Labaien, J., Zugasti, E., De Carlos, X. (2020). *Contrastive Explanations for a Deep Learning Model on Time-Series Data*. In: Song, M., Song, IY., Kotsis, G., Tjoa, A.M., Khalil, I. (eds) Big **Data Analytics and Knowledge Discovery. DaWaK 2020. Lecture Notes in Computer Science**(), vol 12393.This version of the paper has been accepted for publication, after peer review (when applicable) and is subject to Springer Nature's AM terms of use, but is not the Version of Record and does not reflect post-acceptance improvements, or any corrections. The Version of Record is available online at: https://doi.org/10.1007/978-3-030-59065-9_19

Contrastive Explanations for a Deep Learning Model on Time-Series Data

Jokin Labaien[1,*] Ekhi Zugasti[2]
Xabier De Carlos[1]

[1]Ikerlan Technology Research Centre,
Basque Research and Technology Alliance (BRTA),
P° J.M. Arizmediarrieta, 2, 20500 Arrasate-Mondragón, Spain
{jlabaien,xdecarlos}@ikerlan.es

[2]Data Analysis and Cybersecurity Group, Mondragon University,
20500 Arrasate-Mondragón, Spain
ezugasti@mondragon.edu

*Corresponding author

Abstract

In the last decade, with the irruption of Deep Learning (DL), artificial intelligence has risen a step concerning previous years. Although Deep Learning models have gained strength in many fields like image classification, speech recognition, time-series anomaly detection, etc. these models are often difficult to understand because of their lack of interpretability. In recent years an effort has been made to understand DL models, creating a new research area called Explainable Artificial Intelligence (XAI). Most of the research in XAI has been done for image data, and little research has been done in the time-series data field. In this paper, a model-agnostic method called Contrastive Explanation Method (CEM) is used for interpreting a DL model for time-series classification. Even though CEM has been validated in tabular data and image data, the obtained experimental results show that CEM is also suitable for interpreting deep learning models that work with time-series data.

1 Introduction

Nowadays, many systems are monitored by multiple sensors, which provide data on how the system is evolving, and consequently research on temporal data has increased in recent years. DL algorithms are becoming really powerful, also for time-series data, in which the Long Short-Term Memory (LSTM) networks [9] are a key part of many state-of-the-art architectures. LSTMs are capable of preserving information from long term dependencies, and have proven to be very effective in processing temporal data [12].

Even though DL has gained strength, DL models are often considered black-boxes due to the lack of interpretability. Due to this problem, a new research area has been created, called Explainable Artificial Intelligence (XAI), which is focused on DL model interpretation. A lot of work has been done in recent years on interpretation issues, as many techniques have been proposed to facilitate the understanding of DL models [1, 2], such as LRP [3], SHAP [10], LIME [11], Integrated Gradients [14], CEM [6], etc. These methods have been applied especially to image data [13, 15], and more research is needed for time-series data [7, 8].

CEM [6] is a perturbation-based method that provides local explanations. Although CEM offers two ways to interpret a model, either using pertinent negatives (PN) or pertinent positives (PP), in this paper, PNs are used. Unlike other methods, the idea behind this paper is that applying CEM to time series data can allow us to give explanations such as "this time series

is classified as class y because a particular point or group of points have value v (PP) when they should have value w (PN)". The authors apply CEM to a time series classification problem, concluding that this kind of explanation is viable in time series data.

This article is structured as follows. Section 2 provides the theoretical background of the DL methods used in the experiment and it briefly explains the CEM method. Section 3 describes the methodology used in the experiments. Section 4 describes the dataset used, the experimental framework and the hyperparameters of the models. Section 5 discusses the experimental results. Finally, in Sect. 6 the conclusions and future works are exposed.

2 Background

2.1 Long Short-Term Memory Networks

Long Short-Term Memory (LSTM) [9] networks are a variation of the standard Recurrent Neural Networks (RNN). Traditional RNNs have shown to be useful in many problems, but they suffer from the vanishing gradient and exploding gradient problem, which leads to failing to detect long term dependencies in practice. To overcome this problem, LSTMs include some cells in their internals that control how to maintain the information in memory for long periods of time.

LSTMs, like all the RNNs, have the form of a chain of repeating modules, in which the knowledge is transferred through time, from one module to another. Each module of an LSTM is called "unit", and each "unit" is composed of four neural networks, which interact with each other to process the input data.

