

Time Series Forecasting using RNNs: an Extended Attention Mechanism to Model Periods and Handle Missing Values

Yagmur Gizem Cinar

Univ. Grenoble Alps/CNRS/Grenoble INP, Grenoble, France

YAGMUR.CINAR@IMAG.FR

Hamid Mirisaee

Univ. Grenoble Alps/CNRS/Grenoble INP, Grenoble, France

HAMID.MIRISAE@IMAG.FR

Parantapa Goswami

Univ. Grenoble Alps/CNRS/Grenoble INP, Grenoble, France

PARANTAPA.GOSWAMI@IMAG.FR

Eric Gaussier

Univ. Grenoble Alps/CNRS/Grenoble INP, Grenoble, France

ERIC.GAUSSIER@IMAG.FR

Ali Ait-Bachir

Coservit, Grenoble, France

A.AIT-BACHIR@COSERVIT.COM

Vadim Strijov

Computing Center of the Russian Academy of Sciences, Moscow, Russia

STRIJOV@CCAS.RU

Abstract

In this paper, we study the use of recurrent neural networks (RNNs) for modeling and forecasting time series. We first illustrate the fact that standard sequence-to-sequence RNNs neither capture well periods in time series nor handle well missing values, even though many real life times series are periodic and contain missing values. We then propose an extended attention mechanism that can be deployed on top of any RNN and that is designed to capture periods and make the RNN more robust to missing values. We show the effectiveness of this novel model through extensive experiments with multiple univariate and multivariate datasets.

ular class of approaches that has recently received much attention for modeling sequences is based on sequence-to-sequence Recurrent Neural Networks (RNNs) (Graves, 2013). Sequence-to-sequence RNNs constitute a flexible class of methods particularly well adapted to time series forecasting when one aims at predicting a sequence of future values on the basis of a sequence of past values. They have furthermore led to state-of-the-art results in different applications (as machine translation or image captioning). We explore here how to adapt them to general time series.

In real life scenarios, time series data often display periods, due for example to environmental factors as seasonality or to the patterns underlying the activities measured (the workload on professional email servers for example has both weekly and daily periods). A first question one can ask is thus: *Can sequence-to-sequence RNNs model periods in time series?* Furthermore, missing observations and gaps in the data are also common, due to *e.g.* lost records and/or mistakes during data entry or faulty sensors. In addition, for multivariate time series, underlying variables can have mixed sampling rates, *i.e.* different variables may have different sampling frequencies, resulting again in values missing at certain times when comparing the different variables. The standard strategy to tackle missing values is to apply numerical or deterministic approaches, such as interpolation and data imputation methods, to explicitly estimate or infer the missing values, and then apply classical meth-

1. Introduction

Forecasting future values of temporal variables is termed as *time series prediction* or *time series forecasting* and has applications in a variety of fields, as finance, economics, meteorology, or customer support center operations. A considerable number of stochastic (De Gooijer & Hyndman, 2006) and machine learning based (Bontempi et al., 2013) approaches have been proposed for this problem. A partic-

ods for forecasting. In the context of sequence-to-sequence RNNs, a standard technique, called padding, consists in repeating the hidden state of the last observed value over the missing values. All these methods however make strong assumptions about the functional form of the data, and imputed values, especially for long gaps, and can introduce sources of errors and biases. The second question we address here is thus: *Do sequence-to-sequence RNNs handle well missing values?*

To answer the above questions, we first conduct in Section 2 simple experiments to illustrate how state-of-the-art sequence-to-sequence RNNs behave on time series. Even though our approach is general in the sense that it can be applied to any sequence-to-sequence RNN, we consider in this study bidirectional RNNs (Schuster & Paliwal, 1997) based on Long Short-Term Memory (LSTM) networks (Hochreiter & Schmidhuber, 1997) with peephole connections (Gers et al., 2002), known to perform very well in practice (Graves, 2013), and we also make use of the attention mechanism recently introduced in (Bahdanau et al., 2014). We then present in Section 3 two extensions of the attention mechanism to better handle periods and missing values, prior to describe their application to multivariate time series (Section 4) and to evaluate their impact on several univariate and multivariate time series in Section 5. We finally discuss the related work in Section 6. In the remainder, we use the term RNNs to refer to sequence-to-sequence RNNs.

2. How do RNNs behave on time series? A study of periodicity and missing values

We focus at first on univariate time series even though most of the elements we discuss apply directly to multivariate time series (we will consider multivariate time series in a second step). As mentioned before, time series forecasting consists in predicting future values from past, observed values. The time span of the past values, denoted by T , is termed as *history*, whereas the time span of the future values to be predicted, denoted by T' , is termed as *forecast horizon* (in multi-step ahead prediction, which we consider here, $T' > 1$). The prediction is modeled as a regression-like problem where the goal is to learn the relation $\mathbf{y} = r(\mathbf{x})$ where $\mathbf{y} = (x_{T+1}, \dots, x_{T+i}, \dots, x_{T+T'})$ is the output sequence and $\mathbf{x} = (x_1, \dots, x_j, \dots, x_T)$ is the input sequence. Both input and output sequences are ordered and indexed by time instants.

For clarity's sake, and without loss of generality, for the input sequence $\mathbf{x} = (x_1, \dots, x_j, \dots, x_T)$, the output sequence \mathbf{y} is rewritten as $\mathbf{y} = (y_1, \dots, y_i, \dots, y_{T'})$.

RNNs rely on three parts, one dedicated to encoding the input, and referred to as *encoder*, one dedicated to construct-

ing a summary of the encoding, which we will refer to as *summarizer*, and one dedicated to generating the output, and referred to as *decoder*. The encoder represents each input x_j , $1 \leq j \leq T$ as a *hidden state* $\vec{\mathbf{h}}_j = f(x_j, \vec{\mathbf{h}}_{j-1})$, $\vec{\mathbf{h}}_j \in \mathbb{R}^n$, where the function f corresponds here to the non-linear transformation implemented in LSTM with peephole connections (Gers et al., 2002). The equations of LSTM with peephole connections are given in Eq. 1. For bidirectional RNNs (Schuster & Paliwal, 1997), the input is read both forward and backward, leading to two vectors $\vec{\mathbf{h}}_j = f(x_j, \vec{\mathbf{h}}_{j-1})$ and $\overleftarrow{\mathbf{h}}_j = f(x_j, \overleftarrow{\mathbf{h}}_{j+1})$. The final hidden state for any input x_j is constructed simply by concatenating the corresponding forward and backward hidden states, i.e. $\mathbf{h}_j = [\vec{\mathbf{h}}_j; \overleftarrow{\mathbf{h}}_j]^T$, where now $\mathbf{h}_j \in \mathbb{R}^{2n}$.

$$\begin{aligned} \mathbf{i}_j &= \sigma(\mathbf{W}_i x_j + \mathbf{U}_i \mathbf{h}_{j-1} + \mathbf{C}'_i \mathbf{c}'_{j-1}) \\ \mathbf{f}_j &= \sigma(\mathbf{W}_f x_j + \mathbf{U}_f \mathbf{h}_{j-1} + \mathbf{C}'_f \mathbf{c}'_{j-1}) \\ \mathbf{c}'_j &= \mathbf{f}_j \mathbf{c}'_{j-1} + \mathbf{i}_j \tanh(\mathbf{W}_c x_j + \mathbf{U}_c \mathbf{h}_{j-1}) \\ \mathbf{o}_j &= \sigma(\mathbf{W}_o x_j + \mathbf{U}_o \mathbf{h}_{j-1} + \mathbf{C}'_o \mathbf{c}'_{j-1}) \\ \mathbf{h}_j &= \mathbf{o}_j \tanh(\mathbf{c}'_j) \end{aligned} \quad (1)$$

The summarizer builds, from the sequence of input hidden states \mathbf{h}_j , $1 \leq j \leq T$, a context vector $\mathbf{c} = q(\{\mathbf{h}_1, \dots, \mathbf{h}_j, \dots, \mathbf{h}_T\})$. In its most simple form, the function q just selects the last hidden state (Graves, 2013): $q(\{\mathbf{h}_1, \dots, \mathbf{h}_j, \dots, \mathbf{h}_T\}) = \mathbf{h}_T$. More recently, in (Bahdanau et al., 2014), an attention mechanism is used to construct different context vectors \mathbf{c}_i for different outputs y_i ($1 \leq i \leq T'$) as a weighted sum of the hidden states of the encoder representing the input history: $\mathbf{c}_i = \sum_{j=1}^T \alpha_{ij} \mathbf{h}_j$, where α_{ij} are the *attention weights*. They are computed as:

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{j'=1}^T \exp(e_{ij'})} \quad (2)$$

where

$$e_{ij} = a(\mathbf{s}_{i-1}, \mathbf{h}_j) = \mathbf{v}_a^T \tanh(\mathbf{W}_a \mathbf{s}_{i-1} + \mathbf{U}_a \mathbf{h}_j) \quad (3)$$

with a being a feedforward neural network with weights \mathbf{W}_a , \mathbf{U}_a and \mathbf{v}_a trained in conjunction with the entire encoder-decoder framework. \mathbf{s}_{i-1} is the hidden state obtained by the decoder (see below) at time $(i-1)$. One can note that a scores the importance of the input at time j (specifically its representation \mathbf{h}_j by the encoder) for the output at time i , given the previous hidden state \mathbf{s}_{i-1} of the decoder. This allows the model to concentrate or *put attention* on certain parts of the input history to predict each output.

