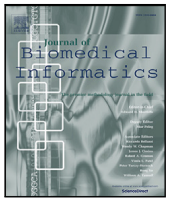
Journal of Biomedical Informatics 139 (2023) 104320

Contents lists available at ScienceDirect

Journal of Biomedical Informatics

journal homepage: www.elsevier.com/locate/yjbin



Original Research

Medical multivariate time series imputation and forecasting based on a recurrent conditional Wasserstein GAN and attention

Sven Festag *, Cord Spreckelsen

Institute of Medical Statistics, Computer and Data Sciences, Jena University Hospital, Germany
SMITH consortium of the German Medical Informatics Initiative, Germany

ARTICLE INFO

Keywords:
GAN
Time series
Imputation
Forecast
Machine learning

ABSTRACT

Objective: In the fields of medical care and research as well as hospital management, time series are an important part of the overall data basis. To ensure high quality standards and enable suitable decisions, tools for precise and generic imputations and forecasts that integrate the temporal dynamics are of great importance. Since forecasting and imputation tasks involve an inherent uncertainty, the focus of our work lay on a probabilistic multivariate generative approach that samples infillings or forecasts from an analysable distribution rather than producing deterministic results.

Materials and Methods: For this task, we developed a system based on generative adversarial networks that consist of recurrent encoders and decoders with attention mechanisms and can learn the distribution of intervals from multivariate time series conditioned on the periods before and, if available, periods after the values that are to be predicted. For training, validation and testing, a data set of jointly measured blood pressure series (ABP) and electrocardiograms (ECG) (length: $1,250 \triangleq 10\ s$) was generated. For the imputation tasks, one interval of fixed length was masked randomly and independently in both channels of every sample. For the forecasting task, all masks were positioned at the end.

Results: The models were trained on around 65,000 bivariate samples and tested against 14,000 series of different persons. For the evaluation, 50 samples were produced for every masked interval to estimate the range of the generated infillings or forecasts. The element-wise arithmetic average of these samples served as an estimator for the mean of the learned conditional distribution. The approach showed better results than a state-of-the-art probabilistic multivariate forecasting mechanism based on Gaussian copula transformation and recurrent neural networks. On the imputation task, the proposed method reached a mean squared error (MSE) of 0.057 on the ECG channel and an MSE of 28.30 on the ABP channel, while the baseline approach reached MSEs of 0.095 (ECG) and 229.1 (ABP). Moreover, on the forecasting task, the presented system achieved MSEs of 0.069 (ECG) and 33.73 (ABP), outperforming the recurrent copula approach, which reached MSEs of 0.082 (ECG) and 196.53 (ABP).

Conclusion: The presented generative probabilistic system for the imputation and forecasting of (medical) time series features the flexibility to handle masks of different sizes and positions, the ability to quantify uncertainty due to its probabilistic predictions, and an adjustable trade-off between the goals of minimising errors in individual predictions and minimising the distance between the learned and the real conditional distribution of the infillings or forecasts.

1. Introduction

In the biomedical domain, many measured values are part of time series representing the development of the measurand over time. Prominent examples of the temporal documentation of repeated readings in the clinical context are bedside monitors that continuously output health-related data of patients. Moreover, biomedical laboratories produce many time series data collections for research and care purposes. Other sectors producing time series data are hospital logistics and stock management that for example record temporal inventory data.

* Corresponding author at: Institute of Medical Statistics, Computer and Data Sciences, Jena University Hospital, Germany.
E-mail addresses: sven.festag@med.uni-jena.de (S. Festag), cord.spreckelsen@med.uni-jena.de (C. Spreckelsen).

https://doi.org/10.1016/j.jbi.2023.104320
Received 13 December 2022; Received in revised form 8 February 2023; Accepted 10 February 2023
Available online 13 February 2023

Nearly all employees working in these domains heavily rely on time series data and forecasts when it comes to making informed diagnoses, answering research questions, allotting hospital beds etc. Thus, time intervals of missing values in such series pose serious problems and can even be a risk to the health of patients. The reasons for missing time steps are manifold. They might arise due to lacking patient compliance, technical defects, human reading or notation errors and similar.

A well-performing imputation and forecasting system for biomedical time series data not only improves patient care and diagnostics but also aids basic research. For this reason, we developed and evaluated a machine learning system based on a generative adversarial neural network (GAN) that can learn temporal characteristics of biomedical domains and impute or forecast with estimates sampled from a conditional distribution incorporating the learned characteristics as well as the context intervals before and possibly after the missing values. There exist already many application-specific deterministic temporal imputation and forecasting methods suggested for the medical domain. In contrast to these, our proposed approach is a generic probabilistic system with as few as possible presumptions. It was assessed on a publicly available data set containing bivariate series of blood pressure and ECG measurements.

2. Related work

Many imputation and forecasting approaches have been developed and used in various domains, from very simple ones like mean imputation, over more complex statistical methods, to approaches based on Deep Learning. A summary of these methods can be found in the review papers by Fang and Wang [1] regarding the imputation task and the article published by Lara-Benítez et al. [2] regarding the forecasting task. In the latter work, the superiority of the long short-term network (LSTM) topology for deterministic forecasting has been empirically shown. In the end, the authors suggest focussing further studies on probabilistic models, GANs and encoder–decoder-based networks, which were not investigated by them. All these suggestions are integrated into the new model presented in the following.

The present section focuses on publications describing methods using GANs for time series imputation and forecasting. A comprehensive summary of related work can be found in our review [3].

The underlying idea of training a GAN, i.e. a generator and a critic in an adversarial fashion, was made famous by Goodfellow et al. in 2014 [4]. Although it was originally presented as a tool for image synthesis, it was adapted quickly for all kinds of data generation tasks [5,6]. Already in 2014, the concept of GANs was extended to conditional GANs that can not only learn to synthesise data samples but also to include context information into this task [7]. This extension is helpful for the time series imputation/forecasting problem, as conditional GANs can be trained to fill areas of missing values based on the learned conditional probability distribution, where the conditioning factors are the historic (and future) contexts as well as other clinical information regarding diagnoses, medication and laboratory findings. The terms historic and future context refer to the data points of the series that lie before or after the interval of missing values, respectively. Luo et al. were one of the first to present a multi-stage GAN-based system for the imputation of medical time series [8]. They even presented a new gated recurrent unit cell, the GRUI, that is specialised in the imputation task. They evaluated the method only on a medical data set with real missing values, such that no direct comparison (prediction vs. real) was possible.

A subsequent publication by the authors describes an improved version of their imputation system that learns in an end-to-end fashion and can directly impute values in a single-stage process [9]. The same medical data set as in their previous experiments was used, leading to the same limitation.

Some of the authors of the previously described articles recently published another text about how they improved their original work [10]. For this version, they applied professor forcing [11] during the training of a generator that works like an autoencoder. Still, no complete medical data set was used that would allow a direct evaluation.