One of the keys of the success of the LSTMs is the cell state (C_t). The function of the cell state is to transfer relative information throughout the entire sequence chain. As it is shown in Fig. 1, the cell state passes from one unit to another with some minor linear interactions, facilitating the learning of long-term dependencies. Moreover, each LSTM unit has its own hidden state (h), that manages the internal state of the unit. The cell state information is updated by three gates: forget gate (f_t), input gate (i_t) and output gate (o_t). The forget gate decides what information of the cell state to keep each time, forgetting the unnecessary information. The input gate i_t decides which values of the cell state have to be updated, and these values are updated by adding the new information stored in the candidate vector $\tilde{C}_t$ and forgetting the information decided to forget by the forget gate f_t. Finally, it has to be decided what is going to be the output of the unit, and the output gate o_t and the hidden state h_t are created. The whole process is summarized in Eq. (1).

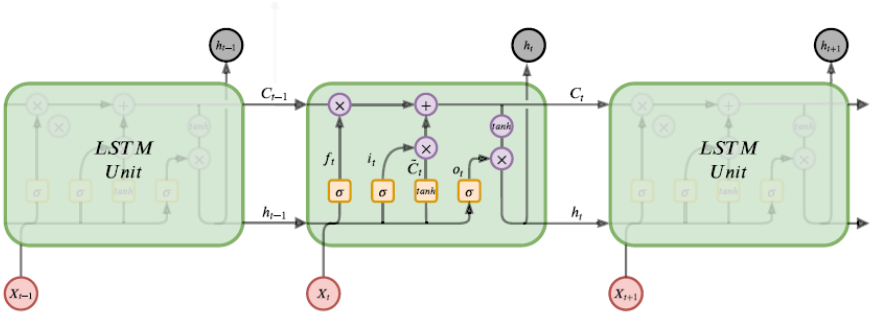

Figure 1: A visualization of an LSTM cell.

$$
\begin{aligned}
f_t &= \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \\
i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \\
\tilde{C}_t &= \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \\
C_t &= f_t * C_{t-1} + i_t * \tilde{C}_t \\
o_t &= \sigma(W_o[h_{t-1}, x_t] + b_o) \\
h_t &= o_t * \tanh(C_t)
\end{aligned}
\tag{1}
$$

W_x, b_x being the weights and the biases of each gate respectively, $x \in \{f, i, C, o\}$.

2.2 Autoencoders

Autoencoders (AE) [4] are neural networks trained unsupervisedly to reconstruct the input data. The AEs consist of two neural networks: an encoder and a decoder. The AEs can be either multilayer perceptrons, convolutional neural networks, recurrent neural networks, etc.

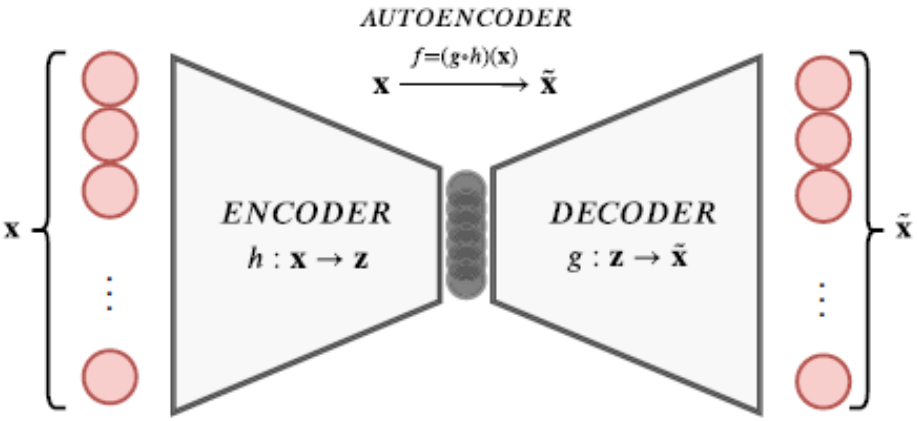

Figure 2: A general structure of an AE.