The decoder parallels the encoder by associating each output y_i , $1 \leq i \leq T'$ to a hidden state vector $\mathbf{s}_i = g(y_{i-1}, \mathbf{s}_{i-1}, \mathbf{c}_i)$ where $\mathbf{s}_i \in \mathbb{R}^n$ and y_{i-1} denotes the output at time $i-1$. The function g corresponds again to an

LSTM with peephole connections, with an explicit context added (Graves, 2013). The equations of LSTM with peephole connections and context are given in Eq. 4. Each output is then predicted in sequence through:

$$y_i = \mathbf{W}_{\text{out}} \mathbf{s}_i + b_{\text{out}},$$

where b_{out} is a scalar and $\mathbf{W}_{\text{out}} \in \mathbb{R}^n$.

$$\begin{aligned} \mathbf{i}_i &= \sigma(\mathbf{W}_i y_{i-1} + \mathbf{U}_i \mathbf{s}_{i-1} + \mathbf{C}'_i \mathbf{c}'_{i-1} + \mathbf{C}_i \mathbf{c}_i) \\ \mathbf{f}_i &= \sigma(\mathbf{W}_f y_{i-1} + \mathbf{U}_f \mathbf{s}_{i-1} + \mathbf{C}'_f \mathbf{c}'_{i-1} + \mathbf{C}_f \mathbf{c}_i) \\ \mathbf{c}'_i &= \mathbf{f}_i \mathbf{c}'_{i-1} + \mathbf{i}_i \tanh(\mathbf{W}_{c'} y_{i-1} + \mathbf{U}_{c'} \mathbf{s}_{i-1} + \mathbf{C}_{c'} \mathbf{c}_i) \quad (4) \\ \mathbf{o}_i &= \sigma(\mathbf{W}_o y_{i-1} + \mathbf{U}_o \mathbf{s}_{i-1} + \mathbf{C}'_o \mathbf{c}'_{i-1} + \mathbf{C}_o \mathbf{c}_i) \\ \mathbf{s}_i &= \mathbf{o}_i \tanh(\mathbf{c}'_i) \end{aligned}$$

2.1. RNNs and periodicity

To illustrate how RNNs behave w.r.t. periods in time series, we retained two periodic time series, fully described in Section 5 and representing respectively (a) the electrical consumption load in Poland over the period of 10 years (1/1/2002-7/31/2012), and (b) the maximum daily temperature, again in Poland, over the period of 12 years (1/1/2002-7/31/2014). The first time series, referred to as *PSE*, has two main periods, a daily one and a weekly one, whereas the second time series, referred to as *PW*, has a yearly period. Figure 1 (left and middle) displays the autocorrelation (Chatfield, 2016) for each time series and shows these different periods (note that the term *original* indicates that we used the dataset as it is, without any modification). We ran the RNN described above, with (RNN-A) and without (RNN) the attention mechanism (in the latter case, the context is taken to be the last hidden state). The experimental setting retained is the one described in Section 5.

We also added to the input sequence the time stamp (or position) information j , to help the RNNs capture potential periods. When the positions are added, each input at time j contains two values: x_j and j , $1 \leq j \leq T$. The position is of course not predicted in the output. The results we obtained, evaluated in terms of mean squared error (MSE) and displayed in Figure 1 (right), show that adding the time stamps in the input does not improve the RNNs: the results slightly degrade on *PSE* (from 0.039 to 0.053) and are almost the same on *PW*. Furthermore, on these time series, the use of the attention mechanism has almost no impact: the MSE remains almost the same when the attention mechanism is used.

To see whether standard RNNs are able to capture periods in time series, we plot, in Figure 2, the weights obtained by the attention mechanism. As mentioned before, the attention mechanism aims at capturing the importance of the input at time j for predicting the output at time i . Were

standard RNNs able to capture periods, we should see that the corresponding weights in the attention mechanism are higher. However, as one can note, this is not the case: for RNNs with time stamp information, a higher weight is put on the first instances, which do not correspond to the actual periods of the times series. Higher weights are put with standard RNNs on the more recent history on *PSE*. This seems more reasonable, and leads to better results, even though the period at one day is missed. On *PW*, the weights with standard RNNs are uniformly distributed on the different time stamps. This shows that current RNNs do not always detect (and make us of) the periods underlying the data.

2.2. RNNs and missing values

We now turn to the problem of missing values and use this time a degraded version of the above time series in which 15% of the original values are missing, either in consecutive sequences (to form gaps in the data) or randomly (we refer again the reader to Section 5 for further detail on these datasets). As mentioned before, a standard technique to handle such missing values in RNNs, called padding, consists in repeating the hidden state of the last observed value over the missing values. Another standard approach on temporal data is to reconstruct missing values using interpolation. Among the existing interpolation methods, we experimented with three methods from three different categories, namely linear interpolation, non-linear spline interpolation and kernel based Fourier transform interpolation (Meijering, 2002). We measured their reconstruction ability on all four datasets we used for experiments (described in Section 5.1) in terms of MSE, calculated between the interpolated and original datasets. In all the cases, we found that linear interpolation provided the minimum reconstruction error. We thus compare here the use of linear interpolation and padding on RNNs, with and without the attention mechanism. The results obtained, displayed in Figure 3 (left), show that the linear interpolation tends to outperform the padding technique on both datasets¹, particularly on *PSE15*. Furthermore, as before, the attention mechanism brings here no improvement.

Looking at the weights obtained by the attention mechanism (Figure 3 (middle and right)) for both *PSE* and *PW*, one can see that, not surprisingly, the missing input data sometimes get high weights and are primarily used to predict output values, despite their lack of reliability. In particular, for *PW*, the missing values corresponding to the gap comprised in between 12 and 8 months in the history get very high values. This indicates that current RNNs are not entirely robust to missing values as an important part of

¹ We obtained the same results on all collections, with different levels of missing values.

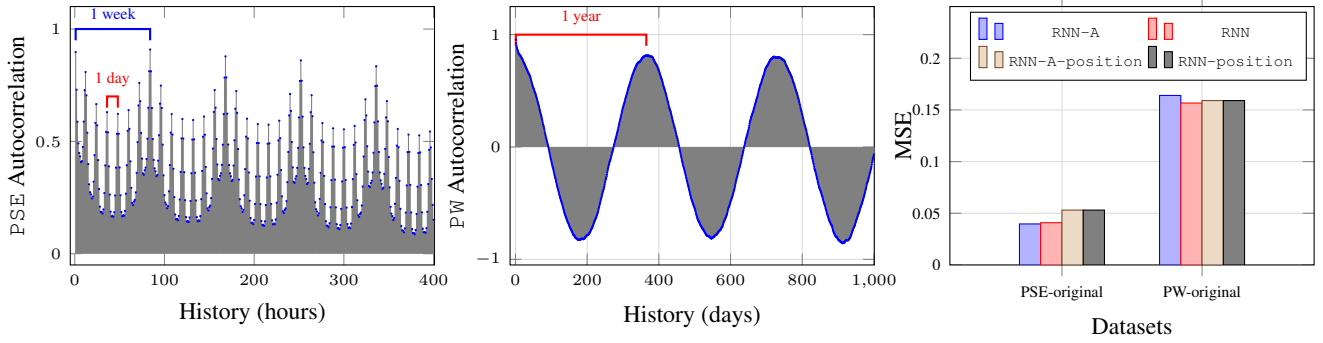


Figure 1. Autocorrelation for PSE and PW. A time span of 5 weeks is considered for PSE (left), and of 2.5 years for PW (middle). Comparison of RNNs with and without attention, and with and without time stamp (or position) information (right).

