Predictive models on 1D signals in a small-data environment

Fabio Coppini, Yiye Jiang, Sonia Tabti

▶ **To cite this version:**

Fabio Coppini, Yiye Jiang, Sonia Tabti. Predictive models on 1D signals in a small-data environment. [Research Report] IMB - Institut de Mathématiques de Bordeaux. 2021. hal-03211100

HAL Id: hal-03211100
https://hal.science/hal-03211100v1

Submitted on 28 Apr 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Predictive models on 1D signals in a small-data environment

Fabio Coppini*[1], Yiye Jiang*[2], Sonia Tabti[3]

SEME Bordeaux, 26-30 oct. 2020 Talence (France)

Abstract

This report is concerned by one project done during the Semaine d'Études Mathématiques et Entreprises (SEME) at the Institut de Mathématiques de Bordeaux. The subject, proposed by the company FieldBox.ai, concerns the use of machine learning algorithms and data augmentation techniques, applied to small datasets composed of 1D signals measurements. The target variable is supposed to be continuous, i.e., a regression problem. By first reviewing the literature and existing methods on data augmentation, we propose two procedures to tackle this problem: one allows to create synthetic observations for a specific range of target values and it is based on a perturbation method in Fourier/Wavelet space; the other is based on neural networks and uses a particular version of the Variational Autoencoder known as LSTM-VAE. Our methods are applied to an open dataset, available at the UCI repository, and show encouraging results for a common class of machine learning algorithms.

Keywords: small-data, machine learning, inferring techniques, data augmentation, imputation, Variational Autoencoder, LSTM, Fast Fourier Transform, Discrete Wavelet Transform, time series, class imbalance

[1]Laboratoire de Probabilités Statistique et Modélisation (UMR 8001), Université de Paris, France
[2]Institut de Mathématiques de Bordeaux (UMR 5251), Université de Bordeaux, France
[3]Lead Data Scientist at FieldBox.ai, Bordeaux, France
*Authors contributed equally.

Subject proposed by FieldBox.ai

In many industrial use cases, we dispose of a set of multiple input 1D signals collected from different sensors (pressure, temperature, ...) and we need to predict a target variable that can be discrete (0: failure, 1: no failure) if it is a classification problem, or continuous if it is a regression problem (eg: a measure of product quality). Although much Machine Learning research work is dedicated to big data processing, in industry, especially after the data cleaning process, it often happens that the dataset is small. After a short bibliographic study on building accurate models with small time series/ 1D signals datasets, the first objective of this work will be to identify existing Python packages or implement some methods to tackle this problem. For instance, using oversampling to generate more data is a good start. The second objective will be to compare the performances of a standard predictive model and the proposed methods in order to assess the improvement they could bring.

Acknowledgments

The team is indebted towards several persons and structures. In random order, we would like to thank: professor Edoardo Provenzi and the Institut de Mathématiques de Bordeaux[1] for the organization, the kind hospitality and the support the global situation notwithstanding; the company FieldBox.ai[2] and, especially, the lead data scientist Sonia Tabti not only for proposing us the subject, but also for insightful discussions during and after the week; the agency AMIES[3] for making this experience possible and for its effort in merging mathematicians and real world problems.

We also would like to thank all the other participants for the pleasant time we had together and the fruitful exchanges.

This report not only concerns the work done during the SEME in Bordeaux, where the PhD student Julien Bidan also contributed, but also, and mainly, on successive investigations and tries. Further ideas came up during discussions between the first two authors who decided to write them down and who equally contributed in writing this report.

[1] `https://www.math.u-bordeaux.fr/imb/spip.php`
[2] `https://www.fieldbox.ai/`
[3] `https://www.agence-maths-entreprises.fr/`

Contents

1 Introduction

Data Science and Machine Learning have gained an exponentially growing attention in the last fifteen years and a lot of effort has been put in deploying algorithms capable of crunching massive datasets.

Nowadays, much of the attention is focused on big data and deep neural networks that take days to be trained; however, from a pure industrial viewpoint, most of the common situations that a small to medium-size company needs to face, concern datasets with relatively few observations and classical machine learning algorithms. The aim of this work is to address such small-data framework for a particular kind of data input.