In Fig. 2 the structure of an AE is shown. The encoder can be defined as a function $\phi : x \in \mathbb{R}^n \rightarrow z \in \mathbb{R}^d$, where $d < n$, that tries to compress the data of the input layer into a lower dimensional latent vector, containing the most important features of it. On the other hand, the decoder can be defined as a function $g : z \in \mathbb{R}^d \rightarrow \tilde{x} \in \mathbb{R}^n$ that takes the compressed latent vector of the input and it decompresses into features that closely matches the original input data. So, summarizing, an AE can be seen as a function f that maps an input $x \in \mathbb{R}^n$ into its reconstruction $\tilde{x} \in \mathbb{R}^n$, trying to find the best parameters of the network for minimizing a loss function $L(x, \tilde{x})$, called reconstruction loss, that commonly is a Mean Squared Error (MSE) or a Mean Average Error (MAE).

2.3 Contrastive Explanation Method

Contrastive Explanations Method (CEM) [6] is a perturbation-based model-agnostic method that provide local explanations. The method consists of solving two different optimization problems, one for finding Pertinent Negatives (PN) and the other for finding Pertinent Positives (PP).

Finding Pertinent Negatives. Let x_0 be an input of a black-box model f, $f(x_0)$ the prediction given and y_0 its corresponding class. Let $AE(\cdot)$ be an autoencoder trained for reconstructing an input. Denoting X/x_0 to the space of missing parts with respect to x_0, finding

pertinent negatives consist of finding an interpretable minimal perturbation $\delta \in X/x_0$ such that $\arg\max_i [f(x_0)]_i \neq \arg\max_i [f(x_0+\delta)]_i$. For finding pertinent negatives, the authors propose solving this optimization problem:

$$\min_{\delta \in X/x_0} c \cdot f_\kappa^{\text{neg}}(x_0, \delta) + \beta\|\delta\|_1 + \|\delta\|_2^2 + \gamma\|x_0 + \delta - AE(x_0+\delta)\|_2^2 \tag{2}$$

where $c, \beta, \gamma \geq 0$ are regularization parameters. The second and the third terms, $\beta\|\delta\|_1$ and $\|\delta\|_2^2$ respectively, are jointly called the elastic net regularizer and they are used for efficient feature selection in high-dimensional learning spaces. The third term $\gamma\|x_0 + \delta - AE(x_0+\delta)\|_2^2$ ensures that the modified input $x_0 + \delta$ is close to the data manifold. The first term $f_\kappa^{\text{neg}}(x_0, \delta)$ is defined in this way:

$$f_\kappa^{\text{neg}}(x_0, \delta) = \max\{[f(x_0+\delta)]_{y_0} - \max_{i \neq y_0}[f(x_0+\delta)]_i, -\kappa\} \tag{3}$$

where $[f(x_0+\delta)]_i$ is the score given by the model f to the i-th class prediction. Introducing $f_\kappa^{\text{neg}}(x_0, \delta)$ to the Eq. 2, ensures $x_0 + \delta$ to be predicted as a different class than y_0. The parameter $\kappa \geq 0$ is a confidence parameter to control the separation between $[f(x_0+\delta)]_{y_0}$ and $\max_{i \neq y_0}[f(x_0+\delta)]_i$.

Finding Pertinent Positives. For finding pertinent positives, let $X \cap x_0$ be the space of existing components of x_0 and let $\delta \in X \cap x_0$ be an interpretable perturbation such that removing it from x_0 the prediction is still the same, i.e $\arg\max_i [f(x_0)]_i = \arg\max_i [f(\delta)]_i$. To this end, similar to finding pertinent negatives, the following optimization problem needs to be solved:

$$\min_{\delta \in X \cap x_0} c \cdot f_\kappa^{\text{neg}}(x_0, \delta) + \beta\|\delta\|_1 + \|\delta\|_2^2 + \gamma\|\delta - AE(\delta)\|_2^2 \tag{4}$$

where the first term $f_\kappa^{\text{neg}}(x_0, \delta)$ is defined in this way:

$$f_\kappa^{\text{neg}}(x_0, \delta) = \max\{\max_{i \neq y_0}[f(\delta)]_i - [f(\delta)]_{y_0}, -\kappa\} \tag{5}$$

To solve the optimization problems (2) and (4), a projected fast iterative shrinkage-thresholding algorithm, called FISTA [5], is used.

3 Methodology

In this paper a model proposed for a multiclass time-series classification problem is interpreted using CEM. As described in the previous section, the CEM method consist of solving the optimization problems of Eqs. (2) and (4). Thus, first of all, a classification model f has to be proposed and an AE needs to be defined for ensuring that the changed input is close to the data manifold. Therefore, in this section the proposed classification model and the AE are exposed and the way that in which CEM is used is described.