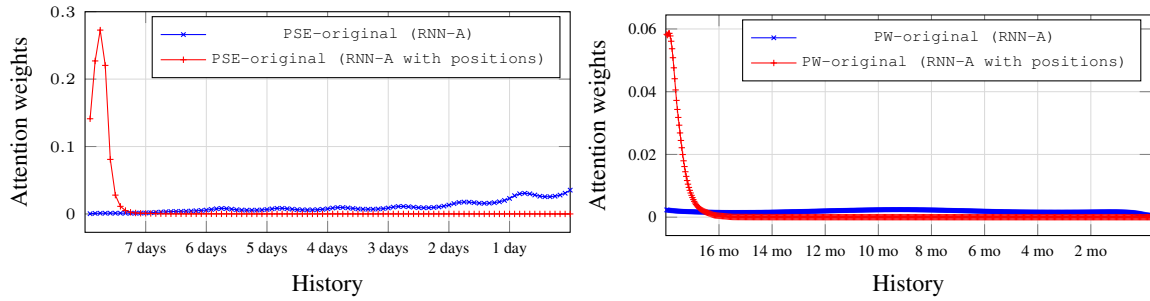


Figure 2. Weights of the attention mechanism for the RNNs without and with time stamp (or position) information on both PSE (left) and PW (right). The weights are averaged over all test examples (see Section 5.3).

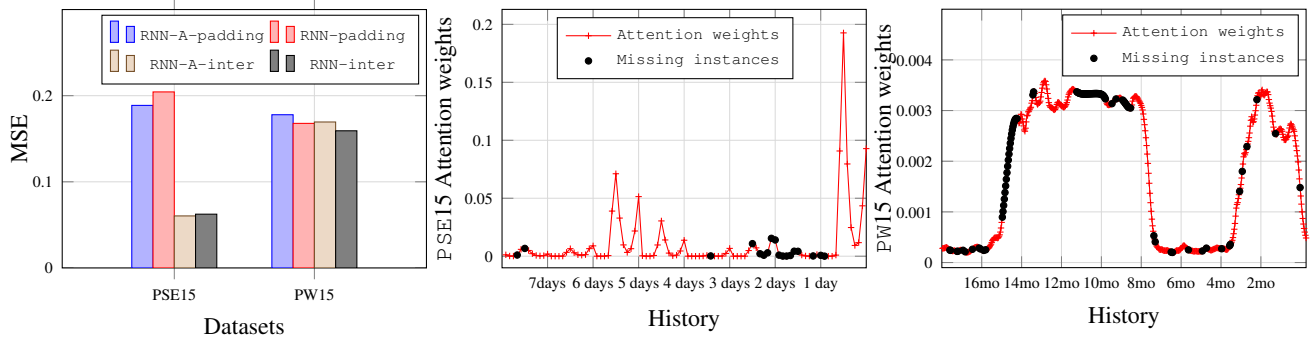


Figure 3. Comparison of padding and linear interpolation on PSE and PW with 15% missing values (left). Examples of attention weights on missing values for PSE (middle) and PW (right).

their decision can be based on them.

3. Extended attention mechanism

We present here two simple extensions of the attention mechanism that allow one to model periods and better handle missing values in RNNs.

3.1. Modeling periodicity

With a history size T and a forecast horizon T' , the possible periods to rely on for prediction range in the set $\{1, \dots, T + T' - 1\}$. We here explicitly model all possible periods throughout a real vector, which we will refer to as τ , of dimension $T + T' - 1$. This vector is used to encode the importance of all possible periods, and to decrease or increase the importance of the corresponding input to predict the current output. To this end, we modify the original attention mechanism to reweigh the attention weights as follows:

$$e_{ij,\tau} = \mathbf{v}_a^T \tanh(\mathbf{W}_a \mathbf{s}_{i-1} + \mathbf{U}_a \mathbf{h}_j) \times (\boldsymbol{\tau}^T \boldsymbol{\Delta}(i, j)) \quad (5)$$

where $\boldsymbol{\Delta}(i, j) \in \mathbb{R}^{T+T'-1}$ is a binary vector that is 1 on dimension $(i - j)$ and 0 elsewhere. We will refer to this model as RNN- τ .

As one can note, $\tau_{(i-j)}$ will either decrease or increase the weight e_{ij} computed by the original attention mechanism. Since τ is learned along with the other parameters of the RNN, we expect that $\tau_{(i-j)}$ will be high for those values of $(i - j)$ that correspond to periods of the time series. We will see in Section 5 that this is indeed the case.

3.2. Handling missing values

Provided their proportion is not too important, otherwise the forecasting task is less relevant, missing values in time series can be easily identified by considering that the most common interval between consecutive points corresponds to the sampling rate of the time series. Points not present at the expected intervals are then considered as missing. This strategy works well in practice, provided the sampling rate of the time series does not change much over time. It is thus possible to identify missing values in time series, and then use padding or interpolation techniques to represent them. Padded or interpolated inputs should however not be treated as standard inputs as they are less reliable than other inputs. Furthermore, be it with padding or interpolation techniques, when the size of a gap, *i.e.* consecutive missing values, is important, the further away the missing value is from the observed values (the last one in case of padding, the last and next ones in case of interpolation), the less confident one is in the padded hidden states or interpolated values. It is thus desirable to decrease the importance of missing values according to their position in gaps.

We propose to do so by another modification of the attention mechanism that again reweighs the attention weights. We consider two reweighing scheme: a first one that penalizes missing values further away from the last observed value through a decaying function, *a priori* adapted to padding, and a second one that penalizes missing values depending on whether they are at the beginning, middle or end of a gap, *a priori* adapted to interpolation methods that rely on values before and after the gap. The first reweighing scheme corresponds to the following equation, that extends Eq. 5 and relies on an exponential decaying function:

$$e_{ij,\tau\mu,1} = e_{ij,\tau} \times f_1(\mu, j) \quad (6)$$

with:

$$f_1(\mu, j) = \begin{cases} \exp(-\mu(j - j_{\text{last}})) & \text{if } j \text{ is missing} \\ 1 & \text{otherwise} \end{cases}$$

where j_{last} denotes the last observed value before j and μ is an additional parameter that is learned with the other parameters of the RNN. $e_{ij,\tau}$ is given by Eq. 5. We will refer to this model as RNN- $\tau\mu-1$.

Similarly, the second reweighing scheme is defined as follows:

$$e_{ij,\tau\mu,2} = e_{ij,\tau} \times (1 + \boldsymbol{\mu}^T \mathbf{Pos}(j; \theta^g)) \quad (7)$$

where $\mathbf{Pos}(j; \theta^g)$ is a three-dimensional vector which is null if j is not missing; otherwise, denoting the length of the gap by $|g|$, if $\frac{j-j_{\text{last}}}{|g|} < \theta_1^g$, then the first coordinate of $\mathbf{Pos}(j; \theta^g)$ is set to 1, if $\theta_1^g < \frac{j-j_{\text{last}}}{|g|} < \theta_2^g$ then the second coordinate is set to 1 and if $\frac{j-j_{\text{last}}}{|g|} > \theta_2^g$ then the third coordinate is set to 1. Note that if only one value is missing, then $|g| = 1$.

$\boldsymbol{\mu}$ is now a three-dimensional vector, learned with the other parameters of the RNN, where each of the coordinates aim at reweighing the impact of missing values on the prediction task according to their position in gaps. The values θ_1^g and θ_2^g are hyper-parameters; they are set here in such a way that they divide each gap g into three equal parts corresponding to the beginning of the gap, its middle and its end. We will refer to this model as RNN- $\tau\mu-2$. As one can note, if there is no gap in the data (*i.e.* no missing values), then both Eq. 6 and Eq. 7 reduce to Eq. 5.

Figure 4 illustrates the modifications brought to the attention mechanism to model periods and handle missing values (the reweighing scheme of RNN- $\tau\mu-1$ is used but can nevertheless be replaced by the one of RNN- $\tau\mu-2$).

