MTEX-CNN: Multivariate Time series EXplanations for Predictions with Convolutional Neural Networks

Roy Assaf*, Ioana Giurgiu, Frank Bagehorn and Anika Schumann
IBM Research, Zurich
roa@zurich.ibm.com

***Abstract*—In this work we present MTEX-CNN, a novel explainable convolutional neural network architecture which can not only be used for making predictions based on multivariate time series data, but also for explaining these predictions. The network architecture consists of two stages and utilizes particular kernel sizes. This allows us to apply gradient based methods for generating saliency maps for both the time dimension and the features. The first stage of the architecture explains which features are most significant to the predictions, while the second stage explains which time segments are the most significant. We validate our approach on two use cases, namely to predict rare server outages in the wild, as well as the average energy production of photovoltaic power plants based on a benchmark data set. We show that our explanations shed light over what the model has learned. We validate this by retraining the network using the most significant features extracted from the explanations and retaining similar performance to training with the full set of features.**

***Keywords*-Convolutional Neural Network, Explainable Machine Learning, Multivariate Time Series, Deep Learning**

I. Introduction

In order to accurately use multivariate time series data to predict future events, both the time and interaction dynamics between the different features have to be considered. However, this presents a challenge for traditional machine learning approaches whereby the time dynamics have to be manually extracted through feature engineering [1].

Deep neural networks (DNNs) are state-of-the-art in predictive tasks for temporal data [2], [3] and can learn meaningful representations from the data without the need for manual feature engineering. In this work we use a DNN architecture based on convolutional neural networks (CNNs). This is because, unlike commonly used recurrent neural network (RNN) architectures, CNNs learn to search for local patterns and do not need to look at the entire sequence. This makes them cheaper to train and can additionally fully benefit from GPU parallelisation all the while matching the performance of RNN-based architectures [4], [5]. This choice is further motivated by the explainability method that we use and which is mainly applicable to CNNs.

However, these networks are considered as black box, and suffer from lack of explainability, which is understanding the reasons for the model's behavior [6]. While explanatory artificial intelligence (XAI) [7], [8] has seen many recent advancements, the bulk of the literature focuses on image data, and explaining multivariate time series data remains largely unexplored.

To this end, we propose MTEX-CNN, a novel and explainable CNN-based architecture which can predict future events based on multivariate temporal data and explain these predictions. Our explanations are *spatial-temporal*, in that they can be used to understand which features during which time intervals are responsible for a given prediction, as well as explaining during which time segments was the joint contribution of all features most important for that prediction.

The contribution of this work is threefold: First, we propose a CNN architecture that allows us to use a gradient based explainability method to explain which features were most significant to the predictions, and which time segments were the most significant; Second, we apply our method on two use cases, and present explanations of the predictions along with a thorough discussion of the results; Third, we show that by retraining the network using a subset of the original features that the network deems most significant, we retain similar performance to training with the full set of features.

II. Related Work

There have been some attempts for providing explainability to DNNs in the context of time series data. For example in [9], the authors use a deep convolutional auto-encoder on f-MRI time series data. After training the model, the filters are extracted and visualized at each layer and then interpreted based on theoretical models of brain activities. In [3], using activation maximization the feature maps are visualized in an attempt to obtain an intuition about the patterns learned by the network. However, these works do not explain why the network made a specific decision for a particular data sample. And as the number of patterns gets larger which is usually the case in DNNs, such approaches become difficult to interpret.

Albeit these works, the two main methods used for explainability are perturbation-based methods, and gradient based methods. Perturbation-based methods can be considered model agnostic [10], it tends to be slow and does

*Corresponding author

2374-8486/19/$31.00 ©2019 IEEE
DOI 10.1109/ICDM.2019.00106

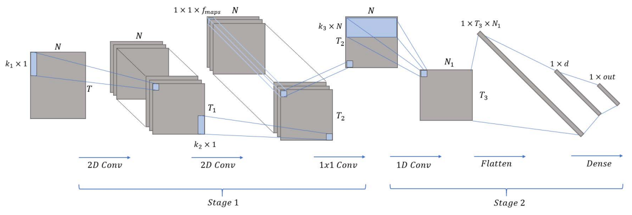

Figure 1. The two stage MTEX-CNN architecture as configured for the use cases.

not work well for deep neural networks due their non-linearities, which makes the result of these explanations highly dependent on the features that are altered [7]. On the other hand, gradient-based methods are seen as state-of-the-art when it comes to explaining deep neural networks [7], [8]. These methods work by computing the gradients over the input or some intermediate layer with respect to a given class output. In this work we consider such methods for extracting explanations from the predictions of the network.

