Interpretable Multivariate Time Series Forecasting with Temporal Attention Convolutional Neural Networks

Pantiskas, Leonardos; Verstoep, C.; Bal, Henri

published in
2020 IEEE Symposium Series on Computational Intelligence (SSCI)
2021

DOI (link to publisher)
10.1109/SSCI47803.2020.9308570

document version
Peer reviewed version

DOI: 10.1109/SSCI47803.2020.9308570 Published version: https://ieeexplore.ieee.org/document/9308570

Interpretable Multivariate Time Series Forecasting with Temporal Attention Convolutional Neural Networks

Leonardos Pantiskas
Dept. of Computer Science
Vrije Universiteit
Amsterdam, The Netherlands
l.pantiskas@vu.nl

Kees Verstoep
Dept. of Computer Science
Vrije Universiteit
Amsterdam, The Netherlands
c.verstoep@vu.nl

Henri Bal
Dept. of Computer Science
Vrije Universiteit
Amsterdam, The Netherlands
h.e.bal@vu.nl

***Abstract*—Data in time series format, such as biological signals from medical sensors or machine signals from sensors in industrial environments are rich sources of information that can give crucial insights on the present and future condition of a person or machine. The task of predicting future values of time series has been initially approached with simple machine learning methods, and lately with deep learning. Two models that have shown good performance in this task are the temporal convolutional network and the attention module. However, despite the promising results of deep learning methods, their black-box nature makes them unsuitable for real-world applications where the predictions need to be explainable in order to be trusted. In this paper we propose an architecture comprised of a temporal convolutional network with an attention mechanism that makes predictions while presenting the timesteps of the input that were most influential for future outputs. We apply it on two datasets and we show that we gain interpretability without degrading the accuracy compared to the original temporal convolutional models. We then go one step further and we combine our configuration with various machine learning methods on top, creating a pipeline that achieves interpretability both across timesteps and input features. We use it to forecast a different variable from one of the above datasets and we study how the accuracy is affected compared to the original black-box approach.**

***Index Terms*—forecasting, multivariate time series, interpretability, attention mechanism, temporal convolutional neural network**

I. Introduction

Time series data are pervasive in many areas in life, such as health, industry and finance. Moreover, with the further diffusion of Internet of Things and 5G, the amount of data collected will increase dramatically, creating new opportunities for knowledge extraction and applications [1].

One important task with respect to time series is the forecasting of their future values, enabling planning of future actions. By forecasting, we mean that given the past values of one or more time series as input, we try to predict the future values of one or more time series. When we have more than one time series as input or output, we refer to the problem as multivariate forecasting. The forecasting output can be the future values of the input time series or it can be the future values of an independent time series which is correlated with the input variables.

Lately, deep learning (DL) methods have been increasingly employed to tackle the above task. Some models have emerged as the de facto approach for processing sequences, namely recurrent neural networks (RNNs) and their more complex variants such as long short-term memory (LSTM) networks [2]. However, recently Bai et al. [3] brought focus on an approach named temporal convolutional network (TCN), showing that it holds promising potential in sequence modeling, while dealing with some of the issues of the recurrent networks. Another approach comes from the natural language sequence processing area with the attention mechanism [4], originally used to focus on the most important parts of an input sentence in a translation task. This model has also been applied to time series forecasting and showed promising performance [5].

Although the interest in machine learning (ML) and DL research for time series forecasting has been increasing, there is still a debate about whether more complex models perform better than simpler models or well-studied statistical methods [6], [7]. As an example, the winner of the 2018 M4 forecasting competition used a hybrid DL - statistical model and the majority of the top accurate submissions were combinations of methods [8]. This leads Makridakis et al. to conclude that the combination of methods is a promising direction for forecasting. In addition, in [6], the same authors find that DL models such as RNNs and LSTMs achieve lower accuracy than most simple statistical methods. In the same work, two more points are stressed: the need for users of complex models to understand how their predictions arise and the importance of providing confidence intervals for the predictions of a model.

Since ML and DL models are being used more extensively in areas where they can drive crucial decisions with significant impact, such as healthcare and industry, they should provide justification for their outputs, something that is also pushed by regulations [9]. In addition, the interpretability of models can enable additional applications. For example, feature importance attribution can lead to knowledge and insight extraction from data [10], as well as reduction of uninformative or noisy

sources of data. These factors have driven researchers to come up with methods for explaining opaque models, as well as strive for stricter definitions surrounding interpretability [11]–[13]. Although methods such as SHAP [14], which calculate feature importance, have become popular for explaining complex black-box models, the majority are after-the-fact explanations of inherently closed models, which can present challenges. Instead, we need models that are explainable by design and offer sufficient basis about their own predictions [15].

Inspired by the above concepts, we present an interpretable method for multivariate time series forecasting using deep learning. Our key contributions are:

- An architecture that provides per instance visualization of the focus of a TCN on the input time series and highlights the important timesteps for the production of each step of the output. The configuration presented is modular, inherently interpretable, comprised of already known DL units and is kept as simple as possible, in the spirit of keeping the model complexity low as well.
- A demonstration of the ability of the above architecture to handle multiple time series of different type in a multitask problem fashion.

