# Automatic data imputation in time series processing using neural networks for industry and medical datasets

Juan Ignacio Porta[1,2], Martín Ariel Domínguez[1,3], and Francisco Tamarit[1,2]

[1] FaMAF, Universidad Nacional de Córdoba, Córdoba, Argentina
[2] Instituto de Física Enrique Gaviola, IFEG (UNC-CONICET), Córdoba, Argentina
[3]Sección de Ciencias de la Computación
juan.porta@mi.unc.edu.ar,
{martin.dominguez, francisco.tamarit}@unc.edu.ar

**Abstract.** Time series classification and regression techniques help solve problems in many knowledge areas, including medicine, electronics, industry, and even music. When we apply them to real-life issues, a common obstacle is the lack of data in intervals within a time series. Usually, to solve it, the missing data is populated with information highly dependent on available datasets, which requires prior analysis. This paper addresses the problem in a novel way, automatically filling the missing data using a mixture of techniques and letting the prediction model decide which filling is better. We tested our approach for classification in industrial and medical datasets and for regression, we used a dataset cointaining COVID-19 information.
Our results are very competitive, and our approach improves the state-of-the-art models. We obtain better performance in all the experiments for the selected quality measures. Most importantly, the improvement is more statistically significant when the amount of missing data is higher.

**Keywords:** Time Series Classification, regression, Deep Learning, Convolutional networks, Long Short-Term memory

## 1 Introduction

Classifying a time series into specific categories predefined by experts has been a task of great interest in different areas of knowledge, such as medicine, music, and electronics, among others, [36, 23, 28]. This type of technique is highly versatile and applicable to endless tasks that range from abnormalities detection in electrocardiograms [28] to detecting failures in electronic components during production[23]. There are different approaches to time series classification, which vary from classical statistical methods to rule-based systems. This research line experienced a re-growth with the increase of computing capability and the creation of specific hardware that enabled significant improvement in the training of deep learning models. In particular, models based on recurrent

Long Short-Term Memory (LSTM)-type networks [16], and Convolutional Neural Networks (CNN) [8] have shown the highest performances for this task. In 2016, Cui, Chen, and Chen [8] addressed this problem with a novel method that incorporates a peculiar feature engineering. They used CNN architectures and fed it simultaneously with different preprocessed versions of a given time series.

In this work, we tackle a common obstacle that time-series datasets usually present in natural environments: time intervals with missing data. This problem is particularly relevant when dealing with sensor measurements, which can fail for various reasons, including the lack of power, signal, and memory, among others. In addition, it is common for missing data to appear in the health area, as in the case of discontinued records of data on patients, either due to irregularities in their measurements or absences. Many classical statistical methods overcome this problem, such as interpolation or forward filling, but these approaches strongly depend on the specific dataset analyzed.

Inspired by the work of Cui, Chen, and Chen [8], we present two models that take time series as input: one for classification and one for regression. The models presented can develop the tasks based on an input with missing data, generate different filling treatments in parallel, and take advantage of the information from each of these inputs to maximize performance in the specific task.

For the classification task, we test our model using two industrial datasets (Wafer and Ford). We also include a medical dataset, Mimic-III, which, unlike the previous ones, incorporates multivariable time series. Both methods use deep learning to choose and extract relevant information. Starting with a complete-time series, we emulate data loss by randomly sampling time intervals and removing them from the time series in different sizes. Our model outperforms other methods, and the performance is even more expressive as the number of missing data increases.

Additionally, we address the data imputation problem for regression. A common task in time series is regression; that is, given a series, one must try to predict a value in a region of $\mathbb{R}^n$. Regression is a widespread task in areas such as public health, econometrics, and meteorology. The best tools to learn regression models for time series are recurrent neural networks, such as LSTM [11] or GRU [7]. To address the lack of data intervals in regression time series, we develop a regression model based on an LSTM, similar to the one we use for the classification task. To test our approach, we use a dataset of COVID-19 cases in Argentina. We took the ideas from [1] where the authors use forecasting methods and interpolation to predict the total number of people infected and the number of active cases for COVID-19 propagation in a global dataset. We construct an iterative method based on cubic spline interpolation and Euler's method, which is an improvement over the two latter methods. Our model outperforms state-of-the-art methods, although the task is more complex than in classification, and therefore, the model is more sensitive to missing data.

This paper is structured as follows. The next section analyzes the related work for time series classification and data imputation. In Section 3, we describe the datasets used for both classification and regression. In Section 4, we address

the problem of TSC with missing data and the problem of TSR with missing data. In particular, we define a baseline model adapted from [8] and a new Automatic Data imputation model, and we analyze the experimental setup of our models and the baseline models. In Section 5, we describe the experimental results and compare the performance of both models. Finally, in 6, we present our conclusion and possible extensions of this work.