III. MTEX-CNN

This section presents the proposed multivariate time series explanations architecture MTEX-CNN. The design of the network allows us to use a gradient based explainability method on the first stage of the network to explain which features were most significant to the predictions, and on the second stage to explain which time segments were the most significant.

A. Architecture

Stage 1 consists of stacked 2D convolution layers and terminates with one 1×1 convolution layer. The number of 2D convolution layers used may vary depending on the application. For demonstration purposes we will consider two stacked 2D convolution layers which reflects the network architecture that is implemented for the use cases. The network architecture can be seen in Figure 1.

For a generic 2D convolution filter $\mathfrak{F}^1 : \{0, \ldots, k-1\}$, the convolution operation C on each element $t : \{0, \ldots, T-1\}$ of the time series x_n. This operation is applied for every n, where $n : \{0, \ldots, N-1\}$ denotes a specific time series. Note that $k \leq T$. In the MTEX-CNN we intentionally restrict the kernel size of these 2D convolution layers to $k \times 1$, where k specifies the receptive field of the network over the time dimension. This is necessary and plays a key role in extracting the feature attention of the network, i.e., the contribution of the individual features with respect to a particular prediction. As a result of using such a kernel size, the network is forced to learn some A number of these kernels over all features. At inference the network computes A kernels over its input which result in feature maps $f_{maps} = \{1, \ldots, A\}$, where each feature map reflects the level of activation of its corresponding kernel indicating the response of that kernel pattern over the different features of the multivariate time series.

Note that we use half padding and a stride size set to (2×1). The half padding keeps the output of each convolution equal to the input, and the stride size keeps the feature size intact and reduces dimensionality of the time axis by a factor of 2, i.e., $T_2 = \frac{T}{4}$ in Figure 1. As a result the output of two 2D convolution layers should be equal to $(T_2 \times N)$.

As mentioned, the 2D convolution layers will be followed by a 1×1 convolution layer. This kind of convolutional layer is also called network in network (NIN) [11] and is in use in many state-of-the-art networks [12]. In particular, the 1×1 convolution applies a 3D filter $\mathfrak{F}^2$ of size $[1 \times 1 \times f_{maps}]$ where f_{maps} is the number of feature maps of the last layer of the stacked 2D convolutions. The 1×1 convolution filter is applied iteratively over one of the $T_2 \times N$ different slices trough the volume dimension of the feature maps and an element wise product is computed between that slice and the f_{maps} values in the filter. This learns to "compress" the feature maps, i.e. f_{maps} is reduced to 1. This allows us to apply a 1D convolution filter in stage 2 of the network.

Stage 2 consists of stacked 1D convolution layers followed by two dense layers, one of which is used as the output layer of the network. The number of 1D CNN layers used may vary depending on the application. Again, for demonstration purposes we will consider one 1D convolution layer similar to the network architecture implemented later for the use cases.

The 1D convolution layer consists of a $k \times n$ filter and therefore slides over the time axis only. This filter is crucial since we know that the interaction between the different time series plays an important role. The 1D convolution will allow us to account for such interactions with focus on the time aspect of the data. This operation is applied for every $t : \{0, ..., T-1\}$ and $o : \{0, ..., N-1\}$. Note that also here applies $k \leq T$.
Here, o, n are iterators over the same dimension and are used to avoid confusion. In our architecture: $k = k_3$ and $T = T_2$. The output of the 1D convolution layers is flattened and fed to the dense layers. The dense layer takes this 1D vector as input and is then connected to an output softmax activated dense layer which assigns probabilities of belonging to different classes.