4. Multivariate Extension

We now consider the multivariate extension of the above model. As each variable in a K multivariate time series can

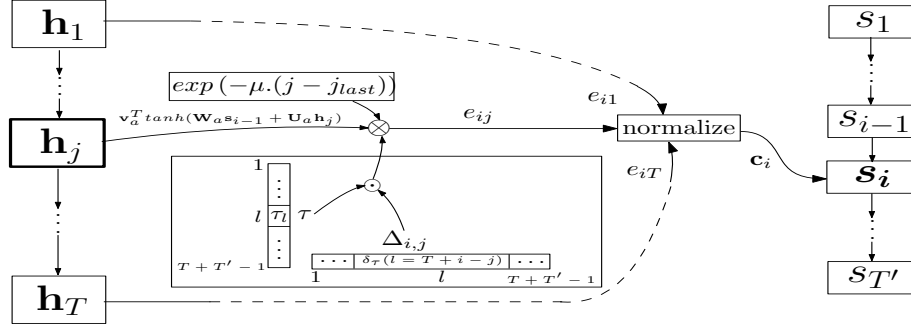


Figure 4. RNN- τ_μ -1 mechanism (Eq. 6). This figure shows the step of predicting the output y_i at time i , given the input history vector $\mathbf{x} = [x_1, \dots, x_j, \dots, x_T]$. For RNN- τ_μ -2, we replace $\exp(\cdot)$ according to Eq. 7.

Table 1. Datasets.

Name	Type	#Instances	History size	Forecast horizon	Sampling rate
Polish Electricity (PSE)	Univariate	46379	96	4	2 hours
Polish Weather (PW)	Multivariate	4595	548	7	1 days
Air Quality (AQ)	Multivariate	9471	192	6	1 hour
Household Power Consumption (HPC)	Multivariate	17294	96	4	2 hours

have its own periods, we propose here to apply the RNN with the extended attention mechanism described above on each variable k , $1 \leq k \leq K$, of the time series. We thus construct, for each variable k , context vectors on the basis of the following equations:

$$\alpha_{ij}^{(k)} = \frac{\exp(e_{ij}^{(k)})}{\sum_{j'=1}^T \exp(e_{ij'}^{(k)})} \quad (8)$$

where $e_{ij}^{(k)}$ is given by Eqs. 3, 5, 6 or 7. The context vector for the i^{th} output of the k^{th} variable is then defined by $\mathbf{c}_i^{(k)} = \sum_{j=1}^T \alpha_{ij}^{(k)} \mathbf{h}_j^{(k)}$, where $\mathbf{h}_j^{(k)}$ is the encoder hidden state at time stamp j for the k^{th} variable.

Lastly, to predict the output in the multivariate case while taking into account potential dependencies between different variables, we concatenate the context vectors from the different variables into a single context vector \mathbf{c}_i that is used as input to the decoder, the rest of the decoder architecture being unchanged:

$$\mathbf{c}_i = [\mathbf{c}_i^{(1)\top} \dots \mathbf{c}_i^{(K)\top}]^\top$$

As each $\mathbf{c}_i^{(k)}$ is of dimension $2n$ (that is the dimension of the encoder hidden states), \mathbf{c}_i is of dimension $2Kn$.

We now turn to the experimental validation of the proposed RNN.

5. Experiments

In this section, we first describe the datasets used in this study and explain the experimental settings prior to presenting the results we obtain using the proposed methods.

5.1. Datasets and settings

We retained four widely used and publicly available (Datasets) that are described in Table 1. The values for the history size were set so as they encompass the known periods of the datasets. They can of course be tuned by cross-validation if one does not want to identify the potential periods by checking the autocorrelation curves. In general, the forecast horizon should reflect the nature of the data and the application one has in mind, with of course a trade off between long forecast horizon and prediction quality. For this purpose, the forecast horizons of PSE, PW, AQ and HPC are chosen as 8 hours, 1 week, 6 hours and 8 hours respectively. In Table 1, one can also find the sampling rate for each dataset.

Note that for PW, we selected the Warsaw metropolitan area which covers only one recording station. For the univariate experiments, from PW we selected the *max temperature* series, from AQ we picked *PT08.S4(NO2)* and from HPC we selected the *global active power* variable.

To implement the RNN models discussed before, we used theano² (Theano Development Team, 2016) and Lasagne³ on a Linux system with 256GB of memory and 32-core Intel Xeon @2.60GHz. All parameters are regularized and learned through stochastic backpropagation with adaptive learning rate for each parameter (Kingma & Ba, 2014), the objective function being the MSE on the output. The hyperparameters we considered for the different RNNs are given in Table 2. To tune the hyperparameters, we performed a two-level grid search, where in the first level we tuned the mini-batch size by using 10 random combinations of

² <http://deeplearning.net/software/theano/>

³ <https://lasagne.readthedocs.io>

the hyperparameters. Within the set $\{1, 32, 64, 128, 256, 512\}$, we retained the mini-batch size of 64 as it performs best. Then, in the second level, we tuned all the hyperparameters shown in Table 2 and picked the top 4 settings according to the RNN model with standard attention mechanism. We use these top 4 settings for all the RNN-based models we are evaluating, selecting the best setting for each model on the validation set. Note that in all those top 4 settings, the learning rate, regularization coefficient and regularization type are the same (shown in the bold face in the upper part Table 2). In other words, the top 4 settings differ in number of attention units and number of RNN units, all the other hyperparameters being the same. Lastly, the objective function for all RNN-based models is the MSE.

In order to assess whether the proposed methods are robust to missing values, we introduced different levels of missing values in the datasets with the following strategy: for each dataset, we added 5%, 10%, 15% and 20% percent of missing values. As the datasets should contain both random missing values and gaps, we introduced half gaps and half random missing values. For instance, to add 10% of missing values, we add 5% of random missing values and 5% of gaps. Furthermore, as the gaps could be of different sizes, we let the length of gaps vary from 5 to 100, picked at random. Note that for HPC and AQ which already contain missing values, we first introduced new missing values until we reach half of the desired percentage, then we introduce the gaps. In our experiments, the percentage of missing values introduced are shown as a number along with the dataset name; the original dataset is named `original`. Also note that the time series AQ already has 5% of missing values, so we discard introducing the 5% missing values and we start with 10%.

We compare the following methods discussed before, namely standard RNN, RNN-A, RNN- τ , RNN- $\tau\mu-1/2$, with gradient boosted trees (GBT) and random forests (RF) as these ensemble methods have been shown to provide state-of-the-art results on various forecasting problems and to outperform moving-average methods (e.g. (Kane et al., 2014a)). We apply grid search over the number of trees $\{500, 1000, 2000\}$ and the number of features $\{n_{\text{features}}, \sqrt{n_{\text{features}}}, \log_2(n_{\text{features}})\}$ for RF, and over learning rate $\{0.01, 0.05, 0.1, 0.25\}$ for GBT. As stated in Section 2, we use linear interpolation and padding for the datasets with missing values.

For evaluation, we use MSE and symmetric mean absolute percentage error (SMAPE). MSE is defined as

$$MSE = \frac{1}{N} \sum_i (y_i - \hat{y}_i)^2$$

and gives the average L_2 distance between the predicted value \hat{y}_i and the true value y_i , where N is the total number

Table 2. Hyperparameters of the grid search.

Hyperparameter	Values
Learning rate	$\{0.01, \mathbf{0.001}, 0.0001\}$
Regularization type	$\{L_1, \mathbf{L_2}\}$
Regularization coeff	$\{0.001, \mathbf{0.0001}, 0.00001\}$
# RNN units	$\{128, 256\}$
# Attention units	$\{128, 256\}$

of predictions, including all horizons. SMAPE is define as

$$SMAPE = \frac{1}{N} \sum_{i=1}^N \frac{|\hat{y}_i - y_i|}{(|y_i| + |\hat{y}_i|)/2}$$

and calculates the L_1 error divided by the average of predicted and true value. SMAPE is a scaled L_1 error.

Lastly, we divided the datasets by retaining the first 75% of each dataset for training-validation and the last 25% for testing. We applied 5-fold cross validation on the training-validation sets for RF and GBT. For RNN-based methods, we divided the training-validation sets into first 75% for training (56.25% of the data) and last 25% for validation (18.75% of the data).