The architecture is evaluated on two multivariate datasets, regarding Air Quality [16] and Water Quality variables, of which the latter has been approached with a TCN in [17]. It is important to note that our suggested architecture does not employ a new model built from scratch, nor do we try to achieve state of the art in the datasets under consideration. Instead, the goal is to integrate interpretability to an existing TCN model with as few changes as possible and without significantly affecting the model accuracy. For that purpose we do not do any feature selection or transformation, or hyperparameter tuning for our method, in order to present as fair metrics as possible.

Moreover, we combine our DL architecture with inherently interpretable ML methods on top, creating a pipeline that is shown fully in Fig. 1. In this, the DL part predicts the future values of the inputs, which are then used as features to the ML methods to predict the final target variable. We apply this pipeline to the Water Quality dataset, choosing one of its variables (dissolved oxygen) as the target output. Although this DL-ML pipeline is not necessary for our suggested mechanism to function, it gives valuable insights for our work. First, we measure the effect on accuracy for an applied real-world use case while introducing importance attribution both across timesteps and features. Moreover, we get an implicit estimation of the forecasting accuracy of our model for the intermediate outputs of the variables whose metrics were not in the baseline, such as pH. Finally, as we discuss later, we verify the validity of the attention distribution patterns.

The rest of the paper is organized as follows: Section 2 presents related work on the main concepts of our approach. Section 3 presents our proposed architecture. Section 4 presents details about the datasets, preprocessing and implementation of the model, as well as the results. Finally, in section 5 we conclude the paper and discuss future directions.

II. Related Work

A. Multivariate Time Series Forecasting

Due to the increasing research interest in DL, more and more models are being proposed that partially or fully utilize DL components for multivariate time series forecasting [5]. The DL models range from simple versions of recurrent neural networks, such as LSTMs, to complex approaches combining the above elements with fully connected neural networks and convolutional neural networks, such as LSTNet [18] and DeepAR [19]. The majority of those works however does not touch at all upon the issue of interpretability of the model.

Temporal Convolutional Networks: Although there have been works that have used 1-dimensional convolutions as components of the overall model, here we focus on works utilizing the general architecture referred to as temporal convolutional network by Bai et al. in [3], with the distinctive elements of residual blocks, 1-d causal convolutions and dilation of the convolution filter across levels. In [20], Borovykh et al. utilize the aforementioned elements to create a network for one-step ahead forecast of financial time series, while in [21], a TCN block has been used as a component of the proposed multi-head model. In [17], which is the work using the Water Quality data, a TCN is used to predict dissolved oxygen and temperature simultaneously. Our approach can forecast multiple variables of different type, fully exploiting the potential of the TCN model.

B. Attention Mechanism

The family of attention mechanisms has been introduced mainly in relation to natural language processing tasks [4], but has evolved beyond this domain into the rest of the DL areas. Regarding multivariate time series forecasting, in [22] the authors propose an encoder-decoder LSTM framework with an intermediate multimodal attention mechanism which focuses on different input periods. In [23] the authors present a convolutional filter based attention mechanism that selects the most relevant steps to pass to a subsequent RNN cell. Other approaches include [24], where the authors apply a Transformer architecture [25] to time series forecasting and [26], where a self-attention layer serves for the learning of temporal patterns by the model. The attention mechanism in hierarchical format has also been used to differentiate between focus across timesteps and across inputs, in works such as [27] and [28]. Although the model performance may be benefited, in the majority of the works the attention mechanism is not used to give insights about the model predictions. In addition, in the works above, the attention module is an intermediate component which outputs a "context" vector which is then usually fed to the rest of the network. Thus, the attention output is removed from the final output by a number of complex stages and its value is not immediately obvious. In contrast, the prediction and the explanation in our approach come directly from the attention mechanism, making them

tightly coupled, as we will see in subsection III-B. Another important advantage is the beneficial match of the attention mechanism with the large receptive field of the TCN [3], which further enables the network to take into account timesteps deep into the input past. Finally, the importance of the input features in the case of our full pipeline is achieved through well-studied machine learning methods.

C. Interpretability Through Visualization Methods

In the time series classification domain, there has been significant work when it comes to visualizing important parts of the input, resulting in methods such as class activation mapping [29]. In the medical domain we see examples such as [30] and [31], where the weights of attention mechanisms are used to indicate the importance of the various input features in each time step for the corresponding classification tasks. When it comes to forecasting however, there has been little work. Two notable examples are [26] and [24], where the visualization of the learned attention weights can indicate persistent temporal patterns. Our approach gives instead per instance intuitive visualization of the focus of the network on the input for each time step of the output. This provides a fine-grained level of detail, which allows us to understand when the network succeeds, and equally importantly, when the network fails to meaningfully focus on the input.

III. Proposed configuration

Our suggested architecture is shown in Fig. 1. The main DL pipeline, which is the configuration applied to both datasets is in the dotted frame (left), while the end-to-end topology that is used for the extended analysis of the Water Quality variable includes an ML pipeline (dashed frame, right).

