P2ExNet: Patch-based Prototype Explanation Network

Dominique Mercier[1,2], Andreas Dengel[1,2], and Sheraz Ahmed[2]

[1] Technische Universitt Kaiserslautern, 67663 Kaiserslautern, Germany
{mercier}@rhrk.uni-kl.de
[2] German Research Center for Artificial Intelligence (DFKI), Kaiserslautern, Germany
{dominique.mercier, andreas.dengel, sheraz.ahmed}@dfki.de

Abstract. Deep learning methods have shown great success in several domains as they process a large amount of data efficiently, capable of solving complex classification, forecast, segmentation, and other tasks. However, they come with the inherent drawback of inexplicability limiting their applicability and trustworthiness. Although there exists work addressing this perspective, most of the existing approaches are limited to the image modality due to the intuitive and prominent concepts. Conversely, the concepts in the time-series domain are more complex and non-comprehensive but these and an explanation for the network decision are pivotal in critical domains like medical, financial, or industry. Addressing the need for an explainable approach, we propose a novel interpretable network scheme, designed to inherently use an explainable reasoning process inspired by the human cognition without the need of additional post-hoc explainability methods. Therefore, class-specific patches are used as they cover local concepts relevant to the classification to reveal similarities with samples of the same class. In addition, we introduce a novel loss concerning interpretability and accuracy that constraints P2ExNet to provide viable explanations of the data including relevant patches, their position, class similarities, and comparison methods without compromising accuracy. Analysis of the results on eight publicly available time-series datasets reveals that P2ExNet reaches comparable performance when compared to its counterparts while inherently providing understandable and traceable decisions.

Keywords: Deep Learning · Convolutional Neural Networks · Time-Series Analysis · Data Analysis · Explainability · Interpretability.

1 Introduction

Nowadays, deep neural networks are very popular and used in many different domains including image processing, natural language processing, and time-series processing. Tough these deep networks have achieved high performance, they are still back boxes in nature which makes it difficult to understand why and how a decision was concluded. In particular, this black box nature hinders the use of these models in critical domains like medical, autonomous driving, industrial, financial, etc. and raises the need for interpretability methods to provide intuitive and understandable explanations so that these models can be used in these critical domains as well, where explanations play a key role [16].

The existing methods for interpreting decisions of deep learning models are mostly not applicable to image modalities. In particular, image concepts are intuitive by default [25]. Besides the image domain, there is only a limited amount of work for the time-series domain as the modalities are complex and usually not directly interpretable for a human. Nevertheless, these time-series analysis networks and their explanations are pivotal for their industrial and financial use and therefore we propose P2ExNet as an approach dealing with time-series data.

Also, existing approaches are mostly post-hoc methods that are applied after the classifier to explain their decisions [7]. Intuitively, post-hoc methods keep the network as it is without any changes to the structure, enabling their use on almost every network. Usually, this results in an instance-based local explanation that does not explain any global behavior. In contrast to post-hoc methods, the intrinsic methods focus on model design in a way that can provide a global explanation to achieve an understandable inference process. Ultimately, neither of the two approaches is superior as both have to deal with several limitations regarding the quality, subjectivity [14], the audience, and the domain usage.

To overcome these limitations we provide a network architecture for time-series analysis that is based on the standard deep neural network architecture which provides a global explanation using representative class-specific prototypes and an instance/local explanation using patch-based similarities and class-similarities. To achieve such an architecture, the inference process follows the human-related reasoning process [11]. This process is based on concepts and prototypes [13]. Intuitive class-specific patches are used to explain the network decision. Conversely, our approach is superior compared to existing template matching approaches [5] in the manner of generalization and applicability. Our experiments emphasize the use of our network structure by highlighting the comparable performance when compared to a non-interpretable network of the same size over eight publicly available datasets while preserving an intuitive and traceable explanation.

2 Related Work

In the field of network interpretability, the existing approaches can be classified as post-hoc or intrinsic methods. Depending on the use-case, it is not always possible to use both methods as these methods come with restrictions concerning the data and the network. In the following paragraphs, we address the perspectives, their advantages, and drawbacks.

2.1 Post-hoc

Using post-hoc methods to explain the decisions of deep neural networks, is a very prominent approach as these methods do no modify the network and can provide an instance base explanation. Furthermore, these methods can provide instance-based as well as global explanations resulting in broad applicability.