The three approaches use other context inputs for their critics than for their generators or none at all. According to Fedus et al. this might lead to inaccurate learning [12]. The critics are only trained to differentiate between series with missing values and series that contain synthetic infillings. So the estimated distribution learned by the generator is unclear.

3. Method

We studied the ability of GANs to predict missing and future values when trained on complete time series. Furthermore, we ensured that the critic has access to the same context information as the generator and is trained to differentiate between synthetic and real values for the masked positions.

The developed system is based on a recurrent conditional generative adversarial neural network (rcGAN). Both the generator and the critic consist of two LSTM recurrent neural networks (RNN). Moreover, an attention mechanism is added to both opponents to help them identify relevant context information. Such attention proved useful for the text imputation with GANs [12]. An overview of the topology can be found in Figs. 1 and 2. In the following, the generator, its inputs and the output structure are described in more detail.

The data basis for training and testing consists of multivariate time series $\mathbf{S} \in \mathbb{R}^{n\times d}$ with d time steps and n channels. The actual input to the generator is twofold. On the one hand, it receives a masked version $\hat{\mathbf{S}} \in \mathbb{R}^{n\times d}$ of the series where masked values are replaced by zeros. On the other hand, a mask matrix $\mathbf{M} \in \{0,1\}^{n\times d}$ is handed to the generator indicating which steps were replaced. The positions of the 1s in $\mathbf{M}$ correspond to the masked values in $\mathbf{S}$. The first column vector $\mathbf{M}_{\_1}$ always equals $\mathbf{0}$, since we expect the first step to be available in every channel. During the training and inference for the future prediction task, the masked steps are placed at the end of the series to keep the same input interface throughout all phases.

The generator itself also contains two parts, a context encoder and a conditional decoder as the predictor. At first the input tuple $y = (\hat{\mathbf{S}}, \mathbf{M})$ is processed by the context encoder. It is built from n bidirectional LSTMs with h hidden states [13] that each process one row (channel) $\hat{\mathbf{S}}_{j\_}$ of the time series and the corresponding row in $\mathbf{M}$. Every LSTM ignores the masked time steps during its computation and repeats the last real output step at these positions in the output sequence. The final memory and cell state vectors $\mathbf{ms}_j$ and $\mathbf{cs}_j$ as well as the output sequences of the LSTMs $\mathbf{OE}_{j\_} = \left(\mathbf{oe}_{j,1}, \ldots, \mathbf{oe}_{j,d}\right)$ with $\mathbf{oe}_{j,t} \in \mathbb{R}^h$ contain condensed information about the context surrounding the missing steps. The individual results are concatenated into $\mathbf{ms} = \left[\mathbf{ms}_1^\top; \cdots; \mathbf{ms}_n^\top\right]^\top$, $\mathbf{cs} = \left[\mathbf{cs}_1^\top; \cdots; \mathbf{cs}_n^\top\right]^\top$ and $\mathbf{OE} = \begin{bmatrix}\mathbf{OE}_{1\_}\\ \vdots \\ \mathbf{OE}_{n\_}\end{bmatrix}$.

The conditional predictor makes use of one unidirectional LSTM, a feedforward attention network and a dense layer. The basic input is again the tuple y. The memory and cell states are initialised with noisy variants of $\mathbf{ms}$ or $\mathbf{cs}$, respectively. These versions are produced by adding additive standard uniform noise vectors. This noise input is needed by the predictor to produce probabilistic outputs and for the overall GAN system to train correctly. In every temporal step t, a concatenation of two vectors $\left[\mathbf{a}_t(\mathbf{OE})^\top; \tilde{\mathbf{s}}_t^\top\right]$ is used as the input with

$$\tilde{\mathbf{s}}_{t_j} := \begin{cases} \hat{\mathbf{S}}_{j,t}, \text{ if } \mathbf{M}_{j,t} = 0 \\ \mathbf{od}_{t-1_j}, \text{ if } \mathbf{M}_{j,t} = 1 \end{cases} \quad \text{(cf. Eq. (3))} \tag{1}$$

$$\mathbf{a}_t(\mathbf{OE}) := \sum_{i=1}^{d} \boldsymbol{\alpha}_{t_i}(\mathbf{OE}) \cdot \mathbf{OE}_{\_i} \quad \text{(cf. Eq. (5))} \tag{2}$$

Every intermediate output is further processed by a single dense layer with a linear activation function. Its n-dimensional output $\mathbf{l}_t$ is

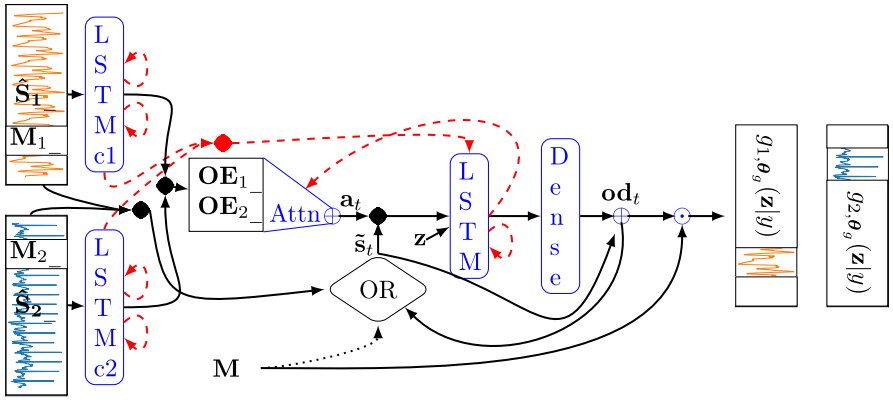


Fig. 1. Topology of the proposed generator network. Filled circles represent concatenations, black lines data flow, and red dashed lines the flow of hidden state values. Each masked input channel $\hat{\mathbf{S}}_i$ is processed recurrently by a bidirectional LSTM. The outputs are concatenated into the joint encoded context series **OE** that in turn is combined by an attention mechanism into a different $\mathbf{a}_t$ (decoder input) for every time step. The second input to the generating decoder at every time step is $\tilde{\mathbf{s}}_t$. It either contains the original time series value if it was not masked or the predicted output of the previous step $\mathbf{od}_{t-1}$. In combination with a noise input $\mathbf{z}$, these values form a single-step input for the unidirectional decoder LSTMs whose outputs are processed by the final dense layer to get the prediction for the next step. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