A. Temporal Convolutional Network Block

The main building block of the DL part of the model is the TCN residual block, as seen in Fig. 2. The structure of the block is similar to [3] and is mainly based on the 1-D causal dilated convolution. This operation means that a filter f with size k ($f \in \mathbb{R}^k$), bias b and dilation factor d is convolved with a 1-D input $x \in \mathbb{R}^L$ to produce an output $x' \in \mathbb{R}^L$, where each element is calculated as:

$$x'_i = \sum_{j=0}^{k-1} f_j \cdot x_{i-d*j} + b \tag{1}$$

In the cases where $i - d * j < 0$ we consider $x_{i-d*j} = 0$. An example of 1-d convolutions with different dilations can be seen in Fig. 3. In each Conv1D layer, a number of F filters is slid across each of the N 1-D sequences of length L in the above manner, producing initially an $N \times L \times F$ matrix. This result is then summed across the dimension of the N inputs, producing an $L \times F$ matrix. Weight normalization [32] is applied on the weights of the Conv1D layers and a channel-wise dropout layer is added for training regularization [33]. This means that during the training phase of the model, each of the F dimensions of the Conv1D output is zeroed out with dropout probability p before being forwarded to the next layer. Finally, a rectified linear unit (ReLU) activation is applied to each of the matrix elements [34]. The residual connection on the right branch of the block is comprised of another weight normalized Conv1D layer. The outputs of the two branches are summed and passed through the final activation function of the block, which is linear for all blocks apart from the last block of the stack which uses ReLU activation.

B. Attention Mechanism

The attention module that we use on top of the TCN to produce the forecast and the interpretation is based on the scaled dot-product attention as described in [25]. The intuition of the attention mechanism is that a Value matrix is transformed based on how a Query and Key matrix match. In our example, the TCN produces the query, while the key-value pair are different transformations of the input. The output O of the block is a function of the Query, Key and Value inputs in this fashion:

$$O = D \cdot V = softmax(\frac{QK^T}{\sqrt{L}}) \cdot V \tag{2}$$

According to (2), the dimension of O is equal to that of Q, and in our case we want to output h predictions for each of the N input time series, i.e. a $O^{N \times h}$ matrix. Thus, we scale down the output of the last TCN block from $L \times F$ to $L \times N$ using a weight normalized Conv1D layer with N filters. Then we apply a dense layer to the transpose of the result to form the query $Q^{N \times h}$. The key and value are formed as follows:

$$\begin{aligned} K^{N \times 1 \times L} &= I^{N \times 1 \times L} \cdot W_K^{N \times L \times L} + b_K^{N \times 1 \times L} \\ V^{N \times 1 \times L} &= I^{N \times 1 \times L} \cdot W_V^{N \times L \times L} \end{aligned} \tag{3}$$

where I is the original time series input and W_K, b_K and W_V are trainable matrices.

Attention Weights Intuition: In order to understand the intuition behind the attention visualization, we can focus on only one of the N time series, ignoring this dimension of the matrices, without loss of generality. As a consequence of (3), we see that each element I_i of the time series contributes to each element of the value matrix V according to the weights of the i_{th} row of the matrix W_V. In the same fashion, each element of the value matrix V contributes to each element j of the final output O according to the j_{th} row of matrix D of (2). This means that we can produce a matrix $A \in \mathbb{R}^{h \times L}$ as so:

$$A^{h \times L} = D^{h \times L} \cdot abs(W_V^{L \times L})^T \tag{4}$$

Each row j of A shows how each output O_j is affected by each original input I_i. We select the absolute values of the weights to portray the attention based on the scale and not the sign of the weights, but this is an implementation choice and can be changed based on the needs of the application.

C. Machine Learning Methods

For the ML part of our analysis, we have chosen to utilize and compare three well known ML methods which provide different forms of interpretability, which are briefly presented

