An XAI Approach on the Capacity of Transformers to Learn Time Dependencies in Time Series Forecasting

Alberto Mino Calero, Adil Rasheed, and Anastasios M. Lekkas
Department of Engineering Cybernetics, Faculty of Information Technology and Electrical Engineering
Norwegian University of Science and Technology (NTNU)
NO-7034, Trondheim, Norway
e-mail: {alberto.m.calero | adil.rasheed | anastasios.lekkas}@ntnu.no

***Abstract*—Despite the traction gained by transformers on time series forecasting tasks, partially based on their success in natural language processing to learn contextual information, their suitability in this domain has been questioned. Several works have found significant performance problems when comparing these models with simpler options leading some people to think transformers are not as fit for time series forecasting as considered before. However, it remains unclear if they can capture long-term dependencies, similarly as they do with contextual information. This study takes on that line of questioning using an explainable artificial intelligence approach. By making use of Shapley additive explanations, the attribution scores assigned to input features are computed, showcasing that a transformer model optimized for time series forecasting is unable to learn long-term time dependencies, and mostly considers the last time steps from the inputs. The analysis, based on interpretable knowledge based on the computed Shapley values, is done in a multivariate forecasting setting that resembles a complex real-world problem, proving that the model is unsuited for the task.**

Keywords-Transformers; Time Series; Deep Learning; XAI; SHAP.

I. Introduction

Time series analysis and forecasting have a major role in research [1] and real-world applications, such as climate science [2], healthcare [3], biology [4], and economics [5]. As technology has progressed, new methods and algorithms have been used to improve the models used. From statistical and regression-based models such as autoregressive integrated moving average (ARIMA) [6][7], the field has seen a transition to machine learning approaches, such as support vector machines [8], nonparametric models like functional decomposition [9], and nonparametric bayesian models [10].

More recently, deep learning has gained more traction over other machine learning techniques thanks to their advantage of being universal approximators [11][12] under the right conditions. With their great versatility and representational power, many neural network architectures have been used for time series forecasting, such as multi-layer perceptions (MLPs) [13], convolutional neural networks (CNNs) [14], and networks designed to solve sequential data problems like recurrent neural networks (RNNs) [15][16] and long-short term memory (LSTM) networks [17]. More recently, transformers [18] have become one of the main technologies used for time series forecasting [19]–[23] thanks to their outstanding capabilities in modeling contextual information and handling sequence data, particularly in natural language processes (NLP) tasks. Despite their popularity, all deep learning models have the disadvantage of being black boxes, a feature highly undesired in safety- and business-critical applications. Even transformers, despite the self-attention mechanism that allows them to weigh the importance of different parts of their input sequential data to make their forecasts, distinguishing them from other types of neural networks, are virtually uninterpretable.

Due to the abrupt increase in the use of transformers for time series forecasting, several works have questioned their effectivity, with results that show how remarkably simple linear models can, indeed, outperform them. To answer this question, performance has been the main element examined to the best of our knowledge, like in [24], using the metrics mean square error (MSE) and mean absolute error (MAE) to reason why transformers may have design flaws to tackle time series forecasting, having been designed for natural language processing (NPL) tasks. Although the inputs of transformers are sequences, like in time series forecasting, one key part of their success is their ability to extract contextual information from the text sequences, which seems not to translate very well to learning time dependencies.

Thanks to the revitalized interest in eXplainable AI (XAI) motivated by the surge in using neural networks, many solutions provide interpretable knowledge and explanations based on many algorithms that exploit either external elements, with primarily model agnostic techniques, or internal elements, which primarily focus on the gradients that connect input and output in these models or attention mechanism in case of transformers. Similarly to other deep learning models, interpretability in transformers can be tackled from those perspectives, which may give insight into their efficacy in time series forecasting by showing if they are learning long-term time dependencies, or if they rather only focus on the last instances from the inputs, leading to short-term dependencies and discardment of the rest of the time series input.

In this work, an XAI-based methodology is proposed to analyze what time dependencies have been learned by a transformer-based model making use of Shapley additive explanations (SHAP) [25] to compute attribution scores of the inputs. By aggregating these scores and examining them with respect to the time series input and output sequences, it is possible to look into how these scores change as the model predicts further into the future. By doing so, an analysis of these changes offers a way to determine what kind of temporal patterns the models have learned, and how long back into the past they are able to look for their predictions.

The paper is organized as follows. Section II presents a brief overview of the state-of-the-art concerning XAI applicable to transformer-based models both broadly and specific to time series forecasting. Next, Section III introduces the methodology used in this work. Details referring to the dataset and the transformer model trained are provided, followed by a description of SHAP, the implementation choice for the computation of the Shapley values, and the framework used to analyze them to assess the suitability of the model for the task. Then, the results and analysis are presented in Section IV, which conveys the findings of this work that the model is unable to learn long-term time dependencies. Finally, V presents the conclusions and suggests future lines of work aimed to better examine this family of models and their suitability for time series forecasting tasks through the lenses of XAI.

II. State-of-the-art on explainability of transformers

Transformers are a type of deep neural network (DNN) characterized by using a self-attention mechanism and an encoder-decoder architecture. Although one self-attention head can learn which parts of the inputs are more important when making predictions, hence holding interpretable knowledge, they typically have many of such self-attention heads, in addition to a very high number of parameters combined with non-linear activation functions, and so they are initially incomprehensible. To tackle them and attempt to explain their inference, there are two main families of algorithms: attribution score and attention-based methods.

In attribution-score methods, the algorithms assign a relevance score to each of the inputs that were processed by the model to reach a certain output. This provides interpretable knowledge that exposes how high or low the contribution of each input feature is for any output. These types of methods rely mostly on input-output mappings, and as such are considered "external", and in many cases are model agnostic, meaning they can be applied to any machine learning model.

Some examples of attribution score-based methods are integrated gradients (IG) [26], layer-wise relevance propagation (LRP) [27] or SHAP [25]. IG works by integrating the gradients of the model's output with respect to the input features along a path from a baseline input to the actual input. LRP propagates the prediction backward in a neural network with designed local propagation rules subjected to a conservation property so that what is received by a neuron is redistributed in equal amounts to the previous layer. SHAP calculates the relevance scores by distributing the outcome of the model among the input features to get the relevance scores, which is done using Shapley values from coalitional game theory. These methods can, usually, be applied to any deep learning model, while others such as SHAP are model agnostic, hence applicable to any machine learning model. Thanks to this versatility, they have been successfully applied on numerous occasions.

