artifacts such as small fluctuations in the time series unfortunately could not be diminished. Regularizing GANs is an ongoing research problem, and the behavior of GANs needs to be further understood. In the appendix a is an overview of different strategies to stabilize GAN training schemes.

V. Supplementary materials

All Python scripts regarding the model architectures, training, loading the dataset, and evaluation are available at https://github.com/lscharwaechter/TransformerGAN.

References

[1] R. Assaf and A. Schumann, "Explainable deep neural networks for multivariate time series predictions," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence Organization, aug 2019. [Online]. Available: https://doi.org/10.24963/Fijcai.2019/932

[2] R. Guidotti, A. Monreale, F. Spinnato, D. Pedreschi, and F. Giannotti, "Explaining any time series classifier," in *Second International Conference on Cognitive Machine Intelligence (CogMI)*. IEEE, (2020). [Online]. Available: https://doi.org/10.1109/cogmi50398.2020.00029

[3] E. Ates, B. Aksar, V. J. Leung, and A. K. Coskun, "Counterfactual explanations for multivariate time series," in *International Conference on Applied Artificial Intelligence (ICAPAI)*. IEEE, (2021). [Online]. Available: https://doi.org/10.1109/icapai49758.2021.9462056

[4] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," 2014. [Online]. Available: https://arxiv.org/abs/1406.2661

[5] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang *et al.*, "Photo-realistic single image super-resolution using a generative adversarial network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4681–4690.

[6] H. Zhang, T. Xu, H. Li, S. Zhang, X. Wang, X. Huang, and D. N. Metaxas, "Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 5907–5915.

[7] E. Brophy, Z. Wang, Q. She, and T. Ward, "Generative adversarial networks in time series: A survey and taxonomy," 2021. [Online]. Available: https://arxiv.org/abs/2107.11098

[8] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2017. [Online]. Available: https://arxiv.org/abs/1706.03762

[9] J. Zhang and Q. Dai, "Latent adversarial regularized autoencoder for high-dimensional probabilistic time series prediction," *Neural Networks*, vol. 155, pp. 383–397, 2022.

[10] A. Makhzani, J. Shlens, N. Jaitly, and I. J. Goodfellow, "Adversarial autoencoders," *CoRR*, vol. abs/1511.05644, 2015. [Online]. Available: http://arxiv.org/abs/1511.05644

[11] S. Wu, X. Xiao, Q. Ding, P. Zhao, Y. Wei, and J. Huang, "Adversarial sparse transformer for time series forecasting," *Advances in neural information processing systems*, vol. 33, pp. 17 105–17 115, 2020.

[12] X. Li, V. Metsis, H. Wang, and A. H. H. Ngu, "Tts-gan: A transformer-based time-series generative adversarial network," in *Artificial Intelligence in Medicine: 20th International Conference on Artificial Intelligence in Medicine, AIME 2022, Halifax, NS, Canada, June 14–17, 2022, Proceedings*. Springer, 2022, pp. 133–143.

[13] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne." *Journal of machine learning research*, vol. 9, no. 11, 2008.

[14] G. Zerveas, S. Jayaraman, D. Patel, A. Bhamidipaty, and C. Eickhoff, "A transformer-based framework for multivariate time series representation learning." New York, NY, USA: Association for Computing Machinery, 2021, p. 2114–2124. [Online]. Available: https://doi.org/10.1145/3447548.3467401

[15] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," 2015. [Online]. Available: https://arxiv.org/abs/1511.06434

[16] Y. Song, D. Demirdjian, and R. Davis, "Tracking body and hands for gesture recognition: Natops aircraft handling signals database," in *2011 IEEE International Conference on Automatic Face Gesture Recognition (FG)*, 2011, pp. 500–506.

[17] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, Y. W. Teh and M. Titterington, Eds., vol. 9. Chia Laguna Resort, Sardinia, Italy: PMLR, 13–15 May 2010, pp. 249–256. [Online]. Available: https://proceedings.mlr.press/v9/glorot10a.html

[18] X. S. Huang, F. Pérez, J. Ba, and M. Volkovs, "Improving transformer optimization through better initialization," in *Proceedings of the 37th International Conference on Machine Learning*, ser. ICML'20. JMLR.org, 2020.