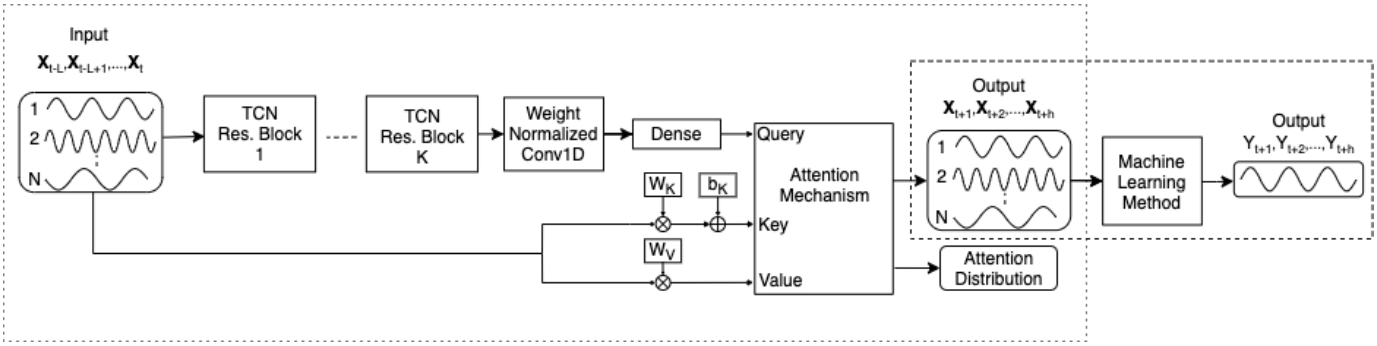


Fig. 1. Proposed configuration

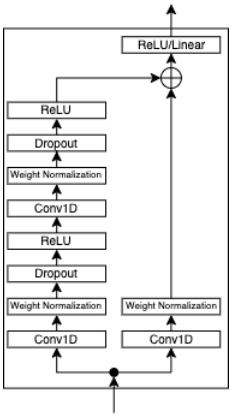


Fig. 2. TCN block structure

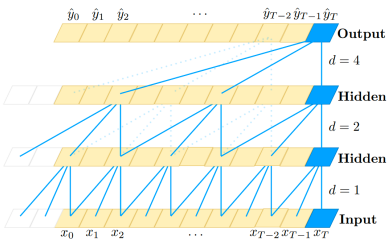


Fig. 3. Dilated convolutions (Source: [3])

in order of increasing complexity. The full description of these ML methods and the presentation of their detailed statistical properties is out of the scope of this paper, but interested readers can follow relevant sources [35].

1) Linear Regression: One of the simplest methods to predict an output is to express it as a weighted sum of the input features such that $y = w_0 + w_1 \cdot x_1 + \cdots + w_n \cdot x_n$. The interpretability of this method comes directly from the weights w_i that show how much a change in one of the features affects the output, given that all other features stay the same.

2) Decision Tree: The decision tree model is a structure where the data are iteratively split in subsets based on some condition relevant to the input features (e.g. if $x_1 < value...$), until the final subsets, or leaf nodes, are created. In the case of regression, the condition of the split can be formed with the aim to minimize the mean squared error or mean absolute error of the predictions in the final nodes. By traversing the tree from the root node following the splits for a given input we arrive at a terminal node that tells us the prediction for this specific input. The decision tree can be visualized and gives an intuition about the model criteria, and the most important features can easily be determined.

3) Generalized Additive Models: In generalized additive models (GAMs) [36], the output y is calculated as

$$g(y) = w_0 + \sum_{i=1}^{n} F_n(x_n), \text{ where } F_n = \sum_{s=0}^{M} \beta_s B_s(x_n) \quad (5)$$

In [37], each F is called a shape function and can be comprised of base functions B_s such as polynomial expressions of a feature. The function g is called the link function and connects the different shape functions. A smoothing term λ is also applied during the fitting of the model to reduce overfitting. It is apparent that simple linear regression is a specific case of a generalized additive model. Since the functions can be very complex, we can visualize how each feature affects the final output by generating the so called partial dependency plots that portray their relationship.

IV. Evaluation

A. Datasets

a) Air Quality: The Air Quality dataset [16] contains hourly averaged values of sensor readings of five chemicals, namely carbon monoxide, non metanic hydrocarbons, benzene, total nitrogen oxides and nitrogen dioxide, from a device placed at road level in an Italian city. The dataset also contains the ground truth for these chemicals, provided by a separate reference analyzer, as well as the temperature and relative and absolute humidity. All these values are recorded from 03/2004 to 04/2005. In our experiment we use all these variables as inputs to predict the future values of the ground truth for carbon monoxide, benzene, total nitrogen oxides and nitrogen dioxide. The reason we drop the non metanic hydrocarbons ground truth time series is its large number of missing values.

b) Water Quality: This dataset comes from the ambient estuarine water quality monitoring data for the Burnett river in Queensland, Australia, retrieved from the Open Data Portal of Queensland Government. The dataset includes measurements of temperature, specific conductivity, pH, dissolved oxygen, turbidity and chlorophylla in water, measured every 30 minutes. In the original approach [17], all variables are used as inputs, and the temperature and dissolved oxygen as outputs. In our suggested formulation, all variables apart from dissolved oxygen are used as inputs to the DL part. The DL output is the predictions for these same variables. Those intermediate predicted values are used as inputs to the ML model along with the following time information for each predicted timestep: time of day, day of the week, day of the month, day of the year, month and quarter of the year. In addition, the ML model has as inputs the value of the dissolved oxygen at the last timestep of the original input window and its mean across the last 48 timesteps of the same input window. The output of the ML model is the future values of the dissolved oxygen time series. Following the example of [17], we use the measurements starting from 03/2014 up to and including 03/2018. More details about the measurements can be found on the official data portal[1].

B. Baselines

For the Water Quality dataset, we use as baseline the work in [17], a TCN with two fully connected networks on top to simultaneously forecast the two output variables of temperature and dissolved oxygen. To the best of our knowledge, no TCN configuration has been applied to the Air Quality dataset. Thus, we set as baseline a model similar to the above, i.e. a TCN with one fully connected network for each of the four output variables on top.

C. Preprocessing and Sample Creation

a) Air Quality: Apart from removing the Non Metanic Hydrocarbons ground truth variable due to a lot of missing values, we remove rows that contained nothing but NaN (Not a Number) values, which signify invalid measurements. We also replace with 0 the missing values that are indicated with the -200 value in the original dataset.

b) Water Quality: The first preprocessing step was inspecting the time series and removing values that are both rare and extreme, which makes it highly probable that they are the result of measurement errors. These include temperatures over 40 degree Celsius for the temperature time series, as well as negative values for the turbidity and chlorophylla series. Those values are replaced with NaN. Following that, we replace with NaN all values that are further than 3 times the standard deviation away from the mean of each times series. In that way we remove data points that may not appear obvious outliers during visual inspection, but statistically are highly improbable. We then fill in all the NaN values with the linear interpolation method of the pandas library [38].