1.1 Small-data environment

Suppose $\mathcal{D}$ is a dataset consisting of n observations and d features, i.e.,

$$\mathcal{D} = (X_1, \ldots, X_d, y) \tag{1.1}$$

where $X_i \in \mathbb{R}^n$ represents the n observations related to the i-th feature, for $i = 1, \ldots, d$, and $y \in \mathbb{R}^n$ stands for n realizations of the target variable. We use the

following notation $X = (X_1, \dots, X_d) \in \mathbb{R}^{n \times d}$ for the $n \times d$ matrix of observations and thus (X, y) for the dataset $\mathcal{D}$.

A big-data environment is usually modeled by the fact that when the number of observations n tends to infinity, the number of features d is fixed or potentially grows slower than n. It is more difficult to define a relation for n and d when talking about *small-data*. Conditioned by industrial applications, we consider a dataset to fit a small-data environment whenever the number of observations per feature, i.e., n/d, is approximately constant. In our setting, we mostly consider the situation $n \approx 500$ and $d \approx 10$.

1.2 1D signals/time-series input

We further restrict our analysis to the case where each single feature is a 1-dimensional (1D) signal, i.e., for $i = 1, \dots, d$, $X_i(\cdot) = X_i(t) \in \mathbb{R}$ with $t > 0$ and the vector of observations satisfies

$$X_i = (X_i(t_1), \dots, X_i(t_n))^\top, \quad \text{for } t_1 < t_2 < \cdots < t_n, \tag{1.2}$$

where $t_i > 0$ are different timestamps at which the signal X_i is measured.

We consider two different scenarios for the target variable: either it does not explicitly depend on the time variable, i.e., it is a pure 1D signal, or it explicitly depends on it and can be treated as a time-series on its own. The proposed solutions in sections 2 and 4 work under the first and the second assumption respectively.

1D signal

The target variable $y(\cdot)$ is supposed to be a function of the features only. In particular, we suppose that

$$\mathbb{R} \ni y(t) = f(X_1(t), \dots, X_d(t)), \quad t > 0. \tag{1.3}$$

Remark 1.1. *The previous assumption assures that y is constant whenever the signals are so. In particular, this hypothesis apriori excludes the tools available from time-series analysis such as autoregression models and stationary analysis.*

Time-series

The target variable $y(\cdot)$ is supposed to be a function of the features and the time. In particular, we suppose that

$$\mathbb{R} \ni y(t) = f(t, X_1(t), \dots, X_d(t)), \quad t > 0. \tag{1.4}$$

1.3 Problem formulation

We would like to answer the following question:

does the 1D signal nature of the problem gives more information that can help building more reliable models?

The approach suggested by FieldBox.ai relies on data augmentation techniques, i.e., in generating new synthetic observations in the train set while maintaining a correct target prediction. One of the main goals of this procedure is to reduce overfitting.

Within this perspective, the main question can be reformulated in the following points:

1. Is it possible to infer new synthetic observations by exploiting the intrinsic time nature of the dataset?

2. How can one adapt the data augmentation techniques to deal with a continuous target variable?

3. Does augmenting the dataset size with new observations help in improving robustness of the model, e.g., reducing overfitting, while only slightly increasing the bias?

Existing results on data augmentation (see next subsection for the known literature) mainly focus on classification problems: from a conceptual viewpoint, the second question represents a challenging issue. The last question poses an interesting problem already known in imputation statistics.

This report proposes a few ideas to tackle previous points, in particular 1. and 2.. Experimental results confirming the choice of this mathematical framework are also provided, we refer to our GitHub repository for the source code, see `https://github.com/fdesmond/seme-ts`.

1.4 Brief state of the art on data augmentation

There is a fast growing literature on data augmentation techniques [7, 5, 13], yet most of it focuses on classification problems and deep learning algorithms, see, e.g., [16, 19, 10], with only few results on the regression case and classical machine learning methods, e.g., [6, 12, 11, 18].