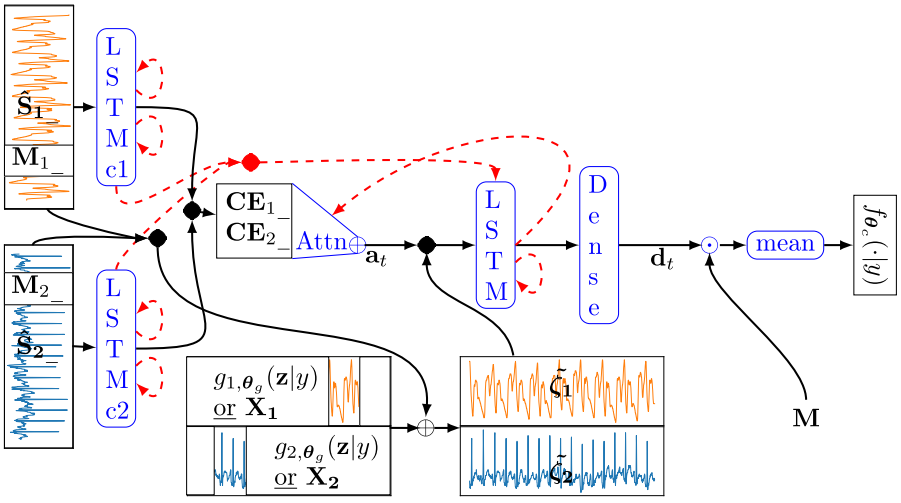


Fig. 2. Topology of the proposed critic network. The encoder part works exactly as in the generator. The decoder, however, is used to score the predictions of the generator or the real step values. Thus, the attention vector is combined with a real time step or a generated one before it is processed by a unidirectional LSTM and a dense layer to get the critic's score.

interpreted as the predicted difference between the time series values at time step $t+1$ and t. To get the final prediction for time step $t+1$, $\mathbf{od}_t$, this difference is added to the step input:

$$\mathbf{od}_t := \tilde{\mathbf{s}}_t + \mathbf{l}_t \tag{3}$$

Even for the non-masked positions, there are such predictions but they are not used for further computations. As described in Eq. (1), the decoder's output $\mathbf{od}_{t-1_j}$ is only reused as $\tilde{\mathbf{s}}_{t_j}$ if time step t is masked in the jth channel of the input series.

By applying an additive attention mechanism that is similar to the one presented by Bahdanau et al. [14] and to the global attention mechanism by Luong et al. [15], the most important elements of the encoder's output are determined and combined into a weighted sum $\mathbf{a}_t(\mathbf{OE})$ in every step. To this end, the weights $\boldsymbol{\alpha}_{t_j}(\mathbf{OE})$ are computed with the help of an attention network for every time step. Let $\mathbf{md}_{t-1}$ and $\mathbf{cd}_{t-1}$ be the memory and cell state of the decoder LSTM at time point $t-1$. The attention weights for time point t can then be computed as follows.

$$\mathbf{k}_{t_i}(\mathbf{OE}) := \mathbf{w}^{(\mathbf{oe},3)\top} \tanh\left(\mathbf{W}^{(\mathbf{oe},1)} \cdot \left[\mathbf{md}_{t-1}^\top; \mathbf{cd}_{t-1}^\top\right]^\top + \mathbf{W}^{(\mathbf{oe},2)} \cdot \mathbf{OE}_{\_i}\right) \tag{4}$$

$$\boldsymbol{\alpha}_{t_i}(\mathbf{OE}) := \frac{\exp\left(\mathbf{k}_{t_i}(\mathbf{OE})\right)}{\sum_{r=1}^{d} \exp\left(\mathbf{k}_{t_r}(\mathbf{OE})\right)} \tag{5}$$

The weight matrices $\mathbf{W}^{(\mathbf{oe},1)}, \mathbf{W}^{(\mathbf{oe},2)}$ and the weight vector $\mathbf{w}^{(\mathbf{oe},3)}$ are part of the learnable parameters of the generator θ_g.

Besides feeding additive noise to the generator via the state initialisation, another noise is added by applying dropout to its predictor LSTM, more precisely to the weights that linearly transform the inputs $\tilde{\mathbf{s}}$. It must be noted that the dropout is applied during training and the inference phase because it is not (only) used for regularisation but to induce noise needed by the generator. A similar setup has already proven successful in the image-to-image translation domain [16]. Both noise sources are summarised by the $\mathbf{z}$ vector in Fig. 1. Since only the masked values need to be imputed or forecasted by the conditional generator, its final output is defined as

$$g_{\theta_g}(\mathbf{z}|y) := \mathbf{M} \odot \left(\mathbf{0}, \mathbf{od}_1, \ldots, \mathbf{od}_{d-1}\right) \tag{6}$$

where $\odot$ defines the element-wise matrix multiplication.

The critic has a similar structure as the generator (cf. Fig. 2), combining a context encoder with a decision network. It also makes use of the input tuple y. If the critic did not know which time steps

are masked and which are real, it would produce inaccurate learning signals for the generator [12]. As in the original GAN framework, the critic feeds alternately on two different input distributions during training. On the one hand, it works on series of original time steps that were measured at the masked positions $\mathbf{X} := \mathbf{M} \odot \mathbf{S}$. On the other hand, it "criticises" the infillings or predictions computed by the generator at these positions $g_{\theta_g}(\mathbf{z}|y)$.

The encoder works exactly like its counterpart in the generator (cf. left parts of Figs. 1 and 2). It uses the same conditioning information y for the two different inputs $\mathbf{X}$ and $g_{\theta_g}(\mathbf{z}|y)$. Hence, its output sequence $\mathbf{CE}$ is the same in both cases.

The role of the critic in the GAN framework is to score the real and probabilistically generated time series values. These scores are in turn used to guide the training of the generator. For this scoring, the critic makes use of a unidirectional LSTM and an attention network. The input in every step can be described as a pair $\left[\mathbf{a}_t(\mathbf{CE})^\top; \tilde{\boldsymbol{\zeta}}_t^\top\right]$, where the weighted attention sums $\mathbf{a}_t(\mathbf{CE})$ are computed as described in Eqs. (2), (4) and (5) with the only difference being that the learnable weights in (4) are changed to $\mathbf{W}^{(\mathbf{ce},1)}, \mathbf{W}^{(\mathbf{ce},2)}$ and $\mathbf{w}^{(\mathbf{ce},3)}$. Furthermore, $\tilde{\boldsymbol{\zeta}}_t$ is set to $\left(\mathbf{X} + (\mathbf{1} - \mathbf{M}) \odot \hat{\mathbf{S}}\right)_{\_t}$ during the training part, where the critic works on the original data or equals $\left(g_{\theta_g}(\mathbf{z}|y) + (\mathbf{1} - \mathbf{M}) \odot \hat{\mathbf{S}}\right)_{\_t}$ during the part, where it is engaged with the generator's output, respectively. The latter summand can be computed from y in both cases.