[1]https://www.data.qld.gov.au/dataset/ambient-estuarine-water-quality-monitoring-data-near-real-time-sites-2012-to-present-day

The preprocessing in both cases is purposefully kept simple and generic. The motivation is that without expert knowledge on the specific domains any more specific preprocessing actions would not be justified, so we utilize methods that can easily be transferred across different domains. To create the samples for training, validation and testing of the end-to-end models, we use the sliding window technique. For the Air Quality dataset, each input window has length 96 and each output window 24. We use the data from the period 10/03/2004 - 10/12/2004 for training and from 11/12/2004 -04/04/2005 as test set. We use 25% of the training period data as validation set. This gives us 4865 training samples and 1621 validation samples. The test days are 114, of 24 measurements each. For the Water Quality dataset, each input window has length 192 and each output 48. We use the data from the period 03/2014 - 03/2017 for training and from 01/04/2017 - 31/03/2018 as test set. We use 10% of the training period data as validation set. This amounts to 41874 training samples and 4652 validation samples. The test days are 333, of 48 measurements each.

Before the start of the training, we shuffle the training samples in order to avoid that the validation set focuses on a specific time of a year that could affect variables such as the temperature. In addition, since the time series differ significantly in scale, we scale them to the 0 - 1 range using the scikit-learn library [39].

In the dissolved oxygen pipeline, the fitting of the ML models is done after the DL model has been trained. We select the DL model with the best validation set performance and we apply it to the whole training and validation set to get the intermediate outputs. After removing overlapping predictions, we have 46560 training samples. The test samples are still 333 days of 48 measurements, i.e. 15984 values.

D. Uncertainty Estimation

In order to get uncertainty estimates for the predictions of the models, we combine dropout [40] and ensemble [41] methods. We train 10 instances of the DL model with different random seeds, so we achieve different weight initialization, as well as different shuffling of the train set. During the evaluation of each of those instances, we enable the dropout layers of the TCN blocks and run the end to end models 20 times. This process gives us 200 samples from which we can get the mean and confidence interval of the predictions at the 95% level, as well as the mean and standard deviation of the metrics.

E. Implementation Details and Hyperparameters

All DL experiments were implemented with Tensorflow 2.1 [42] and executed on an NVIDIA GeForce GTX TITAN X GPU with 12 GB of memory. The training was executed for 120 epochs, with a batch size of 64. The loss is mean squared error and the optimizer is Adam with learning rate 0.001 and weight decay of 0.0001. The TCN for the Air Quality data is comprised of 5 blocks, with dilation factors [1, 2, 4, 8, 16]. The kernel size is 3, and the number of filters is 128. The dropout rate during training and uncertainty estimation is set

to 0.3. The hyperparameters for the Water Quality data are the same ones as in [17]. The TCN is comprised of 7 blocks, with dilation factors [1, 2, 4, 8, 16, 32, 64]. The kernel size is 3, and the number of filters is 64. The dropout rate during training is set to 0.5 and during uncertainty estimation to 0.85. The hyperparameter considered for the decision tree was its depth in the range [4,10]. The GAM model is implemented using the pyGAM library [43]. The hyperparameters considered are the number and order of the spline functions B_s in the ranges [3,5] and [3,10], as well as the smoothing penalty in the logarithmic scale [0.001,1000] with 11 evenly distanced points. The hyperparameters for the decision tree and GAM where chosen by running grid search in the above ranges with 10-fold cross validation on the original training data and keeping the ones with the least average value of mean squared error across runs and the least average difference between training and validation error (to avoid overfitting). This process leads to selecting 6 as tree depth, as well as (0.004,5,7) for λ, order and number of splines for the GAM model. All the training and evaluation code for the experiments, as well as the random seeds used for the training and the resulting weights are made available[2] to enable reproducibility, as well as support further exploratory research [44] on the topics of this work.

F. Results

In Table I we can see the mean and standard deviation of the root mean squared error (RMSE) and mean absolute error (MAE) metrics for all predicted variables on the test sets of both datasets, for the baseline and suggested architecture. These results clearly show that our added interpretability mechanism on top of the original model does not negatively affect the accuracy, while in some cases it even provides better results. We also see that our suggested approach takes more time to train, but in both cases the overhead is an acceptable percentage of the original model training time.

In Table II we can see how the same metrics for dissolved oxygen in the Water Quality dataset are affected as it is generated by the interpretable DL and ML pipeline instead of the original black-box DL model. It is notable that even the simplest approach with linear regression has comparable results with the baseline DL model, with much less deviation among the results. The reason for that can be that the ML output may depend more on features that are fixed, such as the time-related ones, rather than the intermediate uncertain DL outputs. Another noteworthy result is the fact that the GAM model performs worse than linear regression, which may indicate that for the specific set of features, the added complexity does not produce better results. We can however safely conclude that in the context of this real-world problem it is possible to add time and feature intepretability while on the whole achieving same or only slightly worse accuracy.