B. Extracting explanations

When considering CNNs the principle is that deeper layers capture high level representations and global information. Therefore, by backpropagating the gradient of the score for a class label to a feature map layer with respect to every feature map of a specified convolutional layer, we can compute a set of weights capturing the contribution of every feature map with regards to that class. These weights are then used to compute a weighted combination of the feature maps which in turn generates a saliency map that is upsampled to fit the input data. In gradient-based methods for image data [13], this approach directly correlates each spatial location of the data to the attention pattern of the network for a specific class and provides a visual explanation of the network's choice. Specifically, in this work we take inspiration from a backpropagation gradient based method called Grad-CAM [14], which has been successfully applied in image data when using CNN-based networks [13], [14].

Putting things in perspective, we highlight here that the proposed architecture allows us to use Grad-CAM for explaining feature importance at a first stage. This is done by applying Grad-CAM over the output of the last 2D convolutional layer in stage 1. Then, the 1×1 convolution allows us to reduce the dimensionality of the output so that the 1D convolution can process the feature maps generated by the 2D convolutions. This then yields an output which accounts for the interactions between the different features of the multivariate time series. This is key for improving performance, and will allow us to use Grad-CAM once more. This is done to visualize the most influential time steps, where the combined contribution of all features matters most rather than the contribution of each feature in isolation. The combination of both Grad-CAM outputs is what allows us to provide spatio-temporal explanations for multivariate time series.

Consider the output of the last 2D convolution layer in Stage 1 which has produced $f_{maps} = \{1, \ldots, A\}$. We will call $A^g_{i,j}$ the activation of a generic unit g at location $\{i, j\}$. Let $Z = \sum_i \sum_j = N \times T_2$. We want to obtain an importance weight w^c_g associated to each feature map A^g for a class output c. Therefore, we compute the gradient of the score y^c with respect to the feature map A^g which is then globally averaged as $w^c_g = \frac{1}{Z} \sum_i \sum_j \frac{\delta y^c}{\delta A^g_{i,j}}$. We then use w^c_g to compute a weighted combination between all the feature maps for that particular class. A ReLU is then applied as:

$$L^c_{2D} = ReLU\big(\textstyle\sum_g w^c_g A^g\big) \tag{1}$$

The ReLU is used to remove the negative contributions for the sake of having cleaner visualizations. L^c_{2D} is used to find the areas in the input data that have mainly contributed in the decision of the network for class c.

We now implement the above to the feature maps of the 1D convolutions. We define A^o the generic 1D map of dimensions $[T_3 \times 1]$ where $o \in \{0, ..., N_1\}$ and N_1 is the number of maps. We will call A^o_h the activation of unit o at location h, where $h \in \{0, ..., T_3\}$. We want to obtain an importance weight q^c_o associated to each feature map A^o for a class output c. Therefore, we compute the gradient of the score y^c with respect to the feature map A^o which is then globally averaged as $q^c_o = \frac{1}{T_3} \sum_h \frac{\delta y^c}{\delta A^o_h}$. Similar to the 2D case, we then use q^c_o to compute a weighted combination between all the N_1 feature maps for that particular class. A ReLU is then applied as:

$$L^c_{1D} = ReLU\big(\textstyle\sum_{N_1} q^c_o A^o\big) \tag{2}$$

It can be seen that L^c_{2D} and L^c_{1D} are both associated to the same class, however they highlight different areas of the multivariate time series. The former highlights the contribution of time dynamics, while the latter highlights the contribution of the features.

IV. Experiments and Results

We validate our approach on two use cases, namely a benchmark used to predict the average energy production of photovoltaic power plants and a real-world scenario for predicting rare server outages. Explanations for these predictions are subsequently presented and thoroughly discussed and validated.

Regarding the MTEX-CNN architecture and hyperparameters, in stage 1 we use two convolutional layers with half padding. Both use ReLU activation, strides $s = (2, 1)$, and kernel sizes $k = (8, 1), k = (6, 1)$ respectively. The first layer outputs $f_{maps} = 64$ feature maps, and the second layer outputs $f_{maps} = 128$. Both layers are regularized using dropout with rate $r = 0.4$. This is followed by the 1×1 convolutional layer. Stage 2 consists of one convolutional layer with half padding. This uses a ReLU activation, stride $s = 2$, and kernel size $k = (4)$. This convolutional layer is regularized using a dropout with rate $r = 0.4$. It outputs $f_{maps} = 128$ feature maps, these are flattened and fed into a ReLU activated dense layer of dimension $d = 128$. This dense layer has kernel and bias L2 regularization set to 0.2. Finally the output layer consists of a dense layer whose dimension is the number of output classes. This output layer uses softmax activation and no regularization.