As one key aspect of the success of transformers comes from their attention mechanism, this inbuilt attention mechanism can help to look into their behavior. Although still not interpretable, analyzing where transformers focus their attention may provide a better understanding of how they infer. Along this line, several works take advantage of that to build methods that throw some light on their interpretability to provide, to a certain degree, XAI in the context of transformers in the form of attention-based methods. These methods usually provide visual information about which elements of the input sequence the model has learned to pay attention to and rely directly on the attention parameters, although the existence of multiple attention heads complicates the process.

Attention mechanism-based methods require considering the importance of how to properly take advantage of the multi-head self-attention mechanism (MHSA) to get valuable information. In [28], an updated version of LRP designed to work with transformers and MHSA was developed. Instead of propagating scores backward through all layers, their approach focuses on attention head relevance, and although their technique has the initial goal of serving as a means to prune unimportant attention heads, they successfully identify different interpretable roles within the MHSA in the context of automatic translation.

Attention rollout and attention flow are the solutions proposed in [29] to quantify the information flow approximating the attention to the inputs. First, attention rollout traces the flow of information from input tokens to hidden embeddings in the higher layers by propagating attention weights through several network layers. This is done by recursively multiplying the minimum, maximum, or mean of the attention heads of each block with the attention of the previous blocks. Then, attention flow treats the resultant graph as a flow network using the attention weights as edge capacities. By doing this, the maximum attention flow between any of the layers to any of the input nodes can be computed with any maximum flow algorithm, which can serve as an approximation of the attention to input nodes. Nevertheless, the method's speed is prohibiting, hence rather unfeasible for evaluations at a large scale.

Beyond attention [30], based on LRP, proposes assigning attribution scores using the deep Taylor decomposition principle to generate the initial scores and then propagate them through the layers taking into consideration both attention layers and skip connections, both integral parts of transformers architectures. The initial scores are computed via LRP for all attention heads and all layers by integrating the relevance score of each attention head and its gradient with respect to the input features. The weighted attention scores they propose to adopt the notion of positive relevance, hence considering only the positive values that result from the computation, and the method was tested in the NLP model BERT [31] and a visual transformer model.

Beyond intuition [32] is another proposal designed to approximate the contribution of the input tokens. This solution relies on the partial derivative of the loss function with respect to each token in two steps. First, in the attention perception phase, the relationships between input and output for each attention block are unfolded, resulting in the development of two recurrence formulas involving head-wise and token-wise attention maps. The second step is reasoning feedback, which applies the IG algorithm to the last attention map with a noise-decreasing strategy to reduce the gradient self-induced noise.

Gradient self-attention maps (Grad-SAM) [33] uses a gradient-based method applied to the attention matrices to generate rankings for the tokens used in the layers of the encoder and it was tested on the NLP model BERT. This method offers another way to understand how the model reaches an individual prediction by highlighting the inputs that contribute to it the most, and it is based on computing the partial derivatives of the model output with respect to the self-attention blocks. This approach does not ignore negative values resulting from this computation, but to avoid the accumulation of negatives that may cancel the information carried by positive values, they propose using the ReLU function to zero out negatives.

In the context of visual transformers [34], the survey from [35] paints a picture of the current state-of-the-art methods. Understanding visual information has the technical advantage of humans being able to identify patterns easily, and it is only necessary to visualize what the models are paying attention to, to make sense of their inference process. Traditional attribution-score methods, such as layer-wise relevance propagation [27], SHAP [25], local interpretable model-agnostic (LIME) [36], or gradient-weighted class activation mapping (Grad-CAM) [37] can suffice. On the other hand, since the attention mechanism is designed to allow transformers to focus on the relevant part of the input sequence, other methods focus on this element, and in computer vision tasks attention can be directly visualized just by looking into the raw attention of the transformers, or through some function, such as Grad-SAM [33] or beyond intuition [32].

In time series classification [38], attribution and attention methods are the most popular strategies to assign relevance scores to every time point in a time series. Another type of explanation can be extracted based on subsequences by identifying the parts of a sequence responsible for a certain classification outcome, such as PatchX [39] and data-agnostic classification method via shapelets (DASH) [40]. The third general type of explanation in this domain is based on instances and relies on the complete time series to extract reasons for the classification outcomes, such as multi-operator temporal decision trees [41] and dual prototypical shapelet networks [42].

In time series forecasting, [43] presents an experimental comparison between XAI methods based on saliency maps applied to several deep learning architectures including transformers. The methods tested include IG [26], SmoothGrad [44], DeepLIFT [45], gradient and deep SHAP [25], and feature ablation [46], among others less up-to-date. The conclusions point out a general lack of high-quality interpretable knowledge from these methods when applied to multivariate time series data while succeeding when the time series is represented as images or univariate and propose a two-step temporal saliency rescaling to improve the results. They hypothesize that the main reason for this lack of quality is the combination of temporal and feature domains. Previous studies [47] provided comparisons between LIME, LRP, DeepLIFT, Saliency, and SHAP for time series classification with DNNs, CNNs, and ResNet, with SHAP performing more robustly than the rest on ResNet. They showed a decrease in accuracy when perturbing subsequences in univariate time series classification datasets, which was assumed to be indicative of the alteration of parts that were important for the models' internal inference, and SHAP seemed to perform better than the rest for the more advance Resnet architecture.

III. Methodology

The methodology applied in this work is described in this section, and it is based on the goal of analyzing if a machine learning model, in this case a Transformer, is able to learn time dependencies in a time series forecasting setup using attribution scores computed with the Shapley additive explanations method. Henceforth, it can be generalized to any other model as long as the XAI method used to compute the attribution scores is model agnostic, such as SHAP, or is applicable to the model in question.

A. Data

In this work, the data comes from a collection of the currency exchange rates of eight countries, including Australia, British, Canada, Switzerland, China, Japan, New Zealand, and Singapore, between 1990 and 2016, gathered daily. This data has been used for time series analysis benchmarks of different models, transformer-based architectures included, in works such as [48] and [24]. The dataset has eight input features and 7588 total time steps with a time resolution of one day. Regarding the data splitting, a standard distribution was used, forming subsets of 70%, 10%, and 20% of the total samples for training, validation, and testing, respectively.

B. Transformer model

This work analyzes the transformer implementation used in [20], [24]. The implementation details contain several differences with respect to the actual vanilla transformer [18]. The importance of the differences becomes fundamental to analyzing the models through XAI lenses because of a series of technical issues and can be summarized as follows.