Data augmentation is usually sought for one of the following reasons:

- to reduce class imbalance, especially in classification problems whenever one or more labels are highly underrepresented [4];

- to avoid overfitting in neural networks when the training dataset is not large enough, e.g., in image recognition [16];

- to improve robustness of certain classifiers, by interpolation and/or extrapolation of observations in the training set [5];

- to reduce the curse of dimensionality, e.g., by adding points in sparse regions [11].

From an abstract viewpoint, data agumentation techniques fall into the two main categories: domain independent techniques, i.e., methods that apply to roughly any problem; and domain dependent techniques, where the dataset domain (e.g., images, time-series, etc) is taken into account for a tailor-made augmentation.

Examples of domain independent techniques are given by bootstrapping techniques, with our without adding a noise perturbation, see, e.g., [4, 3], but also the use of autoencoders, e.g., [5, 15]. In such cases, the idea is to create a new observation by slightly perturbing an existing one and to assign it the same label: the perturbation is usually not performed on the input data itself, but on a transformed version of it in some abstract space. This space, usually called *feature space*, can be again domain independent (e.g., in the case of autoencoders, dimensionality reduction techniques, etc.), or use information from the data domain (e.g., the Fourier space for time-series).

Looking at domain dependent techniques, we've found only two references citing 1D signals [18, 19]. Most of the results concern data augmentation for time-series classification problems [10, 9, 8, 17, 1, 14] as shown below.

Time-series data augmentation

Following [19], we recognize the following approaches that are used for generating new time-series from existing ones:

- perturb the signal in a suitable feature space (e.g., time domain, frequency domain, time-frequency domain);
- decompose the signal as a time-series with trend, seasonality and residual and bootstrap the residual;
- fit a statistical model on the signal (e.g, ARMA, ARCH, etc.) and perturb its parameter space;
- use learning methods (Autoencoder, Generative Adversarial Network, Reinforcement Learning, etc.).

The new time-series are usually associated to the same label of the parent.

As already anticipated in Section 1.3, a key difficulty of our case is the fact that we are dealing with a regression problem and that there is no label to group the inputs into *similar* objects. Moreover, much of the cited results assume the time-series to be an observation on itself, while for us, the signals represent the features of the input data, i.e., a column and not a row in the dataset $\mathcal{D}$! Fortunately, we can still manage to reproduce some of the ideas present in the literature, we refer to Section 2 (in particular to subsection 2.3) and Section 4.

Remark 1.2. *Interpolation is another possible approach: to add observations by interpolating points in each signal X_i and in y. However, this method requires the signals, and the output, to be regular, an hypothesis that we do not want to assume. We refer to [14] for some (not so satisfactory) result in this direction.*

1.5 Proposed approaches

To tackle this problem, we propose two solutions that are presented in sections 2 and 4. The experimental results of the first methodology are discussed right after in Section 3, while we refer to GitHub page[4] for the second one.

[4] `https://github.com/fdesmond/seme-ts`

2 Quantile-based augmentation in feature space

2.1 Augmentation procedure

The main principle of the proposed method is the following one: define a time window, i.e., a subset of $\{t_1, \ldots, t_n\}$, and randomly select some of the features among $\{X_1, \ldots, X_d\}$; on the time window, replace the chosen features with a suitable perturbed version of them, but keep the other features and the target variable unchanged. Finally, append perturbed and non-perturbed features (together with the target variable) to the original dataset.

To further augment the dataset, on can iterate this procedure with another time window and new features.

Algorithm 1 General procedure

Denote by $\bar{d} \in \{1, \ldots, d\}$ the number of features to be perturbed and by $\sigma > 0$ the strength of the perturbation one wants to apply.

procedure DFAUG($\mathcal{D}$, $\bar{d}$, σ)
  $\bar{\mathcal{D}} \leftarrow$ SLICE($\mathcal{D}$) ▷ slice the input dataset
  $\mathcal{D}_{\texttt{aug}} \leftarrow$ AUG($\bar{\mathcal{D}}, \bar{d}, \sigma$) ▷ obtain synthetic observations
  return $\mathcal{D} \cup \mathcal{D}_{\texttt{aug}}$ ▷ concatenate the observations
end procedure

Observe that this procedure can be iterated to further augment the dataset by choosing new $\bar{d}$ and σ. For such implementation, we refer to the GitHub page.