5.2. Overall results on univariate time series

Figure 5 and Figure 6 illustrate the univariate experiments with the interpolation and padding methods, respectively, using the MSE and the SMAPE as evaluation measures. We nevertheless focus our discussion on MSE as it is the metric based on which the problem is optimized.

As one can observe in Figure 5, for the AQ dataset, the three proposed models, *i.e.* RNN- τ and RNN- $\tau\mu-1/2$, yield better MSE results than the other methods. The same conclusion could be drawn from the SMAPE figure of the same dataset. Furthermore, as conjectured in Section 3, RNN- $\tau\mu-2$ tends to perform better than RNN- $\tau\mu-1$ on these interpolated data. This fact can be seen in Figure 6 where the MSE and SMAPE results are illustrated for the padded data. One can observe in Figure 6 that, in general, RNN- $\tau\mu-1$ performs better in terms of MSE, as anticipated in Section 3, and is very close to RNN- $\tau\mu-2$ in terms of SMAPE. Note that, in Figure 6, we removed RF and GBT as padding cannot be used (they are always interpolated with missing values). One can also observe that on the original dataset, with only 5% missing values, RNN- τ has the lowest MSE both in padded and interpolated data.

In Figure 5, similar results, both in terms of MSE and SMAPE, are observed on PSE, with again a slight advantage for RNN- $\tau\mu-2$ over RNN- $\tau\mu-1$, the difference with the other methods being important when the proportion of missing values increases, *e.g.* for PSE20. On the other hand, one can see in Figure 6 that when the data is

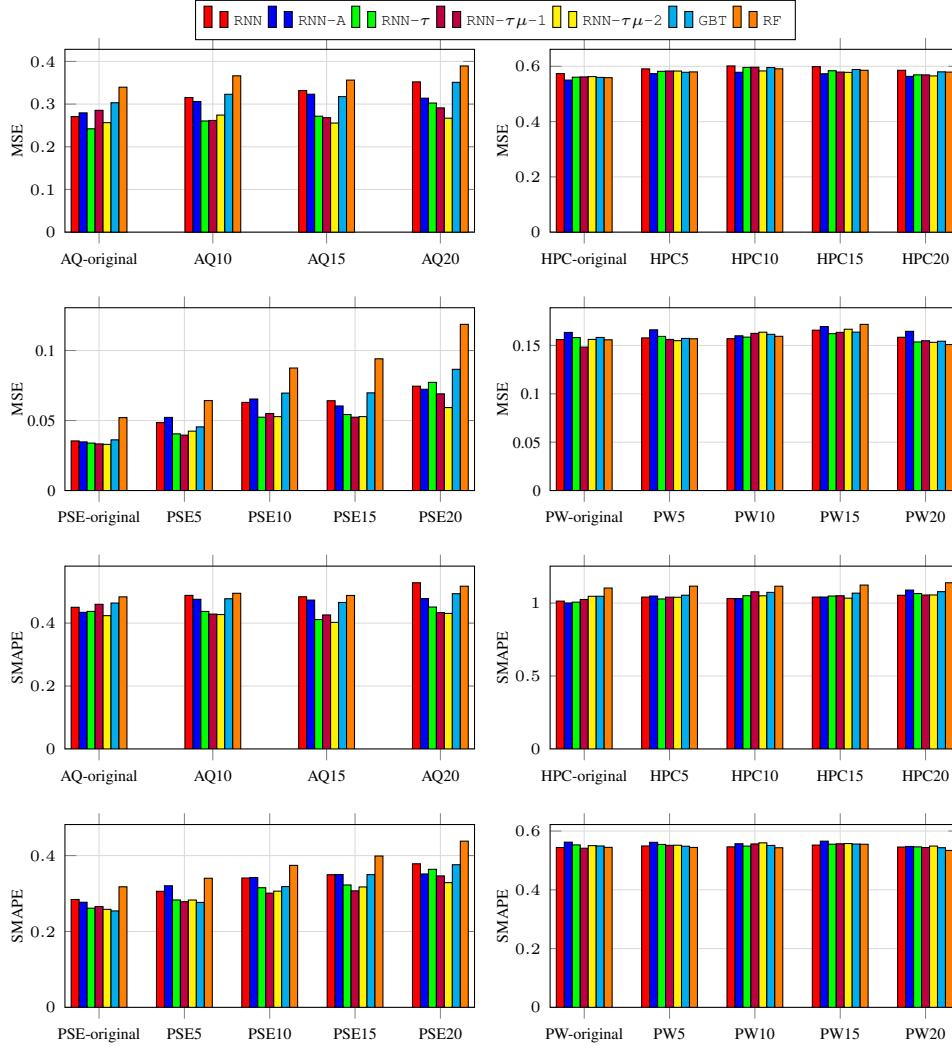


Figure 5. MSE and SMAPE metrics for interpolated data.

padded and there is considerable amount of missing values, *i.e.* PSE20, RNN- $\tau\mu-1$ outperforms RNN- $\tau\mu-2$, both with MSE and SMAPE, as conjectured in Section 3. The baselines, RF and GBT, do not behave well on AQ and PSE, and yield worse results than the RNN-based methods. Moreover, RNN-A yields better results w.r.t. RNN for interpolated data when we have the maximum amount of missing values, both in terms of MSE and SMAPE.

The situation is more contrasted on HPC, where there is no clear difference between the methods for the interpolated data. In this case, the proposed methods are comparable to the standard attention mechanism. When there are many missing values (HPC15 and HPC20), RNN- $\tau\mu-2$ is slightly better than the other methods. However, in case of padded data, all three proposed methods outperform RNN and RNN-A, particularly when we increase the missing values.

Similar to HPC there is no clear difference for PW between the methods in case of interpolated data. However, if we compare the RNN-based methods, we see that except for PW10, the proposed methods outperform RNN and RNN-A, particularly in case of PW20. The situation is a bit better in case of padded data; for instance, there is a clear improvement over the MSE on PW10 and slight improvement on SMAPE on PW15.

5.3. RNN- τ and periodicity

As mentioned before, RNN- τ aims at detecting the periodicity of the data, which is done via the τ vector explained in Section 3.1. Here, we show the effectiveness of this approach by illustrating how the attention weights of the original attention model behave compared to those of RNN- τ . Obviously, we expect RNN- τ to put higher weights on the inputs corresponding to periods. To illustrate this point, we