In Fig. 4 we can see the different visual results for the prediction of dissolved oxygen on a sample day. The similarity between the graphs for linear regression and GAM reinforces our conclusion about the unnecessary added complexity, while the decision tree graph form is the result of its quantized output options for the regression problem.

[2]https://github.com/lpphd/multivariate-attention-tcn

TABLE I
TEST SET METRICS AND TRAINING TIME FOR THE BASELINES AND PROPOSED ARCHITECTURE

Air Quality dataset			
		Baseline	Proposed model
CO	RMSE	1.56 ± 0.10	1.32 ± 0.04
	MAE	1.14 ± 0.07	0.93 ± 0.03
Benzene	RMSE	7.59 ± 0.31	7.02 ± 0.52
	MAE	5.69 ± 0.27	4.85 ± 0.25
NO_X	RMSE	244.43 ± 22.63	214.49 ± 10.31
	MAE	180.62 ± 15.56	152.25 ± 7.46
NO_2	RMSE	52.35 ± 13.29	48.57 ± 1.53
	MAE	41.83 ± 12.66	36.94 ± 1.18
Training time (sec)		477 ± 7	524 ± 11
Water Quality dataset			
		Baseline	Proposed model
Temperature	RMSE	0.59 ± 0.07	0.50 ± 0.02
	MAE	0.39 ± 0.04	0.33 ± 0.02
Training time (sec)		4554 ± 143	5310 ± 262

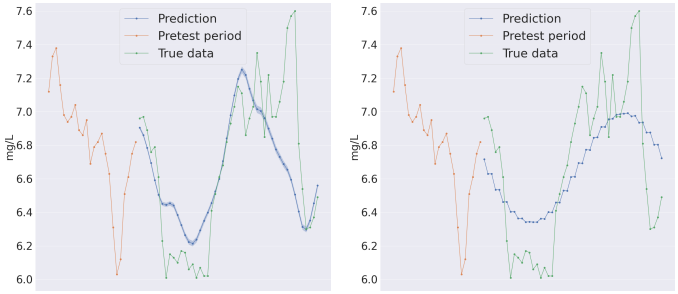
TABLE II
DISSOLVED OXYGEN TEST SET METRICS FOR THE BASELINE AND PROPOSED ARCHITECTURE WITH ML METHODS

Method	Dissolved Oxygen	
	RMSE	MAE
Baseline	0.50 ± 0.09	0.30 ± 0.05
Proposed model with linear regression	0.49 ± 0.00	0.34 ± 0.00
Proposed model with decision tree	0.55 ± 0.02	0.37 ± 0.01
Proposed model with GAM	0.52 ± 0.01	0.36 ± 0.01

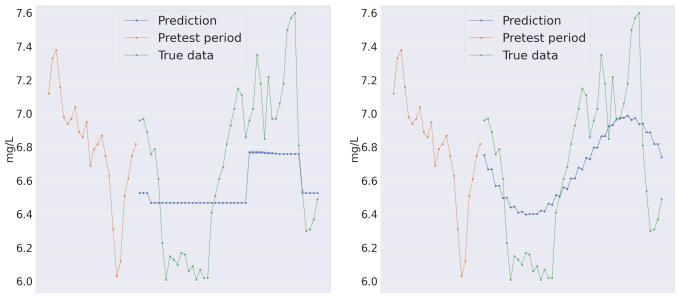
G. Interpretability Examples

In Fig. 5 we can see two representative intuitive visualizations of the attention distributions generated along the forecasting for two sample days, one for the temperature variable from the Water Quality dataset and the other for the benzene variable from the Air Quality dataset. These portray the relationship of one sample step of the output to the input of the specific time series.

In order to generate these visualizations, we select the row of matrix A, described in subsection III-B, that corresponds to the output element we want to study and we portray its values as a heatmap, after scaling them to the range 0-1. Thus, the darker the background at an input step, the higher its contribution to the value of the output step. Regarding the temperature variable, we can see that for this specific sample step (23rd) the model heavily focuses on the last few steps of the input, as well as the area approximately one day before the required prediction to construct the output. The attention for the sample step (13th) of the benzene variable follows

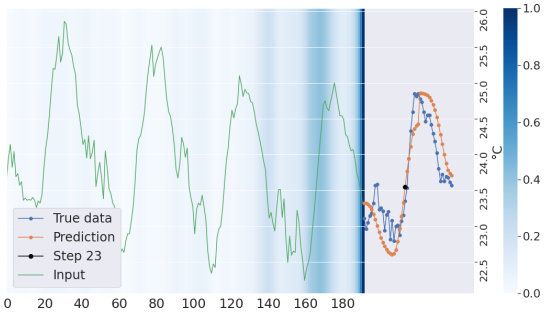


(a) Baseline DL model (b) Linear Regression

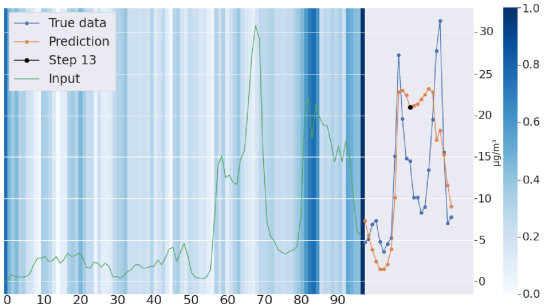


(c) Decision Tree (d) Generalized Additive Model

Fig. 4. Dissolved oxygen forecast visualizations for 21/04/2017



(a) Temperature attention distribution



(b) Benzene attention distribution

Fig. 5. Attention distributions for temperature and benzene variables (Figure best viewed in color)

a similar pattern, although it clearly presents more complex behavior, which may indicate the more complex nature of this time series.