The general procedure DFAUG is summarized in Algorithm 1 with the main function AUG in Algorithm 2. The functions SLICE and DISTORT, corresponding to the slicing and perturbation part respectively, are discussed in the following subsections 2.2 and 2.3.

Algorithm 2 Augmentation function

For $\mathcal{D} = (X_1, \ldots, X_d, y)$ a dataset and $\bar{d} \in \{1, \ldots, d\}$. The coefficient $\sigma > 0$ stands for the distortion magnitude applied on the selected time-series.

function AUG($\mathcal{D}$, $\bar{d}$, σ)
  $i_1, \ldots, i_{\bar{d}} \in \{1, \ldots, d\}$ ▷ sampled without replacement
  for $k = 1, \ldots, \bar{d}$ **do**
    $X_{i_k} \leftarrow$ DISTORT(X_{i_k}, σ) ▷ perturb the selected feature
  end for
  return $(X_1, \ldots, X_d, y)$ ▷ output the synthetic observations
end function

Observe that if $\bar{d} = d$, then all the features are perturbed.

2.2 Slicing

Slicing consists in selecting a subset of the rows in the input dataset: in our case, this is equivalent to retaining certain timestamps, e.g., by considering $\{\tilde{t}_1, \dots, \tilde{t}_{\tilde{n}}\} \subset \{t_1, \dots, t_n\}$ for $\tilde{n} \leq n$, one obtains the following dataset

$$\mathcal{D}_{\text{sliced}} = \big(X_1(\tilde{t}_i), \dots, X_d(\tilde{t}_i), y(\tilde{t}_i)\big)_{i=1,\dots,\tilde{n}}\,.$$

A naive method to slice a dataset is to consider a specific time window, i.e., a subset of consecutive timestamps $\{t_j \text{ for } j \in J \subset \{1, \dots, n\}\}$. However, this procedure did not yield significant improvements in our setting and we implemented a more sophisticated procedure that we explain hereafter.

In the classification case, one can apply the augmentation procedure separately on each class, in the regression case this is clearly not possible since the target variable apriori takes an infinite number of values and no class is defined. However, if we group observations for which the target variable is not varying too much, we can treat them as belonging to a same abstract class and augment this class by keeping the target values within the range defined by the class itself. There are many different ways to implement this idea, we propose the following procedure.

Select the timestamps t_i for which the target variable is varying within a certain range, i.e., for some constant $c \in \mathbb{R}$ and $\varepsilon > 0$, let $J_c^\varepsilon \subset \{1, \dots, n\}$ be such that

$$|y(t_i) - c| < \varepsilon, \qquad \text{for } i \in J_c^\varepsilon. \tag{2.1}$$

By choosing different c and ϵ, one can get a partition of $\{1, \dots, n\}$ in terms of intervals as J_c^ε. Equivalently, one can partition the range of y with real intervals $I_c^\varepsilon \subset \mathbb{R}$ and obtain J_c^ε as the indices of the timestamps t_i for which $y(t_i) \in [c-\varepsilon, c+\varepsilon] =: I_c^\varepsilon$. This can be obviously done for a general interval $I \subset \mathbb{R}$, see Algorithm 3.

Observe that, if the intervals are suitably chosen, e.g., select I to be a quantile interval of the empirical distribution of y, this procedure allows to create fictitious time-series for which the output is approximately constant.

Algorithm 3 Slicing function

For I a chosen interval range in $\mathbb{R}$.

```
function SLICE(D, I)
    D ← D[y ∈ I]          ▷ only select the observations for which y(t_i) ∈ I
    return D
end function
```

The function DFAUG is designed for one specific interval, but if one iterates it on the whole partition, then the new augmented dataset keeps the empirical distribution of the output vector unchanged.

