Learning representations for multivariate time series with missing data using Temporal Kernelized Autoencoders

Filippo Maria Bianchi\textsuperscript{a}, Lorenzo Livi\textsuperscript{c}, Karl Øyvind Mikkelsen\textsuperscript{a}, Michael Kampffmeyer\textsuperscript{a}, Robert Jenssen\textsuperscript{a,b}

\textsuperscript{a}Machine Learning Group, UiT–The Arctic University of Norway, [http://site.uit.no/ml/]
\textsuperscript{b}Norwegian Computing Center, Oslo, Norway
\textsuperscript{c}Department of Computer Science, College of Engineering, Mathematics and Physical Sciences, University of Exeter, Exeter EX4 4QF, UK

Abstract

Learning compressed representations of multivariate time series (MTS) facilitate the analysis and process of the data in presence of noise, redundant information, and large amount of variables and time steps. However, classic dimensionality reduction approaches are not designed to process sequential data, especially in the presence of missing values. In this work, we propose a novel autoencoder architecture based on recurrent neural networks to generate compressed representations of MTS, which may contain missing values and have variable lengths. Our autoencoder learns fixed-length vectorial representations, whose pairwise similarities are aligned with a kernel function that operates in input space and handles missing values. This, allows to preserve relationships in the low-dimensional vector space even in presence of missing values. To highlight the main features of the proposed autoencoder, we first investigate its performance in controlled experiments. Successively, we show how the learned representations can be exploited both in several benchmark and real-world classification tasks on medical data. Finally, based on the proposed architecture, we conceive a framework for one-class classification and imputation of missing data in time series extracted from ECG signals.

Keywords: Representation learning; Autoencoders; Recurrent neural networks; Kernel methods; Multivariate time series.

1. Introduction

Real-valued multivariate time series (MTS) allow to characterize the evolution of complex systems and their analysis is the core component in many research fields and application domains \cite{1}. MTS analysis should account for relationships across variables and time steps, and, at the same time, deal with unequal time lengths and missing data \cite{2}. Missing values, commonly found in real-world data such as electronic health records (EHR) \cite{3}, are usually filled with imputation techniques before processing the data. However, unless they are missing completely at random \cite{4}, imputation destroys information of potential interest contained in the missingness patterns. Furthermore, especially for large fractions of missing values, each imputation method can introduce a strong kind of bias that influences the analysis outcome \cite{5}. A data driven approach has been recently proposed to learn when to switch between two particular types of imputation \cite{6}, but it relies on strong assumptions that are suitable only for specific applications.

MTS data often contain noise, redundant information, and can be characterized by a large amount of variables and time steps. In those cases, extracting the relevant information and generating compressed representations using dimensionality reduction techniques facilitate the analysis and processing of the data \cite{2} \cite{7} \cite{8}. Dimensionality reduction has been a fundamental research topic in machine learning \cite{9} \cite{12}. However, classic approaches are not designed to process sequential data, especially in the presence of missing values.

In this paper, we propose a novel neural network architecture called Temporal Kernelized Autoencoder (TKAE) to learn compressed representations of real-valued MTS with unequal lengths and missing data. Our model is based on a deep Autoencoder \cite{13} with recurrent layers, which generates a fixed-size vectorial
representation of the input MTS. To better capture time dependencies, we implement the encoder with a bidirectional recurrent neural network \[14\]. The final states of the forward and backward network are combined by a dense nonlinear layer that reduces the dimensionality of the representation.

To avoid the undesired biases introduced by imputation, we train our architecture with a loss function that preserves pairwise similarities in the learned representations, even in presence of missing data. This is achieved by a kernel alignment procedure \[15\], which matches the dot products of the representations with a kernel similarity defined in the input space. To this end, we consider the recently-proposed Time series Cluster Kernel (TCK) \[16\], which has been shown to capture well the relationships between MTS containing missing data.

The proposed architecture serves different purposes. When represented as ordinary vectors, MTS can be processed by non-sequential classification or unsupervised machine learning algorithms \[17\], and their indexing and retrieval is more efficient \[18, 19\]. Furthermore, the dimensionality of the data is reduced and, accordingly, models can potentially be trained with less samples. Contrarily to other nonlinear dimensionality reduction techniques, the decoder provides an explicit mapping back to the input space. This can be used, for example, to design an anomaly detector based on the reconstruction error of inputs \[20\], or to implement an imputation method that leverages the generalization capability of the decoder reconstruction, rather than relying on a-priori assumptions that may introduce stronger biases \[21\].

TKAE is evaluated through several experiments on synthetic and benchmark datasets, and MTS extracted from real-world EHR. We first investigate the advantages of using recurrent layers to generate compressed MTS representations with respect to other dimensionality reduction methods. Then, we show the benefit of the kernel alignment for learning representations in classification settings with missing data. Finally, we exploit the capability of the TKAE decoder to impute missing data and build a one-class classifier for anomaly detection.

2. Methods

Sec. 2.1 and 2.2 provide the required background, describing respectively the Autoencoder (AE) and TCK, the selected kernel similarity for MTS with missing values. The details of our methodological contribution are presented in Sec. 2.3.

2.1. Autoencoder

The AE is a neural network traditionally conceived as a non-linear dimensionality reduction algorithm \[13\], which has been further exploited to learn representations in deep architectures \[22\] and to pre-train neural network layers \[23\]. An AE simultaneously learns two functions; the first one, called encoder, is a mapping from an input domain, \( \mathbb{R}^{D_x} \), to a hidden representation (code) in \( \mathbb{R}^{D_z} \). The second function, decoder, maps from \( \mathbb{R}^{D_z} \) back to \( \mathbb{R}^{D_x} \). The encoding and decoding functions are defined as

\[
    z = \phi(x; \theta_E); \quad \tilde{x} = \psi(z; \theta_D),
\]