The output of every LSTM step is further processed by a dense layer with a linear activation to get the scoring vector $\mathbf{d}_t$ of the critic. To achieve a single scalar critic output $f_{\theta_c}(\mathbf{X}|y)$ or $f_{\theta_c}(g_{\theta_g}(\mathbf{z}|y)|y)$ the mean of the individual scores at the masked positions is computed:

$$f_{\theta_c}(\cdot|y) := \frac{1}{\sum_{t=1}^{d} \|\mathbf{M}_{\_t}\|_1} \sum_{t=1}^{d} \mathbf{M}_{\_t}^\top \mathbf{d}_t \tag{7}$$

During preliminary experiments with the classic conditional GAN optimisation

$$\min_{\theta_g} \max_{\theta_c} \Big\{ \mathbb{E}_{\substack{y\sim p_{\text{con}}(y) \\ \boldsymbol{X}\sim p_{\text{data}}(\boldsymbol{X}|y)}} \Big[\log f_{\theta_c}(\boldsymbol{X}|y)\Big] + \mathbb{E}_{\substack{y\sim p_{con}(y) \\ \mathbf{z}\sim p_{\text{noise}}(\mathbf{z})}} \Big[\log\Big(1 - f_{\theta_c}(g_{\theta_g}(\mathbf{z}|y)|y)\Big)\Big] \Big\} \tag{8}$$

which is based on the Jensen–Shannon divergence [4,7], we encountered unstable optimisation processes probably due to uninformative loss gradients. This finding is in line with the ones by Zhou et al. and Zhang et al. [10,17]. The three probability density functions in this "min–max" formulation correspond to the two unknown real-world distributions of the time series context information ($p_{\text{con}}(y)$) and of the measured values at the masked positions conditioned on the context information ($p_{\text{data}}(\mathbf{X}|y)$) as well as the distribution of the noise induced into the generator ($p_{\text{noise}}(z)$).

To circumvent the problem during optimisation, the Wasserstein-1 algorithm was applied for the actual training [18]. This means, in every training iteration the critic estimates the (scaled) Wasserstein-1 distance between the real conditional time series distribution and the one implicitly defined by the generator. To this end, it iteratively approximates

$$\max_{\theta_c} \Big\{ \mathbb{E}_{\substack{y\sim p_{\text{con}}(y) \\ \boldsymbol{X}\sim p_{\text{data}}(\boldsymbol{X}|y)}} \Big[f_{\theta_c}(\boldsymbol{X}|y)\Big] - \mathbb{E}_{\substack{y\sim p_{\text{con}}(y) \\ \mathbf{z}\sim p_{\text{noise}}(\mathbf{z})}} \Big[f_{\theta_c}(g_{\theta_g}(\mathbf{z}|y)|y)\Big] \Big\} \tag{9}$$

while clipping all the critic's weights to $[-0.1, 0.1]$ to approximately enforce the Lipschitz constraint [18]. The generator's task is to optimise this estimate by minimising the latter part:

$$\min_{\theta_g} \Big\{ -\mathbb{E}_{\substack{y\sim p_{con}(y) \\ \mathbf{z}\sim p_{\text{noise}}(\mathbf{z})}} \Big[f_{\theta_c}(g_{\theta_g}(\mathbf{z}|y)|y)\Big] \Big\} \tag{10}$$

In addition to minimising the estimated Wasserstein-1 distance, the generator also tries to minimise the expected Huber loss $\mathcal{H}$ ($\delta = 1$) between predicted and real step values at the masked positions:

$$\min_{\theta_g} \Big\{ \mathbb{E}_{\substack{y\sim p_{\text{con}}(y); \mathbf{z}\sim p_{\text{noise}}(\mathbf{z}) \\ \mathbf{X}\sim p_{\text{data}}(\mathbf{X}|y)}} \Big[\mathcal{H}(\mathbf{X}, g_{\theta_g}(\mathbf{z}|y))\Big] \Big\} \tag{11}$$

This optimisation process was implemented as shown in Algorithm 1.

Table 1
Quantitative summary of the data sets utilised during the experiments.

Set	Persons	Samples	Frequency	Size	Channels
Training T	275	26,939			
Validation V	61	5,972	125 Hz	10 s	ABP, ECG
Test U	65	6,062			

4. Experiments

In this section, the data collection process, the preprocessing as well as the parameter settings are described. Moreover, we detail the experiments conducted to generate baseline results. All approaches were implemented in Python 3.

4.1. Data set

All data used in the described experiments were taken from the "Autonomic Aging" set [19] hosted in the publicly available PhysioNet framework [20]. We only used data recorded by the MP150 (ECG100C, BIOPAC systems inc.) and the CNAP 500 (CNSystems Medizintechnik GmbH) systems. A summary of the employed data sets is given in Table 1.

The training, test and validation sets contain preprocessed bivariate series of non-invasively measured arterial blood pressure in mmHg (ABP) and electrocardiograms written from lead II in mV (ECG). Every sample consists of an interval spanning 10 s with a digitisation frequency of 125 Hz, which leads to series vectors of size $d = 1,250$. If the series are longer than 10 s they are subdivided into several samples. The various intervals of one volunteer do not overlap and no person is represented in more than one of the three sets. In total, the training set T consists of 26,939 samples from 275 adult volunteers. The test U and validation V sets comprise 6062 (65 volunteers) or 5972 (61 volunteers) samples, respectively. The exclusion criteria were the existence of NaN values, less than 10 individual step values in one channel and negative blood pressure values.

The mask matrix $\mathbf{M}$ of every series $\mathbf{S}$ contains exactly 30 neighbouring 1s in every channel. That means, in the masked series $\hat{\mathbf{S}}$ exactly one subinterval of 0.24 s is missing per channel. During the experiments regarding the imputation task, we chose these areas uniformly at random. The placement of the missing intervals was done independently in the two channels. During the forecasting experiments, the missing parts were all placed exactly at the end of the series. To balance the focus between both input channels, they are normalised individually in each sample based on the means and variances determined on the training set.

4.2. Baseline: GPVAR estimator

For the baseline experiments, we used the multivariate forecasting approach GP-Copula that is based on gaussian process vector autoregression with gaussian copula transformation and a recurrent neural network. The system was suggested by David Salinas et al. and outperformed linear auto-regressive models as well as pure recurrent neural networks in several real-world experiments [21]. Similar to the proposed rcGAN system it is a probabilistic system that can be used to sample several forecasts for the same context information. The approach was integrated into the Gluon Time Series Toolkit [22] under the name of "GPVAR estimator" that we used for our experiments. Since the model only supports forecasting, we adapted the imputation task accordingly. During training, every bivariate series was handed to the model in two versions. In the first version, the "forecast" mask for *both* channels was set to the mask position given for the first channel and in the second version to the position for the second signal. The context information after the missing positions were not used. During inference, the model was also queried twice for every imputation. At

Algorithm 1 rcGAN with hybrid optimisation of Wasserstein-1 metric and Huber loss. cf. [18].