## 2 Related Work

In this Section we describe the most relevant previous work related to our models.

Formally, a time series is a set of real numbers measured in chronological order. We denote a time series as $T = \{V_{t_1}, ..., V_{t_n}\}$, where $V_i = (v_1, \ldots, v_k)$ is a vector of dimension $k$ of $v_i$ values at time $t_i$ with data recorded at $n$ different times for each series.

A labeled time series dataset is denoted as $D = \{(T_j, y_j)\}_{j=1}^{N}$, which contains $N$ time series with their corresponding tags. In this paper, we consider classification and regression tasks, then, for classification, $y_i$ is a categorical value at $C = \{1, ..., c\}$ where $c \in \mathbb{Z}^+$ is the number of different labels or categories. In contrast, for regression task, $y_i$ is a number belonging to an interval of $\mathbb{R}$.

As Cui, Chen, and Chen mentioned, many specific difficulties arise when dealing with time series that make it very difficult for machine learning models to deal with the classification tasks. Among the most relevant ones, we can mention, for instance, the need for features at different time scales, strongly depending on the dataset used and the handling of random noise.

### 2.1 Cui Chen and Chen MCNN

To address these issues, the authors introduced a Multi-scale Convolutional Neural Network (MCNN). They have addressed the TSC problem using CNN and obtained excellent results compared with previous ones. They compared their results with Dynamic Time Warping (DTW) with a warping window constraint set through cross-validation(DTW CV) [25], Fast Shapelet (FS) [26], Symbolic Aggregate approXimation (SAX) with Vector Space (SV) model [34], Bag-of-SFA-Symbols (BOSS) [32], Shotgun Classifier (SC) [31], Time Series based on a Bag-of-Features (TSBF) [5], Proportional Elastic Ensemble (PROP) [3], 1-NNBag-Of-SFA-Symbols in Vector Space (BOSSVS) [33], Learn Shapelets Model (LTS) [10] and the Shapelet Ensemble (SE) model [4]. The proposed model by Cui, Chen, and Chen, proved to be very powerful for carrying out this difficult task. The architecture proposed can be divided into three different stages:

1. *Transformation Stage*: from the input time series, they build three different transformations, namely, (a) a time compression, (b) spectral transformations in frequency and, (c) a direct mapping, and these three series feed the local stage. Each one of these transformations is called *a branch*.

2. *Local convolution stage*: for each of the three branches, we apply sequentially several convolutional layers to implement a feature extraction. At this stage, the convolutions are independent of each branch.
3. *Global convolution stage*: finally, they concatenate the output of the convolutional layers into a single layer, on which new convolutional nuclei are applied, ending in a completely connected architecture with two linear output neurons with softmax activation functions, which is a function that normalizes the outputs of the linear layer.

It is worth mentioning that the results obtained in [8] outperform all previous models at that time.

### 2.2 State-of-the-art for Data Filling

The problem of filling data in a time series has been widely studied in the literature, the most relevant ones are: soft impute [22] xgboost-based model [39] ST-MVL [37], adversarial training [19]. Next, we describe more in the deep Soft Impute and Xgboost-based Model used in our approach. .

- **Soft Impute** [22]: In this work, the authors replace the missing elements with those obtained from a soft threshold Singular Value Decomposition (SVD). With the help of warm starts, they efficiently calculate a complete solution regularization path on a grid of regularization parameter values. This model shows great performance as a filling data technique, showing better performance than Hard-impute [22], SVT [6] and OptSpace [17], and a similar performance to MMMF [35].
- **Xgboost-based Model** [39]: this approach combines an unsupervised prefilling strategy with a supervised machine learning approach, using extreme gradient boosting (XGBoost). This combination leverage both types of context for imputation purposes: Contextual information in individuals and laboratory test variables. The resulting model outperforms 3D-MICE [21], and even a combination of local mean and soft impute in most datasets presented.

## 3 Dataset

This section describes the datasets used in this work. The first dataset we use is the Wafer dataset generated by R. Olszewski. This time series dataset consists of measurements obtained during the semiconductor circuit production process and binary classification of it: normal or abnormal. The dataset essentially consists of 152 temporary features. This corpus is split in two with distribution not suitable to test the models; consequently, we generate a new distribution. We join the previous sets and create a new partition of 80% of the data for training (5731 entries), and the resulting 20 % is assigned as the test set (1433). The dataset is highly unbalanced, with 10.7% of the training set and 12.1% of the test set belonging to the abnormal class.