[19] J. Yoon, D. Jarrett, and M. Van der Schaar, "Time-series generative adversarial networks," *Advances in neural information processing systems*, vol. 32, 2019.

[20] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014. [Online]. Available: https://arxiv.org/abs/1412.6980

[21] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein gan," 2017. [Online]. Available: https://arxiv.org/abs/1701.07875

[22] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," *Advances in neural information processing systems*, vol. 29, 2016.

[23] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," 2017. [Online]. Available: https://arxiv.org/abs/1706.08500

[24] A. Borji, "Pros and cons of gan evaluation measures," 2018. [Online]. Available: https://arxiv.org/abs/1802.03446

[25] M. Shokoohi-Yekta, B. Hu, H. Jin, J. Wang, and E. Keogh, "Generalizing dtw to the multi-dimensional case requires an adaptive approach," *Data mining and knowledge discovery*, vol. 31, pp. 1–31, 2017.

[26] F. Bahrpeyma, M. Roantree, P. Cappellari, M. Scriney, and A. McCarren, "A methodology for validating diversity in synthetic time series generation," *MethodsX*, vol. 8, p. 101459, 2021. [Online]. Available: https://doi.org/10.1016/j.mex.2021.101459

[27] T. Lucas, "Deep generative models: over-generalisation and mode-dropping," Ph.D. dissertation, Artificial Intelligence [cs.AI]. Université Grenoble Alpes, 2020. NNT: 2020GRALM049. tel-03102554.

[28] L. Metz, B. Poole, D. Pfau, and J. Sohl-Dickstein, "Unrolled generative adversarial networks," 2016. [Online]. Available: https://arxiv.org/abs/1611.02163

[29] Z. Lin, A. Khetan, G. Fanti, and S. Oh, "Pacgan: The power of two samples in generative adversarial networks," 2017. [Online]. Available: https://arxiv.org/abs/1712.04086

[30] M. Arjovsky and L. Bottou, "Towards principled methods for training generative adversarial networks," *arXiv preprint arXiv:1701.04862*, 2017.

[31] K. Roth, A. Lucchi, S. Nowozin, and T. Hofmann, "Stabilizing training of generative adversarial networks through regularization," *Advances in neural information processing systems*, vol. 30, 2017.

[32] R. Yang, D. M. Vo, and H. Nakayama, "Stochastically flipping labels of discriminator's outputs for training generative adversarial networks," *IEEE Access*, vol. 10, pp. 103 644–103 654, 2022.

[33] T. White, "Sampling generative networks," *arXiv preprint arXiv:1609.04468*, 2016.

[34] S. Jelassi, A. Mensch, G. Gidel, and Y. Li, "Adam is no better than normalized SGD: Dissecting how adaptivity improves GAN performance," 2022. [Online]. Available: https://openreview.net/forum?id=D9SuLzhgK9

A. Appendix

a. Methods for stabilizing GAN training

Training GANs is notoriously hard. Common problems include mode collapse ("Helvetica scenario"), vanishing gradients, and furthermore there is no guarantee of convergence after a certain amount of training time. The following list gives an incomplete overview of different techniques to counteract these phenomena.

Mode collapse

- The generator must not trained too much without updating the discriminator, as this could result in mode collapse [4]
- Apply batch normalization to all layers except the output layers of the generator and the discriminator to prevent the generator from mode collapse [15]
- The Wasserstein loss supports the discriminator to reject samples the generator stabilizes on. So the generator has to learn something new [21]
- Incorporating label information in the adversarial regularization helps to construct different modes [10]
- Updating the generator not only using the current discriminator's output but also the outputs of future discriminator versions (Unrolled GANs) inhibits the optimization for a single discriminator and forces the generator to diversify its output [28]
- Modifying the discriminator to make decisions based on multiple samples of the same class (true/fake) at once helps to identify homogeneous packs as artificial [29]