A. Use Case: Photovoltaic Plant Average Power Production

Here we apply MTEX-CNN to the multivariate time series dataset of the multi-plant PV energy forecasting challenge [15]. This is a multivariate time series where each time step represents hourly aggregated observations, and each day is represented by 19 time steps (PV plants are active from 02:00 to 20:00). Each time step consists of 7 features related to weather conditions, and 2 features collected from sensors placed on the plants. We use these features to predict the average energy generated over a period of 4 days ($\approx$80 time steps) in kW. The average power output is bucketed into 6 classes, $[(0-50), (50-100), (100-150), (150-200), (200-250), (250-300)]$, where the number of samples for each is $[178, 572, 698, 727, 959, 303]$ respectively.

After training, the network reaches 90% overall F1 score on all classes in validation, and 88% in testing. This can be seen in Table I, when all 9 features (100%) are used. After generating the predictions, we are able to visualize the network's attention on time and features. Here, a high

	% of features contribution											
	55%			75%			85%			100%		
	Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1
0-50	0.97	0.81	0.88	0.91	0.86	0.89	0.89	0.89	0.89	0.91	0.86	0.89
50-100	0.88	0.79	0.83	0.93	0.82	0.87	0.94	0.85	089	0.89	0.91	0.9
100-150	0.81	0.87	0.84	0.84	0.94	0.89	0.87	0.91	0.89	0.9	0.89	0.89
150-200	0.82	0.89	0.85	0.87	0.85	0.86	0.86	0.86	0.86	0.86	0.88	0.87
200-250	0.87	0.85	0.86	0.85	0.91	0.88	0.84	0.91	0.87	0.87	0.91	0.89
250-300	0.8	0.77	0.78	0.88	0.72	0.79	0.84	0.7	0.77	0.88	0.75	0.81
AVG	0.85	0.84	0.84	0.87	0.87	0.87	0.87	0.87	0.87	**0.88**	**0.88**	**0.88**

Table I
PREDICTION ACCURACY FOR ENERGY PRODUCTION ON THE PHOTOVOLTAIC DATASET, WHEN 55%, 75% AND 85% MOST CONTRIBUTING FEATURES ARE USED AS OPPOSED TO ALL FEATURES.

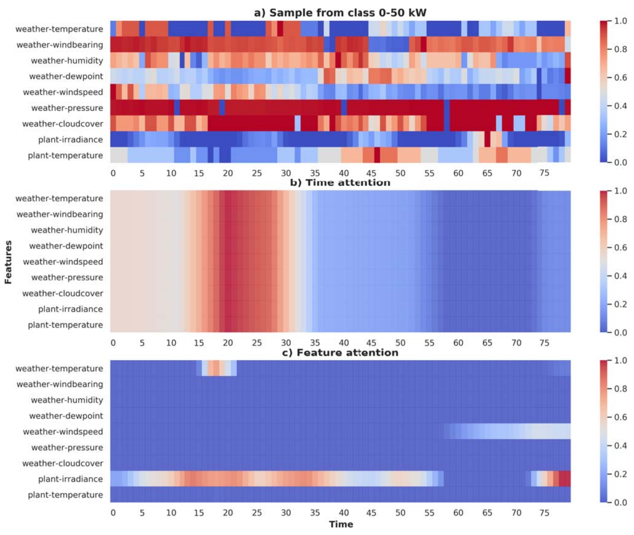

Figure 2. Time and feature attention corresponding to a prediction for a sample of class 0-50 kW

network attention is visualized in red, whereas little or no attention is shown in blue.

In Figure 2 we show an example where the network has successfully predicted the energy generation as belonging to class 0-50 kW. We notice from c) that the network puts considerable attention on the PV plant irradiance feature when it is very low. The network also considers the weather temperature and the wind speed at a time step where these are, as well, low. In b), which corresponds to the joint contribution of all features, the network shows more attention to the first half of the sample (representing two days), which seems to correspond to unfavorable weather conditions for PV energy generation (high cloud coverage, low plant temperature, and low weather temperature).