- Concerning the architecture, this transformer version contains an additional temporal embedding that receives and handles the information about the time marks of each sequence element.
- For the inputs, the transformers take 3 elements in addition to the expected time series sequence. First, the decoder takes the last L_{dec} pieces of the time series input. This reduced time series acts as an enhanced "start

sequence" token, supplying additional information for the decoder. Then, the time marks of the time series input on one hand, and the time marks of the enhanced start sequence token followed by the time marks that characterized each time step of the expected prediction horizon on the other. These two time mark sequences are used by the encoder and the decoder respectively, and are treated as the other two inputs. The time mark sequences encode information about the dates in a four-values format. This format is a hyperparameter, and in these experiments, the selected "hourly" encodes and normalizes the hour of the day, the day of the week, the day of the month, and the day of the year.

- In regards to the outputs and in connection to the training and inference processes, the model performs a direct multi-step prediction, hence providing the whole expected sequence in a single step.

1) Training and performance: Regarding training, 10 models were optimized for a prediction horizon $Z_p = 96$, chosen from the set of values $96, 192, 336, 720$ as it yielded the best performance. Since it is a multivariate forecasting task and 10 models were trained, each model's performance mean squared error MSE_{exp} is computed with (2), where y is the prediction value, $\hat{y}$ the true value, N_f the total number of features predicted, N_p the number of predictions, and the subscripts i, j, and k are feature, prediction, and experiment, respectively (also in (1)). Then, the aggregated MSE (MSE_{agg}) is simply an average of the MSE_{exp} of the 10 models following (1).

$$MSE_{agg} = \frac{\sum_{k=1}^{N_{exp}} MSE_{exp_k}}{N_{exp}} \tag{1}$$

$$MSE_{exp} = \frac{\sum_{j=1}^{N_p} \sum_{i=1}^{N_f} (y_{i,j} - \hat{y}_{i,j})^2}{N_f N_p} \tag{2}$$

As the loss function, the MSE is used as the standard for regression tasks, which is averaged among the 8 output features since the multivariate forecast of the model. The models are trained with different seeds to get rid of possible underperforming combinations of initialized parameters, using the adaptive moment estimation (ADAM) optimizer with input sequences of length $Z_i = 96$, again yielding the best results from the set $96, 192, 336, 720$. The models were trained in an NVIDIA RTX 3090 GPU. For the rest of the hyperparameters, the default values from [24] were initially used, and for this work, a random search for optimization purposes was performed over batch size, learning rate, training epochs, and early stopping hyperparameters, arriving at respective values of 32, 0.0001, 10, and early stopping with patience 3.

Since the goal is to ascertain if transformers can learn time dependencies, the best-performing model is the one chosen to be analyzed by SHAP. Note that this work will not examine model performance in depth as it falls out of the scope.

C. SHAP

The original kernel and deep implementations from [25], referred to hereinafter as Kernel and Deep, were used to examine the inner workings of transformers with the SHAP approach as computationally efficient algorithms to approximate Shapley values, particularly so in higher dimensions. As shown by [25], the Shapley values approximated with SHAP using Kernel perform slightly better than those approximated with Deep (DeepLift-based SHAP). Because of the substantially high memory and computation requirements of Kernel, a base test was made to compare both results and determine how much Kernel suffers by leaving out part of the training data as background data for the algorithm, to make an informed decision about which algorithm to use for the analysis of the transformers with SHAP. The L2 distance and symmetric mean absolute percentage error (SMAPE) were used as distance measurements. L2 serves as a standard distance measurement between real value vectors to capture important deviations in feature attribution, while SMAPE is a scale-independent and commutative measurement that offers a better understanding of the distance regardless of the small scale of the values.

With the larger amount of data manageable with the computation resources used in this work for a single time step of the prediction horizon (32 training sequences for Deep, 16 for Kernel), an $L2 = 0.0036$ and *SMAPE* $= 175.48\%$ were reached between Deep and Kernel. With the minimum reduction of the background data for the analysis of the whole prediction horizon (32 training sequences for Deep, 4 for Kernel), an $L2 = 0.0046$ and *SMAPE* $= 184.38\%$ were computed instead. These are calculated for the feature attribution of the first time step forecast. While the difference between *SMAPE* error is only 4.95%, the percentage difference in $L2$ ascends to 24.81%, suggesting the Kernel implementation suffers when the background data to integrate out features decreases. Henceforth, the choice was made to use the results of the DeepLift-based Shapley values (DeepSHAP) for the analysis.

For the use of the original DeepSHAP algorithm implementation [25], the critical adaptations of this algorithm to function on the model and data used for this work were implemented in the form of wrappers and data refactorization to meet the requirements of the DeepExplainer interface. To achieve this, the model is wrapped in a function that only needs the time series input sequence so that SHAP focuses on extracting the feature importance of this data instead of over every input the model actually needs. This is done by making the rest of the inputs independently accessible by the model outside the DeepSHAP algorithm. Therefore, the time series input sequence is used as input data for the model to calculate the Shapley values. Regarding the rest of the inputs, the time series sequence subset needed by the decoder is handled by the wrapper, which already receives the whole time series input sequence. The time marks, on the other hand, must remain unchanged during the computation because it is not desirable in this setting to alter the possible time dependencies that the model might have learned and may be related to the temporal embeddings. For this reason, this part of the inputs is accessible by the model through external variables, but not modified by SHAP.

The data refactorization involves the input sequence and the output but does not affect the process other than reshaping the sequences so that DeepSHAP can process them. The time series input sequences are reshaped to a format admissible by SHAP before the algorithm is called and shaped back to the format required by the model within the model wrapper sent to the SHAP algorithm. Regarding the outputs, the wrapper is configured to output a single time step within the prediction horizon, which can be selected beforehand. Therefore, to analyze the whole prediction horizon SHAP needs to be computed for each time step. These changes allow the extraction of the attribution scores of the features examined in this work in a compatible way with SHAP, while also reducing the computation requirements, which would otherwise be prohibited in terms of memory due to the high memory use, which increases exponentially with the dimensionality of the data. To analyze the impact of each input feature of each time step of the input sequence, each feature from each time step needs to be considered as an individual input, hence the dimensionality of the input features analyzed by SHAP is the number of input features times the length of the input sequence. Note none of these changes have any effect on the actual transformer model architecture or parameters behind the wrapper, hence the inference process performed by the transformer remains unaltered.