where \( x \in \mathbb{R}^{D_x}, z \in \mathbb{R}^{D_z}, \) and \( \tilde{x} \in \mathbb{R}^{D_x} \) denote a sample from the input space, its hidden representation, and its reconstruction given by the decoder, respectively. The encoder \( \phi(\cdot) \) is usually implemented by stacked dense layers of neurons equipped with sigmoidal activation functions. The decoder \( \psi(\cdot) \) often has an architecture symmetric to the encoder that operates in reverse direction; when inputs are real-valued vectors, decoder squashing nonlinearities are often replaced by linear activations \[24\]. Finally, \( \theta_E \) and \( \theta_D \) are the trainable parameters of the encoder and decoder, respectively. In this work, we focus on AEs implemented with fully connected layers (later, also recurrent layers will be introduced). In those cases, the parameters are the connection weights and biases of each layer, i.e., \( \theta_E = \{W_E, b_E\} \) and \( \theta_D = \{W_D, b_D\} \). AEs are trained to minimize the discrepancy between the input \( x \) and its reconstruction \( \tilde{x} \). In case of real-valued inputs, this is usually achieved by minimizing a loss \( L_r \) implemented as the empirical Mean Squared Error (MSE).
In this paper, we focus on AEs with a “bottleneck”, which learn an under-complete representation of the input, i.e., $D_z < D_x$, retaining as much useful information as possible to allow an accurate reconstruction \[13\]. The learned lossy, compressed representation of the input can be exploited, e.g., in clustering and visualization tasks \[25\], or to train a classifier \[26\]. The bottleneck already provides a strong regularization as it limits the variance of the model. However, further regularization can be introduced by tying encoder and decoder weights ($W_D = W_E^T$) or by adding a $\ell_2$ norm penalty to the loss function

$$L = L_r + \lambda L_2 = \text{MSE}(x, \tilde{x}) + \lambda \|W\|_2^2,$$

where $L_2$ is the $\ell_2$ norm of all model weights $W = \{W_D, W_E\}$ and $\lambda$ is the hyperparameter controlling the contribution of the regularization term.

Recurrent neural networks (RNNs) are models excelling in capturing temporal dependencies in sequences \[27, 28\] and are at the core of seq2seq models \[29\]. The latter learns fixed-size representations of sequences with unequal lengths and, at the same time, generates variable-length outputs. The main applications of RNN AEs based on the idea of seq2seq have been on text \[30\], speech \[31\], and video data \[32\]. However, few efforts have been devoted so far in applying these architectures to real-valued MTS.

Modern seq2seq architectures implement a powerful mechanism called attention, which provides an inductive bias that facilitate modelling long-term dependencies and grants a more accurate decoding if the length of the input sequences varies considerably \[33–35\]. Rather than learning a single vector representation for the whole input sequence, a model with attention maintains all the encoder states generated over time, which are combined by a time-varying decoding vector at each decoding step. Therefore, models with attention provide a representation that is neither compact nor of fixed size and, henceforth, are not suitable for our purposes.

2.2. Time Series Cluster Kernel

The Time series Cluster Kernel (TCK) \[16\] is an algorithmic procedure to compute kernel similarities among MTS containing missing data. The method is based on an ensemble learning approach that guarantees robustness with respect to hyperparameters. This ensures that the TCK also works well in unsupervised settings, the ones in which AE usually operates, where it is not possible to tune the hyperparameters by means of supervised cross-validation. The base models in the ensemble are Gaussian mixture models (GMMs), which are fit to the dataset of MTS using a range of numbers of mixture components. By fitting GMMs with different numbers of mixtures, the TCK procedure generates partitions at different resolutions that capture both local and global structures in the data.

To further enhance diversity in the ensemble, each partition is evaluated on a random subset of MTS samples, MTS attributes (variates) and time segments, using random initializations and randomly chosen hyperparameters. This also contributes to provide robustness with respect to the selection hyperparameters, such as the number of mixture components. To avoid imputation, missing data are analytically marginalized away in the likelihoods. To obtain the GMM posteriors, the likelihoods are multiplied with smooth priors, whose contribution becomes stronger as the percentage of missingness increases. TCK is then built by summing up, for each partition, the inner products between pairs of posterior assignments corresponding to different MTS. The details of TCK are provided in Appendix A.

2.3. Temporal Kernelized Autoencoder

The Temporal Kernelized Autoencoder (TKAE) is a novel AE architecture designed for MTS of unequal length with missing values. A schematic representation of TKAE is provided in Fig. 1. We assume MTS to be represented as a matrix $X \in \mathbb{R}^{V \times T_x}$, where $V$ denotes the number of variables and $T_x$ is the number of time steps that may vary in each MTS. Analogously to seq2seq \[29\], in TKAE the dense layers of standard AEs are replaced by recurrent layers, which process inputs sequentially and update their hidden state at each time step $t$ according to the following map,

$$h_t = \phi(x_t, h_{t-1}, \theta_E),$$

3
aligned, by means of a cost term $L_r$ to reproduce inputs by minimizing the reconstruction error $L$. To compensate for these shortcomings, imputation injects biases in data that may negatively affect the quality of the representations and conceal those are filled beforehand with some imputed value (0, mean value, last observed value) [41]. However, speed, providing a practical motivation for our choice.

The decoder operates according to the following map,

$$\hat{x}_t = \psi(h_t, x_{t-1}, \theta_D),$$

where $\theta_E$ is the set of trainable parameters of the recurrent units. The recurrent layers are composed of either gated recurrent unit (GRU) [36] or long short-term memory (LSTM) [37] cells. The choice of the cell is usually guided by the task at hand [38].

Conventional RNNs make use of previous inputs to build their current hidden representation [14]. However, in applications like MTS classification where the whole input is available at once, it is also possible to exploit the information contained in future inputs to generate the current state of the network. For this reason, the encoder is implemented as a stacked bidirectional RNN [39] consisting of two RNNs working in parallel, each one with $M$ layers of $D_z$ cells and transition function [3]. While one RNN captures input dependencies going backward in time, the other processes the same input but in reverse order, thus modeling relationships that go forward in time. After the whole input is processed, the final states of the forward and backward RNN are denoted as $h_f^T$ and $h_b^T$, respectively. While $h_f^T$ depends on the future ones, hence their combination can capture a wider range of temporal dependencies in the input. In TKAE, the combination is implemented with a dense nonlinear layer $\ell$, which produces an output vector $z \in \mathbb{R}^{D_z}$. The latter, is the fixed-size, vectorial representation of the MTS.

TKAE is trained end-to-end by means of stochastic gradient descent with scheduled sampling [40]. More specifically, during training the decoder input at time $t$ is, with probability $p_s$, the decoder output at time $t-1$ (inference mode) and with probability $1-p_s$ the desired output at time $t-1$ (teacher forcing). Since the desired output is not available during the test phase, the decoder generates test data operating only in generative mode ($p_s = 1$). In most of our experiments, scheduled sampling improved the training convergence speed, providing a practical motivation for our choice.

Analogously to standard AEs, the RNNs in TKAE cannot process data with missing values. Therefore, those are filled beforehand with some imputed value (0, mean value, last observed value) [41]. However, imputation injects biases in data that may negatively affect the quality of the representations and conceal potentially useful information contained in the missingness patterns. To compensate for these shortcomings,
we introduce a kernel alignment procedure \cite{15}, which allows to preserve the pairwise similarities of the inputs, encoded as a positive semi-definite matrix $K$, in the learned representations. $K$ can be any positive semi-definite matrix and is selected according to which properties one wants to embed into the representations. In our case, by choosing the TCK matrix as $K$, the relationships of the learned representations will also account for missing data.