A straightforward way to partition the dataset with respect to the target variable, is to compute the quantile intervals associated to y and to later apply the augmentation procedure to each interval. In the case where the target variable distribution is not uniform, one can apply the augmentation procedure more times to the sliced datasets

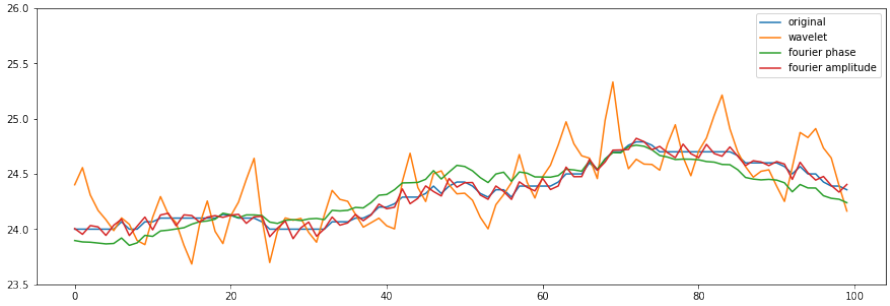

Figure 1: *An example of the three different distortions applied to a same signal.*

which correspond to the observations having a lower empirical frequency, e.g., related to *rare events*, thus providing a way to reduce – some sort of – imbalance in the dataset.

Remark 2.1. *Observe that, while the time order in the sliced dataset is maintained, the sampled t_j are not necessarily consecutive as in the naive time-window selection. It is possible to break the time structure since we are supposing that y does not explicitly depend on the time variable.*

2.3 Distortion

To perturb the selected features, we have decided to work in a suitable *feature space*. In order to exploit the 1D signal nature of the inputs, we have focused on two classical representation space for signals:

1. Fourier space;
2. Wavelet space.

In the frequency domain, we have tested different perturbations both at the level of the phase and at the level of the amplitude. The resulting signal, which has been obtained by inverting back the one perturbed in feature space, shows non-local distortions depending on the most affected modes.

Concerning the Wavelet space, the perturbation has been applied on the first half-plane only, showing the most encouraging results. During hyperparameter tuning, we have found that a mix of these three procedures yields the best results.

We refer to Figure 1 for a example with the three perturbations.

3 Experimental setting

3.1 Appliances energy prediction Data Set

A suitable dataset satisfying the hypothesis in subsection 1.2 is given by Appliances energy prediction Data Set. The dataset is used to create regression models of appli-

ances energy use in a low energy house. It consists, among other features, of several measurements of temperature, humidity and pressure in the different rooms.

We select 25 out of the 29 features in the original dataset, which are lights, T1, RH_1, T2, RH_2, T3, RH_3, T4, RH_4, T5, RH_5, T6, RH_6, T7, RH_7, T8, RH_8, T9,RH_9, T_out, Press_mm_hg, RH_out, Windspeed,Tdewpoint, Appliances. Additionally, we construct two temporal variables: daytime (t_d), dayofweek (t_w), which mark the time point when the observation is recorded in a day and the day it is recorded in a week respectively. To capture the nonlinear trends of time series, we also add the polynomial $p(t_d, t_w, m)$ regressor in the predictive model to capture the nonlinear trend, which is motivated from the polynomial regression. The order m is set as 12.

The target variable is the energy appliance (energy use in Wh). The entire dataset has 19735 observations, denoted as $\mathcal{D}_A$. We refer to the GitHub page related to this project for more details:
`https://github.com/fdesmond/seme-ts`.

3.1.1 Subsampling

We simulate the small-data environment. To this end, we subsample the 5% of $\mathcal{D}_A$, this yields the small-data set $\mathcal{D}_B$. Train-test split is performed with a proportion of $0.75 : 0.25$, so that the small input dataset on which we will train the machine learning algorithms is given by 740 observations.

Remark 3.1. *The train-test split procedure is performed at the level of $\mathcal{D}_A$ so that the test set (train set respectively) in $\mathcal{D}_B$ is actually a 5%-sampling of the test set (train set respectively) in $\mathcal{D}_A$.*

3.1.2 Hyperparameters

The augmented dataset, which we call $\mathcal{D}_C$, is obtained by firstly partitioning the observations into different subsets corresponding to the target interval quantiles; and then by applying the augmentation procedure to different portions of the dataset and with different distortion levels.