D. Shapley values analysis

In the analysis, the features that significantly impact the model outputs according to the attribution scores computed by Shapley values are examined. These values are computed by the SHAP algorithm using the full training set as background data as described in [25]. The Shapley values are studied by looking at the features with a higher accumulated impact on the outputs from several individual time steps of the prediction horizon. This is done by considering each feature from each time step of the input sequence on its own and accumulating their impact by adding the absolute values that each input feature has for all the outputs in the chosen prediction horizon slice. This way, it can be seen which input features and from which input time steps have a higher impact on the model's outputs.

Secondly, the evolution of the attribution scores is investigated throughout time steps from input and output sequences. To look at the overall evolution of the impact of all the input features and time steps, looking separately into specific output features of the model allows to obtain more general information about how the impact of input features from different time steps varies depending on the prediction horizon.

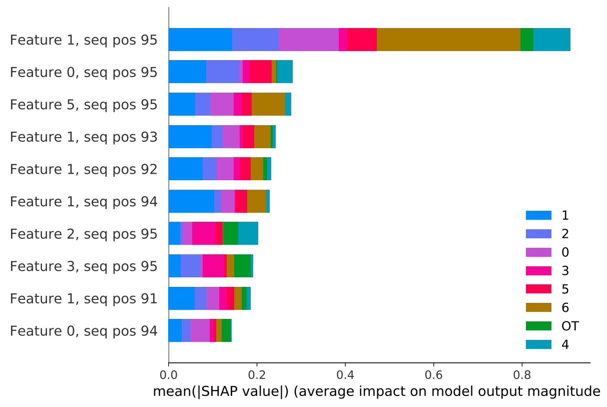
Ultimately, the global distribution of the accumulated impact from all features is studied. This is done by applying a sum reduction over the absolute values of the impact of all the input features over all the output features. Since the impact from the features computed by attribution scores that SHAP provides measures how much changes in the values cause changes in the prediction, using sum reduction over their absolute values will provide practical information about the patterns learned by the model without mismanaging or losing the SHAP values meaning. Hence, looking at the impact that each time step of the input sequence has over each time step of the prediction horizon yields the most global vision. By analyzing this, it is possible to ascertain what time dependencies the model has learned from a wider point of view, as well as detect possible anomalies or interesting behaviors.

IV. RESULTS AND ANALYSIS

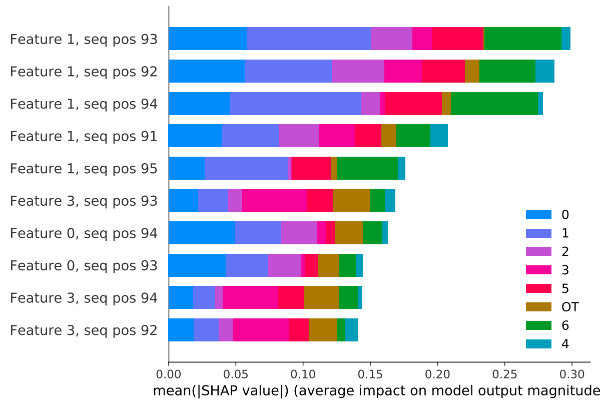
The 10 instances of the transformer trained reached an average performance of $MSE = 0.748$, with the best-performing model chosen for the analysis with SHAP having a performance of $MSE = 0.601$.

The input features with the highest impact on the model's output for several time steps located at the beginning and ending parts of the prediction horizon can be seen in Figure 1. In Figure 1a, it can be seen how the Shapley values reveal that feature attribution is placed directly on the time steps from the input sequence that are directly adjacent to the starting point of the prediction horizon. This behavior of the attribution scores replicates in the whole prediction horizon the model is trained for ($Z_p = 96$) except in the last time step. Note that the second to last time step also behaves similarly, as displayed by Figure 1b, although with more evenly distributed attribution scores across a wider range of input features from the last elements of the input sequence. The only deviation from this pattern can be seen in Figure 1c, which displays the attribution score for the last time step of the prediction horizon, and where it can be observed that 6 out of the 10 most impactful features come from the intermediate input sequence elements instead of the last. Nevertheless, as this does not replicate in any other time step of Z_p, it seems likely to be an outlier prediction situation. Note that there is no evident anomaly in the input sequence that explains this event either.

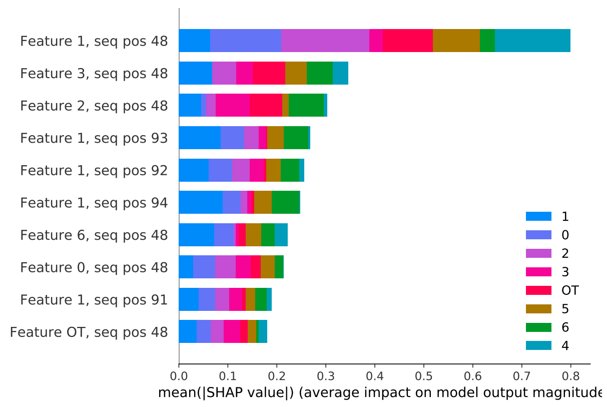
The distribution of the attribution scores across the entirety of the input sequence is shown in Figure 2 for four time steps across diverse lengths of the Z_p. As can be seen, the model is influenced mostly by the last elements of the input sequence length, except in the last time step. These attribution scores, however, present some changes after the first forecasted time steps, especially observable in Figure 2b and Figure 2c, with the attribution scores being slightly more evenly distributed. For some output features, such as Feature 1 and 2, this happens across the second half of the input sequence, while for others, such as Feature 3 and 4, it is distributed between the ending and the beginning of the input sequence. Despite these changes, notice that most of the importance is still placed in the last time steps of the input sequence, especially so if the comparison is made between the initial, intermediate, and ending parts of the input sequence. Moreover, as was displayed by Figure 1c, Figure 2d shows that there is a notable anomaly on the attribution score placement for the last forecasted time step, where most of the impact, and across all output features, is placed on an intermediate, single time step (48). Regardless of the shifts in the feature importance distribution as the model



(a) Predicted time step $Z_p = 0$



(b) Predicted time step $Z_p = 94$



(c) Predicted time step $Z_p = 95$

Figure 1. SHAP values of the 10 features with the highest impact for multiple times steps of the prediction horizon. Seq. pos. indicates the time step of the input feature, i.e. its location in the time series input sequence, while the name of each feature is enumerated from 0 to 6 plus the OT feature. Note that OT is one of the eight countries, named differently to be used as the default output target for the univariate implementation.