Kernel alignment is implemented by an additional regularization term in the loss function \cite{2}, which becomes

$$L = L_r + \lambda L_2 + \alpha L_k.$$  \hspace{1cm} (5)

$L_k$ is the kernel alignment cost, which computes the normalized Frobenius norm of the difference between two matrices: $K$, the prior kernel matrix, and $ZZ^T$, the dot product matrix between the encodings $z$ of the input MTS. The $L_k$ term is defined as

$$L_k = \frac{\|ZZ^T - K\|_F}{\|K\|_F},$$  \hspace{1cm} (6)

where $Z \in \mathbb{R}^{N \times D_z}$ is the matrix of hidden representations relative to the $N$ MTS in the dataset (or, more specifically, in the current mini-batch). Finally, $\alpha \geq 0$ is a hyperparameter controlling the contribution of the alignment cost in the loss function.

3. Experiments

We perform three sets of experiments. First, in Sec. 3.1 we investigate advantages and shortcomings of an AE with recurrent layers to learn MTS representations, with respect to other dimensionality reduction methods. In this case, we do not use kernel alignment ($\alpha = 0$) and we refer to TKAE simply as TAE. Secondly, in Sec. 3.2 we show how kernel alignment improves the learned representations when MTS contain missing data. Lastly, in Sec. 3.3 we present two case-studies, where TKAE is used for one-class classification and for imputing missing data. Both applications exploit not only the TKAE hidden representation but also its decoder, as the results are computed in the input space.

In the following, we compare the proposed architecture with baseline methods for dimensionality reduction, such as a standard AE and PCA; the learned compressed representations have the same dimensionality in all models taken into account. In each experiment, we train the models for 5000 epochs with mini-batches containing 25 MTS using the Adam optimizer \cite{42} with initial learning rate 0.001. We independently standardize each variate of the MTS in all datasets. In each experiment and for each method, we identify the optimal hyperparameters with $k$-fold cross-validation evaluated on the reconstruction error (or, in general, on the unsupervised loss function) and we report the average results on the test set, obtained in 10 independent runs. We consider only TKAE models with maximum 3 hidden layers of either LSTM or GRU cells, as deeper models generally improve performance slightly at the cost of greater complexity \cite{43}.

3.1. Evaluation of TAE representations

In this section, we compare the compressed representations yielded by TAE, a standard AE, and PCA, to investigate which are the types of data that are better represented when processed by a recurrent architecture. Beside the benchmark datasets, we also consider synthetic data to study the properties of the different architectures in a controlled setting. We let $D_x$ be the input dimensionality; in TAE $D_x = V$, as it processes recursively each single time step. On the other hand, the MTSs must be unrolled when processed by PCA and AE since they expect vectors as inputs rather than sequences. Therefore, in AE and PCA $D_x = V \cdot T$ and then the reconstructed outputs are folded back to match the original shape of the input sequence. We let $D_z$ be the size of the compressed representations, which corresponds to the number of RNN cells in each TAE layer, the size of the innermost layer in AE, and the number of principal components in PCA, respectively. In all experiments we use an AE architecture with 3 hidden layers, $\{D_x, 30, D_z, 30, D_x\}$; the number of neurons in the intermediate layers (30) has been set after preliminary experiments and is not a sensitive hyperparameter (comparable results were obtained using 20 or 40 neurons). As measure of performance, we consider the MSE between the original test elements and their reconstructions produced by each model.
**Time series with different frequencies.** Here, we show the capability of TAE to compress periodic signals having different frequencies and phases. We generate a dataset of sinusoids $y(t) = \sin(a \cdot t + b)$, where $a, b$ are drawn from $\mathcal{N}(0, 1)$ and $t \in [0, 100]$. The proposed task is closely related to the multiple superimposed oscillators, studied in pattern generation and frequency modulation [44]. The training and test set contain 200 and 1000 samples, respectively. We let $D_z = 5$ and the optimal configurations are: AE with nonlinear decoder and $\lambda = 0.001$; TAE with 2 layers of LSTM cells, $\lambda = 0$, and $p_s = 1.0$. The reconstruction MSE on the test set is 0.41 for PCA, 0.212 for AE, and 0.013 for TAE.

Both PCA and AE process the entire time series at once. This may appear an advantage with respect to TAE, which stores information in memory for the whole sequence length before generating the final representation. Nonetheless, TAE yields a better reconstruction since the AE (and PCA) architecture is unsuitable for this task. Indeed, in AE the time step $t_i$ in each MTS is always processed by the same input neuron. For periodic signals, the training procedure tries to couple neurons associated to time steps with the same phase, by associating similar weights to their connections. However, these couplings always change if inputs have different frequencies (see Fig.2). Therefore, the standard AE training never converges as it is impossible to learn a model that generalizes well for each frequency. On the other hand, thanks to its recurrent architecture, TAE can naturally handle inputs of different frequencies since there is no pairing between structural parameters and time steps.

Fig. 3 shows the reconstruction of one sample time series. The lower quality of the reconstruction yielded by AE and PCA can be immediately noticed. Additionally, since those methods are unable to reproduce the dynamics of each sample, they rather adopt a more conservative setting and output signals with lower amplitudes that are closer (in a mean square sense) to the “average” of all the random sinusoids in the dataset.

**Time series of different lengths.** While TKAE can process MTS with different lengths, standard AE and PCA require inputs of fixed size. The common workaround, also followed in this work, is to pad the shorter MTS with zeros [45]. To systematically study the performance of the different methods when MTS have fixed or variable length, we generate data by integrating the following system of first-order Ordinary
Differential Equations (ODE):

\[
d\frac{dy}{dt} = A \tanh(y(t)),
\]

where \(y \in \mathbb{R}^V\), \(A \in \mathbb{R}^{V \times V}\) is a matrix with 50% sparsity and elements uniformly drawn in \([-0.5, 0.5]\). To guarantee system stability, we set the spectral radius of \(A\) to 0.8. \(\tanh(\cdot)\) is applied component-wise and introduces nonlinear dependencies among the variables. A MTS \(x \in \mathbb{R}^{T \times V}\) is obtained by integrating (7) for \(T\) steps, starting from a random initial condition \(y(0)\). Since a generic deterministic dynamical system can be described by a ODE system, these synthetic MTS represent well many case-studies.