We choose to partition the dataset into 9 intervals corresponding to the quantile intervals of y: the augmentation procedure is thus applied to vectors of dimension[5] approximately 80.

3.2 Model fitting and evaluation

The same machine learning algorithm (e.g., k-NN, Linear Regression, etc.) is fit to the train sets in $\mathcal{D}_A$ and $\mathcal{D}_B$ and on $\mathcal{D}_C$. For each algorithm, we thus obtain three regression models `A`, `B` and `C` corresponding to the three training datasets.

The evaluation is done by comparing the models `B` and `C` on the test set of $\mathcal{D}_A$: we choose to use the root mean squared error ($RMSE$) and the R^2 score as metrics.

[5]The dimension here is given by: # number of observations train set / # number of quantile interval.

3.3 Results

The results are given in Table 3.3. See also Figures 2 and 3.

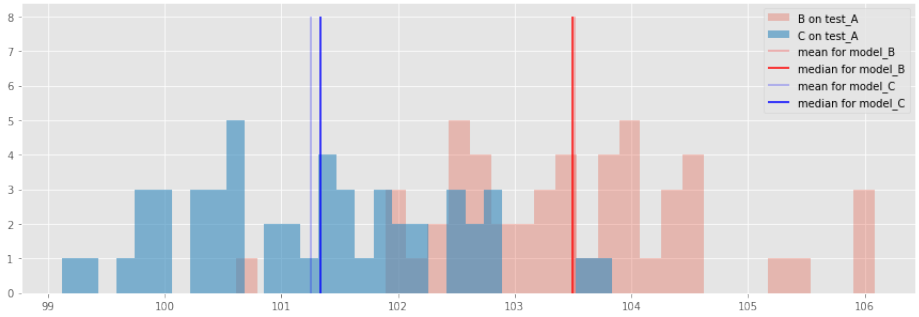

Figure 2: *Histogram of RMSE for* `B` *and* `C` *(*50 *realizations of* $\mathcal{D}_B$*).* In red-scale `B` and in blue-scale `C`.

Table 3.3 reports the prediction performance of model `B` and `C`, measured in *RMSE* and R^2 score. It shows clearly that the model trained on the augmented training set improves the performance. More specifically, Figure 2 plots the *RMSE* distributions of 50 model `B`'s and `C`'s trained on 50 independently sampled small datasets and their corresponding augmented datasets. We can see that the histogram of model `C` is on the left of the one of model `B`, which means `C` exhibits a better prediction performance than model `B`, on unseen data. Figure 3 shows the representative predictions of model `B`, and model `C`, where in several cases model `C` is able to better predict the spikes present in the test-set of $\mathcal{D}_A$. These support the effectiveness of the proposed data augmented approaches.

	RMSE	**R^2**
`A` on `test` $\mathcal{D}_A$	92.25	0.26
`B` on `test` $\mathcal{D}_B$	100.83(13.74)	0.074(0.041)
`C` on `test` $\mathcal{D}_B$	99.14(13.38)	0.103(0.061)
`B` on `test` $\mathcal{D}_A$	103.52(1.11)	0.071(0.020)
`C` on `test` $\mathcal{D}_A$	101.26(1.10)	0.111(0.019)

Table 1: *Average prediction metrics (*50 *runs).* We repeat 4 times Fourier augmentation, 2 times Wavelet augmentation to expand `train` $\mathcal{D}_B$ (740 samples), to $\mathcal{D}_C$ (6540 samples). The numbers in parentheses represent 1 standard deviation.