The following explored dataset is the **FordA** and **FordB** dataset, which is a dataset initially used in a competition at the IEEE World Congress on Computational Intelligence, 2008. Its objective is to classify the existence or absence of a particular symptom in an . Each time series consists of a record of engine noise and a binary objective of 1 or -1.

The last classification dataset to use is the Medical Information Mart for Intensive Care III database version 1.4 (MIMIC III v1.4) which includes information on 46,520 patients who were admitted to various ICUs of BIDMC in Boston, Massachusetts from 2001 to 2012 [15] [24] [9]. Adult patients who were diagnosed with sepsis-3 were included in the study. The criterion used was the same as in Hou, Nianzong, et al. (2020) [12]. Patients were divided into two groups based on whether they were dead or alive within 30 days, and a binary variable was generated based on this. A total of 4559 sepsis-3 patients are included in our study, in which 889 patients died and 3670 survived after 30 days.

Finally, to test our approach for regressor models, we use a COVID-19 dataset from Argentina [2]. These time series include the number of new daily cases and the average age of people infected. The time frame considered was between March 1, 2020, and August 1, 2021. Taking this freely available data, we generate 400 series of 30 days in length. We split it into two subsets for training a test dataset with 80 and 20 percent, respectively. The target is to predict the number of cases 15 days in the future.

## 4 Experimental Setup

This section describes the pipeline, and the network architecture that we propose to address the imputation of missing data in time series processing. We also clarify the differences in our approach with that introduced in [8], which inspired our work.

Based on the first two datasets previously described, each series randomly removes missing data taking into account different percentages.
Let $D = \{(V_{t_1}, y_{t_1}), \ldots, (V_{t_N}, y_{t_N})\}$ be a labeled time series. We build different datasets, with different percentages of missing data from the original dataset $D$. We call $D_{-f}$ the dataset resulting from deleting a percentage $f$. We construct $D_{-f}$ by generating sets of disjoint time intervals $I_f = [t^l_{i_1}, t^r_{i_1}] \ldots [t^l_{i_k}, t^r_{i_k}]$ where $t_i < t^l_{i_1} < {ti_1}^r < \ldots < {ti_k}^l < {ti_k}^r < t_N$ and the resulting series $D_{-f}$ has an $f$ percentage of less vectors than $D$. This deletion is carried out by replicating real-life sensor time series: it has one or more missing subsets of data that are continuous over time, and it has an average controlled amount of fewer data.

For Wafer and Ford datasets, we tested our model for $D_{-f}$ with percentage $f \in \{10, 30, 50, 60, 70, 80, 90\}$. The percentage $f$ of data deletion is randomly selected, then, to validate the statistical significance of our results, for each percentage $f$, we build 10 random deleted datasets. Because of how data erasure is designed, the type of missing data that our models address is missing at random (MAR) because the data deletion is random, but missing data cannot appear in the first or last values of the series. [27]

The MIMIC dataset is more complex than previous datasets, and it has real missing data; then, the data deletion was carried out only until having 30% of missing data. Although in the case of medical datasets, it makes more sense to find missing data in all variables simultaneously, as in certain variables, we also trained on the mimic-III dataset with independent deletion per column. This idea tries to test the model on a dataset multivariable that may present this characteristic more frequently in sensor systems. To do so, we apply the deletion of the univariable datasets to each column of the mimic-III dataset.

In the COVID-19 dataset, the data deletion was carried out only until having 30% of missing data.

### 4.1 MCNN Baseline Model

Before we introduce our model, let us briefly describe the modification implemented on the model presented in [8], which we will call our baseline model. This modification seeks to handle smaller datasets and thus avoid overfitting. First, we eliminate the convolutional layers in the global stage, leaving only a single one, a fully connected global layer. The base model is structured as follows:

- **Coord Conv Channel**: A general problem that convolutional networks present when analyzing time series, compared to recurrent networks (RNN), is that the former lacks a structure that allows full exploitation of the time relationship between the values of the time series. A suitable property of the convolutional network is that it uses the "closeness" in time of two values when they are at a distance less than the size of the convolutional nucleus. However, when this does not happen, the layers have no information on where in the series it is located. For regular series, this can be of great importance. Therefore, we propose to use a variant of the idea stated in [20]. In this work, the authors added a channel to the network's entrance, which provides spatial information in convolutional neural networks of images, while our proposal provides temporal information and allows the model to locate itself in time. We call this the time *CoordConv* channel.
- **Transformation stage**: In the transformation stage, downsampling is performed every five temporary events to generate a temporarily compressed version of the original series. We also perform a moving average with a window of 5 consecutive events to generate a smoothing transformation in frequency.
- **Local convolution stage**: each branch of Local convolution stage in figure 1 consists of 3 convolutional layers in parallel of 32 kernels, with a kernel height of 1, 3, and 5 respectively, each with batch normalization [14] and followed by a linear layer of 10 neurons each with a dropout of 0.3 (figure 2). The number of branches is 12, 4 for each of the transformed series because the baseline model has the same number of parameters as the one that we will present later.
- **Global stage**: We concatenate the outputs by using a linear layer of 2 neurons, followed by a softmax.