We generate two different datasets of MTS with \(V = 10\) variables, each one with 400 and 1000 samples for training and test set, respectively. The first, \(\text{ODEfix}\), contains MTS with same length \(T = 90\), while in the second, \(\text{ODEvar}\), each MTS has a random length \(T \in [30, 90]\). We let \(D_z = 10\) and compare the reconstruction MSE of PCA, AE, and TAE. The optimal configurations for this task are: AE with \(\lambda = 0.001\) and linear decoder; TAE with 1 LSTM layer, \(\lambda = 0.001\), and \(p_s = 0.9\). The average results for 10 independent random generations of the data (\(A\)) and initialization of AE and TAE are reported in Tab. 1.

Table 1: Average reconstruction MSE of MTS with fixed (\(\text{ODEfix}\)) and variable (\(\text{ODEvar}\)) length.

| Dataset | PCA MSE | AE MSE | TAE MSE |
|---------|---------|--------|---------|
| ODEfix  | 0.018   | 0.004  | 0.060   |
| ODEvar  | 0.718   | 0.676  | 0.185   |

In \(\text{ODEfix}\), both AE and PCA yield almost perfect reconstructions, which is expected due to simplicity of the task. However, they perform worse in \(\text{ODEvar}\) despite the presence of many padded values and a consequent lower amount (in average) of information to encode in the compressed representation. On the other hand, TAE naturally deals with variable-length inputs, since once the input sequence terminates its state is no longer updated, as well as its model weights during the training.

**Dealing with large number of variates and time steps.** In order to test the ability to learn compressed representations when the number of variates in the MTS increases, starting from (7) we generate 4 datasets \(\text{ODE5, ODE10, ODE15, and ODE20}\), obtained by setting \(V = \{5, 10, 15, 20\}\). The number of time steps is fixed to \(T = 50\) in each dataset. We let \(D_z = 10\); TAE is configured with 2 layers of LSTM and \(p_s = 0.9\); \(\lambda\) is 0.001 in both AE and TAE. We also include in the comparison an AE with tied weights in the (nonlinear) decoder, which has fewer parameters. Reconstruction errors are reported in Tab. 2. We notice that AE performs well on MTS characterized by low dimensionality, but performance degrades when \(V\) assumes larger values. Since AE processes MTS unrolled into a unidimensional vector, the input size grows quickly as \(V\) increases (one additional variable increases the total input size by \(T\)). Accordingly, the number of parameters in the first dense layer scales-up quickly, possibly leading to model overfit. We also notice that the tied weights regularization, despite halving the number of trainable parameters, degrades performance in each case, possibly because it hinders too much the flexibility of the model. On the other hand, in TAE the model complexity changes slowly, as only one single neuron is added for an additional input dimension. As a consequence, TAE is the best performing model when MTS have a large number of variates.

Table 2: Average reconstruction MSE on the ODE task for different values of \(V\), obtained by TAE, AE, AE with tied weights (\(\text{AE (tw)}\)) and PCA. For AE and TAE we report the number of trainable parameters (\(\#\text{par}\)). Best results are in bold.

| Dataset | TAE MSE | AE MSE | AE (tw) MSE | PCA MSE |
|---------|---------|--------|-------------|---------|
|         | \#par   | \#par  | \#par       |         |
| ODE5    | 0.019   | 0.04   | 0.014       | 0.007   |
|         | 6130    | 31170  | 15870       |         |
| ODE10   | 0.060   | 0.04   | 0.071       | 0.018   |
|         | 6780    | 31370  | 31370       |         |
| ODE15   | 0.072   | 0.10   | 0.153       | 0.174   |
|         | 7430    | 92170  | 46870       |         |
| ODE20   | 0.089   | 0.12   | 0.181       | 0.211   |
|         | 8080    | 122670 | 62370       |         |
To study the performance as the length of MTS increase, we generate 8 datasets with the ODE system \( \text{(7)} \) by varying \( T \in \{50, 75, 100, 125, 150, 175, 200\} \), while keeping fixed \( V = 15 \). In Fig. \( \text{4} \) we report the reconstruction errors and note that TAE performs poorly as \( T \) increases. Especially if there are no temporal patterns in the data that can be exploited by the RNNs to model the inputs, like in this case, TAE is more effective when MTSs are short.

![Figure 4: Reconstruction MSE when increasing length \( T \) of MTS in ODE15. TAE performance decreases for large \( T \).](image)

**Evaluation on benchmark classification datasets.** Here we consider several classification benchmarks from [46], UCR, and UCI repositories\(^1\), whose details are reported in Tab. \( \text{3} \). The datasets have been selected in order to cover a wide variety of cases, in terms of training/test sets size, number of variates \( V \), number of classes and (variable) lengths \( T \) of the MTS. We evaluate the quality of the representations yielded by TAE, standard AE, and PCA by classifying them with a \( k \)NN classifier configured with \( k = 3 \) and Euclidean distance. For each method the results, which include also the reconstruction MSE, are reported in Tab. \( \text{4} \). On the majority of the datasets TAE produces compressed representations that not only best describe the classes of the original inputs (in terms of classification accuracy), but also provide the most accurate reconstruction. In few cases, however, PCA achieves higher performance, which is even superior to the one obtained by the standard AE. Such a result may appear contradictory at first, since the AE is a non-linear extension of PCA and should perform at least equally well; when properly trained. Nevertheless, in MTS datasets having large dimensionality (i.e., many variates) but few samples, it is difficult to obtain a good fit of the parameters in models such as AE and TAE.

| Dataset  | \( V \) | Train   | Test    | Classes | \( T_{\text{min}} \) | \( T_{\text{max}} \) | Source |
|----------|--------|---------|---------|---------|----------------|------------------|--------|
| ECG      | 1      | 500     | 4500    | 5       | 140            | 140              | UCR    |
| ECG2     | 2      | 100     | 100     | 2       | 39             | 152              | UCR    |
| Libras   | 2      | 180     | 180     | 15      | 45             | 45               | [15]   |
| Char.Traj.| 3     | 300     | 2558    | 20      | 109            | 205              | UCI    |
| Wafer    | 6      | 298     | 896     | 2       | 104            | 198              | UCR    |
| Jp. Vow. | 12     | 270     | 370     | 9       | 7              | 29               | UCI    |
| Arab. Dig.| 13    | 6600    | 2200    | 10      | 4              | 93               | UCI    |
| Auslan   | 22     | 1140    | 1425    | 95      | 45             | 136              | UCI    |