1) Feature importance and validation: For the sake of further validating the explanations obtained with MTEX-CNN, we generate predictions over the validation set and collect only the true positive sample indices. Saliency scores are then computed for all the samples that correspond to these indices. We then compute a sum over the batch and time axis for every feature. This then yields a scoring

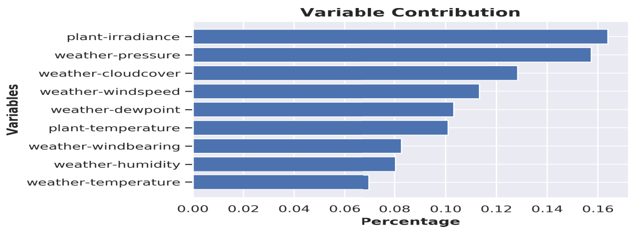

Figure 3. Feature importance across all true positive samples in the PV dataset.

	% of KPIs contribution											
	55%			75%			95%			100%		
	Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1
No outage	0.99	1	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
Outage	0.36	0.17	0.23	0.42	0.3	0.35	0.37	0.35	0.36	0.44	0.34	0.38
AVG	0.98	0.98	0.98	0.98	0.99	0.98	0.98	0.98	0.98	0.98	0.99	0.98

Table II
PREDICTION ACCURACY ON THE SERVER OUTAGE PREDICTION DATASET, WHEN 55%, 75% AND 95% MOST CONTRIBUTING KPIS ARE USED AS OPPOSED TO USING ALL KPIS

across all features, i.e., features that contributed most towards correct classifications would have a higher score, as seen in Figure 3. Note that the irradiance feature is the most contributing feature, followed by cloud coverage and weather temperature. This makes a lot of sense since these are the most important factors that lead to power generation in photovoltaic plants.

The benefit from running this validation experiment is twofold. First, it allows to obtain a global explanation for our model from a features perspective. Second, it provides means for further feature selection prior to training the model. In fact, as shown in Table I, we retrain the model with a subset of the features that represent 55%, 75% and 85% of the overall contribution. These correspond to 4, 6 and 7 features, respectively. As before, we report precision, recall and F1 score for each of the 6 output classes. The expectation is that the model performance drops when less features are used. However, if MTEX-CNN correctly identifies the most contributing of features, we expect the drop in accuracy will be small. Indeed, when 75% and 85% of the features are used, the model achieves 87% performance (as opposed to 88% when trained on all features). When less than half of the features are considered, the performance drop is only 3%.

B. Use Case: Server Outage Prediction

Here we tackle the problem of predicting and explaining rare server outages events. Because of the huge imbalance between the outages and non-outages classes, on such problems models typically yield many false positives. Therefore to trust the predictions, support teams need to understand how these were derived from the model and so it is imperative that the predictions are explainable. The data consists of key performance indicators (KPIs) that are collected periodically by Operating System (OS) level monitoring agents in a large IT environment comprised of several thousand servers. These are typical indicators that reflect performance of the CPU, memory, storage and network components. The time series consist of 47 KPIs, each time step represents an aggregate over 15 minute interval observations. For the prediction task, we use 216 time steps (2 days and 6 hours) to predict outages over the next 7 days.

First, we report the accuracy of the model in Table II. Next, we show the explanations generated by MTEX-CNN for 2 examples, namely a true positive and a false positive.

1) Example #1 – True Positive: The model predicts a server outage with 98.9% probability. We validate the prediction via a corresponding outage ticket, which mentions that the server could not be reached via external communication. The heat maps for the time and feature attentions are shown in Figure 4. The KPIs show that the server has a large paging file, as well as a high number of allocated pages in both the non-paged and the paged pool. Around 3 hours before the outage, both the maximum CPU and the number of pages read or written from or to the disk cache start to show a significant increase. This hints towards a situation where an application requests more pages to be stored in the memory than the OS can provide, resulting in the OS paging out memory areas used by other applications. The attention maps show that the model pays special attention to the last 2 hours before the outage and, in particular, to the KPI describing how often the cache read-write operations went over a certain defined threshold. Some attention is paid to the CPU related features, namely the CPU maximum and minimum values.