Instance-based: A widespread instance-based post-hoc class of approaches in the field of image domain are so-called back-propagation methods [4]. These approaches produce heat-maps highlighting the most important/sensitive parts concerning the network decision/loss. There exist enhancements that evolved [27] and take various aspects into account to improve the expressiveness and consistent behavior. Another post-hoc instance-based class of methods are the layer-wise relevance propagation methods [3,10], producing results that are close to the heat-maps but are more stable. In particular, the image domain explored different methods to visualize the activations [24] or make use of the gradients [18] or saliency [21] to produce heat-maps for instances. However, in the case of the time-series modalities, there exists only a limited amount of work e.g. [20].

Global: In contrast to instance-based methods described earlier, there exist attempts to compute a global behavior based on the influence of the samples [12, 23]. Using the influence, these methods provide an idea of helpful and harmful samples to detect outliers and debug datasets. Another approach is to attach an interpretable architecture to the already-trained network. As presented in [15], the attachment of an autoencoder before the network including a customized loss function for the autoencoder can enhance the interpretability. The adoption of this approach for the time-series domain with a customized loss function was presented by Siddiqui et al. [19].

2.2 Intrinsic

Intrinsic methods approach the problem from a different perspective by incorporating the interpretability directly. Therefore, they modify the model architecture by introducing interpretable layers [26]. A drawback of these approaches is that the restricted learning process can harm the performance. An intuitive interpretable layer solution are prototype layers to explain model decision [2]. Mainly, there are two types of prototypes that have shown to provide reasonable explanations. First, providing a class prototype that covers the complete input [8,13] and second multiple patches [6].

2.3 Limitations of existing methods

Even though there exists work to explain the network decisions, most of the approaches are limited to image modalities [17]. Furthermore, there is ongoing research investigating the consistency, expressiveness, and subjectivity of these explanations. This includes findings that prove the inconsistency of saliency-based methods [22] and measurements to prove the expressiveness [1]. In addition, methods that use sparsity constraints suffer from the same problems concerning their consistency.

3 P2ExNet: The proposed approach.

This section provides an insight into the proposed method framework where it first provides a motivation followed by the general architecture structure, the mathematical background, and the training procedure.

3.1 Motivation: An understandable reasoning behavior.

Inspired from the human reasoning behavior we aligned our framework to rely on implicit knowledge about objects/examples already seen before, which is well-aligned with humans. Precisely, abstract concepts including class typical features are compared against the new instance. The knowledge about these concepts is denoted as prototypical knowledge and covered by the analogical process to map the new to the existing knowledge [9]. Following this process, the proposed method uses shallow representations namely prototypes encoding class-specific concepts and provides the decision based on similarity.

3.2 Architecture

Inspired by the work of Gee et al. [8], we combined an autoencoder with a prototype network. The autoencoder consists of several convolutional and max-pooling layers serving as a feature encoding network to provide a latent representation that encodes the important features of an input sequence. This latent representation is fed forward to a custom prototype layer to generate prototypes. Motivated by the work of Chen et al. [6], we use multiple prototypes to represent a sample rather than a single one for the complete sample. Essentially, the prototype layer has randomly initialized variables representing patch prototypes of user-defined size. Larger sizes will result in composed concepts whereas smaller sizes result in more basic concepts. On top of the prototype layer, we attached a prototype-weight layer to encourage class-specific prototypes and weight their position within the sample to cover the local importance. Finally, a soft-max classification evaluates the similarity scores produced by the prototype layer multiplied with the prototype-weights as shown in Figure 1.

3.3 Mathematical background

Our method uses a novel combined loss that captures several aspects enabling the network to produce a meaningful set of patch prototypes, based on the losses proposed by [6, 8]. For the following equations we denote the encoder, decoder and prototype/classification network as e, d and c. Further, we say that S represents the patches and P the prototypes.