This visualization mechanism is important in portraying the focus of the model when it succeeds, and equally importantly, the inability of the model to correctly construct the output, which can indicate that it is unsuitable for a specific time series. A robust empirical confirmation of the validity of this focus in the case of the Water Quality dataset is that we have used the attention pattern of the temperature variable as guideline to construct two features for the dissolved oxygen time series as input for the ML model, namely its last known value and the average value of its last period, as discussed in subsection IV-A. The accuracy that is achieved for the dissolved oxygen is an affirmation of the usefulness of the visualization. Even in the cases that the attention focus is more complex and not immediately obvious, the fact that we have this information at our disposal allows us to perform fine-grained analysis, such as discovering common patterns across multiple days of forecasting.

V. Conclusion

In this paper we proposed an architecture comprised of an attention mechanism combined with a temporal convolutional network that can produce per-instance prediction interpretation for multivariate time series forecasting by portraying, for each time series, the focus of the model on its input when producing each step of its output. We tested this mechanism on two multivariate datasets and we showed that the attention mechanism can be trivially added on top of an existing temporal convolution model with minimal or no changes and it adds interpretability without reducing the accuracy of the model, and in some cases even increasing it. We also presented the intuitive visual explanations of the attention distribution. In addition, we presented this architecture in the context of an end-to-end forecasting pipeline, by combining it with each of three inherently explainable ML methods on top. We used this pipeline to tackle a real-world forecasting problem and predict a target variable that was originally produced from a deep learning, black-box model. We compared the performance between these methods and the original approach and we showed that they give similar results in terms of metrics. Thus, we get comparable results with the added advantage of interpretability both across timesteps and features to the end-to-end model for the final target variable.

Future work includes further research regarding the combination of attention mechanisms with other types of models to achieve inherent and not post-hoc interpretability. In addition, we plan to explore an interpretable mechanism of evaluating the effect of all inputs to the output of a specific time series, by exploring the internal workings of the TCN model.

Acknowledgments

This work has been conducted as part of the Just in Time Maintenance project funded by the European Fund for Regional Development. We would also like to thank Dr. Mark Hoogendoorn for his helpful comments on the paper draft.

References

[1] M. Mohammadi, A. I. Al-Fuqaha, S. Sorour, and M. Guizani, "Deep learning for IoT big data and streaming analytics: A survey," *IEEE Commun. Surv. Tutorials*, vol. 20, no. 4, pp. 2923–2960, 2018.

[2] H. Hewamalage, C. Bergmeir, and K. Bandara, "Recurrent neural networks for time series forecasting: Current status and future directions," *CoRR*, vol. abs/1909.00590, 2019.

[3] S. Bai, J. Z. Kolter, and V. Koltun, "An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling," *arXiv:1803.01271 [cs]*, 2018.

[4] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *International Conference on Learning Representations (ICLR)*, 2015.

[5] B. Lim and S. Zohren, "Time series forecasting with deep learning: A survey," *CoRR*, vol. abs/2004.13408, 2020.

[6] S. Makridakis, E. Spiliotis, and V. Assimakopoulos, "Statistical and Machine Learning forecasting methods: Concerns and ways forward," *PLOS ONE*, vol. 13, no. 3, pp. 1–26, 2018.

[7] B. N. Oreshkin, D. Carpov, N. Chapados, and Y. Bengio, "N-BEATS: neural basis expansion analysis for interpretable time series forecasting," in *International Conference on Learning Representations (ICLR)*, 2020.

[8] S. Makridakis, E. Spiliotis, and V. Assimakopoulos, "The M4 Competition: Results, findings, conclusion and way forward," vol. 34, no. 4, pp. 802–808, 2018.

[9] B. Goodman and S. Flaxman, "European union regulations on algorithmic decision-making and a right to explanation," *AI Magazine*, vol. 38, no. 3, pp. 50–57, 2017.

[10] M. Du, N. Liu, and X. Hu, "Techniques for interpretable machine learning," *Commun. ACM*, vol. 63, no. 1, pp. 68–77, 2020.

[11] L. H. Gilpin, D. Bau, B. Z. Yuan, A. Bajwa, M. Specter, and L. Kagal, "Explaining explanations: An overview of interpretability of machine learning," in *IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, 2018, pp. 80–89.

[12] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi, "A survey of methods for explaining black box models," *ACM Comput. Surv.*, vol. 51, no. 5, 2018.

[13] F. Doshi-Velez and B. Kim, "Towards A Rigorous Science of Interpretable Machine Learning," *arXiv preprint arXiv:1702.08608*, 2017.

[14] S. M. Lundberg and S. Lee, "A unified approach to interpreting model predictions," in *Advances in Neural Information Processing Systems (NIPS)*, 2017, pp. 4765–4774.