2) Example #2 – False Positive: The model predicts an outage with a probability of 66.0%, when no related outage tickets exist for the server during the prediction window. As before, we show the corresponding heat maps in Figure 5. The model pays attention to the last several hours, mostly to the same swapping related feature as before, and to a lesser degree to the CPU related features. The KPIs show that the server periodically performs several large swapping operations, with some of them lasting up to several hours. Each time, the situation normalizes again without the server running into an outage. The last such swapping happens at the very end of the time series, while at the same time the maximum CPU is lower than in our first example. Given this, and the fact that the swapping correlates with lower CPU utilization, domain experts would conclude that such behavior is consistent with workloads being run periodically on the server. However, because such examples are rare in the data set, the model wrongly predicts an outage based on similarities with true positives.

3) KPI importance and validation: Figure 6 shows the importance of the top 9 KPIs relative to the true positive samples. Note that the most contributing KPIs are paging

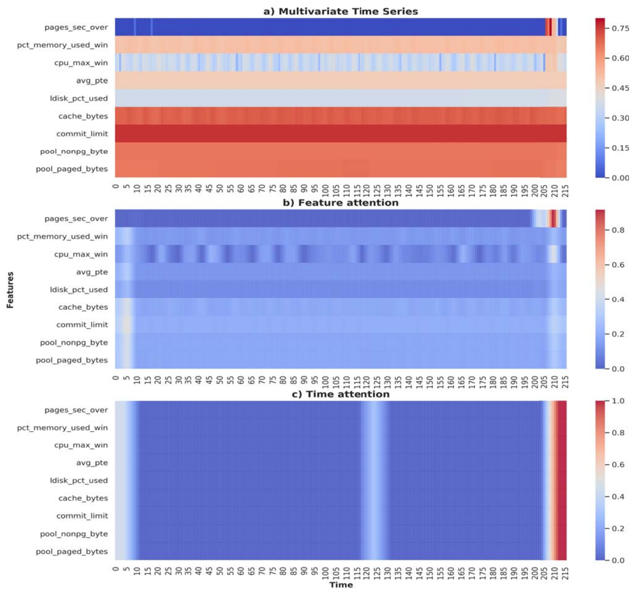

Figure 4. Time and feature attention corresponding to a true positive sample.

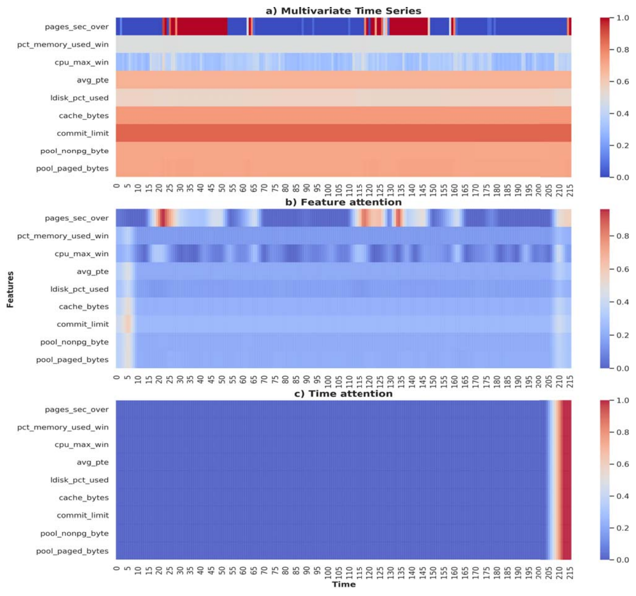

Figure 5. Time and feature attention corresponding to a false positive sample.

related. This matches the previous explanations of server issues caused by massive memory page in/out operations. The first non-paging KPI is the logical disk usage, in agreement with the observation, that high disk usage or even disk full events can be primary causes for outages.

By using this ranking, we conduct the same experiment as on the photovoltaic dataset and retrain our model with a subset of the features. More specifically, we consider the

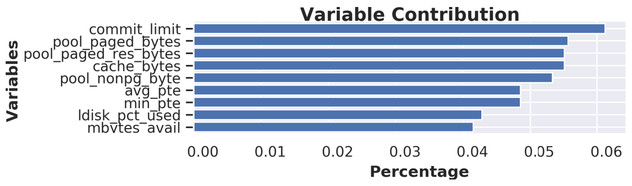

Figure 6. Feature importance across all true positive samples in the server outage prediction data set.