Require: η: learning rate of RMSprop, m: mini-batch size; n_{critic}: number of critic iterations to approximate Wasserstein distance; α, β: weighting hyperparameters to combine equations (10) and (11); $\boldsymbol{\theta}_{c,0}$, $\boldsymbol{\theta}_{g,0}$: initial learnable parameters;

$\boldsymbol{\theta}_g \leftarrow \boldsymbol{\theta}_{g,0}$; $\boldsymbol{\theta}_c \leftarrow \boldsymbol{\theta}_{c,0}$; $\mathbf{rms}_c \leftarrow \mathbf{0}$; $\mathbf{rms}_g \leftarrow \mathbf{0}$

while stop criterion not reached **do**

  for t = $1, \dots, n_{critic}$ **do**

    Sample mini-batches $\left(y_i\right)_{i=1}^{m} \sim p_{\text{con}}(y)$; $\left(\mathbf{z}_i\right)_{i=1}^{m} \sim p_{\text{noise}}(\mathbf{z})$

    Sample m "dependant" $\mathbf{X}_i \sim p_{\text{data}}(\mathbf{X}|y_i)$

    $\mathbf{grad} \leftarrow \nabla_{\theta_c} \frac{1}{m} \left[-\sum_{i=1}^{m} f_{\theta_c}(\mathbf{X}_i|y_i) + \sum_{i=1}^{m} f_{\theta_c}\left(g_{\theta_g}(\mathbf{z}_i|y_i)|y_i\right) \right]$

    $\mathbf{rms}_c \leftarrow 0.9 \cdot \mathbf{rms}_c + 0.1 \cdot \mathbf{grad} \odot \mathbf{grad}$

    $\boldsymbol{\theta}_c \leftarrow \boldsymbol{\theta}_c - \eta \cdot \frac{\mathbf{grad}}{\sqrt{\mathbf{rms}_c + 10^{-7}}}$

    $\boldsymbol{\theta}_c \leftarrow \text{clip}(\boldsymbol{\theta}_c, [-0.1, 0.1]^d)$

  end for

  Sample mini-batches $\left(y_i\right)_{i=1}^{m} \sim p_{\text{con}}(y)$; $\left(\mathbf{z}_i\right)_{i=1}^{m} \sim p_{\text{noise}}(\mathbf{z})$

  Sample m "dependant" $\mathbf{X}_i \sim p_{\text{data}}(\mathbf{X}|y_i)$

  $\mathbf{grad} \leftarrow \nabla_{\theta_g} \frac{1}{m} \left[-\alpha \cdot \sum_{i=1}^{m} f_{\theta_c}\left(g_{\theta_g}(\mathbf{z}_i|y_i)|y_i\right) + \beta \cdot \mathcal{H}(\mathbf{X}_i - g_{\theta_g}(\mathbf{z}_i|y_i)) \right]$

  $\mathbf{rms}_g \leftarrow 0.9 \cdot \mathbf{rms}_g + 0.1 \cdot \mathbf{grad} \odot \mathbf{grad}$

  $\boldsymbol{\theta}_g \leftarrow \boldsymbol{\theta}_g - \eta \cdot \frac{\mathbf{grad}}{\sqrt{\mathbf{rms}_g + 10^{-7}}}$

end while

first, the channel with the earliest mask position was filled in. For this prediction, the context of the other channel up to this first mask position was used as well. Afterwards, the model was queried a second time with the now partially imputed series to get the prediction for the channel with the latter mask.

The systems were trained for 105 epochs with batches of size 1000. It was optimised with Adam (initial learning rate: 0.001, weight decay: 10^{-8}) and a learning rate reduction approach instead of early stopping. The two corresponding experiments are referred to as GP_{imp} and GP_{fut}

4.3. rcGAN with hybrid wasserstein

The implementation of the rcGAN as well as its optimisation according to algorithm 1 draws upon the machine learning framework TensorFlow.[1] The topology of the rcGAN followed the description given in Section 3. The dimensionality of the used series equated $d \times n = 1,250 \times 2$. The unidirectional LSTMs made use of hidden states of size 10, whereas the bidirectional version had hidden states of size 5 (in each direction). The dropout probability in the generator was set to 5%.

We conducted several experiments for each of the imputations and the future prediction task. All of the corresponding training runs started with a pre-training phase. During this phase, the generator was individually trained on T to minimise $\mathcal{H}(\mathbf{X} - g_{\theta_g}(\mathbf{z}|y))$ directly with RMSprop (batch size: 50, learning rate: 0.001, initialisation: Glorot Uniform (weights), zeros (biases)). This pretraining initially slows down the convergence of the critic's performance and then ensures a stable adversarial learning process. In the main training phase, the hybrid Wasserstein training outlined in algorithm 1 was conducted ($\eta = 0.005, m = 50, n_{critic} = 5, \alpha = 1, \beta = 0.5$). The learnable parameters of the generator were initialised with the values learned during pre-training ($\boldsymbol{\theta}_{g,0}$), whereas the initial values for the critic $\boldsymbol{\theta}_{c,0}$ were determined with the Glorot Uniform approach (and zero biases). After every epoch, the performance of the generator (MSE) was evaluated on V. During this stage, only one imputation or prediction sample was generated for every input series. In the end, the weights $\boldsymbol{\theta}_g$ and $\boldsymbol{\theta}_c$ were reset to the versions leading to the best validation performance (early stopping), before the final evaluation was conducted on U. Based on the test set we also evaluated the statistic characteristics of the adversarially trained generator with a Monte Carlo average.

[1] https://tensorflow.org/

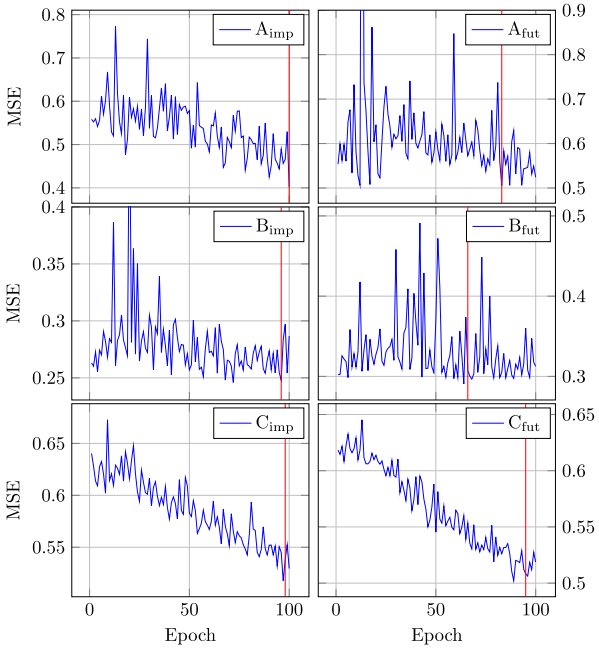


Fig. 3. Development of the mean squared errors reached on the test set throughout the experiments (A), (B) and (C). The red vertical lines mark the epochs at which the smallest errors were reached on the validation set. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

The different experiments are summarised in the following. In the first round, round (A), we pre-trained the generators for 5 epochs, before shifting to the GAN trainings for 100 epochs. This was executed independently for the imputation (A_{imp}) and the future prediction (A_{fut}) tasks. For the second group (B) we elongated the pre-training to 105 epochs for each task, to analyse the influence of the non-probabilistic part of the training on the overall performance.