[15] C. Rudin, "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead," *Nature Machine Intelligence*, vol. 1, no. 5, pp. 206–215, 2019.

[16] S. De Vito, E. Massera, M. Piga, L. Martinotto, and G. Di Francia, "On field calibration of an electronic nose for benzene estimation in an urban pollution monitoring scenario," *Sensors and Actuators B: Chemical*, vol. 129, no. 2, pp. 750–757, 2008.

[17] Y. Zhang, P. J. Thorburn, and P. Fitch, "Multi-task Temporal Convolutional Network for Predicting Water Quality Sensor Data," in *International Conference on Neural Information Processing ICONIP*, 2019, pp. 122–130.

[18] G. Lai, W. Chang, Y. Yang, and H. Liu, "Modeling long- and short-term temporal patterns with deep neural networks," in *ACM SIGIR Conference on Research & Development in Information Retrieval (SIGIR)*, 2018, pp. 95–104.

[19] "DeepAR: Probabilistic forecasting with autoregressive recurrent networks," *International Journal of Forecasting*, vol. 36, no. 3, pp. 1181–1191, 2020.

[20] A. Borovykh, S. Bohte, and K. Oosterlee, "Conditional time series forecasting with convolutional neural networks," *arXiv preprint arXiv:1703.04691*, 2017.

[21] R. Wan, S. Mei, J. Wang, M. Liu, and F. Yang, "Multivariate Temporal Convolutional Network: A Deep Neural Networks Approach for Multivariate Time Series Forecasting," *Electronics*, vol. 8, no. 8, p. 876, 2019.

[22] C. Fan, Y. Zhang, Y. Pan, X. Li, C. Zhang, R. Yuan, D. Wu, W. Wang, J. Pei, and H. Huang, "Multi-horizon time series forecasting with temporal attention learning," in *ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*, 2019, pp. 2527–2535.

[23] S.-Y. Shih, F.-K. Sun, and H.-y. Lee, "Temporal pattern attention for multivariate time series forecasting," *Machine Learning*, vol. 108, no. 8-9, pp. 1421–1441, 2019.

[24] S. Li, X. Jin, Y. Xuan, X. Zhou, W. Chen, Y. Wang, and X. Yan, "Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting," in *Advances in Neural Information Processing Systems Systems (NIPS)*, 2019, pp. 5244–5254.

[25] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems Systems (NIPS)*, 2017, pp. 5998–6008.

[26] B. Lim, S. Ö. Arik, N. Loeff, and T. Pfister, "Temporal fusion transformers for interpretable multi-horizon time series forecasting," *CoRR*, vol. abs/1912.09363, 2019.

[27] Y. Qin, D. Song, H. Chen, W. Cheng, G. Jiang, and G. W. Cottrell, "A dual-stage attention-based recurrent neural network for time series prediction," in *International Joint Conference on Artificial Intelligence (IJCAI)*, 2017, pp. 2627–2633.

[28] R. Sen, H. Yu, and I. S. Dhillon, "Think globally, act locally: A deep neural network approach to high-dimensional time series forecasting," in *Advances in Neural Information Processing Systems Systems (NIPS)*, 2019, pp. 4838–4847.

[29] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Learning deep features for discriminative localization," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 2921–2929.

[30] E. Choi, M. T. Bahadori, J. Sun, J. Kulas, A. Schuetz, and W. F. Stewart, "RETAIN: an interpretable predictive model for healthcare using reverse time attention mechanism," in *Advances in Neural Information Processing Systems Systems (NIPS)*, 2016, pp. 3504–3512.

[31] M. Rosnati and V. Fortuin, "MGP-AttTCN: An Interpretable Machine Learning Model for the Prediction of Sepsis," *CoRR*, vol. abs/1909.12637, 2019.

[32] T. Salimans and D. P. Kingma, "Weight normalization: A simple reparameterization to accelerate training of deep neural networks," in *Advances in Neural Information Processing Systems Systems (NIPS)*, 2016, p. 901.

[33] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[34] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *International Conference on Learning Representations (ICLR)*, J. Fürnkranz and T. Joachims, Eds., 2010, pp. 807–814.

[35] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, ser. Springer Series in Statistics. Springer, 2009.

[36] T. J. Hastie and R. J. Tibshirani, *Generalized additive models*. CRC press, 1990, vol. 43.

[37] Y. Lou, R. Caruana, and J. Gehrke, "Intelligible models for classification and regression," in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2012, pp. 150–158.

[38] Wes McKinney, "Data Structures for Statistical Computing in Python," in *Proceedings of the 9th Python in Science Conference*, 2010, pp. 56 – 61.

[39] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[40] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *International Conference on Machine Learning (ICML)*, vol. 48, 2016, pp. 1050–1059.

[41] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," in *Advances in Neural Information Processing Systems (NIPS)*, 2017, pp. 6402–6413.

[42] M. Abadi *et al.*, "Tensorflow: A system for large-scale machine learning," in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, 2016, pp. 265–283.

[43] D. Servén and C. Brummitt, "pygam: Generalized additive models in python," 2018.

[44] X. Bouthillier, C. Laurent, and P. Vincent, "Unreproducible research is reproducible," in *International Conference on Machine Learning (ICML)*, 2019, pp. 725–734.