In particular, TAE does not obtain the best classification results only on three of the benchmark datasets: ECG, Character Trajectories, and Wafer. However, in these cases the classification accuracy achieved by TAE is only slightly inferior to the competitors and it can be shown that TAE learns compressed representations which are qualitatively comparable to the ones of the other methods. This is illustrated in Fig. \( \text{5} \) which depicts the first two principal components of the learned representations; for TAE and AE, the components

\(^1\)www.cs.ucr.edu/~eamonn/time_series_data, archive.ics.uci.edu/ml/datasets.html
Table 4: Average reconstruction errors (MSE) and classification accuracy on the compressed representations (ACC) on benchmark datasets. We report the optimal hyperparameters found with cross-validation. For AE: $\lambda$ of $l_2$ regularization and tied weights in the decoder (tw). For TAE: type of cell, number of layers, probability of scheduled sampling ($p_s$), and $\lambda$ of $l_2$ regularization.

| Dataset | $D_z$ | PCA MSE ACC | AE MSE ACC $\psi$ tw | TAE MSE ACC cell $p_s$ $\lambda$ |
|---------|------|-------------|----------------------|-------------------------------------|
| ECG     | 10   | 0.026 0.934 | 0.027 0.937 lin. 0 no | 0.064 0.931 GRU×1 1.0 0          |
| ECG2    | 10   | 0.100 0.801 | 0.132 0.809 sig. 0.001 no | 0.131 0.819 LSTM×2 1.0 0.001     |
| Libras  | 5    | 0.199 0.577 | 0.173 0.583 sig. 0.001 no | 0.110 0.661 LSTM×2 0.9 0.001     |
| Char. Traj. | 10 | 0.094 0.922 | 0.116 0.903 lin. 0.001 no | 0.122 0.912 LSTM×3 1.0 0.001     |
| Wafer   | 10   | 0.139 0.928 | 0.163 0.910 lin. 0 no | 0.088 0.919 LSTM×2 1.0 0         |
| Jp. Vow. | 10  | 0.186 0.929 | 0.216 0.927 lin. 0.001 no | 0.177 0.932 LSTM×2 0.8 0.001     |
| Arab. Dig. | 15 | 0.542 0.965 | 0.554 0.926 sig. 0 no | 0.418 0.984 GRU×2 1.0 0          |
| Auslin  | 10   | 0.245 0.538 | 0.338 0.344 lin. 0 yes | 0.213 0.680 LSTM×2 1.0 0         |

Figure 5: First two principal components of ECG, Character Trajectories (CHAR), and Wafer (WAF) datasets. PCA is computed on (i) input space, (ii) AE and (iii) TAE hidden representations. Samples are colored according to their class.

are computed in their hidden-state space. It is possible to observe that the representations yielded by TAE are characterized by a well-defined structure and the class separation (qualitatively speaking) is comparable to the other methods.

3.2. Kernel alignment to handle missing data

So far, we have investigated the effectiveness of a recurrent AE architecture to generate compressed representations of MTSs, with respect to baseline methods for dimensionality reduction. Hereinafter, we demonstrate the effect of kernel alignment when MTS contain missing data. Specifically, we compare the representations learned by TAE ($\alpha = 0$) and TKAE ($\alpha \neq 0$) in a synthetic and real classification problem.

First, we consider the Jp. Vow. dataset (see Tab. 3). This dataset does not originally contain missing data, but similarly to previous studies [16, 46] we inject missing data in a controlled way by randomly removing a certain percentage of elements from the dataset. We vary such percentage from 10% to 90%, evaluating each time the reconstruction MSE and classification accuracy of TAE and TKAE encodings using $k$NN with $k = 3$. We apply zero imputation to replace missing data in the MTS. TAE and TKAE are
configured with 2 LSTM cells, \( p_s = 0.9 \) and \( \lambda = 0.001 \). In TKAЕ, \( \alpha = 0.1 \). In Fig. 6 we show the kernel matrix \( \mathbf{K} \) yielded by TCK and the dot products \( \mathbf{ZZ}^T \) of the representations of the test set when 70% of the data are missing. \( \mathbf{Z} \) is very similar to the TCK matrix, as they are both characterized by a block structure indicating that intra-class similarities in the 9 classes are much higher than inter-class similarities.

Fig. 7 shows how the classification accuracy and reconstruction error of TAE and TKAЕ vary as we increase the amount of missing data. The classification accuracy (blue lines) does not decrease in TKAЕ when the data contain up to 50% missing values and is always higher than in TAE. When 90% of the data are missing, TKAЕ still achieves a classification accuracy of 0.7, while for TAE it drops to 0.1. We also observe that the alignment does not compromise input reconstruction, since the MSE in TAE and TKAЕ (red lines) is almost identical. As a side note, the reconstruction MSE decreases for higher amount of missingness since there are more imputed values (which are constants), hence less information to compress.

In a second experiment, we analyze medical data obtained from EHR in the form of blood measurements of patients undergoing a gastrointestinal surgery at the University Hospital of North Norway in 2004–2012 [47]. Each patient is represented by a MTS of 10 blood sample measurements collected for 20 days after surgery. We consider the problem of classifying patients with and without surgical site infections from their blood samples. The dataset consists of 883 MTS, of which 232 pertain to infected patients. The original MTS contain missing data, corresponding to measurements not collected for a given patient, which are replaced with mean-imputation.

Performance is assessed by classifying the representations generated by TAE and TKAЕ. We also include in the comparison the representations generated by PCA and a standard AE. We let \( D_z = 10 \); TAE and TKAЕ are configured with 2 layers of 10 GRU cells, \( \lambda = 0 \) and \( p_s = 0.9 \); we set \( \alpha = 0.1 \) in TKAЕ. AE is configured with \( \lambda = 0.001 \) and a linear decoder without tied weights. Since the dataset is imbalanced, beside classification accuracy in Tab. 5 we also report the F1 score. We can observe that TKAЕ representations achieves the best accuracy and F1 score. Fig. 8 depicts the first two principal components of the
| Method | Accuracy | F1 score |
|--------|----------|----------|
| PCA    | 0.835    | 0.651    |
| AE     | 0.846±0.013 | 0.675±0.034 |
| TAE    | 0.853±0.021 | 0.682±0.022 |
| TKAE   | 0.899±0.022 | 0.802±0.047 |

Table 5: Classification of the blood data. F1 score is calculated considering infected as “positive” class.

representations learned by TAE and TKAE and their densities, computed with a kernel density estimator. It is possible to recognize the effect of the kernel alignment, as the densities of the components relative to different classes become more separated.

![PCA and density of the two principal components of the representations yielded by TAE and TKAE on blood sample data.](image)

(a) TAE

(b) TKAE

Figure 8: PCA and density of the two principal components of the representations yielded by TAE and TKAE on blood sample data.