3.1 Classification Model

The model proposed for the classification part is a combination of an LSTM and a Fully Connected Layer (FCN). The LSTM is used for processing the data and the FCN is used for classifying it [16]. Figure 3 shows an outline of the model architecture. Firstly, the input of the model is processed by the LSTM layer in time-steps. In this way, in each time-step, the LSTM uses what it has learned before in addition to the current input to update the current hidden state. The last hidden state (h_n) of the network is the input of the FCN. The activations a of the FCN are computed as follows

$$a = W_{fc} \cdot h_n^T + b_{fc} \tag{6}$$

where W_{fc} and b_{fc} are the weights and the biases learned during training process, and h_n^T denotes the transpose of the last hidden state. After the activations are calculated, a softmax function is applied, returning the target as a probability vector $\hat{y}$, in which each value $\hat{y}_i \in \hat{y}$ is computed as follows

$$\hat{y}_i = \frac{e^{a_i}}{\sum_j e^{a_j}} \tag{7}$$

and denotes the probability that the input belongs to class i. Afterwards, the index of the maximum probability value is taken as the class predicted by the model. In training time, the weights of the model are adjusted by minimizing the Categorical Crossentropy Loss (CCE) in batches.

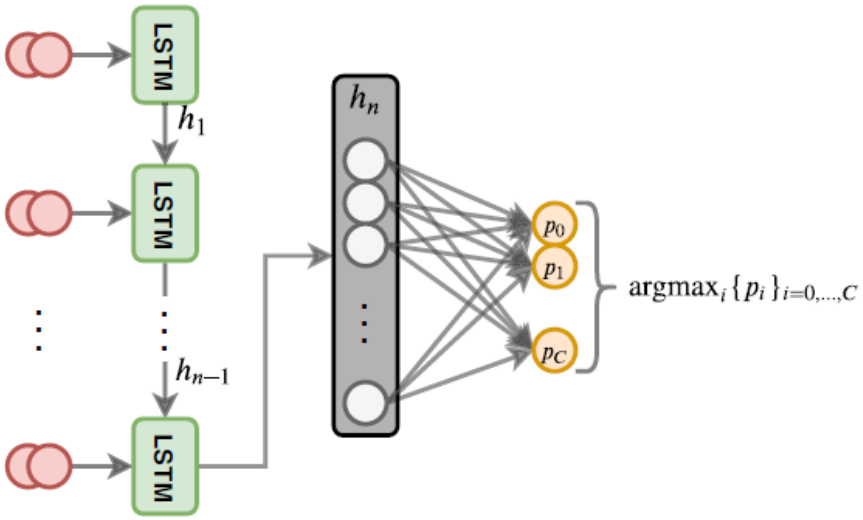

Figure 3: Proposed classification model.

3.2 Autoencoders

As seen in Eq. (2) and (4), the optimization problems include a term regarding an AE. This term ensures that the modified input is close to the data manifold. The architecture proposed for the AE is based on LSTMs, since they have demonstrated to work well when processing time-series data, and FCNs. The proposed AE can be seen in Fig. 4. First, the input data is processed with the LSTM. Then, the last hidden state of the LSTM has the information of the whole input and it is encoded into a lower-dimensional vector by the FCN. Then, the encoded vector is repeated to feed the decoder, which is another LSTM. Finally, the hidden states of the decoder are connected to an FCN layer, that converts the hidden states of the encoder into arrays with the same dimension as the inputs, using a sigmoid activation function. In this case, the weights of the model are adjusted for minimizing the Mean Squared Error (MSE).

3.3 Contrastive Explanations Method

As stated above, in this paper CEM PNs are studied. In this scenario, the x_0 of Eq. (2) is a multivariate time-series x, and the δ denotes the PN that makes the predicted class to change (i.e. $\arg\max_i [f(x)]_i \neq \arg\max_i [f(x+\delta)]_i$), being f the classification model. Moreover, the AE of Eq. (3) denotes the AE proposed above.