Vanishing gradients

- Using a modified min-max loss, the generator is trained to maximize $log(D(G(z)))$ to prevent the GAN from vanishing gradients and getting stuck in the early stages of training [4]
- The Wasserstein loss helps to prevent vanishing gradients by providing information even when the discriminator is trained to optimality [21]

Convergence

- Adding noise to the discriminator's input helps to stabilize GAN training and improve convergence [30]
- Regularizing the weight updates of the discriminator by using the norm of the gradients increases training stability [31]
- Label noise, i.e. occasionally flipping the labels, balances the training and prevents the discriminator from being too strong and outperforming the generator [32]

Practical considerations

- Sampling from a Gaussian distribution instead of a Uniform distribution during training and afterwards interpolating via a circle from point A to point B instead of a straight line results in sharper samples [33]

- Smoothing the labels when training the discriminator reduces the vulnerability to adversarial samples [22]
- Normalized stochastic gradient descent ascent nSGDA is a simpler alternative to Adam and reaches similar or even better performance [34]

Further regularization steps regarding GAN training can be found in [22].

b. Figures

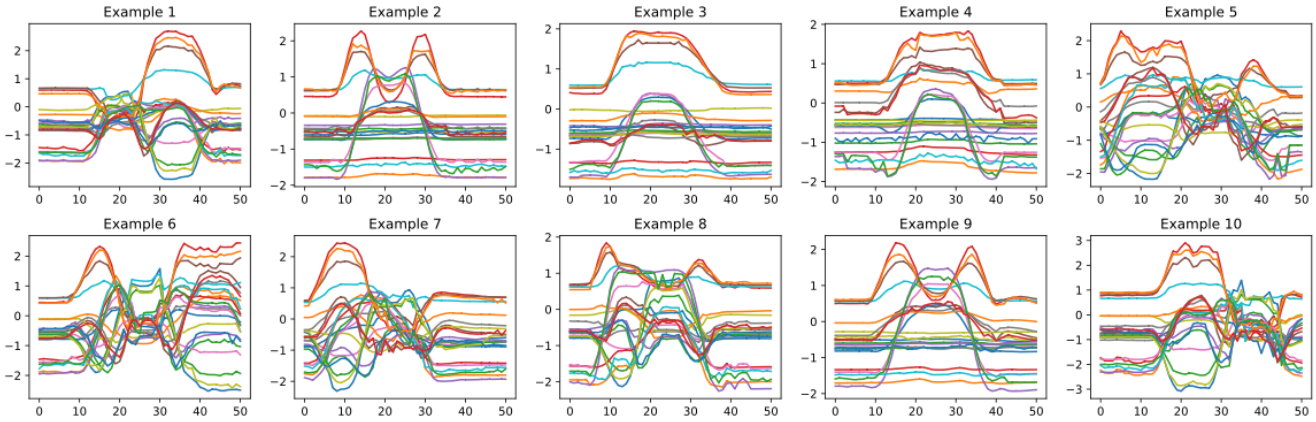

Fig. 3: Ten random examples of time series signals from the NATOPS validation set

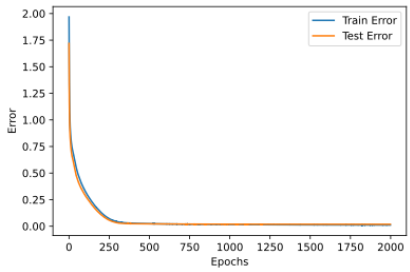

Fig. 4: Errors of the transformer model without GAN regularization (TAE)

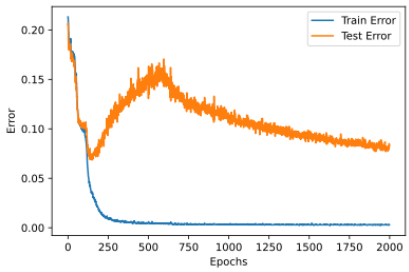

Fig. 5: Errors of the convolutional model without GAN regularization (CAE)