3.3. Case studies

Table 6: MSE and Pearson correlation (CORR) of the MTS where missing values are imputed using different methods, with respect to the original MTS (without missing values).

| Dataset | Mean Imp. | LOCF | DAE | TKAE |
|---------|-----------|------|-----|------|
|         | MSE | CORR | MSE | CORR | MSE | CORR | MSE | CORR | MSE | CORR |
| ECG     | 0.883 | 0.792 | 0.393 | 0.884 | 0.157±0.004 | 0.956±0.001 | 0.151±0.003 | 0.956±0.001 |
| Libras  | 0.505 | 0.666 | 0.085 | 0.949 | 0.050±0.001 | 0.970±0.001 | 0.029±0.002 | 0.978±0.002 |
| Wafer   | 0.561 | 0.695 | 0.226 | 0.911 | 0.199±0.017 | 0.935±0.004 | 0.093±0.007 | 0.964±0.003 |
| Jp. Vow.| 0.502 | 0.699 | 0.084 | 0.954 | 0.132±0.001 | 0.926±0.000 | 0.114±0.003 | 0.938±0.001 |

**Imputation of missing data with TKAE.** In presence of missing data, the reconstruction MSE of the loss function can be modified to account only for non-imputed values,

$$L_r = -\sum_t \frac{((x_t - \tilde{x}_t)m_t)^2}{\sum_t m_t}, \quad (8)$$

where $m_t = 0$ if $x_t$ is imputed and 1 otherwise. In this way, the decoder is not constrained to reproduce the values that are imputed and, instead, freely assigns values to the entries that are originally missing. Thus, we can exploit the generalization capability of the decoder to provide an alternative form of imputation,
which depends on nonlinear relationships among the whole training data. A similar principle is followed by
denoising AEs (DAEs) \cite{24}, as they try to reconstruct the original input from a corrupted version where
some entries are randomly removed, hence implementing a form of imputation \cite{48, 49}.

We randomly remove approximately 50% of the values from 4 datasets in Tab. 3 and compare the
capability of TKAE to reconstruct missing values with respect to other imputation techniques. As baseline,
we consider mean imputation, last occurrence carried forward (LOCF), and DAE imputation \cite{21}. In Tab. 6
we report MSE and Pearson correlation (CORR) of the MTS with imputed missing values, with respect
to the original MTS. For TKAE and DAE, we use the same configurations from Tab. 4 but in TKAE we
replace the $L_r$ term with (8) and we set $\alpha = 0.1$. In DAE, we apply a stochastic corruption of the inputs,
by setting input values with probability 0.5 to 0. From the results, we observe that TKAE achieve a more
accurate reconstruction of the true input in 3 of the 4 datasets. In Jp. Vow. instead, LOCF imputation
allows to retrieve missing values with the highest accuracy. This can be explained by the fact that the MTS
in the dataset contain very similar values that are repeated for several time intervals. However, it is possible
to notice that also in this case TKAE performs better than DAE.

**One-class classification with TKAE.** One-class classification and anomaly detection are applied in
several domains, including healthcare, where non-nominal samples are scarce and often unavailable during training \cite{50}. The methods based on dimensionality reduction procedures, such as AEs and energy based
models \cite{20, 51} rely on the assumption that anomalous samples do not belong to the subspace containing
nominal data, which is learned during training. Therefore, the representations generated by the trained model
for samples of a new, unseen class will arguably fail to capture important characteristics. Consequently, for
those samples an AE would yield large reconstruction errors, which we consider as the classification scores
for the new class.

| Method   | AUC  |
|----------|------|
| OCSVM    | 0.713|
| IF       | 0.662±0.01 |
| PCA      | 0.707|
| AE       | 0.712±0.001 |
| TKAE     | **0.732±0.006** |

Table 7: AUC obtained by different one-class classification methods in detecting the MTS of
atrial fibrillation class, which is not present in the training set.

We process time series of peak-to-peak intervals extracted from ECGs in the 2017 Atrial Fibrillation
challenge \cite{52}, which are divided in 4 classes: normal (N), atrial fibrillation (A), other symptoms (O) and
noisy records (\sim). By following a commonly used procedure \cite{53}, we simulate missing data by randomly
removing approximately 50% of the entries in each MTS and then we exclude samples of class A from the
training set (which are then considered as non-nominal). We evaluated the performance of TKAE, AE, and
PCA in detecting class A in a test set containing samples of all classes (N,A,O,\sim). As performance measure,
we considered the area under ROC curve (AUC) and compared the performance also with two baseline
classifiers: one-class SVM (OCSVM) and Isolation Forests (IF). The optimal configurations are: $D_z = 10$;
TKAE with 1 layer of GRU cells, $p_s = 0.9$, $\lambda = 0$, and $\alpha = 0.2$; AE with non-linear decoder, no tied weights,
and $\lambda = 0$; OCSVM with rbf kernel width $\gamma = 0.7$ and $\nu = 0.5$; IF with contamination 0.5. Results in Tab. 7
show that TKAE scores the highest AUC.

4. Conclusion

We proposed the Temporal Kernelized Autoencoder, an RNN-based model for representing MTS with
missing values as fixed-size vectors. MTS with missing values are commonly found in domains such as
healthcare and are caused by measurement errors, incorrect data entry or lack of observations. Through a
kernel alignment with the Time series Cluster Kernel, a similarity measure designed for MTS with missing
data, we learned compressed representations that better preserve the original input pairwise relationships in
presence of missing values.
We demonstrated how the recurrent architecture of TKAE excels in encoding short MTS with many variables, characterized by different lengths and periodicity. We showed that the representations learned by TKAE can be used both in supervised and unsupervised tasks. Experimental results, contrasted with other dimensionality reduction techniques on several datasets of MTS, showed that the TKAE representations are classified more accurately, especially in presence of missing data.

To further evaluate the capabilities of the TKAE architecture, we considered two applications that exploit the decoder, which is learned as part of the optimization. Specifically, we built a framework based on dimensionality reduction and inverse mapping to the input space for imputing missing data and for one-class classification. We showed that TKAE is able to outperform competing methods on those tasks.

Acknowledgments

This work was partially funded by the Norwegian Research Council FRIPRO grant no. 239844 on developing the Next Generation Learning Machines. The authors would like to thank Arthur Revhaug, Rolv-Ole Lindsetmo and Knut Magne Augestad, all clinicians currently or formerly affiliated with the gastrointestinal surgery department at the University Hospital of North Norway, for preparing the blood samples dataset.