Since the changed sample's prediction has to be different to the original class, i.e $\arg\max_i [f(x)]_i \neq \arg\max_i [f(x+\delta)]_i$, $\max_{i \neq y} [f(x+\delta)] > [f(x+\delta)]_y$. Therefore, $[f(x+\delta)]_y - \max_{i \neq y} [f(x+\delta)]_i \in$

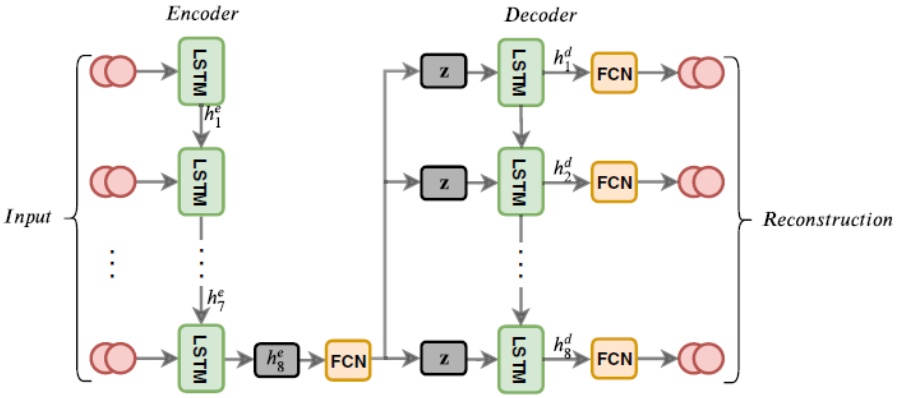

Figure 4: Proposed LSTM-FCN AE.

$[-1, 0)$, and thus, κ has to be chosen in the range $[0, 1]$. In the experiments, a set of different γ and κ parameters have been proved and it is concluded that the best results for this case study are given by $\gamma = 0.2$ and $\kappa = 0.5$.

4 Experimental Framework

4.1 PenDigits Dataset Description

In this work, a public time-series dataset, named PenDigits, is used. The PenDigits dataset D is a handwritten digit classification dataset. Each data sample $X_i \in D$ is a 2-dimensional multivariate time-series, denoted as $X_i = \{x_1^{(i)}, x_2^{(i)}\}$, where $x_1^{(i)} \in \mathbb{R}^8$ denotes the trajectory of the pen across the coordinate x of a digital screen and $x_2^{(i)}$ denotes the trajectory of the pen across the coordinate y. Each sample is labeled with a single class label, representing the digit drawn. In Fig. 5 a sample of the dataset is showed.

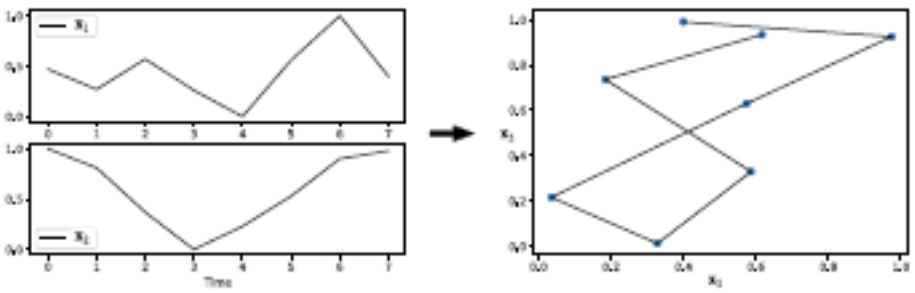

Figure 5: A sample of PenDigits dataset.

The dataset was crated by 44 writers and it is divided in two sets: training set and testing set. The training test is composed by 7,494 different samples and the testing set is composed by 3,498 different samples.

4.2 Framework and Hyperparameters

The experiments are run on an Nvidia-Docker container that uses ubuntu 18.04. The models were implemented using the Keras library of Tensorflow. The Tensorflow version used in this case is TensorFlow 2.1.0. The optimization of both models was performed using Adam. The models

have been trained in minibatches of 32 samples. The training process has stopped at epoch 163 for the classification model and at epoch 134 for the AE, because EarlyStopping has been used. Moreover, the learning rate at the beginning has been set to 0.1 and has been decreased every epoch using exponential decay. For training the models an NVIDIA TITAN V GPU has been used, with a memory of 12GB, in an Intel i7-6850K 3.6 GHz machine with 32GB of DDR4 RAM.