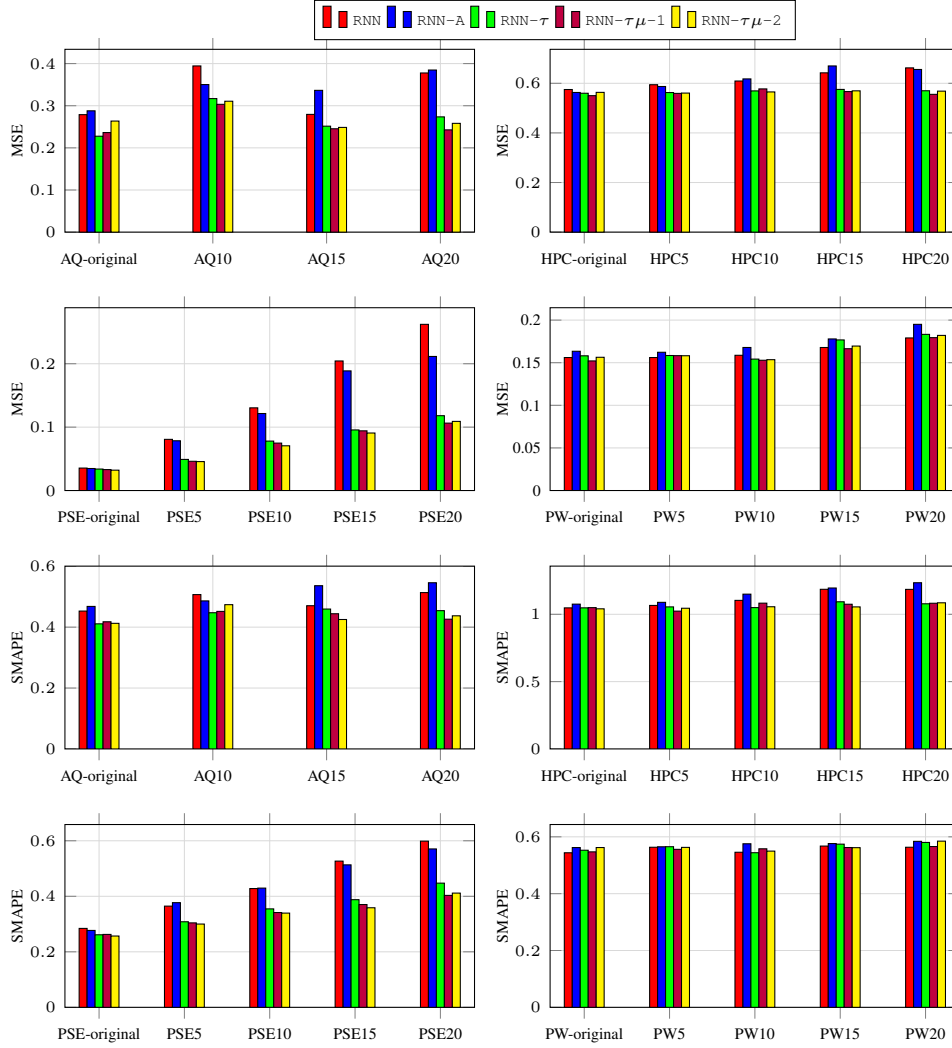


Figure 6. MSE and SMAPE metrics for padded data.

chose the PSE dataset which has two periods, weekly and daily (see Section 2). To observe how RNN- τ behaves, we average all the attention weights of all test examples and all forecast horizons. Figure 7 (left) shows the average attention weights of PSE with RNN-A and RNN- τ . As one can observe, RNN-A fails to capture both periods (1 day and 7 days) while RNN- τ can effectively spot those two periods by giving them higher weights. We also illustrate the *learned* τ values of PSE in Figure 7 (right), which shows that the highest values are assigned to the period of the data, *i.e.* 7 days and 1 day.

5.4. RNN- $\tau\mu$ and missing values

The RNN- $\tau\mu$ -1/2 models are designed to capture periodicity and to prevent the attention mechanism from assigning high weights to the missing instances. As noted before, RNN- $\tau\mu$ -2 with interpolated data is overall the best per-

forming method. To show the effectiveness of this method in handling the missing values, we investigated the attention weights it assigned to missing instances. For each dataset, we averaged the attention weights of RNN- $\tau\mu$ -2 and compared them to those of RNN-A. Figure 8 illustrates the average attention weight and the confidence intervals for all datasets.

As one can observe, in 11 cases out of 15, RNN- $\tau\mu$ -2 provides attention weights which are lower on missing values than those of RNN-A. In average, the attention weights that RNN- $\tau\mu$ put on missing values is **48%** smaller than that of RNN-A, which shows how the μ vector can efficiently take care of missing values inside the data, regardless of its position, by not putting much weight on them.

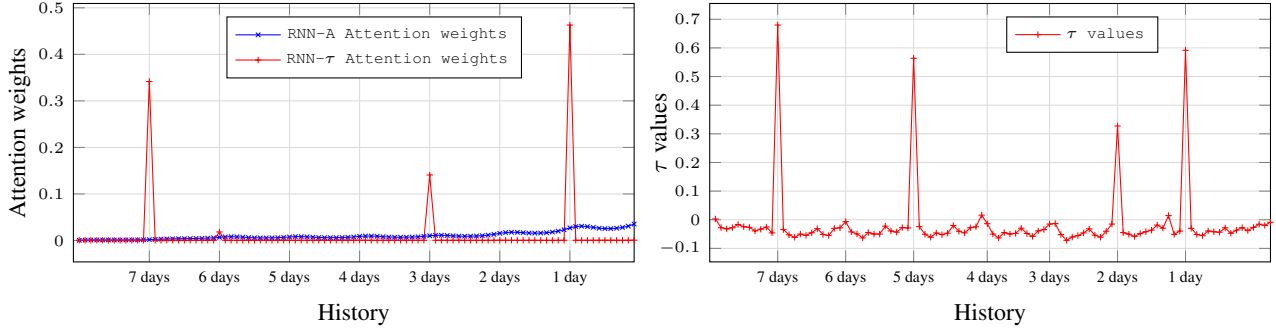


Figure 7. Comparison of averaged attention weights of PSE-original using RNN-A and RNN- τ (left). The τ values learned with RNN- τ method on PSE-original (right).

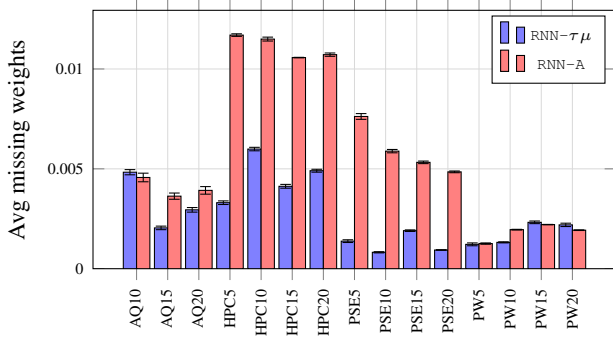


Figure 8. Comparison of average weights of PSE-original using RNN-A and RNN- $\tau\mu$.

5.5. Results on multivariate time series

As mentioned in Section 4, the proposed methods can be extended to multivariate time series. Here, we show how this extension can effectively outperform the state-of-the-art methods as well as the standard attention mechanism. To illustrate that, we pick all four global time series of HPC, namely global active power, global reactive power, voltage and global intensity. We predict the first one which is important for e.g. monthly electricity bills. For AQ, we selected the four variables associated to real sensors (thus excluding nominal sensors), namely C6H6(GT), NO2(GT), CO(GT) and NOx(GT) and predict the first one, C6H6(GT), as it is the most important for NMHC-related air pollution (Vito et al., 2009). Note that this differs from the univariate case in which we focused on the nominal sensor PT08.S4(NO2) to observe the effect of different amounts of missing values and gaps. Also note that all the time series selected from AQ have 18% of missing values; for HPC we consider 5, 10 and 15% of missing values.

Figure 9 (left) shows the results of our experiments on these multivariate series. We only show GBT here since it generally outperforms RF. As one can see in this figure, RNN- τ and RNN- $\tau\mu$ -1/2 provide better results than RNN in all cases. Compared to RNN-A, this is also the true in three cases out of five, the methods being comparable in the last

two cases. In all cases, RNN- τ outperforms GBT and, except for AQ dataset, RNN- $\tau\mu$ -1/2 always provides better results than the other approaches.

Similarly with what is happening on the univariate case, RNN- $\tau\mu$ -1/2 behave better when the amount of missing values increases, with again a *slight* preference for RNN- $\tau\mu$ -2, as conjectured in Section 3. Furthermore, relying on several variables for the prediction improves the results over the univariate case. As illustrated in Figure 9 (right), the results obtained with the multivariate extension are systematically and significantly better than the ones obtained with the univariate case (the same variable is predicted in both cases).

6. Related Work

The notion of stochasticity of time series modeling and prediction was introduced long back (Yule, 1927). Since then various stochastic models have been developed, notable among these are autoregressive (AR) (Walker, 1931) and moving averages (MA) (Slutzky, 1937) methods. These two models were combined in a more general and more effective framework, known as autoregressive moving average (ARMA), or autoregressive integrated moving average (ARIMA) when the differencing is included in the modeling (Box & Jenkins, 1968). Vector ARIMA or VARIMA (Tiao & Box, 1981) is the multivariate extension of the univariate ARIMA models where each time series instance is represented using a vector.