Appendix A. Details of the TCK algorithm

A MTS $\mathbf{X} \in \mathbb{R}^{V \times T}$ is represented by a sequence of $V$ univariate time series (UTS) of length $T$, $\mathbf{X} = \{x_v \in \mathbb{R}^T | v = 1, \ldots, V\}$, being $V$ and $T$ the dimension and length of $\mathbf{X}$, respectively. Given a dataset of $N$ samples, $\mathbf{X}^{(n)}$ denotes the $n$-th MTS and a binary MTS $\mathbf{R}^{(n)} \in \mathbb{R}^{V \times T}$ describes whether the realisation $x_v^{(n)}(t)$ in $\mathbf{X}$ is observed ($r_v^{(n)}(t) = 1$) or is missing ($r_v^{(n)}(t) = 0$).

**DiagGMM.** The TCK kernel matrix is built by first fitting $G$ diagonal covariance GMM (DiagGMM) to the MTS dataset. Each DiagGMM $g$ is parametrized by a time-dependent mean $\mu_{gv} \in \mathbb{R}^T$ and a time-constant covariance matrix $\Sigma_g = \text{diag}\{\sigma_{g1}^2, \ldots, \sigma_{gV}^2\}$, being $\sigma_{gv}^2$ the variance of UTS $v$. Moreover, the data is assumed to be missing at random, i.e. the missing elements are only dependent on the observed values. Under these assumptions, missing data can be analytically integrated away \cite{53} and the pdf for each incompletely observed MTS $\{\mathbf{X}, \mathbf{R}\}$ is given by

$$p(\mathbf{X} | \mathbf{R}, \Theta) = \sum_{g=1}^{G} \theta_g \prod_{v=1}^{V} \prod_{t=1}^{T} \mathcal{N}(x_v(t) | \mu_{gv}(t), \sigma_{gv})^{r_v(t)} \quad (A.1)$$

The conditional probabilities follows from Bayes’ theorem,

$$\pi_g = \frac{\theta_g \prod_{v=1}^{V} \prod_{t=1}^{T} \mathcal{N}(x_v(t) | \mu_{gv}(t), \sigma_{gv})^{r_v(t)}}{\sum_{g=1}^{G} \theta_g \prod_{v=1}^{V} \prod_{t=1}^{T} \mathcal{N}(x_v(t) | \mu_{gv}(t), \sigma_{gv})^{r_v(t)}}. \quad (A.2)$$

The parameters of the DiagGMM are trained by means of a maximum a posteriori expectation maximization algorithm, as described in \cite{16}.

**Ensemble generation.** To ensure diversity in the ensemble, each GMM model has a different number of components from the interval $[2, C]$ and is trained $Q$ times, using random initial conditions and hyperparameters. Specifically, $Q = \{q = (q_1, q_2) | q_1 = 1, \ldots, Q, q_2 = 2, \ldots, C\}$ denotes the index set of the initial conditions and hyperparameters ($q_1$), and the number of components ($q_2$). Moreover, each DiagGMM is trained on a subset of the original dataset, defined by a random set of the MTS samples, a random set $\mathcal{V}$ of $|\mathcal{V}| \leq V$ variables, and a randomly chosen time segment $\mathcal{T}, |\mathcal{T}| \leq T$. The inner products of the posterior distributions from each mixture component are then added up to build the final TCK kernel matrix. Details are provided in Alg. 1.
Algorithm 1 TCK kernel training

**Input:** Training set of MTS \( \{X^{(n)}\}_{n=1}^{N} \), Q initializations, C maximal number of mixture components.

1. Initialize kernel matrix \( K = 0_{N \times N} \).
2. for \( q \in Q \) do
3. \hspace{1em} Compute posteriors \( \Pi^{(n)}(q) \equiv (\pi_{1}^{(n)}, \ldots, \pi_{q}^{(n)})^{T} \), by applying maximum a posteriori expectation maximization \([16]\) to the DiagGMM with \( q_2 \) components and by randomly selecting,
   i. hyperparameters \( \Omega(q) \),
   ii. a time segment \( T(q) \) of length \( T_{min} \leq |T(q)| \leq T_{max} \),
   iii. attributes \( V(q) \), with cardinality \( V_{min} \leq |V(q)| \leq V_{max} \),
   iv. a subset of MTS, \( \eta(q) \), with \( N_{min} \leq |\eta(q)| \leq N \),
   v. initialization of the mixture parameters \( \Theta(q) \).
4. Update kernel matrix, \( K_{nm} = K_{nm} + \frac{\Pi^{(n)}(q)^T \Pi^{(m)}(q)}{\lVert \Pi^{(n)}(q) \rVert \lVert \Pi^{(m)}(q) \rVert} \).
5. end for