Each of the models used in this work has different hyperparameters. For the model used for classification, the LSTM layer has 64 units and the FCN has 10 units and softmax activation function. For the AE, the encoder and the decoder LSTMs have 16 units, the FCN of the encoder encodes the last hidden state of the first LSTM into a 4-dimensional vector, therefore, the encoder's FCN uses 4 units. Since the AE has to reconstruct the input data, the last FCNs have 15 units and a sigmoid activation function.

5 Experimental Results

In this work, PenDigits dataset has been used, since it is a simple time-series classification dataset and it is easily interpretable for anyone. Although the main objective of this work is not to propose a classification model, the model used achieves 98.11% of accuracy and a micro-average F1 score of 0.979 in validation data. On the other hand, the proposed AEs is valid to reconstruct the data, since the MSE for validation data is 0.0064.

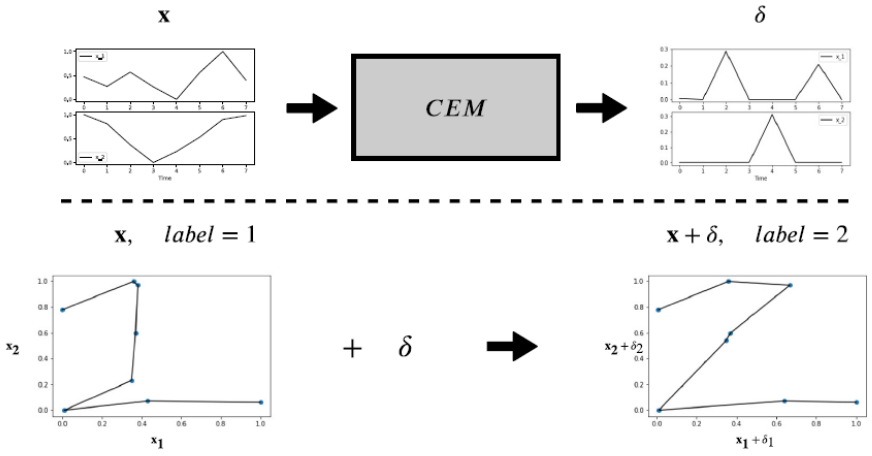

Figure 6: Process used for giving explanations using CEM.

In Fig. 6 the process used for giving the explanations is illustrated. As shown in the figure, an input x representing a one is used to obtain its pertinent negatives δ by optimizing Eq. 2. The pertinent negatives δ represent the changes that need to be made in the input x for the model to classify it as another class. In this example, it can be seen that changing the position among the x axis of the third and seventh points and changing the position among the y axis of the sixth point, x changes from a one to a two.

In this paper, CEM method has been applied to a variety of digits, for example in Fig. 7 the explanations given to 3 samples are illustrated. In the first case, a "1" is changed to a "2" by simply moving the third point from the center to the right, to simulate the curved shape of the "2". In the second case a "5" is converted to a "6" by making three small changes. In this second case, the most significant changes are found in the last two points, and by moving them, CEM has created the rounded shape of the bottom of a "6". In the third case a "4" has been changed to a "9" by changing the first four points. It can be seen that the changes have been done for simulating the rounded shape of top of a "9".

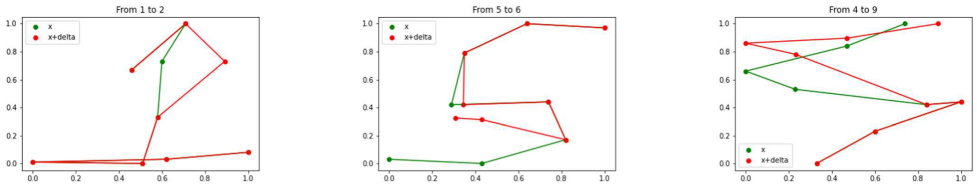

Figure 7: The original samples (green), x, versus the changed ones (red), $x + \delta$.

We have tested the CEM method for 150 samples and in general the modifications are made with sense, causing the label of the input to change. Moreover, the relationship between the label of class x and the label of $x + \delta$ is similar in all the samples. In Table 1 the percentages of the changes between the class of the original samples and the changed ones are given. It can be seen that, for example, generally the digit "4" is related to the digit "9", and the changes are done for changing from one to another, and this happens also for all other classes. For example, the digit "2" is converted into a "1" in the 80% of the cases, the changes in digit "7" make it "1" in 54.55% of the cases, the digit "8" changes into a "5" in the 50% of the cases, etc.