Unlike the original Cui *et al.* model, the applied model works on the complete time series.

### 4.2 LSTM Baseline Model

The model used as a baseline for regression-based in LSTM follows the same ideas of the CNN classification model. It is set up in the following way:

- **Coord Conv Channel**: As in the MCNN model, we add a Time *CoordConv* channel, which allows the model to locate itself in time.
- **Transformation stage**: In the transformation stage, downsampling is performed every five temporary events to generate a temporarily compressed version of the original series. We also perform a moving average with a window of 5 consecutive events to generate a smoothing transformation in frequency.
- **local LSTM stage**: each branch of local recurrent stage in figure 1 consists of 1 LSTM layer, with 100 hidden states, followed by a linear layer of 10 neurons each with a dropout of 0.3 (figure 3). The number of branches is 12, 4 for each of the transformed series because the baseline model has the same number of parameters as the one that we will present later.
- **global stage**: we concatenate the outputs by using a linear layer of 1 neuron.

Keep in mind that both baselines are state of the art in data filling.

### 4.3 Our Approach for Automatic Imputation Model

In this subsection, we describe our model, which, as we will see in short, improves results obtained with the baseline model. As we mentioned previously, our work is based on Cui Chen and Chen's idea; we added new branches, which use different methods of data imputation. We combine the standard techniques of imputation in statistical analysis for time series, such as linear interpolation and forward filling, with the state-of-the-art techniques, such as soft impute, which repeatedly fills in missing values with the current prediction, and then solves an optimization problem on the complete matrix using a soft-thresholded SVD[22], and an xgboost-based model[39]. We combine each of these methods with frequency and time scale transformations. The idea is that the model can combine the robustness provided by multi-scale and multi-frequency to extract information from different methods of data imputation.

In the rest of this section, we describe our extension for CNN and LSTM models for automatic data imputation.

#### CNN imputation data classification model

In Figure 1, we schematize our model architecture. Next, we enumerate how we extend the Cui *et al.* model [8]:

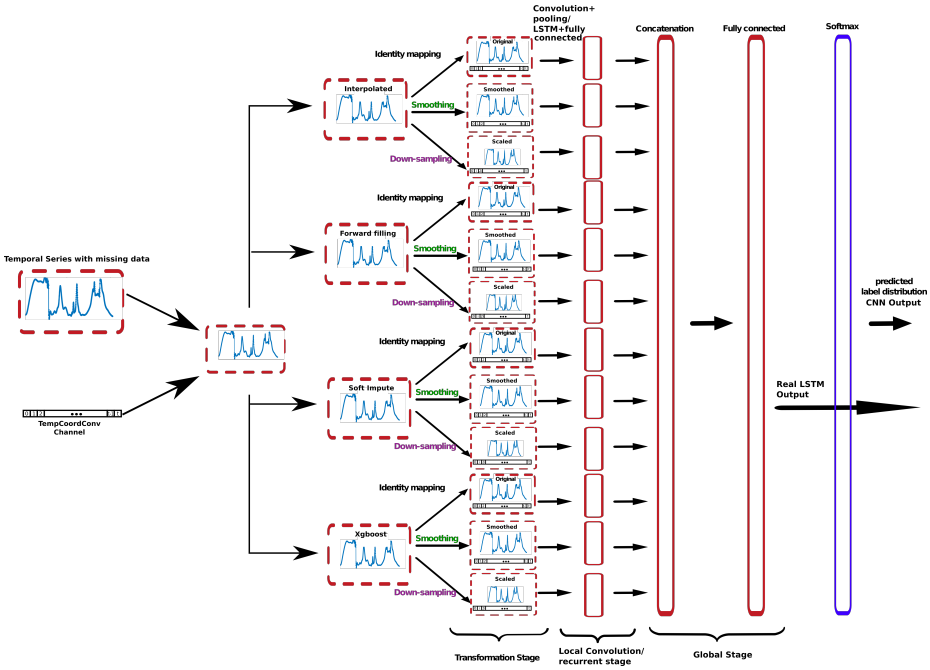


**Fig. 1.** MCNN/LSTM architecture scheme with data imputation.

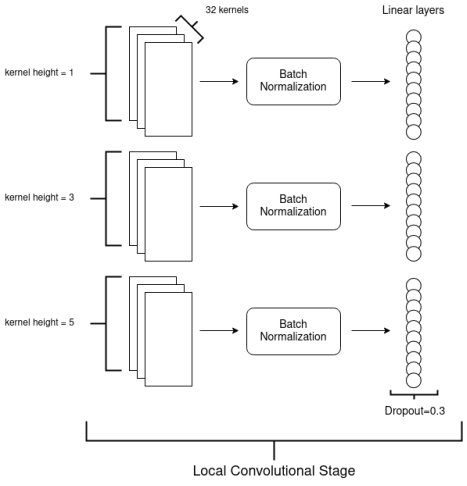


**Fig. 2.** Scheme of one branch in local convolution stage.

– The repeated branches in the baseline model are replaced with the chosen mechanisms for data imputation previously described.
– We add a temporary feature channel. The temporal information added in the new channel is only a list of index $1, 2, ..., n-1, n$, which corresponds with a column of the temporal series, and $n$ is the number of its elements.
– As in the Baseline model, each branch of Local convolution stage in figure 1 consists of 3 convolutional layers in parallel of 32 kernels, with a kernel height of 1, 3, and 5 respectively, each with batch normalization [14] and followed by a linear layer of 10 neurons each with a dropout of 0.3 (figure 2). The number of branches is 12, 4 data filling techniques, and 3 transformations for each of the techniques. 2.

**LSTM imputation data regression model**

In Figure 1, we schematize our model architecture. The differences between the baseline model 4.2 and our model are as follows:

– The repeated branches in the baseline model are replaced with the same mechanisms for data imputation that were applied in the CNN model (forward filling, linear regression, soft impute, and Xgboost).
– We add a temporary feature channel, same as the CNN model.
– As in the baseline model, each branch of local recurrent stage in figure 1 consists of 1 LSTM layer, with 100 hidden states, followed by a linear layer of 10 neurons each with a dropout of 0.3 (figure 3). The number of branches is 12,4 data filling techniques, and 3 transformations for each of the techniques 3.

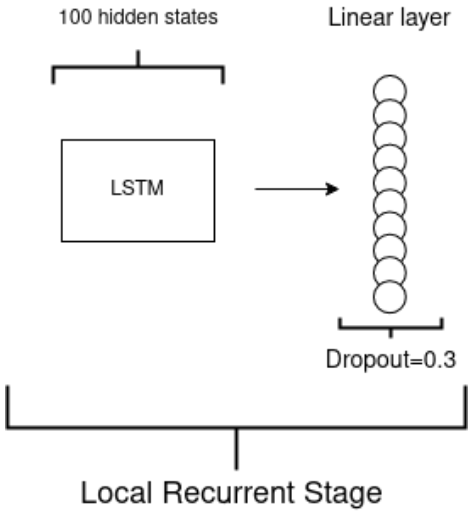


**Fig. 3.** Scheme of one branch in local recurrent stage.

## 5 Results

In this section, we present the evaluation of models proposed in the previous one.

### 5.1 Classification

We perform experiments for classification using "Wafer" and "Ford" datasets, with percentages 10, 30, 50, 60, 70, 80 and 90 of missing data in each time series as we described in Section 3. For mimic we use 10, 20 and 30. It is important to remark that for each percentage of missing data, we generate ten different datasets. In Tables 1, 2, 3 and 4 we compare the results of our data imputation with the baseline model described in the previous section. We report the performance for each percentage $f$ of deleted data by using the standard quality measure. The selected measures were the average of $ROC - AUC$ measure ($\overline{AUC}$), and the standard deviation of $ROC - AUC$ ($\sigma_{AUC}$). The results show that our model exceeds the baseline even more while increasing the percentage of deleted data. It is important to note that the baseline model has a higher standard deviation in the results when the loss of data increases. The pertinent t-tests were carried out for these datasets, and it was found that, for the wafer and ford datasets, the models evaluated with data erasure between 50% and 90% are statistically different in 95% of the cases. In the Mimic-III dataset, it is found that they are different when the models are evaluated with data erasure above 10%.

It is remarkable how for classification models, our approach consistently outperforms the baseline. In simpler datasets, such as wafer or ford, it is found that when we evaluate the models by deleting time intervals up to 70%, the model continues to perform well and is consistently better than the baseline. For higher percentages, the performance decreases, which is to be expected.

For more complex datasets, such as mimic-III, it is observed that when 20% of data is deleted, the performance drops considerably, however our model is still consistently better.

The behavior of the models for those datasets without data deletion is the same, which is to be expected since by not having data deleted, both models are essentially the same.