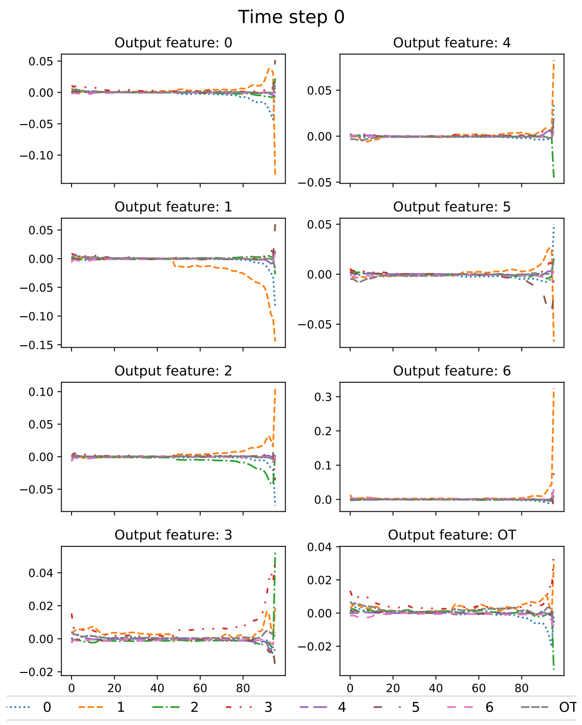
predicts further into the future, the anomaly only happens in the last forecasted time step, complicating the analysis of the event unless it can be explained as an outlier in the prediction, which is a known issue in certain domains as pointed out in [49], [50].

The distribution of the impact from all the input features over all the outputs, for all the time steps from both input sequence and output prediction horizon can be seen in Figure 3. In the figure, it can be easily seen that the transformer model is mostly only affected by the last five elements of the input sequence. As most of the influence is placed upon only a small fraction of the inputs, essentially the last time steps from the input, and across the whole prediction horizon, it can be verified that the model has not learned any long-term time dependencies, which suggests transformers might be fundamentally flawed in dealing with long time series forecasting (LTSF) tasks from an XAI perspective. It can be noted that two anomalies occur in two instances of time. One, as previously spotted in Figure 2d, appears on the last time step of the prediction horizon, while the other does so in the very first, and can also be seen in more detail in Figure 2a. The second anomaly is only in the accumulated value of the impact of the features, which is significantly higher (about three times bigger) with respect to its neighbors on the plot. Nevertheless, it is not an anomaly in the general behavior. On the contrary, the first anomaly of the predicted time steps is indeed of value but also of behavior. Nevertheless, being the only behavioral break of the pattern analyzed across 9216 data points of 3, it is hardly significant enough, and seems most likely an outlier in the prediction or the SHAP values computed.

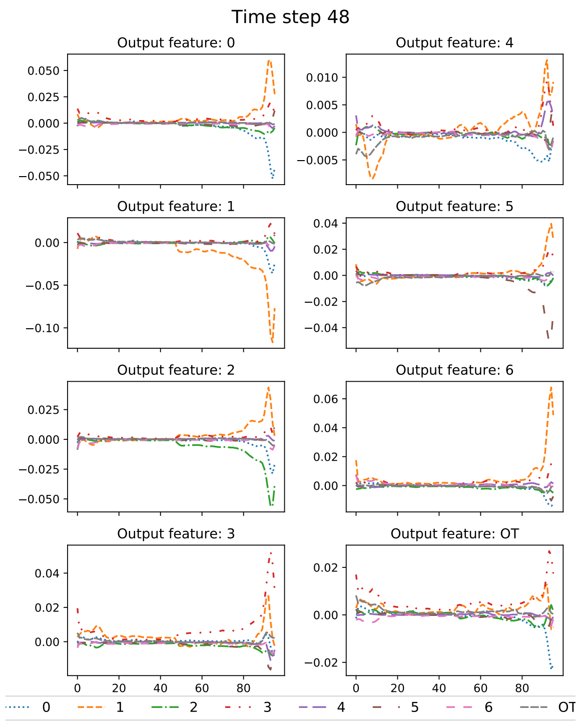
V. Conclusion and Future Work

In this work, a framework is proposed to look into the suitability of transformers for time series forecasting using SHAP to analyze the behavior of the model. It is found that the transformer was not able to learn long-term time dependencies from its training, and according to the attribution scores computed all the impact is placed upon the last time steps of the inputs. These findings suggest that transformers-based models might be particularly prone to disregard most of the time series data used for their training, hence making them not good choices for this particular task. In consequence, developers should be aware that transformers appear to fail to capture the long-term dynamics of time series, and the issue should be investigated when proposing transformer-based models for this type of task, in addition to considering the use of other types of models for LTSF tasks. Furthermore, for applications where the performance of the transformer-based model is the only feature looked for, a similar analysis may be useful in optimizing the training, given that similar conclusions are reached where most of the input sequence has a very low impact on the forecasts, suggesting that decreasing the length of the input sequence would not significantly impact the performance.

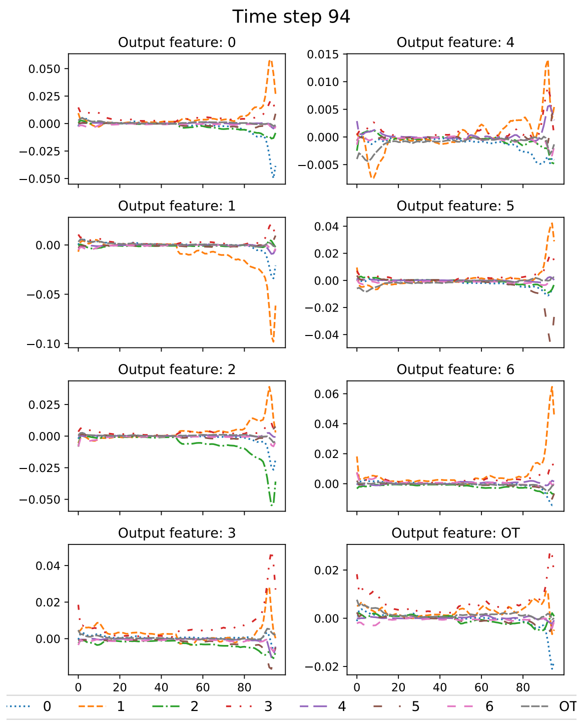
Facing the lack of analysis of this particular issue from an XAI perspective, experimenting further with these models in a wider variety of time series datasets and types of models may yield interesting findings. It can lead to better establishing if the findings of this work are a symptom of a flaw in transformer-based models' design to address LTSF tasks, or if it is just a case of a flawed model not being able to learn from a