features that represent 55%, 75% and 95% of the overall contribution. This corresponds to 11, 18 and 31 features, respectively. We report the precision, recall and F1 score for both classes in Table II. While for the no outage class, using less KPIs has no impact, precision and recall decreases as we decrease the number of KPIs, by at most 8% and 17%, respectively. The most robust accuracy is obtained when 18 features (38%) are used instead of the full set of 47, since F1 score only drops by 3%. The reduction to only 11 of the KPIs causes the model to miss certain outages that are signaled by the excluded features and results in a significantly lower recall. Being able to achieve comparable performance while using a fraction of the initial data set has a multitude of advantages. First, support teams and domain experts need to inspect fewer KPIs and validate less complex explanations. Second, it provides an opportunity for business to monitor fewer sensor metrics, but optimize those with regards to the configuration like thresholds. Third, it allows to gain faster and deeper insight into servers' behavior.

V. Conclusion

In this paper, we proposed MTEX-CNN, a novel CNN architecture for explaining prediction tasks based on multivariate time series data. This employs a combination of 2D convolutions with $k \times 1$ kernels in order to capture feature relevance. Further, it is combined with 1D convolutions with $k \times N$ kernels. This in turn captures a combined contribution of all features over time rather than the contribution of each feature in isolation. A gradient-based method is implemented to back-propagate gradients relative to a specific class and visualize the attention patterns of the network over multivariate time series. MTEX-CNN is an end-to-end explainable CNN network that can simultaneously visualize the network's attention over both time and feature dimensions. We demonstrated our approach and explanations on two use cases. In addition, we showed that by retraining the network using a subset of the original features that the network deems most significant, we retain similar performance to training with the full set of features.

Acknowledgments

This work received funding from the EU H2020 project ROMEO (grant agreement No. 745625), and from SERI (Swiss State Secretariat for Education, Research and Innovation). The dissemination of results herein reflects only the authors' view.

References

[1] T. Górecki and M. Łuczak, "Multivariate time series classification with parametric derivative dynamic time warping," *Expert Systems with Applications*, vol. 42, no. 5, pp. 2305–2312, 2015.

[2] H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, "Deep learning for time series classification: a review," *arXiv preprint arXiv:1809.04356*, 2018.

[3] Y. Zheng, Q. Liu, E. Chen, Y. Ge, and J. L. Zhao, "Exploiting multi-channels deep convolutional neural networks for multivariate time series classification," *Frontiers of Computer Science*, vol. 10, no. 1, pp. 96–112, 2016.

[4] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin, "Convolutional sequence to sequence learning," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 1243–1252.

[5] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," *arXiv preprint arXiv:1803.01271*, 2018.

[6] L. H. Gilpin, D. Bau, B. Z. Yuan, A. Bajwa, M. Specter, and L. Kagal, "Explaining explanations: An approach to evaluating interpretability of machine learning," *arXiv preprint arXiv:1806.00069*, 2018.

[7] M. Ancona, E. Ceolini, C. Öztireli, and M. Gross, "A unified view of gradient-based attribution methods for deep neural networks," in *NIPS 2017-Workshop on Interpreting, Explaining and Visualizing Deep Learning*. ETH Zurich, 2017.

[8] J. Adebayo, J. Gilmer, M. Muelly, I. Goodfellow, M. Hardt, and B. Kim, "Sanity checks for saliency maps," in *Advances in Neural Information Processing Systems*, 2018, pp. 9505–9515.

[9] H. Huang, X. Hu, Y. Zhao, M. Makkie, Q. Dong, S. Zhao, L. Guo, and T. Liu, "Modeling task fmri data via deep convolutional autoencoder," *IEEE transactions on medical imaging*, vol. 37, no. 7, pp. 1551–1561, 2018.

[10] M. T. Ribeiro, S. Singh, and C. Guestrin, "Why should i trust you?: Explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. ACM, 2016, pp. 1135–1144.

[11] M. Lin, Q. Chen, and S. Yan, "Network in network," *arXiv preprint arXiv:1312.4400*, 2013.

[12] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

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

[14] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *2017 IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2017, pp. 618–626.

[15] M. Ceci, R. Corizzo, F. Fumarola, D. Malerba, and A. Rashkovska, "Predictive modeling of pv energy production: How to set up the learning task for a better prediction?" *IEEE Transactions on Industrial Informatics*, vol. 13, no. 3, pp. 956–966, 2017.