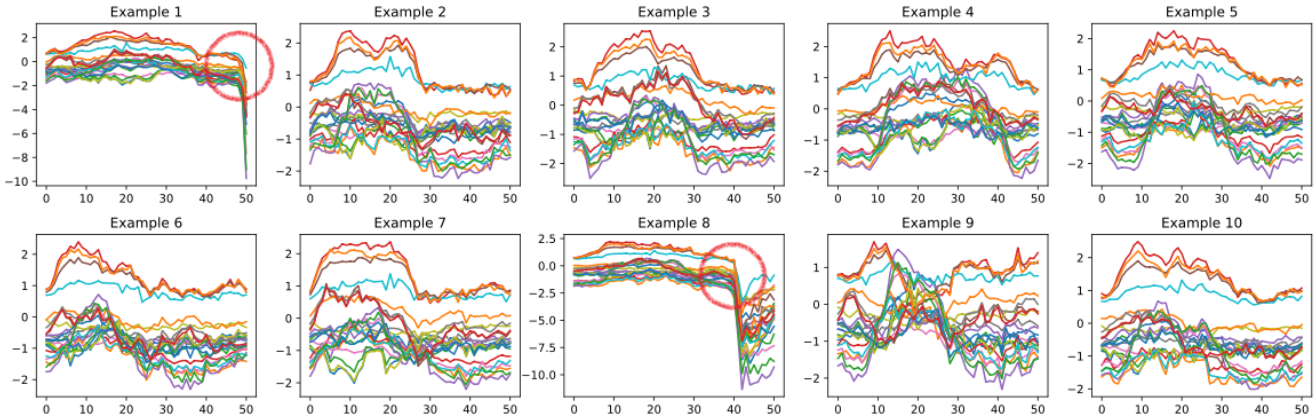

Fig. 6: Ten different time series samples generated by TAE

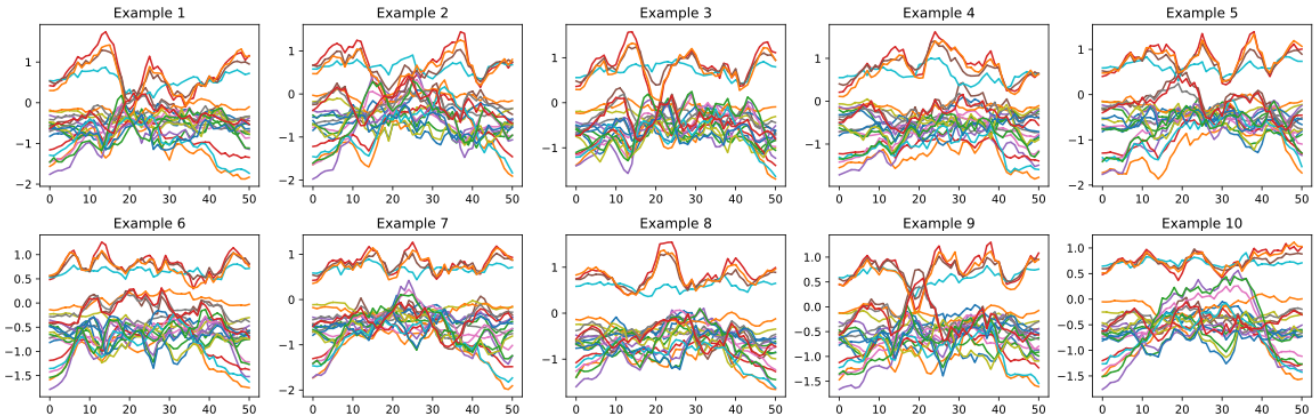

Fig. 7: Ten different time series samples generated by CAE

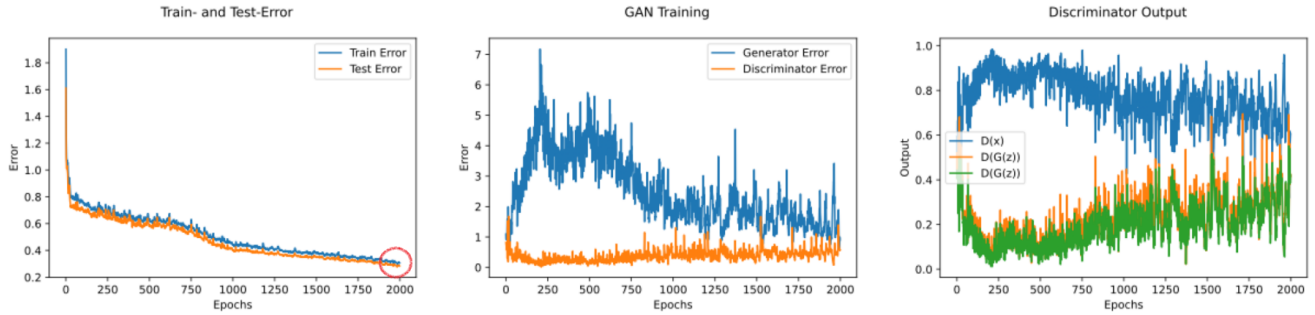