4 LSTM-VAE based model

In this section, we propose another framework for data augmentation, which is based on deep learning technique. The related codes and notebook with the illustrative

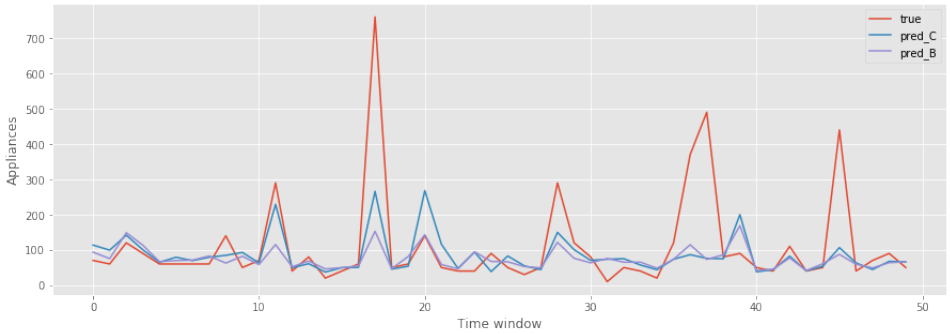

Figure 3: *Predictions comparison.* Comparing a small time-window in $\mathcal{D}_A$ of the true target (in red) with the two predictions by `B` (in purple) and `C` (in blue) respectively.

results can be found in the notebook `DL_aug.ipynb` present in `https://github.com/fdesmond/seme-ts/`.

More specifically, we use the long short-term memory variational autoencoder (LSTM-VAE) introduced in [2]. We decide not to break the temporal order during train-test-split as well as the small data sampling, see again Section 4.2 for the details. Three folds of facts motivate this choice. Firstly, autoencoders areunsupervised models which is well adapted to our problematic. Secondly, the temporal order of observations can be furthermore exploited; many network architectures have been designed for this sequential data type, with one of the most widely used being LSTM. Lastly, Variational AutoEncoder has sampling scheme, from which we can profit, to generate the restored inputs as augmented data, yet of different noisy levels. We will elaborate this point in the following section.

4.1 Data augmentation using LSTM-VAE

The main difference of variational autoencoder compared to the deterministic autoencoder is that, instead of learning a deterministic mapping from each input to its latent vector, it learns the mapping from the input to the posterior distribution of its latent vector. The posterior distributions are always assumed to be multivariate Gussian with diagonal covariance.

To reconstruct an input, VAE samples a latent vector from its posterior distribution, and pass it to the decoder. Thus even for the same input, VAE's ouptut will be different across separate runs. This property can be employed to generate the artificial data associated with the input. Furthermore, for a trained VAE, we can multiply a scale to all the covariance mapped from the inputs, in order to have the smoother/noisier reconstructed signals. These smoother/noisier reconstructions of inputs can also be used as augmented data. We call the scale *smooth level*, which is the hyperparameter of this approach.

Specific to sequential data, we use LSTM layer in the beginning of the encoder, accordingly at the end of the decoder. The network input is a matrix of dimension

timesteps$\times(d+1)$, where d denotes the number of individual time series in equation (1.3), and *timesteps* is another hyperparameter which is the common length of individual sequences. Note that, we include the response variable as well in the input sequences. We aim to let the network analyze its complex relation with all other regressor sequences, and reconstruct the multivariate sequences which can follow this relation. This property is to be explored more closely in the future. For the network architecture used in the illustrative experiment, we refer to the code on GitHub and the notebook cited at the beginning of the section.

4.2 Sampling details and latent space visualization

Since we have LSTM modules in the network, we simply take the first 75% of $\mathcal{D}_A$ as the training set, leaving the rest as test set. For the construction of small data, we take the last 5% of the training set of $\mathcal{D}_A$ as the training set of $\mathcal{D}_B$, while the first 5% of the big test set as the small test set. We train the network on the small training set to generate the augmented data for ML model fitting.

Before using the trained LSTM-VAE to synthesize artificial data, we would like to know how much information it has learnt from the small training set. In addition to examine the reconstruction performance, we propose to inspect the pattern of latent representation of entire training set of $\mathcal{D}_B$. Even though we are not in the classification problem as the common application of VAE, we can still justify the network usefulness by attaching each latent mean with the *day of week* of its input multivariate sequences[6] as classification labels, and verify if the means cluster in the latent space according to their labels.

5 Conclusions

We did not push the analysis further but we believe that these methods can be improved and give even better results.