Distances: Using the L^2 norm we computed the patches to prototypes distances (D_{s2p}), the minimum over the patches (D_{s2p_r}) and the minimal prototype distances D_{p2p_r}. In addition, we calculate the minimum distance to a prototype of

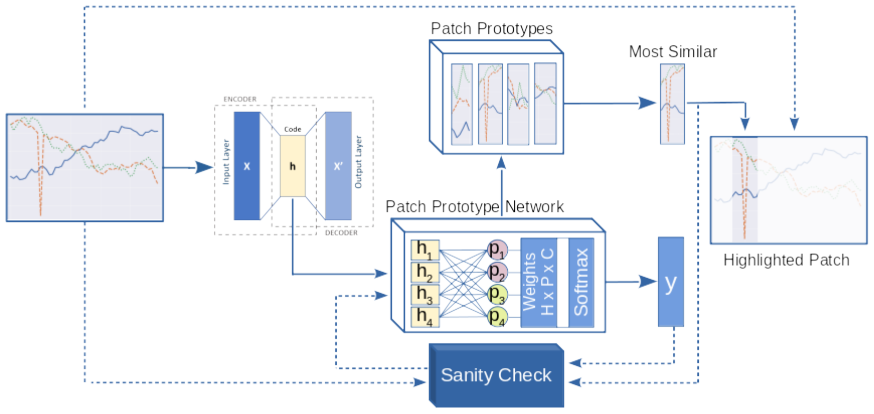


Fig. 1. Inference and testing workflow. Artificially, computed prototypes are evaluated in a similarity-based manner to suggest class-specific patches.

the same class D^i_{clst} and to the other classes D^i_{sep} w.r.t. the label of sample i. The distances are shown in Equations 1 to 6.

$$D_{s2p}(i,j) = L^2(s_i, p_j) \mid i \in S, j \in P \quad (1) \qquad D_{p2p}(i,j) = L^2(p_i, p_j) \mid i, j \in P \quad (2)$$

$$D_{s2p_r}(i) = \min D_{s2p}(i) \mid i \in S \quad (3) \qquad D_{p2p_r}(i) = \min D_{p2p}(i) \mid i \in P \quad (4)$$

$$D^i_{clst} = \{D_{s2p}(j,p) \mid p \in P_{y_i}, j \in S\} \quad (5) \qquad D^i_{sep} = \{D_{s2p}(j,p) \mid p \notin P_{y_i}, j \in S\} \quad (6)$$

Loses: To ensure high-quality prototypes we introduce our novel patch loss shown in Equation 7 which combined several losses.

$$Patch_Loss = \lambda_c H + \lambda_{mse} MSE + \lambda_{p2s} L_{p2s} + \lambda_{s2p} L_{s2p} + \lambda_{div} L_{div} + \lambda_{clst} L_{clst} + \lambda_{sep} L_{sep} \quad (7)$$

This loss is a combination to achieve good accuracy as well as an explanation that does not contain duplicates or prototypes that are not class-specific. Our loss combines the following losses:

– **Autoencoder loss:** MSE is used to encourage reconstruction later used for prototype reconstruction.
– **Classification loss:** To produce logits for the softmax cross-entropy we multiply the reciprocal of D_{s2p} and the prototype-weight layer.
– L_{p2s} **and** L_{s2p}**:** To preserve the relation between the input and the prototypes as shown in Equation 8 and 9.
– L_{div}**:** To ensure diversity among the patch prototypes is computed as shown in Equation 10.
– L_{clst} **and** L_{sep}**:** To encourage the network to learn class-specific prototypes we ensure that prototypes close to samples of different classes are penalized.

$$L_{p2s} = \frac{1}{P}\sum_{i=0}^{P} \min\{D^{j}_{s2p_r}(i) \mid 0 \leq j < n\} \tag{8}$$

$$L_{s2p} = \frac{1}{n}\sum_{i=0}^{n} \min D^{i}_{s2p_r} \tag{9}$$

$$L_{div} = \frac{1}{\log(1 + \frac{1}{P}\sum_{i=0}^{n} D^{i}_{p2p_r})} \tag{10}$$

$$L_{clst} = \frac{1}{n}\sum_{i=0}^{n} \min D^{i}_{clst} \tag{11}$$

Our proposed final loss is shown in Equation 7 is a linear combination taking into account previously mentioned aspects ensuring meaningful, diverse, and class-specific patch prototypes. By default, we set all lambda values except λ_c to one to find the best compromise between the objectives preserving a good accuracy.

3.4 Training process

The proposed network can be trained in two stages. In the first stage, we fix the weights of the pre-initialized prototype-weight layer to ensure class-specific prototypes. We then train the network until it converges. In the second learning phase, all weights except the prototype-weighting layer are frozen and the network learns to adjust the prototype weights resulting in a fine-tuning to adjust the prototype class affiliation using the previously trained latent representation.