To analyse the influence of different building blocks in the presented system on performance and training behaviour, we also conducted ablation studies. Thus, we evaluated the deterministic performances of the generators after the pre-training of (B) in experiments (B_{det}) and repeated the imputation experiment of round (A) without the additive noise input to the initial states in (A_{no}).

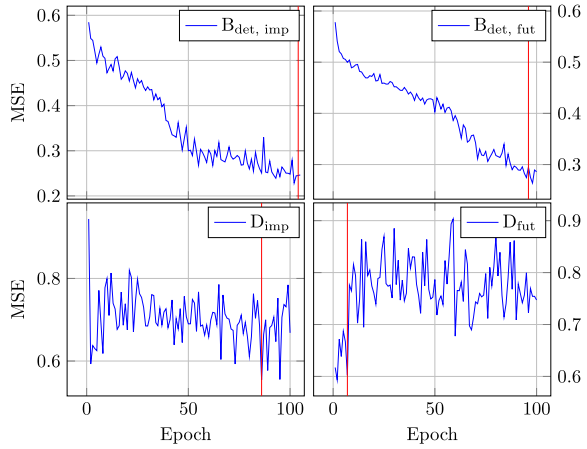


Fig. 4. Development of the mean squared errors reached on the test set throughout the experiments (B_{det}) and (D). The errors reached during (D) cannot be compared with the other runs as they correspond to the more complex task of predicting/imputing 250 steps instead of 30. The red vertical lines mark the epochs at which the smallest errors were reached on the validation set. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Furthermore, in phase (C) we conducted experiments similar to the ones described in (A) but with univariate versions of the data sets. To this end, all samples were subdivided into two individual series containing only ABP or ECG signals. With this trial the utilisation of information regarding temporal inter-channel dynamics for imputations and predictions by the rcGAN was studied. In the last round (D) the ability of the system to learn to impute and predict missing intervals of different sizes was inspected. Two rcGANs were pre-trained for 50 and trained for 100 epochs on the bivariate data sets with masks of size 250 (3.3 s). This system was tested against the original test set with masks of size 30 (D_{30}) and against a test set version with masks of size 250 (D_{250}).

5. Results

Figs. 3 and 4 summarise the training dynamics of the experiments detailed in Section 4.3. Every plot shows the development of the mean squared error between the measured step values of the test samples and the predicted/imputed ones by the generator over the training epochs. For this error computation, the generator produced only one prediction per sample. The vertical red line notes the epoch which led to the smallest MSE on the validation set. During training, the errors are computed on the normalised series.

Table 2 contains the results of the Monte Carlo simulations. For these, we sampled 50 infillings or predictions for every masked interval with the help of the trained generators or GPVAR models, respectively. Afterwards, the lower (0.25) and upper (0.75) quartiles were computed element-wise. Moreover, the arithmetic mean of all 50 series was computed. Thus, there were three imputations/forecasts for every sample in the end summarising the estimated range of possible outcomes and their mean. Before the error computation, the still normalised samples

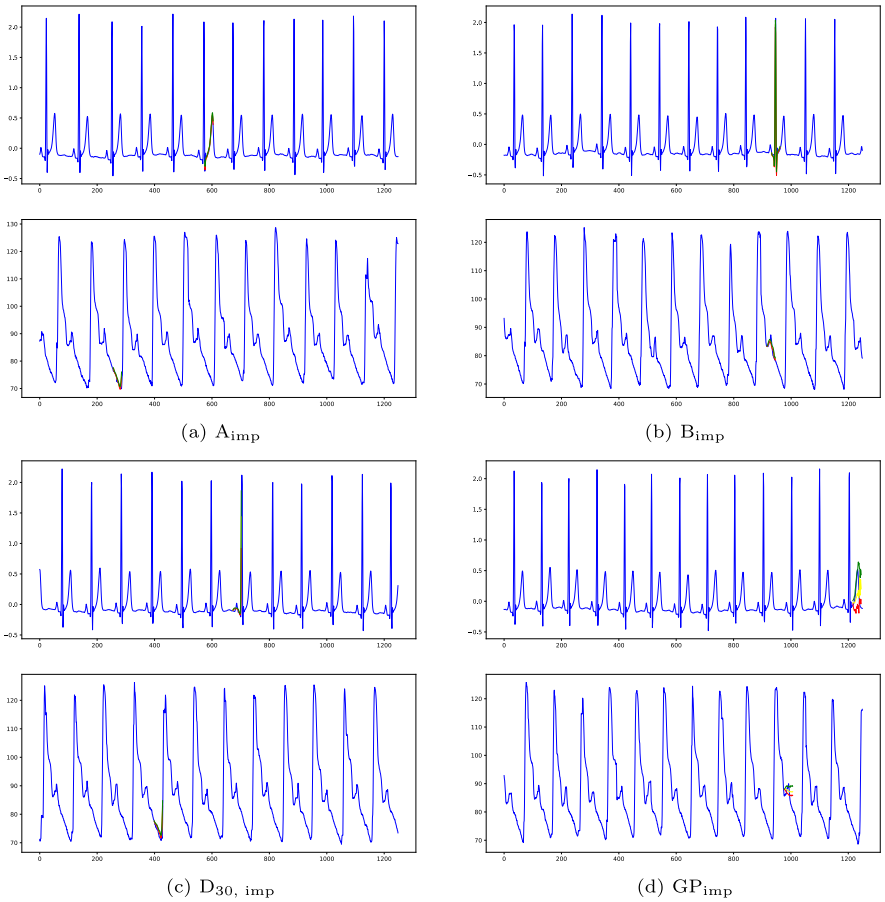


Fig. A.5. Exemplary short imputations (30 steps) produced during the experiments. The blue line depicts the full measured series, while the green, red and yellow lines correspond to the upper or lower quartile or the mean imputation, respectively. The upper series represent the ECG signals in mV and the lower ones depict the ABP data in mmHg. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Table 2
Summary of the de-normalised mean squared error achieved during the experiments with respect to the test set. For the quartile columns only the lower or upper quartile of the 50 predictions were considered for every test sample. For the mean column, the average of all predictions was computed before the error was calculated for the time series. The best two results with the smallest error between the mean estimator prediction and the measured series are marked in bold (imputation) or italics (forecasting).