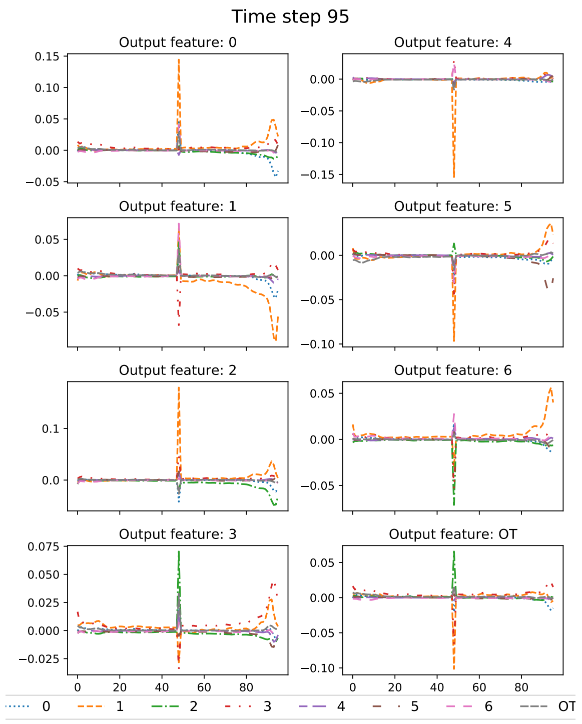
(a) Forecasted time step 0. Most of the input influence is placed over the last elements of the input time series sequence.



(b) Forecasted time step 48. The behavior is similar to Figure 1a, but the influence is less concentrated on the last time steps.



(c) Forecasted time step 94. The behavior is consistent with every previous forecasted time step, and nearly identical to Figure 2b.



(d) Forecasted time step 95. The behavior changes and most of the influence is placed on the middle input features of the time series. Note that it is the only time step where this behavior occurs.

Figure 2. Distribution of the attribution scores across all the elements of the input sequence for the forecasted time steps 0, 48, 94, and 95. Each subfigure depicts the impact evolution across the input time steps for each of the 8 output features, which are divided into different plots, each one depicting 8 signals representing each one of the 8 input features from the input time series

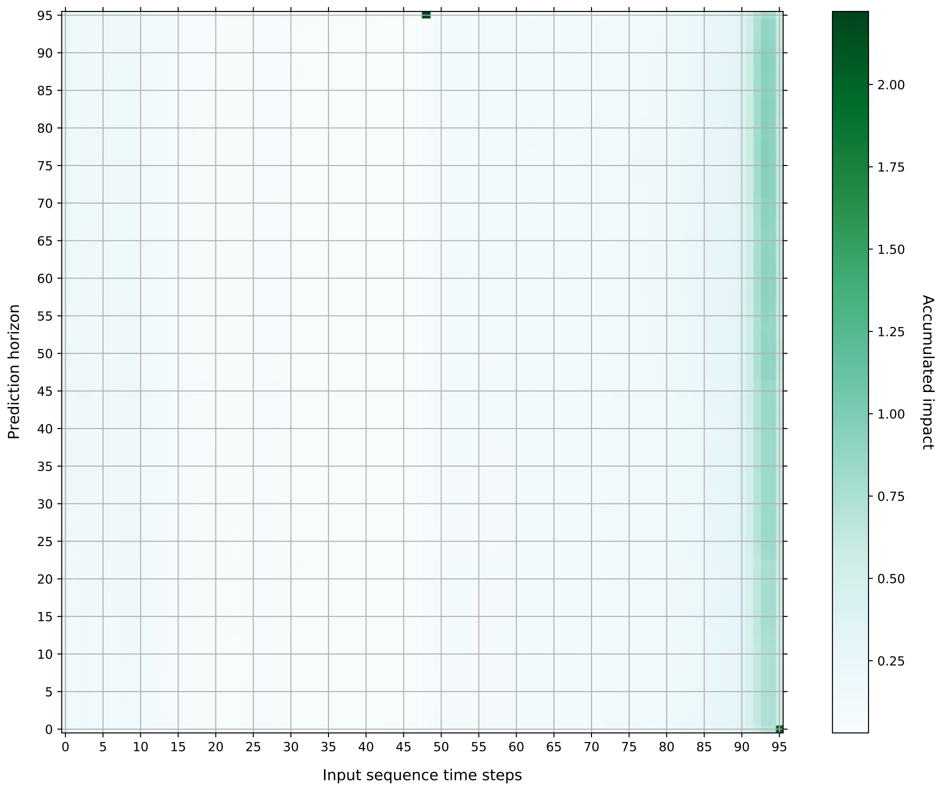


Figure 3. Accumulated impact of all the input features aggregated by the aggregation of the absolute values of their attribution scores over all the output features for each time step of the input sequence and of the output prediction horizon.

dataset. Furthermore, analyzing more recent transformer-based models' designs from this XAI perspective can lead to more impactful and relevant findings regarding model performance and suitability for the LTSF task. Additionally, looking more deeply into these findings with different XAI methods might yield a solution to exactly where the problem is with these types of models, so they can be adapted and improved.

ACKNOWLEDGEMENTS

This work has been funded by the Research Council of Norway through the EXAIGON project, project number 304843.

REFERENCES

[1] G. Mahalakshmi, S. Sridevi, and S. Rajaram, "A survey on forecasting of time series data," in *2016 International Conference on Computing Technologies and Intelligent Data Engineering (ICCTIDE'16)*, Jan. 2016, pp. 1–8. DOI: 10.1109/ICCTIDE.2016.7725358.

[2] M. Mudelsee, *Climate time series analysis. Classical statistical and bootstrap methods.* (Atmospheric and Oceanographic Sciences Library), Second Edition. Springer, 2014, vol. 51, ISBN: 978-3-319-04449-1. DOI: 10.1007/978-3-319-04450-7.

[3] E. J. Topol, "High-performance medicine: The convergence of human and artificial intelligence," *Nature Medicine*, vol. 25, no. 1, pp. 44–56, Jan. 2019, Publisher: Nature Publishing Group, ISSN: 1546-170X. DOI: 10.1038/s41591-018-0300-7.