4 Datasets

We used eight publicly available time-series datasets to emphasize the broad applicability of our approach and examine possible limitations. As a representative set, we used seven different datasets from the UCR Time Series Classification Repository[3] and a point anomaly dataset proposed in [20]. To have better coverage of different types, we selected different datasets based on different characteristics concerning the number of classes, channels, and time-steps to have several different conditions and show the prototypes. However, we focus on classification datasets including n-way classification and anomaly detection.

5 Experiments

In this section, we present our results concerning the interpretable results, performance, applicability, and resource consumption for our proposed approach highlighting a comparable performance while producing interpretable results.

5.1 P2ExNet: Instance-based evaluation

The proposed method provides the possibility to identify and highlight the parts of the input that were most relevant for the classification. Besides, it provides

[3] http://www.timeseriesclassification.com/

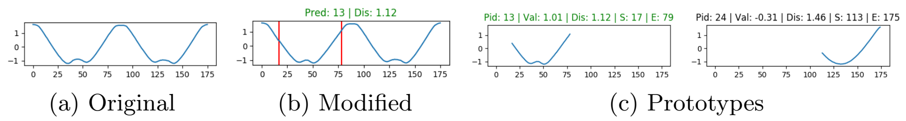


Fig. 2. Adiac dataset prototype explanation. a) shows the original series. b) shows the series with the prototype between the red bars. c) shows two prototypes.

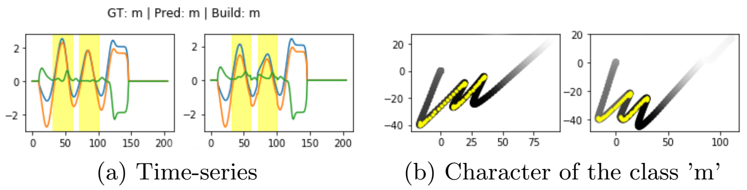


Fig. 3. Character dataset prototype explanation. a) shows the original series and the series with the prototypes. b) shows the character output and the modified character.

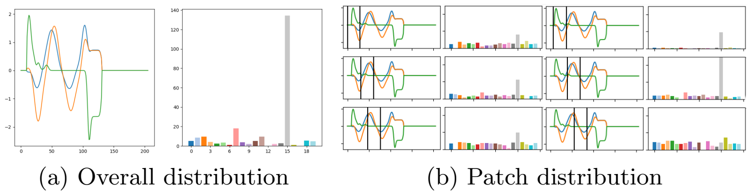


Fig. 4. Class and prototype distribution. a) shows the class similarities. b) shows some patches and the corresponding class similarities.

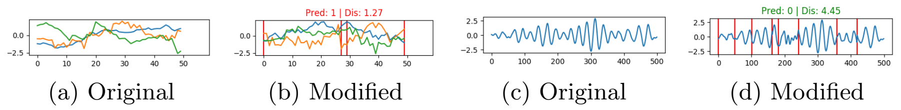


Fig. 5. Prototype substitution. a) and c) show original time-series. b) and d) show the corresponding modified samples and their re-classification.

prototypes along with a sample containing the prototypes to compare it to the original input. Figure 2 shows highlighted regions that were important for the inference on the ADIAC dataset sample. This explanation including the original sample of the adiac dataset, a modified version, and two prototypes. In the modified version shown in Figure 2b we replaced the part between the two red lines with the most important patch prototype to show how close it is to the original part. Figure 2c shows two prototypes. The value of each prototype denoted as 'Val' highlights its contribution towards the classification result. Similarly,

Dataset	**Classes**	**Length**	**Channel**	**CNN**	**P2ExNet**
Anomaly [20]	2	50	3	**99.79**	93.79
FordA	2	500	1	85.44	**89.32**
Devices	7	96	1	55.42	**62.53**
Adiac	37	176	1	**63.54**	60.15
Crop	24	46	1	68.27	**68.54**
50words	13	270	1	76.84	**81.98**
PenDigits	10	8	2	**94.29**	93.95
Character	20	206	3	**96.53**	91.78

Table 1. Accuracy comparison. A comparison of interpretable and the corresponding non-interpretable counterpart.