	MSE 0.25 quartile		MSE mean		MSE 0.75 quartile	
	ECG (10^{-2})	ABP	ECG (10^{-2})	ABP	ECG (10^{-2})	ABP
A_{imp}	5.95	28.49	***5.66***	28.30	5.98	29.33
$A_{no,\ imp}$	7.38	28.38	7.04	27.57	7.56	27.88
A_{fut}	7.24	33.66	**6.89**	33.73	7.62	35.08
B_{imp}	3.66	18.88	***3.31***	***18.46***	3.49	19.10
B_{fut}	4.58	23.51	**4.35**	**22.23**	4.62	22.16
C_{imp}	7.67	71.34	7.57	70.82	7.51	74.59
C_{fut}	7.17	63.01	7.01	61.99	6.99	63.75
GP_{imp}	9.53	236.24	9.51	229.01	9.95	305.87
GP_{fut}	11.25	397.57	8.20	196.53	9.08	272.89
$D_{30,\ imp}$	7.34	20.32	7.28	***20.12***	8.49	21.11
$D_{30,\ fut}$	7.98	29.51	7.57	**29.95**	8.33	31.61
$D_{250,\ imp}$	8.50	50.56	8.58	49.71	9.54	52.22
$D_{250,\ fut}$	9.39	109.12	8.91	107.06	9.56	113.88

were de-normalised to be in the same scale as the original input. Table 2 lists the MSE for the three quartiles computed in the different experiments and individually for the two channels. Visualisations of the exemplary outputs can be found in Appendices A.5, B.6, C.7.

For the deterministically trained networks, there exists only one MSE value per channel and experiment. During $B_{det,\ imp}$ MSEs of 0.0379 (ECG) and 18.6353 (ABP) were reached and during $B_{det,\ fut}$ the errors were 0.0429 (ECG) and 22.5195 (ABP).

6. Discussion

The results showed that the presented rcGAN approach can learn the probabilistic imputation of missing areas in multivariate time series as well as the probabilistic forecast into the future. In both tasks, it could outperform the well-established probabilistic GPVAR model. The advantage of the proposed approach over other frameworks like VARMAX and state space models is that it can handle different mask positions for the various input channels. Moreover, it does not depend on domain knowledge about season lengths or transformation functions from latent space to data space etc. It can easily be extended to handle more than two input channels. The results of phase (C) suggest that the model learns to integrate context information of one channel to predict missing values of the other. Interestingly, the models trained for longer masks in phase (D) performed well on smaller masks as well. Thus, the proposed system can be utilised for inputs with missing areas of different sizes during inference without any retraining.

The omission of additional noise input (A_{no}) on top of dropout noise leads to a large relative increase in the error of the ECG signal while it decreases the error in the ABP only slightly. So the additive noise input seems to be advantageous for certain kinds of data. However, the optimal magnitude of noise input needs to be studied in upcoming projects. Another hyperparameter whose influence needs to be studied in the future is the number of pretraining steps. During the deterministic pretraining of the generator, the error can be decreased very fast as the results of B_{det} and B showed. Though, these systems do not learn the estimated real distribution of the signal data, which is needed for the probabilistic imputation and prediction as well as for the corresponding quantification of uncertainty. Hence, the pretraining of the generator up to a certain convergence criterion followed by the adversarial training phase seems to be the best approach for combining both goals. A visual analysis of the produced imputations prompts that the generators struggle to correctly fill the masked areas if they are very close to the beginning of the series.

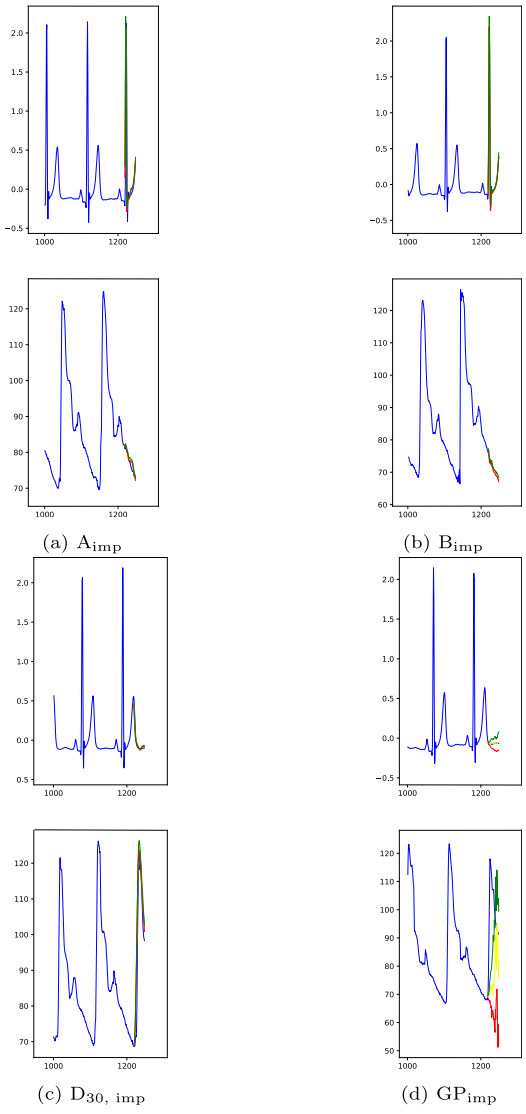
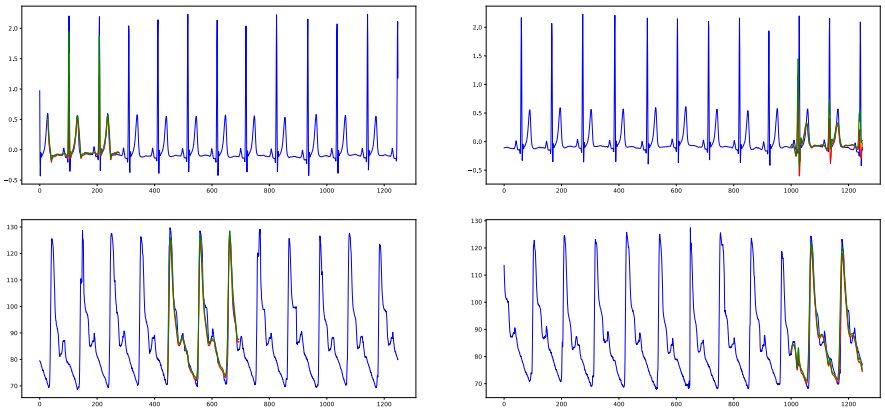


Fig. B.6. Exemplary short forecasts (30 steps) produced during the experiments. The blue line depicts an excerpt of the measured series, while the green, red and yellow lines correspond to the upper or lower quartile or the mean imputation, respectively. The upper series represent the ECG signals in mV and the lower ones depict the ABP data in mmHg. The first 1000 context values are omitted. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

7. Conclusion

We developed a new recurrent conditional GAN based on LSTM cells and an attention mechanism for the probabilistic multivariate time series imputation and forecasting task. When tested on blood pressure and ECG series generated by bedside monitors that each had one interval of missing values at a random position or the end, it showed better results than the GPVAR model.