In the case of the mimic-III dataset with global deletion, it is observed that the behavior of the model is worse than in the single variable datasets. It must be considered that we perform the data deletion, erasing all variables at the same time. This means that we delete patients and this makes the prediction more difficult. Additionally, the target variable is more complex than in previous datasets. In order to simulate a more realistic experiment in which only some features of patients are missing, we create a new datasets with deletion of data by column. Better results are observed since the model can use information from one variable to estimate the behavior at the time when another variable is absent.

| % $f$ of Del. | Our Model | | Baseline | |
|---|---|---|---|---|
| | $\overline{AUC}$ | $\sigma_{AUC}$ | $\overline{AUC}$ | $\sigma_{AUC}$ |
| 0 | 99.2% | 1.0% | 99.2% | 1.0% |
| 10 | 99.1% | 1.1% | 99.1% | 1.0% |
| 30 | 98.3% | 2.3% | 98.1% | 1.2% |
| 50 | 95.3% | 2.3% | 92.8% | 3.7% |
| 60 | 94.8% | 1.5% | 90.5% | 5.3% |
| 70 | 94.1% | 2.2% | 87.1% | 3.1% |
| 80 | 76.1% | 5.1% | 70.5% | 5.7% |
| 90 | 56.7% | 8.4% | 53.0% | 9.6% |

**Table 1.** Performance of our model against the Baseline in Wafer Dataset.

| % $f$ of Del. | Our Model | | Baseline | |
|---|---|---|---|---|
| | $\overline{AUC}$ | $\sigma_{AUC}$ | $\overline{AUC}$ | $\sigma_{AUC}$ |
| 0 | 97.9% | 1.3% | 97.9% | 1.3% |
| 10 | 95.2% | 2.2% | 95.1% | 1.7% |
| 30 | 93.4% | 3.1% | 93.1% | 3.3% |
| 50 | 91.9% | 2.7% | 88.3% | 3.2% |
| 60 | 91.3% | 2.5% | 85.5% | 3.3% |
| 70 | 88.7% | 2.7% | 84.2% | 2.1% |
| 80 | 69.3% | 6.2% | 65.1% | 7.4% |
| 90 | 54.1% | 7.7% | 54.7% | 9.4% |

**Table 2.** Performance of our model against the Baseline in Ford Dataset.

| % $f$ of Del. | Our Model | | Baseline | |
|---|---|---|---|---|
| | $\overline{AUC}$ | $\sigma_{AUC}$ | $\overline{AUC}$ | $\sigma_{AUC}$ |
| 0 | 91.2% | 1.2% | 91.2% | 1.2% |
| 10 | 89.4% | 1.1% | 87.5% | 1.0% |
| 15 | 71.7% | 3.4% | 66.8% | 4.2% |
| 20 | 64.3% | 4.3% | 60.1% | 5.0% |
| 25 | 59.7% | 5.0% | 57.3% | 6.6% |
| 30 | 56.4% | 7.3% | 55.5% | 8.3% |

**Table 3.** Performance of our model against the Baseline in MIMIC-III Dataset.

| % $f$ of Del. | Our Model | | Baseline | |
|---|---|---|---|---|
| | $\overline{AUC}$ | $\sigma_{AUC}$ | $\overline{AUC}$ | $\sigma_{AUC}$ |
| 0 | 91.2% | 1.2% | 91.2% | 1.2% |
| 10 | 90.2% | 1.1% | 89.2% | 1.2% |
| 15 | 75.3% | 2.4% | 72.9% | 2.6% |
| 20 | 68.9% | 2.9% | 64.4% | 3.7% |
| 25 | 60.2% | 4.4% | 59.2% | 4.8% |
| 30 | 56.6% | 7.3% | 55.8% | 7.5% |

**Table 4.** Performance of our model against the Baseline in MIMIC-III Dataset, with independent deletion per column.

### 5.2 Regression

We perform experiments for regression using the COVID-19 dataset, with percentages 10, 20, and 30 of missing data. It is important to remark that for each percentage of missing data, we generate ten different datasets. We report the performance for each percentage $f$ of deleted data by using the standard quality measure. The selected measures were the average of the Percentage Prediction Error measure ($\overline{PPE}$) [1], and the standard deviation of $PPE$ ($\sigma_{PPE}$).