Fig. 8: Error plots of the transformer model regularized with the conventional GAN scheme (TAE-GAN)

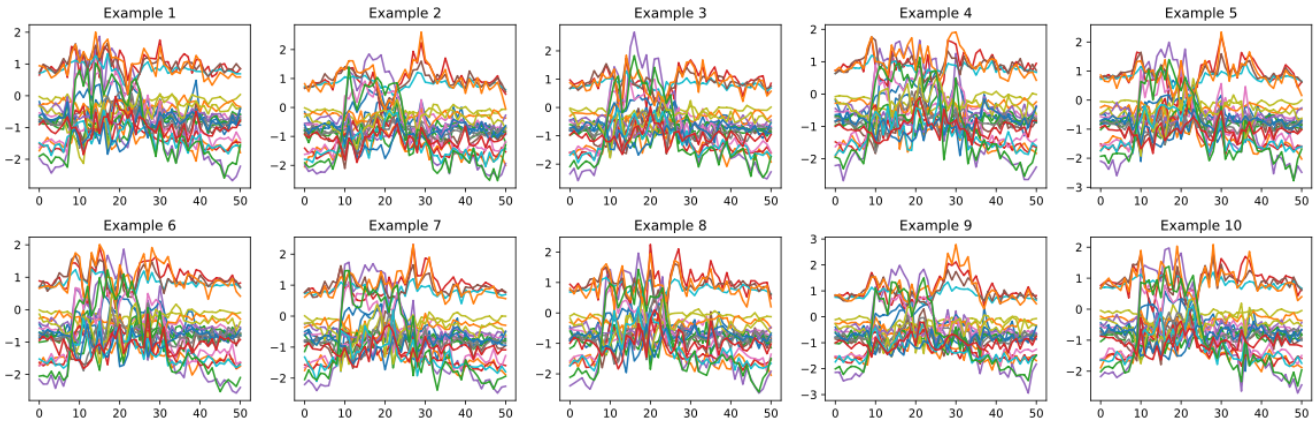

Fig. 9: Ten different time series samples generated by TAE-GAN

Fig. 10: Error plots of the transformer model regularized with the Wasserstein GAN scheme (TAE-WGAN)