Neural networks are regarded as a promising tool for time series prediction (Zhang, 2001; Crone et al., 2011) due to their data-driven and self-adaptive nature, their ability to approximate any continuous function and their inherent non-linearity. The idea of using neural networks for prediction dates back to 1964 where an adaptive linear network was used for weather forecasting (Hu, 1964). But the research was quite limited due to the lack of a training algorithm for general multilayer networks at the time. Since the introduction of backpropagation algorithm (Rumelhart

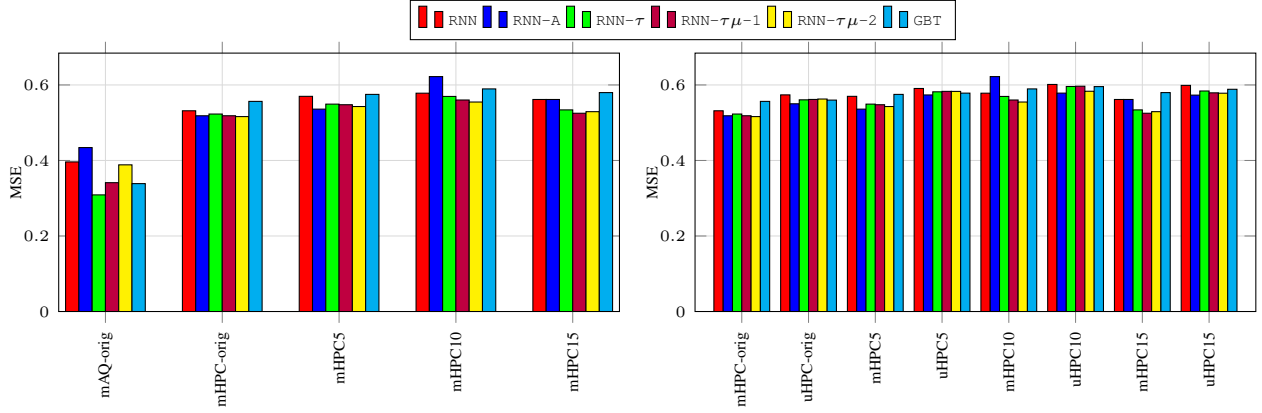


Figure 9. Multivariate extension of the proposed methods on AQ and HPC (left). Comparison of multivariate and univariate time series prediction (right). Multivariate results prefixed with **m** and univariates with **u**.

et al., 1988), there had been much development in the use of neural networks for forecasting. It was shown in a simulated study that neural networks can be used for modeling and forecasting nonlinear time series (Laepes & Farben, 1987). Several studies (Werbos, 1988; Sharda & Patil, 1990) have found that neural networks are able to outperform the traditional stochastic models such as ARIMA or Box-Jenkins approaches (Box & Jenkins, 1968).

With the advent of SVM, it has been used to formulate time series prediction as a regression estimation using Vapnik’s insensitive loss function and Huber’s loss function (Müller et al., 1997). The authors showed that SVM based prediction outperforms traditional neural network based methods. Since then different versions of SVMs are applied for time series prediction and many different SVM forecasting algorithms have been derived (Raicharoen et al., 2003; Fan et al., 2006). More recently, random forests (Breiman, 2001) are used for time series predictions due to their performance accuracy and ease of execution. Random forest regression is used for prediction in the field of finance (Creamer & Freund, 2004) and bioinformatics (Kusiak et al., 2013), and are shown to outperform ARIMA (Kane et al., 2014b).

Traditional neural networks allow only feedforward connections among the neurons of a layer and the neurons in the following layer. In contrast, *recurrent neural networks* (RNN) (Jordan, 1986; Elman, 1990) allow both forward and feedback or recurrent connections between the neurons of different layers. Hence, RNNs are able to incorporate contextual information from past inputs which makes them an attractive choice for predicting general sequence-to-sequence data, including time series (Graves, 2013). In this present study we use RNNs for modeling time series. Early work (Connor et al., 1991) has shown that RNNs (a) are a type of nonlinear autoregressive moving average (NARMA) model and (b) outperform feedforward networks and various types of linear statistical models

on time series. Subsequently, various RNN-based models were developed for different time series, as noisy foreign exchange rate prediction (Giles et al., 2001), chaotic time series prediction in communication engineering (Jaeger & Haas, 2004) or stock price prediction (Hsieh et al., 2011). A detailed review can be found in (Långkvist et al., 2014) on the applications of RNNs along with other deep learning based approaches for different time series prediction tasks.

RNNs based on LSTMs (Hochreiter & Schmidhuber, 1997), which we consider in our study, alleviate the *vanishing gradient* problem of the traditional RNNs proposed. They have furthermore been shown to outperform traditional RNNs on various temporal tasks (Gers et al., 2001; 2002). More recently, they have been used for predicting the next frame in a video and for interpolating intermediate frames (Ranzato et al., 2014), for forecasting the future rainfall intensity in a region (Xingjian et al., 2015), or for modeling clinical data consisting of multivariate time series of observations (Lipton et al., 2015).

Adding attention mechanism on the decoder side of an encoder-decoder RNN framework enabled the network to focus on the interesting parts of the encoded sequence (Bahdanau et al., 2014). Attention mechanism is used in many applications such as image description (Xu et al., 2015), image generation (Karol et al., 2015), phoneme recognition (Chorowski et al., 2015), heart failure prediction (Choi et al., 2016), as well as time series prediction (Riemer et al., 2016) and classification (Choi et al., 2016). Many studies also apply attention mechanism on external memory (Graves et al., 2014; 2016).

The previous work (Riemer et al., 2016) uses attention mechanism to determine the importance of a factor among other factors that affect time series. However, we use extended attention mechanism to model period and emphasize the interesting parts of input sequence with missing values. Our approach is applicable for both univariate and

multivariate time series prediction.

However, as far we know, no work has focused on analyzing the adequacy of RNNs (based or not on LSTMs) for time series, in particular w.r.t. their ability to model periods and handle missing values (for this latter case, some methods based on *e.g.* Gaussian processes have been proposed to handle irregularly sampled data and missing values (Ghassemi et al., 2015), but none related to RNNs). To our knowledge, this study is the first one to address this problem.

7. Conclusion

In this paper we studied the abilities of RNNs for modeling and forecasting time series. We used state-of-the-art RNNs based on bidirectional LSTM encoder-decoder with attention mechanism, and illustrated their deficiencies in capturing the periodicity of the data and in handling properly missing values. To alleviate this, we proposed two architectural modifications over the traditional attention mechanism: a first one to learn and exploit the periodicity in the temporal data (RNN- τ) and a second one to handle both random missing values and long gaps in the data (RNN- $\tau\mu$ -1/2). We further extended the entire framework for multivariate time series.

The experiments we conducted over multiple univariate and multivariate time series demonstrate the effectiveness of these modifications. RNN- τ and RNN- $\tau\mu$ -1/2 are able to perform not only better than traditional RNNs and RNNs with original attention mechanism, but also better than two state-of-the-art baselines based on random forests and gradient boosted trees. Furthermore, by design, RNN- τ can capture several periods in a time series and make use of all of them to forecast new values. We provided experimental illustration to this fact on a time series with both weekly and daily periods. Regarding missing values, the two variants we propose, RNN- $\tau\mu$ -1/2, are well adapted to different techniques used in RNNs to handle missing values: padding and interpolation. We showed that these variants rely more on actual values and less on padded or interpolated values than the original RNNs, thus making the proposed RNN more robust to missing values.

References

- Bahdanau, Dzmitry, Cho, Kyunghyun, and Bengio, Yoshua. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.
- Bontempi, Gianluca, Taieb, Souhaib Ben, and Le Borgne, Yann-Aël. Machine learning strategies for time series forecasting. In *Business Intelligence*, pp. 62–77. Springer, 2013.
- Box, George EP and Jenkins, Gwilym M. Some recent advances in forecasting and control. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 17(2): 91–109, 1968.
- Breiman, Leo. Random forests. *Machine learning*, 45(1): 5–32, 2001.
- Chatfield, Chris. *The analysis of time series: an introduction*. CRC press, 2016.
- Choi, Edward, Bahadori, Mohammad Taha, Sun, Jimeng, Kulas, Joshua, Schuetz, Andy, and Stewart, Walter. Retain: An interpretable predictive model for healthcare using reverse time attention mechanism. In *Advances in Neural Information Processing Systems*, pp. 3504–3512, 2016.
- Chorowski, Jan K, Bahdanau, Dzmitry, Serdyuk, Dmitriy, Cho, Kyunghyun, and Bengio, Yoshua. Attention-based models for speech recognition. In *Advances in Neural Information Processing Systems*, pp. 577–585, 2015.
- Connor, Jerome, Atlas, Les E, and Martin, Douglas R. Recurrent networks and narma modeling. In *NIPS*, pp. 301–308, 1991.
- Creamer, Germán G and Freund, Yoav. Predicting performance and quantifying corporate governance risk for latin american adrs and banks. 2004.
- Crone, Sven F, Hibon, Michele, and Nikolopoulos, Konstantinos. Advances in forecasting with neural networks? empirical evidence from the nn3 competition on time series prediction. *International Journal of Forecasting*, 27(3):635–660, 2011.
- Datasets.
PSE: <http://www.pse.pl>.
PW: <https://globalweather.tamu.edu>.
AQ: <http://archive.ics.uci.edu>.
HPC: <http://archive.ics.uci.edu/>.
- De Gooijer, Jan G and Hyndman, Rob J. 25 years of time series forecasting. *International journal of forecasting*, 22(3):443–473, 2006.
- Elman, Jeffrey L. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.
- Fan, Yugang, Li, Ping, and Song, Zhihuan. Dynamic least squares support vector machine. In *Intelligent Control and Automation, 2006. WCICA 2006. The Sixth World Congress on*, volume 1, pp. 4886–4889. IEEE, 2006.