Figure 3 shows a sample from the character trajectories dataset including the mapping of the time-series back to the character. The black value highlights the pressure of the pen and the yellow part shows the mapping of the prototype back to the input space. When using our approach for debugging, in case of a misclassification the prototypes have a red caption. Furthermore, in Figure 4 the class-wise overall and patch-wise distribution can be used to get additional information about similar classes and important patch positions. Especially, in Figure 4b we show that not all patches have the same importance when it comes to the classification. Replacing the original data with a prototype there are offset sensitive datasets for which the re-classification can change. However, for the classification and the explanation, this is not a problem as it can be solved by using proper re-scaling and adjustment methods to fit the prototypes into the corresponding gap in a proper manner. In Figure 5b such a jump in the orange signal is shown and leads to an anomaly. Although, the classification of the original signal with the network was correct. Furthermore, some datasets are invariant to small offsets as shown in Figure 5d. This emphasizes that rescaling should be done based on the problem task e.g. in a point anomaly task the patches have to perfectly align, whereas in a classification task a single point offset does not make a difference.

5.2 P2ExNet: Evaluation as a classifier

Usually, interpretability, when incorporated directly in the network structure, comes with an accuracy drop. In Table 1 we present the accuracy trade-off highlighting that our structure is on the same level as the non-interpretable counterpart. To create the previously mentioned counterpart the prototype layer was replaced with a dense layer and a cross-entropy loss and a traditional learning procedure were used as suggested by Chen et al. [6]. Furthermore, we removed the decoder as there is no need to restrict the latent representation as no reconstruction is required. We conducted this comparison for all eight datasets showing that P2ExNet achieves comparable and in some cases better performance in comparison to the non-interpretable variant. Evaluating the overall accuracy, the interpretable network has an insignificant performance increase of 0.03%. Each

Dataset	Data replaced	Equal Pred.	P2ExNet Acc.	P2Exnet modified Acc.
Anomaly	71.99	87.43	93.79	**91.78**
	67.32	19.45		22.72
FordA	51.17	99.92	89.32	**89.40**
	44.95	23.09		32.69
Devices	52.36	81.65	62.53	**60.52**
	65.81	49.81		39.11
Adiac	35.22	85.97	60.15	**55.98**
	69.90	9.11		14.84
Crop	50.50	94.08	68.54	**66.94**
	81.12	22.01		23.28
50words	36.43	93.01	81.98	**77.20**
	52.88	62.50		56.98
PenDigits	69.47	99.31	93.95	**93.54**
	68.65	8.83		11.0
Character	18.15	92.93	91.78	**85.30**
	52.90	31.71		32.87

Table 2. Replacement of original patch. The second column shows how much data was replaced with the suggested prototypes proposed by P2ExNet. The third column shows whether the prediction was the same as with the original time-series or not. The fourth column shows the P2ExNet accuracy for the original sample and the last column for the sample replacing the original patch with the suggested patch. The first row of each dataset corresponds to replacements with the most similar prototype whereas the second row with the most different.

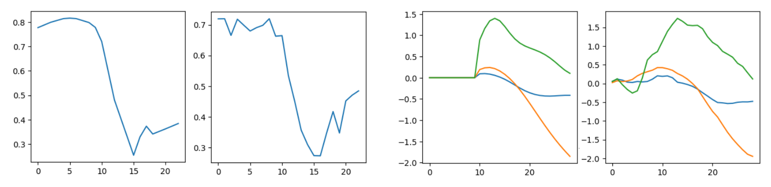
Dataset	P2ExNEt with decoder	P2ExNet without decoder	Improvement
Anomaly	0.6393	**0.4929**	-22.9%
FordA	**0.7018**	1.0315	47.0%
Devices	0.4135	**0.3399**	-17.8%
Adiac	0.538	**0.4993**	-6.2%
Crop	**0.442**	0.4815	8.9%
50words	**0.0413**	0.2086	505.1%
PenDigits	**0.5123**	0.5622	9.7%
Character	**0.0099**	0.5887	5946.5%

Table 3. Closeness of prototypes. Shows the difference between representative and generated latent patch prototypes for P2ExNet with and without the use of the decoder.

network was superior in four out of the eight datasets. For the anomaly dataset there as a difference of 6% accuracy in favor of the non-interpretable network and for the Electric Devices a difference of 7% in favor of the interpretable network.