**Output:** TCK matrix \( K \).
References

[1] C. Chatfield, The Analysis of Time Series: An Introduction, CRC press, 2016.
[2] M. Härdle, L. Härdle, A. Lütkepohl, Analytics ofFinancial Time Series, Springer, 2011.
[3] J. D. Nicoud (Eds.), Artificial Neural Networks — ICANN'97, Springer Berlin Heidelberg, Berlin, Heidelberg, 1997, pp. 583–588.
[4] K. Fukumizu, F. R. Bach, M. Jordan, Dimensionality reduction for supervised learning with reproducing kernel hilbert spaces, Journal of Machine Learning Research.
[5] R. Jenssen, Kernel entropy component analysis, IEEE Transactions on Pattern Analysis and Machine Intelligence 32 (5) (2010) 847–860. doi:10.1109/TPAMI.2009.100
[6] M. Harandi, M. Sitzmann, R. Hartley, Dimensionality reduction on SPD manifolds: The emergence of geometry-aware methods, IEEE Transactions on Pattern Analysis and Machine Intelligence 40 (1) (2018) 48–62. doi:10.1109/TPAMI.2017.2650548
[7] G. E. Hinton, R. R. Salakhutdinov, Reducing the dimensionality of data with neural networks, Science 313 (5786) (2006) 504–507. doi:10.1126/science.1127647
[8] A. Graves, A. R. Mohamed, G. Hinton, Speech recognition with deep recurrent neural networks, in: 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, 2013, pp. 6645–6649. doi:10.1109/ICASSP.2013.6638947
[9] M. Kampffmeyer, S. Løkse, F. M. Bianchi, R. Jenssen, L. Livi, Deep kernelized autoencoders, in: P. Sharma, F. M. Bianchi (Eds.), Artificial Neural Networks — ICANN’97, Springer Berlin Heidelberg, Berlin, Heidelberg, 1997, pp. 583–588.
[10] K. Ø. Mikalsen, F. M. Bianchi, V. Durme, R. Jenssen, Time series cluster kernel for learning similarities between multivariate time series with missing data, Pattern Recognition 76 (2018) 569–581. doi:10.1016/j.patcog.2017.11.030
[11] Z. Xing, J. Pei, E. Keogh, A brief survey on sequence classification, SIGKDD Explor. News. doi:https://doi.org/10.1145/2655048.2655049
[12] Y. Chung, C. Wu, C. Shen, H. Lee, L. Lee, Audio word2vec: Unsupervised learning of audio segment representations using sequence-to-sequence autoencoder, CoRR abs/1603.00982. arXiv:1603.00982
[13] S. Zhai, Y. Cheng, W. Lu, Z. Zhang, Deep structured energy based models for anomaly detection, in: Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48, ICML’16, JMLR.org, 2016, pp. 1100–1109.
[14] B. K. Beaulieu-Jones, J. H. Moore, Missing data imputation in the electronic health record using deeply learned autoencoders, World Scientific, 2016, pp. 1100–1109.
[15] Y. Bengio, Learning deep architectures for AI, Foundation and Trends Machine Learning 2 (1) (2009) 1–127. doi:10.1561/2200000006
[16] D. Erhan, Y. Bengio, A. Courville, P.-A. Manzagol, P. Vincent, S. Bengio, Why does unsupervised pre-training help deep learning?, Journal of Machine Learning Research 11 (2010) 625–660.
[17] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, P.-A. Manzagol, Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criteria, Journal of Machine Learning Research 11 (Dec) (2010) 3371–3408.
[18] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, B. Frey, Adversarial autoencoders, arXiv preprint arXiv:1511.05644.
[19] W. W. Y. Ng, G. Zeng, J. Zhang, D. S. Yeung, W. Pedrycz, Dual autoencoders features for imbalance classification problem, Pattern Recognition.
[20] J. L. Elman, Finding structure in time, Cognitive science 14 (2) (1990) 179–211.
[21] F. M. Bianchi, E. Maiorino, M. C. Kampffmeyer, A. Rizzi, R. Jenssen, Recurrent Neural Networks for Short-Term Load Forecasting: An Overview and Comparative Analysis, Springer, 2017.
[22] I. Sutskever, O. Vinyals, Q. V. Le, Sequence to sequence learning with neural networks, in: Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, K. Q. Weinberger (Eds.), Advances in Neural Information Processing Systems 27, Curran Associates, Inc., 2014, pp. 3104–3112.
[30] S. R. Bowman, L. Vilnis, O. Vinyals, A. M. Dai, R. Józefowicz, S. Bengio, Generating sentences from a continuous space, CoRR abs/1511.06349. arXiv:1511.06349

[31] J. Chung, K. Kastner, L. Dinh, K. Goel, A. C. Courville, Y. Bengio, A recurrent latent variable model for sequential data, in: C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, R. Garnett (Eds.), Advances in Neural Information Processing Systems 28, Curran Associates, Inc., 2015, pp. 2980–2988.

[32] N. Srivastava, E. Mansimov, R. Salakhutdinov, Unsupervised learning of video representations using lstms, in: Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37, ICML'15, JMLR.org, 2015, pp. 843–852.

[33] D. Bahdanau, K. Cho, Y. Bengio, Neural machine translation by jointly learning to align and translate, CoRR abs/1409.0473. arXiv:1409.0473.

[34] Y. Kim, C. Denton, L. Hoang, A. M. Rush, Structured attention networks, in: International Conference on Learning Representations, 2017.

[35] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, I. Polosukhin, Attention is all you need, in: I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett (Eds.), Advances in Neural Information Processing Systems 30, Curran Associates, Inc., 2017, pp. 5998–6008.

[36] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio, Learning phrase representations using RNN encoder-decoder for statistical machine translation, arXiv preprint arXiv:1406.1078.

[37] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Computation 9 (8) (1997) 1735–1780.

[38] J. Chung, C. Gulcehre, K. Cho, Y. Bengio, Empirical evaluation of gated recurrent neural networks on sequence modeling, arXiv preprint arXiv:1412.3555.

[39] M. Schuster, K. K. Paliwal, Bidirectional recurrent neural networks, IEEE Transactions on Signal Processing 45 (11) (1997) 2673–2681.

[40] S. Bengio, O. Vinyals, N. Jaitly, N. Shazeer, Scheduled sampling for sequence prediction with recurrent neural networks, in: Proceedings of the 28th International Conference on Neural Information Processing Systems, MIT Press, Cambridge, MA, USA, 2015, pp. 1171–1179.

[41] J. L. Schafer, J. W. Graham, Missing data: Our view of the state of the art, Psychological Methods.

[42] D. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980.

[43] M. G. Baydogan, G. Runger, Time series representation and similarity based on local autopatterns, Data Mining and Knowledge Discovery 30 (2) (2016) 476–509. doi:10.1007/s10618-015-0425-y

[44] K. Øyvind Mikalsen, C. Soguero-Ruiz, F. M. Bianchi, A. Revhaug, R. Jenssen, An Unsupervised Multivariate Time Series Kernel Approach for Identifying Patients with Surgical Site Infection from Blood Samples, ArXiv e-prints arXiv:1803.07879.

[45] B. K. Beaulieu-Jones, C. S. Greene, Semi-supervised learning of the electronic health record for phenotype stratification, Journal of Biomedical Informatics.

[46] L. Gondara, K. Wang, Multiple imputation using deep denoising autoencoders, arXiv preprint arXiv:1705.02737.

[47] I. Irigoien, B. Sierra, C. Arenas, Towards application of one-class classification methods to medical data, The Scientific World Journal.

[48] M. Sakurada, T. Yaari, Anomaly detection using autoencoders with nonlinear dimensionality reduction, in: Proceedings of the MLSDA 2014 2Nd Workshop on Machine Learning for Sensory Data Analysis, MLSDA’14, ACM, New York, NY, USA, 2014, pp. 4:4–4:11. doi:10.1145/2689746.2689747

[49] G. Clifford, C. Liu, B. Moody, L. Lehman, I. Silva, Q. Li, A. E. Johnson, R. G. Mark, AF classification from a short single lead ecg recording: The physionet computing in cardiology challenge 2017, Computing in Cardiology.

[50] J. L. Peugh, C. K. Enders, Missing data in educational research: A review of reporting practices and suggestions for improvement, Review of Educational Research 74 (4) (2004) 525–556. doi:10.3102/00346543074004525

[51] D. B. Rubin, Inference and missing data, Biometrika 63 (3) (1976) 581–592.