The results in Table 5 show that our model exceeds the baseline consistently while increasing the percentage of deleted data. However, the problem seems to be more sensitive to data loss; for values over 30%, the $PPE$ is too big. To validate our results, we perform t-tests, obtaining that the results of the regression models are statistically different in 95% of the cases in the regression for a value above 10% of the data deletion. In addition, with no data erasure (0%), both the proposed model and the baseline converge to the same model, as we expected.

| % $f$ of Del. | Our Model | | Baseline | |
|---|---|---|---|---|
| | $\overline{PPE}$ | $\sigma_{PPE}$ | $\overline{PPE}$ | $\sigma_{PPE}$ |
| 0 | 26.4% | 14.7% | 26.4% | 14.7% |
| 10 | 31.3% | 23.4% | 39.7% | 26.3% |
| 15 | 38.1% | 27.2% | 47.1% | 31.2% |
| 20 | 53.9% | 39.3% | 60.5% | 35.1% |
| 25 | 64.5% | 41.1% | 70.1% | 35.6% |
| 30 | 80.5% | 42.4% | 86.9% | 46.5% |

**Table 5.** Performance of our LSTM model against the Baseline in COVID-19 dataset.

## 6 Conclusions and Future Work

As a general conclusion, we find that our model is a suitable method for the automatic imputation of missing data in temporal series classification and regression. The results obtained show that our approach outperforms a very competitive method combined with a standard interpolation. Another remarkable property is that our model's imputation is automatic, while this task is usually dependent on the dataset statistical analysis. For the classification task, our results show that our models outperform the state-of-the-art baseline in the three considered dataset, obtaining better ($\overline{AUC}$) and minor variance when the drop of data is higher. We obtain a similar result in the regression task, but the improvement in the quality measure ($\overline{MPE}$) is even more significant.

This research opens many doors to evolve the model. Next, we describe the most relevant ones. First, we can include any new data filling techniques in our model. Another line of investigation, especially with multi-class classification, involves using Generative Adversal Network (GAN) [38] for generating the missing time series data. Finally, we can extend the model to apply it not only as a classifier and regressor of time series, but also as a forecast. In this sense, we could include state of the art models of forecasting and let the learning models choose the best option. Some of the possible combinations to use are: DeepAR [30], which is based on LSTM or GRU neurons; combined models of RNN as in [29], LSTNet [18], which is a mixture of CNN and RNN; or finally, augmented LSTM [13], which combines LSTM with autoencoders. In addition, we can make the approach more complex by exploring mixed problems of regression and forecasting.

## References

1. Comparison of some forecasting methods for covid-19. Alexandria Engineering Journal **60**(1), 1565–1589

2. Covid-19. casos registrados en la república argentina, http://datos.salud.gob.ar/dataset/covid-19-casos-registrados-en-la-republica-argentina
3. Bagnall, A.: Time series classification with ensembles of elastic distance measures. Data Mining and Knowledge Discovery (2014)
4. Bagnall, A., Lines, J., Hills, J., Bostrom, A.: Time-series classification with cote: The collective of transformation-based ensembles. IEEE Transactions on Knowledge and Data Engineering (2015)
5. Baydogan, M., Runger, G., Tuv, E.: A bag-of-features framework to classify time series. IEEE Transactions on Pattern Analysis and Machine Intelligence **35** (2013)
6. Cai, J.F., Candès, E.J., Shen, Z.: A singular value thresholding algorithm for matrix completion. SIAM Journal on optimization **20**(4), 1956–1982 (2010)
7. Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using rnn encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078 (2014)
8. Cui, Z., Chen, W., Chen, Y.: Multi-scale convolutional neural networks for time series classification. ArXiv (2016)
9. Goldberger, A.L., Amaral, L.A., Glass, L., Hausdorff, J.M., Ivanov, P.C., Mark, R.G., Mietus, J.E., Moody, G.B., Peng, C.K., Stanley, H.E.: Physiobank, physiotoolkit, and physionet: components of a new research resource for complex physiologic signals. circulation **101**(23), e215–e220 (2000)
10. Grabocka, J., Schilling, N., Wistuba, M., Schmidt-Thieme, L.: Learning time-series shapelets. In: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 392–401 (2014)
11. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**, 1735–80 (1997)
12. Hou, N., Li, M., He, L., Xie, B., Wang, L., Zhang, R., Yu, Y., Sun, X., Pan, Z., Wang, K.: Predicting 30-days mortality for mimic-iii patients with sepsis-3: a machine learning approach using xgboost. Journal of Translational Medicine **18**(1), 1–14 (2020)
13. Hsu, D.: Time series forecasting based on augmented long short-term memory. arXiv preprint arXiv:1707.00666 (2017)
14. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift (2015)
15. Johnson, A.E., Pollard, T.J., Shen, L., Li-Wei, H.L., Feng, M., Ghassemi, M., Moody, B., Szolovits, P., Celi, L.A., Mark, R.G.: Mimic-iii, a freely accessible critical care database. Scientific data **3**(1), 1–9 (2016)
16. Karim, F., Majumdar, S., Darabi, H., Harford, S.: Multivariate lstm-fcns for time series classification. Neural Networks **116** (2019)
17. Keshavan, R.H., Montanari, A., Oh, S.: Matrix completion from a few entries. IEEE transactions on information theory **56**(6), 2980–2998 (2010)
18. Lai, G., Chang, W.C., Yang, Y., Liu, H.: Modeling long-and short-term temporal patterns with deep neural networks. In: The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval. pp. 95–104 (2018)
19. Lin, S., Wu, X., Martinez, G., Chawla, N.V.: Filling missing values on wearable-sensory time series data. In: Proceedings of the 2020 SIAM International Conference on Data Mining. pp. 46–54. SIAM (2020)
20. Liu, R., Lehman, J., Molino, P., Petroski Such, F., Frank, E., Sergeev, A., Yosinski, J.: An intriguing failing of convolutional neural networks and the coordconv solution. In: Advances in Neural Information Processing Systems 31, pp. 9605–9616. Curran Associates, Inc. (2018)