[4] B. Lim and S. Zohren, "Time-series forecasting with deep learning: A survey," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 379, no. 2194, p. 20 200 209, Feb. 2021, Publisher: Royal Society. DOI: 10.1098/rsta.2020.0209.

[5] J. Feyrer, "Trade and Income—Exploiting Time Series in Geography," *American Economic Journal: Applied Economics*, vol. 11, no. 4, pp. 1–35, Oct. 2019, ISSN: 1945-7782. DOI: 10.1257/app.20170616.

[6] A. C. Harvey, "ARIMA models," in *Time series and statistics*, J. Eatwell, M. Milgate, and P. Newman, Eds., London: Palgrave Macmillan UK, 1990, pp. 22–24, ISBN: 978-1-349-20865-4. DOI: 10.1007/978-1-349-20865-4_2.

[7] K. Yunus, T. Thiringer, and P. Chen, "ARIMA-Based Frequency-Decomposed Modeling of Wind Speed Time Series," *IEEE Transactions on Power Systems*, vol. 31, no. 4, pp. 2546–2556, Jul. 2016, Conference Name: IEEE Transactions on Power Systems, ISSN: 1558-0679. DOI: 10 . 1109 / TPWRS.2015.2468586.

[8] K. W. Lau and Q. H. Wu, "Local prediction of non-linear time series using support vector regression," *Pattern Recognition*,

vol. 41, no. 5, pp. 1539–1547, May 2008, ISSN: 0031-3203. DOI: 10.1016/j.patcog.2007.08.013.

[9] M. G. Frei and I. Osorio, "Intrinsic time-scale decomposition: Time–frequency–energy analysis and real-time filtering of non-stationary signals," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 463, no. 2078, pp. 321–342, Aug. 2006, Publisher: Royal Society. DOI: 10.1098/rspa.2006.1761.

[10] S. J. Gershman and D. M. Blei, "A tutorial on Bayesian nonparametric models," *Journal of Mathematical Psychology*, vol. 56, no. 1, pp. 1–12, Feb. 2012, ISSN: 0022-2496. DOI: 10.1016/j.jmp.2011.08.004.

[11] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural Networks*, vol. 4, no. 2, pp. 251–257, Jan. 1991, ISSN: 0893-6080. DOI: 10.1016/0893-6080(91)90009-T.

[12] Y. Lu and J. Lu, "A Universal Approximation Theorem of Deep Neural Networks for Expressing Probability Distributions," in *Advances in Neural Information Processing Systems*, vol. 33, Curran Associates, Inc., 2020, pp. 3094–3105.

[13] T. Kolarik and G. Rudorfer, "Time series forecasting using neural networks," *ACM SIGAPL APL Quote Quad*, vol. 25, no. 1, pp. 86–94, 1994, ISSN: 0163-6006. DOI: 10.1145/190468.190290.

[14] B. Zhao, H. Lu, S. Chen, J. Liu, and D. Wu, "Convolutional neural networks for time series classification," *Journal of Systems Engineering and Electronics*, vol. 28, no. 1, pp. 162–169, Feb. 2017, Conference Name: Journal of Systems Engineering and Electronics, ISSN: 1004-4132. DOI: 10.21629/JSEE.2017.01.18.

[15] G. Grudnitski and L. Osburn, "Forecasting S&P and gold futures prices: An application of neural networks," *Journal of Futures Markets*, vol. 13, pp. 631–643, Sep. 1993. DOI: 10.1002/fut.3990130605.

[16] H. Hewamalage, C. Bergmeir, and K. Bandara, "Recurrent Neural Networks for Time Series Forecasting: Current status and future directions," *International Journal of Forecasting*, vol. 37, no. 1, pp. 388–427, Jan. 2021, ISSN: 0169-2070. DOI: 10.1016/j.ijforecast.2020.06.008.

[17] U. M. Sirisha, M. C. Belavagi, and G. Attigeri, "Profit Prediction Using ARIMA, SARIMA and LSTM Models in Time Series Forecasting: A Comparison," *IEEE Access*, vol. 10, pp. 124 715–124 727, 2022, Conference Name: IEEE Access, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2022.3224938.

[18] A. Vaswani *et al.*, "Attention is All you Need," in *Advances in Neural Information Processing Systems*, vol. 30, Curran Associates, Inc., Jun. 2017, p. 11.

[19] H. Zhou *et al.*, "Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting," in *Proceedings of the AAAI Conference on Artificial Intelligence*, ISSN: 2374-3468, 2159-5399 Issue: 12 Journal Abbreviation: AAAI, vol. 35, May 2021, pp. 11 106–11 115. DOI: 10.1609/aaai.v35i12.17325.

[20] H. Wu, J. Xu, J. Wang, and M. Long, *Autoformer: Decomposition Transformers with Auto-Correlation for Long-Term Series Forecasting*, arXiv:2106.13008 [cs], Jan. 2022. DOI: 10.48550/arXiv.2106.13008.

[21] G. Woo, C. Liu, D. Sahoo, A. Kumar, and S. Hoi, *ETSformer: Exponential Smoothing Transformers for Time-series Forecasting*, arXiv:2202.01381 [cs], Jun. 2022. DOI: 10.48550/arXiv.2202.01381.

[22] Y. Zhang and J. Yan, "Crossformer: Transformer Utilizing Cross-Dimension Dependency for Multivariate Time Series Forecasting," in *ICLR Proceedings 2023*, Sep. 2022, p. 21.

[23] Q. Wen *et al.*, *Transformers in Time Series: A Survey*, arXiv:2202.07125 [cs, eess, stat], May 2023.

[24] A. Zeng, M. Chen, L. Zhang, and Q. Xu, "Are Transformers Effective for Time Series Forecasting?" *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 9, pp. 11 121–11 128, Jun. 2023, Number: 9, ISSN: 2374-3468. DOI: 10.1609/aaai.v37i9.26317.

[25] S. M. Lundberg and S.-I. Lee, "A Unified Approach to Interpreting Model Predictions," in *Advances in Neural Information Processing Systems*, vol. 30, Curran Associates, Inc., 2017.

[26] M. Sundararajan, A. Taly, and Q. Yan, "Axiomatic attribution for deep networks," in *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ser. ICML'17, Sydney, NSW, Australia: JMLR.org, 2017, pp. 3319–3328.