The main ideas behind our work can be summed up into the following points:

- (procedure 1) Slice the original dataframe based on target-dependent classes;
- (procedure 1) Perturb the signals at the level of the frequency-domain (i.e., in feature space).
- (procedure 2) Exploit the latent space distribution of VAE to draw samples whose distribution is close to the one of the samples in the original dataset.

References

[1] K. Bandara, H. Hewamalage, Y.-H. Liu, Y. Kang, and C. Bergmeir. Improving the Accuracy of Global Forecasting Models using Time Series Data Augmentation. *arXiv:2008.02663 [cs, stat]*, 2020.

[6] We use the average day of week if some sequences across different days in a week.

[2] S. R. Bowman, L. Vilnis, O. Vinyals, A. M. Dai, R. Jozefowicz, and S. Bengio. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*, 2015.

[3] C. Buciluǎ, R. Caruana, and A. Niculescu-Mizil. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '06, pages 535–541, New York, NY, USA, 2006. Association for Computing Machinery.

[4] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research*, 16:321–357, 2002.

[5] T. DeVries and G. W. Taylor. Dataset Augmentation in Feature Space. 2017.

[6] F. Dubost, G. Bortsova, H. Adams, M. A. Ikram, W. Niessen, M. Vernooij, and M. de Bruijne. Hydranet: Data Augmentation for Regression Neural Networks. In D. Shen, T. Liu, T. M. Peters, L. H. Staib, C. Essert, S. Zhou, P.-T. Yap, and A. Khan, editors, *Medical Image Computing and Computer Assisted Intervention – MICCAI 2019*, Lecture Notes in Computer Science, pages 438–446, Cham, 2019. Springer International Publishing.

[7] D. A. v. Dyk and X.-L. Meng. The Art of Data Augmentation. *Journal of Computational and Graphical Statistics*, 10(1):1–50, 2001.

[8] H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller. Data augmentation using synthetic data for time series classification with deep residual networks. *arXiv:1808.02455 [cs]*, 2018.

[9] G. Forestier. Generating synthetic time series to augment sparse datasets. pages pp. 865–870, 2017.

[10] A. L. Guennec, S. Malinowski, and R. Tavenard. Data Augmentation for Time Series Classification using Convolutional Neural Networks. 2016.

[11] G. Hooker and S. Rosset. DARE: Data-Augmented Regression for Extrapolation. page 8.

[12] M. M. Krell, A. Seeland, and S. K. Kim. Data Augmentation for Brain-Computer Interfaces: Analysis on Event-Related Potentials Data. *arXiv:1801.02730 [cs, q-bio]*, 2018.

[13] M. Kuchnik and V. Smith. Efficient Augmentation via Data Subsampling. page 22, 2019.

[14] C. Oh, S. Han, and J. Jeong. Time-Series Data Augmentation based on Interpolation. *Procedia Computer Science*, 175:64–71, 2020.

[15] H. Ohno. Auto-encoder-based generative models for data augmentation on regression problems. *Soft Computing*, 24(11):7999–8009, 2020.

[16] C. Shorten and T. M. Khoshgoftaar. A survey on Image Data Augmentation for Deep Learning. *Journal of Big Data*, 6(1):60, 2019.

[17] S. Smyl and K. Kuber. *Data Preprocessing and Augmentation for Multiple Short Time Series Forecasting with Recurrent Neural Networks*. 2016.

[18] F. Wang, S.-h. Zhong, J. Peng, J. Jiang, and Y. Liu. Data Augmentation for EEG-Based Emotion Recognition with Deep Convolutional Neural Networks. In K. Schoeffmann, T. H. Chalidabhongse, C. W. Ngo, S. Aramvith, N. E. O'Connor, Y.-S. Ho, M. Gabbouj, and A. Elgammal, editors, *MultiMedia Modeling*, Lecture Notes in Computer Science, pages 82–93, Cham, 2018. Springer International Publishing.

[19] Q. Wen, L. Sun, X. Song, J. Gao, X. Wang, and H. Xu. Time Series Data Augmentation for Deep Learning: A Survey. *arXiv:2002.12478 [cs, eess, stat]*, 2020.