Table 1: Percentage of changes from x's class to $x + \delta$'s class.

$x \rightarrow x + \delta$	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	41.67	0	0	0	50	8.33
1	0	0	25	0	0	12.5	0	25	37.5	0
2	0	80	0	0	0	0	0	0	20	0
3	0	21.43	0	0	0	28.57	0	50	0	0
4	14.29	7.14	0	7.14	0	0	0	14.29	7.14	50
5	0	15.38	15.38	7.7	7.7	0	7.7	0	15.38	30.76
6	23.1	7.7	7.7	0	0	7.7	0	0	53.8	0
7	0	54.55	18.18	0	0	0	0	0	27.27	0
8	11.1	16.67	0	0	0	50	0	0	0	22.22
9	0	16.66	0	0	16.66	16.66	0	0	50	0

6 Conclusions

In this work, CEM has been validated in a time-series classification use case. Previously, as far as it is known, CEM has not been used in time-series data, and in this work, it is shown that it can be effectively used in these scenarios to create meaningful explanations. The pertinent negatives given by CEM offers a different way to understand the model's decisions, and unlike other XAI methods such as SHAP, LIME, LRP, etc., that their explanations are based on feature importance, the pertinent negatives explanations provide information of what should be changed in the input to be classified as other class. Looking at the results, it can be seen that the CEM method is also useful to find relationships between different classes, giving an intuition of which class is closer to another.

Although this is a work in progress, in which CEM is validated for a simple classification problem, the idea in the future is to validate for anomaly detection in a sensorized industrial scenario, since it can provide information about what should be changed for not occurring an anomaly, and this can be useful for fixing it.

References

[1] Adadi, A., Berrada, M.: Peeking inside the black-box: a survey on explainable artificial intelligence (XAI). IEEE Access **6**, 52138–52160 (2018)

[2] Arrieta, A.B., et al.: Explainable artificial intelligence (XAI): concepts, taxonomies, opportunities and challenges toward responsible AI. Information Fusion **58**, 82–115 (2020)

[3] Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K.R., Samek, W.: On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. PloS One **10**(7), e0130140 (2015)

[4] Ballard, D.H.: Modular learning in neural networks. In: AAAI, pp. 279–284 (1987)

[5] Beck, A., Teboulle, M.: A fast iterative shrinkage-thresholding algorithm for linear inverse problems. SIAM Journal on Imaging Sciences **2**(1), 183–202 (2009)

[6] Dhurandhar, A., et al.: Explanations based on the missing: towards contrastive explanations with pertinent negatives. In: Advances in Neural Information Processing Systems, pp. 592–603 (2018)

[7] Giurgiu, I., Schumann, A.: Additive explanations for anomalies detected from multivariate temporal data. In: Proceedings of the 28th ACM International Conference on Information and Knowledge Management, pp. 2245–2248 (2019)

[8] Guillemé, M., Masson, V., Rozé, L., Termier, A.: Agnostic local explanation for time series classification. In: 2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI), pp. 432–439. IEEE (2019)

[9] Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Computation **9**(8), 1735–1780 (1997)

[10] Lundberg, S.M., Lee, S.I.: A unified approach to interpreting model predictions. In: Advances in Neural Information Processing Systems, pp. 4765–4774 (2017)

[11] Ribeiro, M.T., Singh, S., Guestrin, C.: 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, pp. 1135–1144. ACM (2016)

[12] Sagheer, A., Kotb, M.: Time series forecasting of petroleum production using deep LSTM recurrent networks. Neurocomputing **323**, 203–213 (2019)

[13] Selvaraju, R.R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., Batra, D.: Grad-CAM: visual explanations from deep networks via gradient-based localization. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 618–626 (2017)

[14] Sundararajan, M., Taly, A., Yan, Q.: Axiomatic attribution for deep networks. In: Proceedings of the 34th International Conference on Machine Learning-Volume 70, pp. 3319–3328. JMLR.org (2017)

[15] Van Looveren, A., Klaise, J.: Interpretable counterfactual explanations guided by prototypes. arXiv preprint arXiv:1907.02584 (2019)

[16] Zhao, J., Deng, F., Cai, Y., Chen, J.: Long short-term memory-fully connected (LSTM-FC) neural network for PM2.5 concentration prediction. Chemosphere **220**, 486–492 (2019)