Its advantages are the flexibility to handle masks of different sizes and positions, its ability to quantify uncertainty due to its probabilistic predictions, and the adjustable trade-off between the goals of minimising errors of individual predictions and minimising the distance between the learned and the real conditional distribution of the infillings or forecasts. The system might serve as a building block



(a) $\mathrm{GP}_{\mathrm{imp}}$ (b) $\mathrm{D}_{250,\ \mathrm{imp}}$

Fig. C.7. Exemplary long imputations and forecasts (250 steps) produced during the experiments. The blue line depicts the full measured series, while the green, red and yellow lines correspond to the upper or lower quartile or the mean imputation, respectively. The upper series represent the ECG signals in mV and the lower ones depict the ABP data in mmHg. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

for anomaly detectors, duty schedulers, automatic patient monitoring alarms and disease progression predictors.

CRediT authorship contribution statement

Sven Festag: Conceptualization, Methodology/study design, Software, Formal analysis, Investigation, Writing – original draft, Writing – review & editing, Visualization. **Cord Spreckelsen:** Conceptualization, Validation, Resources, Writing – original draft, Writing – review & editing, Supervision, Funding acquisition.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Sven Festag reports a relationship with RELX Plc that includes: equity or stocks.

Acknowledgements

The project reported here was supported by the German Federal Ministry of Education and Research (grant number 01ZZ1803C) in the context of the Smart Medical Information Technology for Healthcare consortium.

Appendix A. Examples: Imputation

See Fig. A.5.

Appendix B. Examples: Forecasting

See Fig. B.6.

Appendix C. Examples: Long imputation and forecasts

See Fig. C.7.

References

[1] Chenguang Fang, Chen Wang, Time series data imputation: A survey on deep learning approaches, 2020, arXiv:2011.11347.

[2] Pedro Lara-Benítez, Manuel Carranza-García, José C. Riquelme, An experimental review on deep learning architectures for time series forecasting, Int. J. Neur. Syst. 31 (03) (2021) 2130001.

[3] Sven Festag, Joachim Denzler, Cord Spreckelsen, Generative adversarial networks for biomedical time series forecasting and imputation, J. Biomed. Inform. 129 (2022) 104058.

[4] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio, Generative adversarial nets, in: Z. Ghahramani, M. Welling, C. Cortes, N.D. Lawrence, K.Q. Weinberger (Eds.), Advances in Neural Information Processing Systems,Vol. 27, Curran Associates, Inc., 2014, pp. 2672–2680.

[5] Syed Khurram Jah Rizvi, Muhammad Ajmal Azad, Muhammad Moazam Fraz, Spectrum of advancements and developments in multidisciplinary domains for generative adversarial networks (GANs), Arch. Comput. Methods Eng. (2021) 1–19.

[6] Alankrita Aggarwal, Mamta Mittal, Gopi Battineni, Generative adversarial network: An overview of theory and applications, Int. J. Inform. Manag. Data Insights 1 (1) (2021) 100004.

[7] Mehdi Mirza, Simon Osindero, Conditional generative adversarial nets, 2014, arXiv:1411.1784.

[8] Yonghong Luo, Xiangrui Cai, Ying ZHANG, Jun Xu, Yuan xiaojie, Multivariate time series imputation with generative adversarial networks, in: S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, R. Garnett (Eds.), Advances in Neural Information Processing Systems, Vol. 31, Curran Associates, Inc., 2018.

[9] Yonghong Luo, Ying Zhang, Xiangrui Cai, Xiaojie Yuan, E^2GAN: End-to-end generative adversarial network for multivariate time series imputation, in: Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19, International Joint Conferences on Artificial Intelligence Organization, 2019, pp. 3094–3100.

[10] Y. Zhang, B. Zhou, X. Cai, W. Guo, X. Ding, X. Yuan, Missing value imputation in multivariate time series with end-to-end generative adversarial networks, Inform. Sci. 551 (2021) 67–82.

[11] Alex M Lamb, Anirudh Goyal, Ying Zhang, Saizheng Zhang, Aaron Courville, Yoshua Bengio, Professor forcing: A new algorithm for training recurrent networks, in: Advances in Neural Information Processing Systems, Vol. 29, Curran Associates, Inc., 2016.

[12] William Fedus, Ian Goodfellow, Andrew M. Dai, Maskgan: better text generation via filling in the____, 2018, arXiv:1801.07736 [Cs].

[13] Sepp Hochreiter, Jürgen Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780.

[14] Dzmitry Bahdanau, Kyunghyun Cho, Yoshua Bengio, Neural machine translation by jointly learning to align and translate, 2016, arXiv:1409.0473.

[15] Minh-Thang Luong, Hieu Pham, Christopher D. Manning, Effective approaches to attention-based neural machine translation, 2015, arXiv:1508.04025.

[16] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, Alexei A. Efros, Image-to-image translation with conditional adversarial networks, 2018, arXiv:1611.07004.

[17] Kun Zhou, Wenyong Wang, Teng Hu, Kai Deng, Time series forecasting and classification models based on recurrent with attention mechanism and generative adversarial networks, Sensors (Basel) 20 (24) (2020).

[18] Martin Arjovsky, Soumith Chintala, Léon Bottou, Wasserstein generative adversarial networks, in: Proceedings of the 34th International Conference on Machine Learning, PMLR, 2017, pp. 214–223.

[19] Andy Schumann, Karl Bär, Autonomic aging: A dataset to quantify changes of cardiovascular autonomic function during healthy aging, PhysioNet (2021).

[20] Ary L. Goldberger, Luis A.N. Amaral, Leon Glass, Jeffrey M. Hausdorff, Plamen Ch. Ivanov, Roger G. Mark, Joseph E. Mietus, George B. Moody, Chung-Kang Peng, H. Eugene Stanley, PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals, Circulation 101 (23) (2000).

[21] David Salinas, Michael Bohlke-Schneider, Laurent Callot, Roberto Medico, Jan Gasthaus, High-dimensional multivariate forecasting with low-rank Gaussian copula processes, 2019, arXiv:1910.03002 [Cs].

[22] Alexander Alexandrov, Konstantinos Benidis, Michael Bohlke-Schneider, Valentin Flunkert, Jan Gasthaus, Tim Januschowski, Danielle C. Maddix, Syama Rangapuram, David Salinas, Jasper Schulz, Lorenzo Stella, Ali Caner Türkmen, Yuyang Wang, GluonTS: Probabilistic and neural time series modeling in Python, J. Mach. Learn. Res. 21 (116) (2020) 1–6.