[27] S. Bach *et al.*, "On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation," en, *PLOS ONE*, vol. 10, no. 7, e0130140, Jul. 2015, Publisher: Public Library of Science, ISSN: 1932-6203. DOI: 10.1371/journal.pone.0130140.

[28] E. Voita, D. Talbot, F. Moiseev, R. Sennrich, and I. Titov, "Analyzing Multi-Head Self-Attention: Specialized Heads Do the Heavy Lifting, the Rest Can Be Pruned," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, A. Korhonen, D. Traum, and L. Màrquez, Eds., Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 5797–5808. DOI: 10.18653/v1/P19-1580.

[29] S. Abnar and W. Zuidema, "Quantifying Attention Flow in Transformers," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, D. Jurafsky, J. Chai, N. Schluter, and J. Tetreault, Eds., Online: Association for Computational Linguistics, Jul. 2020, pp. 4190–4197. DOI: 10.18653/v1/2020.acl-main.385.

[30] H. Chefer, S. Gur, and L. Wolf, *Transformer Interpretability Beyond Attention Visualization*, en, arXiv:2012.09838 [cs], Apr. 2021.

[31] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in *Proceedings of the 2019 Conference of the North*, Minneapolis, Minnesota: Association for Computational Linguistics, 2019, pp. 4171–4186. DOI: 10.18653/v1/N19-1423.

[32] J. Chen, X. Li, L. Yu, D. Dou, and H. Xiong, "Beyond Intuition: Rethinking Token Attributions inside Transformers," en, *Transactions on Machine Learning Research*, Oct. 2022, ISSN: 2835-8856.

[33] O. Barkan *et al.*, "Grad-SAM: Explaining Transformers via Gradient Self-Attention Maps," in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, Virtual Event Queensland Australia: ACM, Oct. 2021, pp. 2882–2887, ISBN: 978-1-4503-8446-9. DOI: 10.1145/3459637.3482126.

[34] A. Dosovitskiy *et al.*, *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*, arXiv:2010.11929 [cs], Jun. 2021. DOI: 10.48550/arXiv.2010.11929.

[35] R. Kashefi, L. Barekatain, M. Sabokrou, and F. Aghaeipoor, *Explainability of Vision Transformers: A Comprehensive Review and New Perspectives*, en, arXiv:2311.06786 [cs], Nov. 2023.

[36] M. T. Ribeiro, S. Singh, and C. Guestrin, *"Why Should I Trust You?": Explaining the Predictions of Any Classifier*, arXiv:1602.04938 [cs, stat], Aug. 2016.

[37] R. R. Selvaraju *et al.*, "Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization," in *2017 IEEE International Conference on Computer Vision (ICCV)*, ISSN: 2380-7504, Oct. 2017, pp. 618–626. DOI: 10.1109/ICCV.2017.74.

[38] A. Theissler, F. Spinnato, U. Schlegel, and R. Guidotti, "Explainable AI for Time Series Classification: A Review, Taxonomy and Research Directions," *IEEE Access*, vol. 10,

pp. 100 700–100 724, 2022, Conference Name: IEEE Access, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2022.3207765.

[39] D. Mercier, A. Dengel, and S. Ahmed, "PatchX: Explaining Deep Models by Intelligible Pattern Patches for Time-series Classification," in *2021 International Joint Conference on Neural Networks (IJCNN)*, arXiv:2102.05917 [cs], Jul. 2021, pp. 1–8. DOI: 10.1109/IJCNN52387.2021.9533293.

[40] R. Guidotti and A. Monreale, "Designing Shapelets for Interpretable Data-Agnostic Classification," Jul. 2021, pp. 532–542. DOI: 10.1145/3461702.3462553.

[41] V. Shalaeva, S. Alkhoury, J. Marinescu, C. Amblard, and G. Bisson, "Multi-operator Decision Trees for Explainable Time-Series Classification," en, in *Information Processing and Management of Uncertainty in Knowledge-Based Systems. Theory and Foundations*, J. Medina *et al.*, Eds., Cham: Springer International Publishing, 2018, pp. 86–99, ISBN: 978-3-319-91473-2. DOI: 10.1007/978-3-319-91473-2_8.

[42] W. Tang, L. Liu, and G. Long, *Interpretable Time-series Classification on Few-shot Samples*, arXiv:2006.02031 [cs, stat], Jul. 2020. DOI: 10.48550/arXiv.2006.02031.

[43] A. A. Ismail, M. Gunady, H. C. Bravo, and S. Feizi, *Benchmarking Deep Learning Interpretability in Time Series Predictions*, arXiv:2010.13924 [cs, stat], Oct. 2020. DOI: 10.48550/arXiv.2010.13924.

[44] D. Smilkov, N. Thorat, B. Kim, F. Viégas, and M. Wattenberg, *SmoothGrad: Removing noise by adding noise*, arXiv:1706.03825 [cs, stat], Jun. 2017. DOI: 10.48550/arXiv.1706.03825.

[45] A. Shrikumar, P. Greenside, and A. Kundaje, *Learning Important Features Through Propagating Activation Differences*, arXiv:1704.02685 [cs], Oct. 2019. DOI: 10.48550/arXiv.1704.02685.

[46] H. Suresh *et al.*, "Clinical Intervention Prediction and Understanding using Deep Networks," *ArXiv*, May 2017.

[47] U. Schlegel, H. Arnout, M. El-Assady, D. Oelke, and D. A. Keim, *Towards a Rigorous Evaluation of XAI Methods on Time Series*, en, arXiv:1909.07082 [cs], Sep. 2019.

[48] G. Lai, W.-C. Chang, Y. Yang, and H. Liu, "Modeling Long- and Short-Term Temporal Patterns with Deep Neural Networks," in *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, ser. SIGIR '18, New York, NY, USA: Association for Computing Machinery, Jun. 2018, pp. 95–104, ISBN: 978-1-4503-5657-2. DOI: 10.1145/3209978.3210006.

[49] Harald Martens and Tormod Næs, "5.3.2 Outliers in prediction (goodness of fit)," in *Multivariate calibration*, John Wiley & Sons, Aug. 1992, p. 440, ISBN: 978-0-471-93047-1.

[50] J. A. Fernández Pierna, F. Wahl, O. E. de Noord, and D. L. Massart, "Methods for outlier detection in prediction," *Chemometrics and Intelligent Laboratory Systems*, Chemometrics 2002 S.I. Vol. 63, no. 1, pp. 27–39, Aug. 2002, ISSN: 0169-7439. DOI: 10.1016/S0169-7439(02)00034-5.