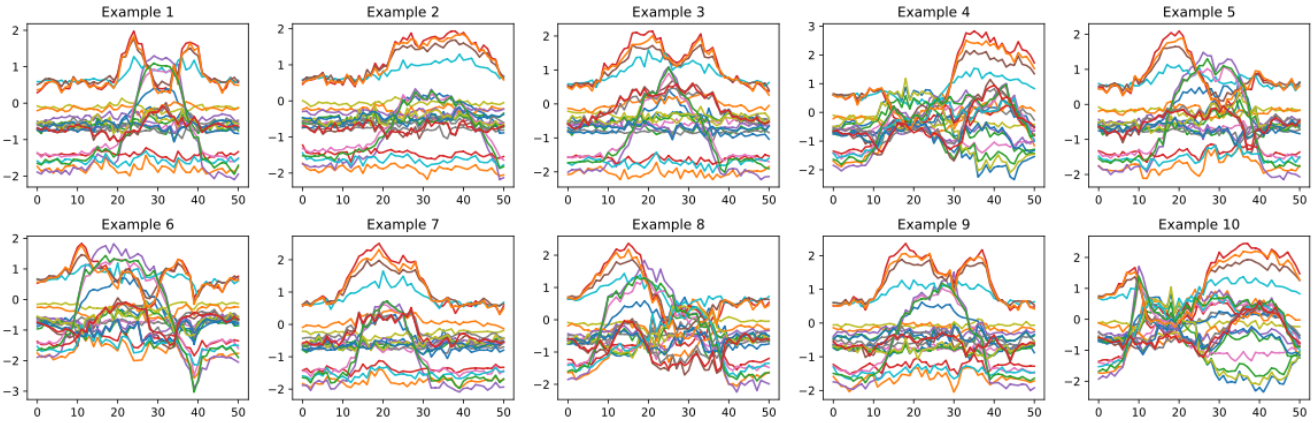

Fig. 11: Ten different time series samples generated by TAE-WGAN

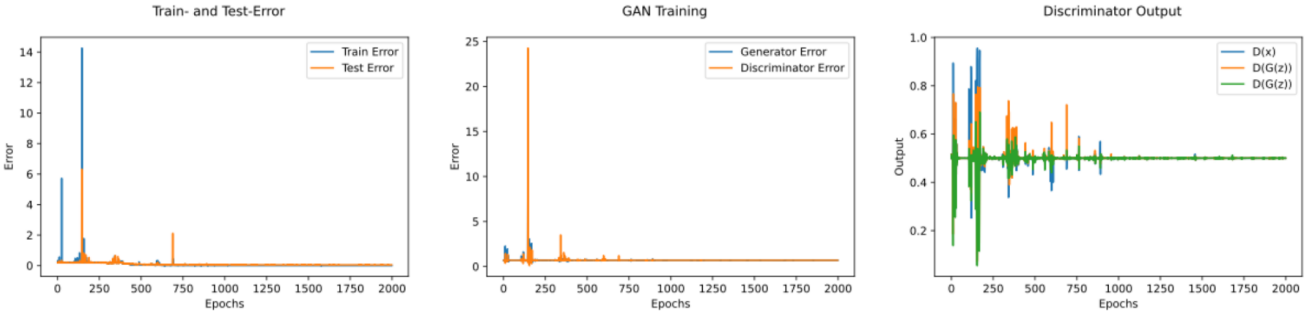

Fig. 12: Error plots of the convolutional model regularized with the convolutional GAN scheme (CAE-GAN)