- Gers, Felix A, Eck, Douglas, and Schmidhuber, Jürgen. Applying lstm to time series predictable through time-window approaches. In *International Conference on Artificial Neural Networks*, pp. 669–676. Springer, 2001.
- Gers, Felix A, Schraudolph, Nicol N, and Schmidhuber, Jürgen. Learning precise timing with lstm recurrent networks. *Journal of machine learning research*, 3(Aug): 115–143, 2002.
- Ghassemi, Marzyeh, Pimentel, Marco AF, Naumann, Tristan, Brennan, Thomas, Clifton, David A, Szolovits, Peter, and Feng, Mengling. A multivariate timeseries modeling approach to severity of illness assessment and forecasting in icu with sparse, heterogeneous clinical data. In *Proceedings of the... AAAI Conference on Artificial Intelligence. AAAI Conference on Artificial Intelligence*, volume 2015, pp. 446. NIH Public Access, 2015.
- Giles, C Lee, Lawrence, Steve, and Tsoi, Ah Chung. Noisy time series prediction using recurrent neural networks and grammatical inference. *Machine learning*, 44(1-2): 161–183, 2001.
- Graves, Alex. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.
- Graves, Alex, Wayne, Greg, and Danihelka, Ivo. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014.
- Graves, Alex, Wayne, Greg, Reynolds, Malcolm, Harley, Tim, Danihelka, Ivo, Grabska-Barwińska, Agnieszka, Colmenarejo, Sergio Gómez, Grefenstette, Edward, Ramalho, Tiago, Agapiou, John, et al. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538(7626):471–476, 2016.
- Hochreiter, Sepp and Schmidhuber, Jürgen. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
- Hsieh, Tsung-Jung, Hsiao, Hsiao-Fen, and Yeh, Wei-Chang. Forecasting stock markets using wavelet transforms and recurrent neural networks: An integrated system based on artificial bee colony algorithm. *Applied soft computing*, 11(2):2510–2525, 2011.
- Hu, Michael Jen-Chao. *Application of the adaline system to weather forecasting*. 1964.
- Jaeger, Herbert and Haas, Harald. Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *science*, 304(5667):78–80, 2004.
- Jordan, Michael I. Serial order: A parallel distributed processing approach. Technical Report 8604, Institute for Cognitive Science, University of California, 1986.
- Kane, Michael J, Price, Natalie, Scotch, Matthew, and Rabinowitz, Peter. Comparison of arima and random forest time series models for prediction of avian influenza h5n1 outbreaks. *BMC bioinformatics*, 15(1):276, 2014a.
- Kane, Michael J, Price, Natalie, Scotch, Matthew, and Rabinowitz, Peter. Comparison of arima and random forest time series models for prediction of avian influenza h5n1 outbreaks. *BMC bioinformatics*, 15(1):276, 2014b.
- Karol, G, Danihelka, I, Graves, A, Rezende, D, and Wierstra, D. Draw: a recurrent neural network for image generation. ICML, 2015.
- Kingma, Diederik and Ba, Jimmy. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- Kusiak, Andrew, Verma, Anoop, and Wei, Xiupeng. A data-mining approach to predict influent quality. *Environmental monitoring and assessment*, 185(3):2197–2210, 2013.
- Laepes, A and Farben, R. Nonlinear signal processing using neural networks: prediction and system modelling. Technical report, Technical report, Los Alamos National Laboratory, Los Alamos, NM, 1987.
- Längkvist, Martin, Karlsson, Lars, and Loutfi, Amy. A review of unsupervised feature learning and deep learning for time-series modeling. *Pattern Recognition Letters*, 42:11–24, 2014.
- Lipton, Zachary C, Kale, David C, Elkan, Charles, and Wetzell, Randall. Learning to diagnose with lstm recurrent neural networks. *arXiv preprint arXiv:1511.03677*, 2015.
- Meijering, Erik. A chronology of interpolation: From ancient astronomy to modern signal and image processing. *Proceedings of the IEEE*, 90(3):319–342, 2002.
- Müller, K-R, Smola, Alexander J, Rätsch, Gunnar, Schölkopf, Bernhard, Kohlmorgen, Jens, and Vapnik, Vladimir. Predicting time series with support vector machines. In *International Conference on Artificial Neural Networks*, pp. 999–1004. Springer, 1997.
- Raicharoen, T, Lursinsap, C, and Sanguanbhoki, P. Application of critical support vector machine to time series prediction. circuits and systems. iscas03. In *Proceedings of the 2003 International Symposium on*, volume 5, pp. 25–28, 2003.
- Ranzato, MarcAurelio, Szlam, Arthur, Bruna, Joan, Mathieu, Michael, Collobert, Ronan, and Chopra, Sumit. Video (language) modeling: a baseline for generative models of natural videos. *arXiv preprint arXiv:1412.6604*, 2014.

- Riemer, Matthew, Vempaty, Aditya, Calmon, Flavio P, Heath III, Fenno F, Hull, Richard, and Khabiri, Elham. Correcting forecasts with multifactor neural attention. In *Proceedings of The 33rd International Conference on Machine Learning*, pp. 3010–3019, 2016.
- Rumelhart, David E, Hinton, Geoffrey E, and Williams, Ronald J. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1, 1988.
- Schuster, Mike and Paliwal, Kuldip K. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.
- Sharda, Ramesh and Patil, R. Neural networks as forecasting experts: an empirical test. In *Proceedings of the International Joint Conference on Neural Networks*, volume 2, pp. 491–494. IEEE, 1990.
- Slutzky, Eugen. The summation of random causes as the source of cyclic processes. *Econometrica: Journal of the Econometric Society*, pp. 105–146, 1937.
- Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May 2016. URL <http://arxiv.org/abs/1605.02688>.
- Tiao, George C and Box, George EP. Modeling multiple time series with applications. *Journal of the American Statistical Association*, 76(376):802–816, 1981.
- Vito, Saverio De, Piga, Marco, Martinotto, Luca, and Francia, Girolamo Di. Co, {NO₂} and {NO_x} urban pollution monitoring with on-field calibrated electronic nose by automatic bayesian regularization. *Sensors and Actuators B: Chemical*, 143(1):182 – 191, 2009. ISSN 0925-4005. doi: <http://dx.doi.org/10.1016/j.snb.2009.08.041>. URL <http://www.sciencedirect.com/science/article/pii/S092540050900673X>.
- Walker, Gilbert. On periodicity in series of related terms. *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character*, 131(818):518–532, 1931.
- Werbos, Paul J. Generalization of backpropagation with application to a recurrent gas market model. *Neural networks*, 1(4):339–356, 1988.
- Xingjian, SHI, Chen, Zhouong, Wang, Hao, Yeung, Dityan, Wong, Wai-Kin, and Woo, Wang-chun. Convolutional lstm network: A machine learning approach for precipitation nowcasting. In *Advances in Neural Information Processing Systems*, pp. 802–810, 2015.
- Xu, Kelvin, Ba, Jimmy, Kiros, Ryan, Cho, Kyunghyun, Courville, Aaron C, Salakhutdinov, Ruslan, Zemel, Richard S, and Bengio, Yoshua. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*, volume 14, pp. 77–81, 2015.
- Yule, G Udny. On a method of investigating periodicities in disturbed series, with special reference to wolfer’s sunspot numbers. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 226:267–298, 1927.
- Zhang, Guoqiang Peter. An investigation of neural networks for linear time-series forecasting. *Computers & Operations Research*, 28(12):1183–1202, 2001.