21. Luo, Y., Szolovits, P., Dighe, A.S., Baron, J.M.: 3d-mice: integration of cross-sectional and longitudinal imputation for multi-analyte longitudinal clinical data. Journal of the American Medical Informatics Association **25**(6), 645–653 (2018)
22. Mazumder, R., Hastie, T., Tibshirani, R.: Spectral regularization algorithms for learning large incomplete matrices. The Journal of Machine Learning Research **11**, 2287–2322 (2010)
23. Mohammed, B., Awan, I., Ugail, H., Younas, M.: Failure prediction using machine learning in a virtualised hpc system and application. Cluster Computing (Jun 2019)
24. Oweira, H., Schmidt, J., Mehrabi, A., Kulaksiz, H., Schneider, P., Schöb, O., Giryes, A., Abdel-Rahman, O.: Comparison of three prognostic models for predicting cancer-specific survival among patients with gastrointestinal stromal tumors. Future Oncology **14**(4), 379–389 (2018)
25. Rakthanmanon, T., Campana, B., Mueen, A., Batista, G., Westover, M.B., Zhu, Q., Zakaria, J., Keogh, E.: Searching and mining trillions of time series subsequences under dynamic time warping. vol. 2012 (08 2012)
26. Rakthanmanon, T., Keogh, E.: Fast Shapelets: A Scalable Algorithm for Discovering Time Series Shapelets, pp. 668–676 (05 2013)
27. RUBIN, D.B.: Inference and missing data. Biometrika **63**(3), 581–592 (12 1976)
28. Salem, A.M., Revett, K., El-Dahshan, E.A.: Machine learning in electrocardiogram diagnosis. In: 2009 International Multiconference on Computer Science and Information Technology (2009)
29. Salinas, D., Bohlke-Schneider, M., Callot, L., Medico, R., Gasthaus, J.: High-dimensional multivariate forecasting with low-rank gaussian copula processes. arXiv preprint arXiv:1910.03002 (2019)
30. Salinas, D., Flunkert, V., Gasthaus, J., Januschowski, T.: Deepar: Probabilistic forecasting with autoregressive recurrent networks. International Journal of Forecasting **36**(3), 1181–1191 (2020)
31. Schäfer, P.: Towards time series classification without human preprocessing. pp. 228 – 242 (01 2014)
32. Schäfer, P.: The boss is concerned with time series classification in the presence of noise. Data Mining and Knowledge Discovery **29** (2015)
33. Schäfer, P.: Scalable time series classification. Data Mining and Knowledge Discovery **30** (2015)
34. Senin, P., Malinchik, S.: Sax-vsm: Interpretable time series classification using sax and vector space model (2013)
35. Srebro, N., Rennie, J.D., Jaakkola, T.S.: Maximum-margin matrix factorization. In: NIPS. vol. 17, pp. 1329–1336. Citeseer (2004)
36. Wyse, L.L.: Audio spectrogram representations for processing with convolutional neural networks. ArXiv (2017)
37. Yi, X., Zheng, Y., Zhang, J., Li, T.: St-mvl: filling missing values in geo-sensory time series data (2016)
38. Yoon, J., Jarrett, D., van der Schaar, M.: Time-series generative adversarial networks. In: Advances in Neural Information Processing Systems 32. Curran Associates, Inc. (2019)
39. Zhang, X., doi, C., Gao, C., B., M., Chen, Y.: Predicting missing values in medical data via xgboost regression. Journal of Healthcare Informatics Research **4** (12 2020)