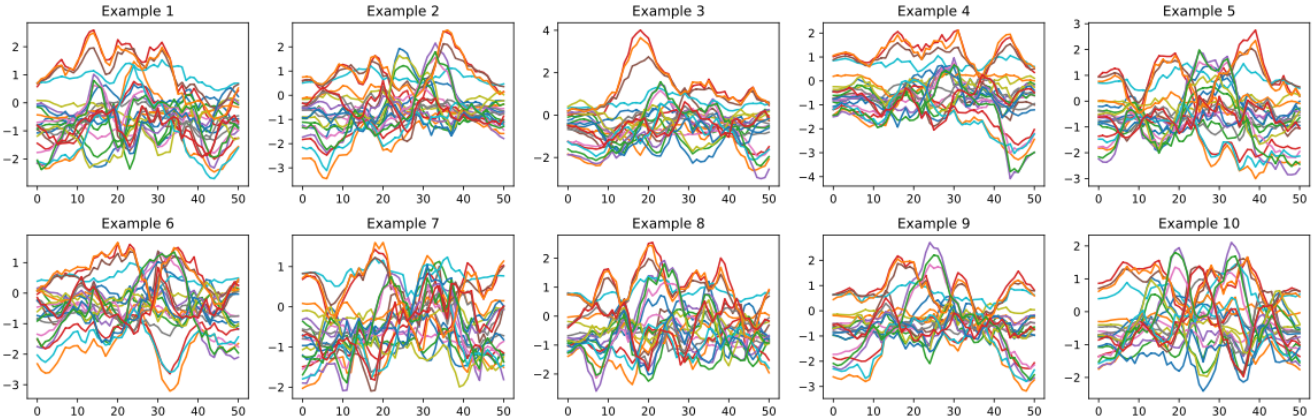

Fig. 13: Ten different time series samples generated by CAE-GAN

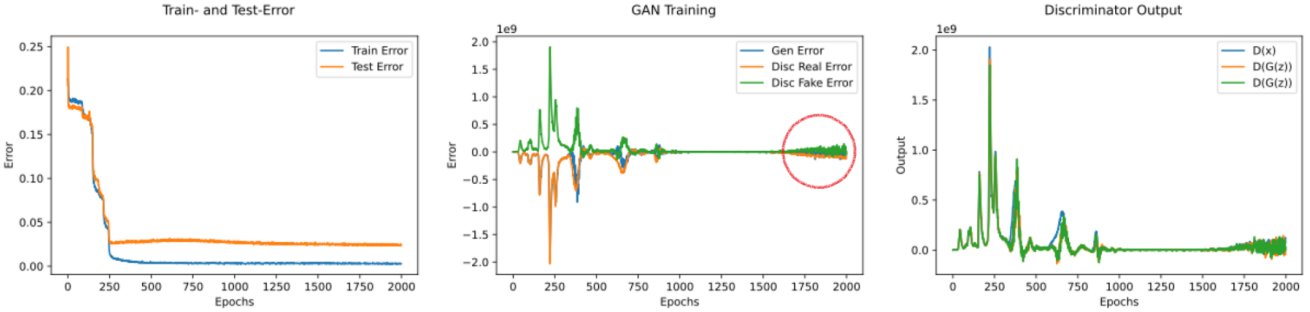

Fig. 14: Error plots of the convolutional model regularized with the Wasserstein GAN scheme (CAE-WGAN)

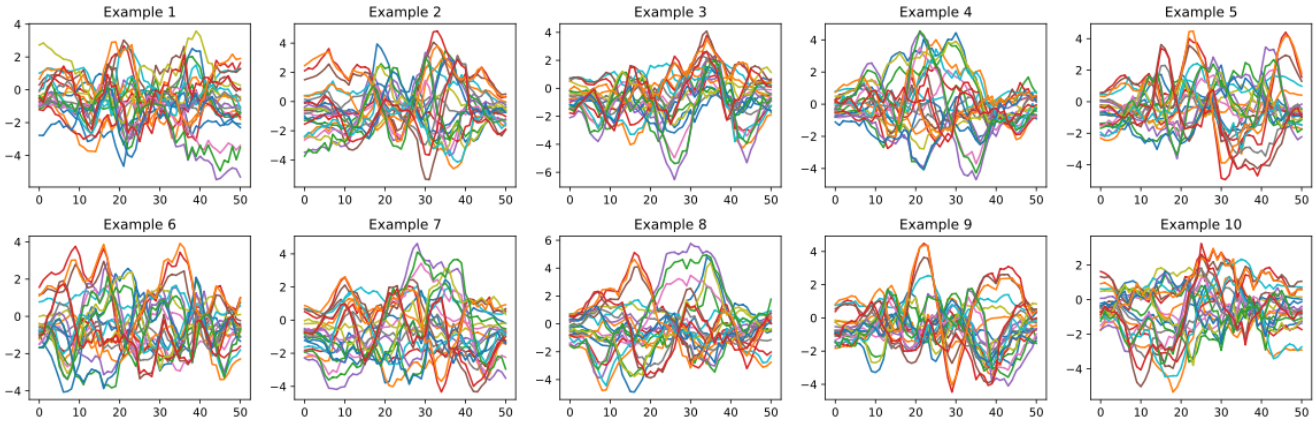

Fig. 15: Ten different time series samples generated by CAE-WGAN