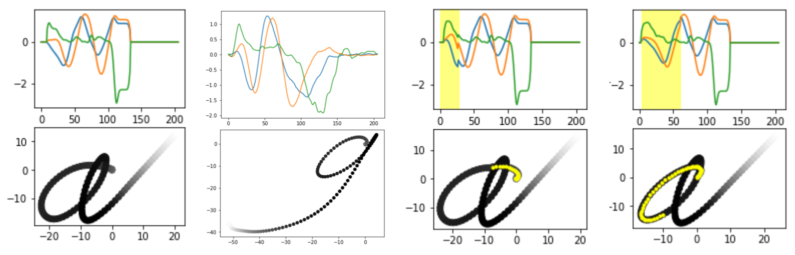
5.3 P2ExNet: Sanity check

To prove the class-specific and meaningful behavior of the prototypes we replaced the original time-series once with the most positive and once with the



(a) Crop dataset (b) Character trajectories dataset

Fig. 6. Prototype comparison. This figure shows the representative patch selected as prototype representative selected based on the distance to the latent prototype and the reconstruction of the latent representation.



(a) Original (b) Gee et al. [8] (c) Chen et al. [6] (d) P2ExNet

Fig. 7. P2ExNet approaches. This figure shows one sample of the character 'a' and the corresponding prototype explanations.

most negative influencing prototypes. In Table 2 we show that the replacement with the most positive prototypes corresponding to the predicted class achieved results close to the default accuracy, whereas the replacement with the best fit prototype of a different class dramatically decreased the performance as the prediction switched showing that our prototypes are class-specific. However, a second sanity check was performed to emphasize the need for the decoder to produce latent representations that are close to the representative prototypes. In Table 3 we show that especially, for the character trajectories, 50words, and the FordA dataset there is a significant difference if the decoder is excluded. Also, we compared the representative and decoded samples and show two samples in Figure 6 highlighting the small difference between the selected representative sample (left) and the decoded one (right).

5.4 Comparison with existing prototype-based approaches

As there already exist methods based on prototypes to explain the network prediction we compared the proposed method against [6] and [8]. Precisely, the explanations and additional outputs are highlighted. In Figure 7 the explanation of each method for a sample of class zero which corresponds to the character 'a' is shown. While [8] explains the class with a prototype providing a single prototype capturing the complete sample, [6] is based on parts of the sample leading to a more detailed explanation. This method provides a patch and allows us to match this patch against a region in the input image. Precisely, this means additional position information for the patch is available. Lastly, our proposed method provides the same information about the location but offers re-scaling as well as an implicit comparison to other prototypes and a class distribution for the complete sample and the patches as shown in Figure 4b. Furthermore, our prototypes are class-specific and can be decoded for a comparison with the representatives enabling easy validation.

6 Conclusion

Summarizing our results we came up with novel network architecture, along with a loss, and training procedure aligned to produce interpretable results and an inference process similar to the human reasoning without a significant drop in performance. Further, we proved that the proposed method can be used for several time-series classification tasks and when excluding the class-specific prototype assignment our approach is suitable to produce prototypes for regression and forecast tasks. Besides, we compared the proposed method with existing prototype-based methods concerning their interpretable output and time consumption finding ours superior in both aspects.

References

1. Alvarez-Melis, D., Jaakkola, T.S.: On the robustness of interpretability methods. arXiv preprint arXiv:1806.08049 (2018)
2. Angelov, P., Soares, E.: Towards explainable deep neural networks (xdnn). arXiv preprint arXiv:1912.02523 (2019)
3. Arras, L., Montavon, G., Müller, K.R., Samek, W.: Explaining recurrent neural network predictions in sentiment analysis. arXiv preprint arXiv:1706.07206 (2017)
4. Bojarski, M., Choromanska, A., Choromanski, K., Firner, B., Jackel, L., Muller, U., Zieba, K.: Visualbackprop: efficient visualization of cnns. arXiv preprint arXiv:1611.05418 (2016)
5. Brunelli, R.: Template matching techniques in computer vision: theory and practice. John Wiley & Sons (2009)
6. Chen, C., Li, O., Tao, C., Barnett, A.J., Su, J., Rudin, C.: This looks like that: deep learning for interpretable image recognition. arXiv preprint arXiv:1806.10574 (2018)
7. Choo, J., Liu, S.: Visual analytics for explainable deep learning. IEEE computer graphics and applications **38**(4), 84–92 (2018)
8. Gee, A.H., Garcia-Olano, D., Ghosh, J., Paydarfar, D.: Explaining deep classification of time-series data with learned prototypes. arXiv preprint arXiv:1904.08935 (2019)
9. Gentner, D., Colhoun, J.: Analogical processes in human thinking and learning. In: Towards a theory of thinking, pp. 35–48. Springer (2010)

10. Gu, J., Yang, Y., Tresp, V.: Understanding individual decisions of cnns via contrastive backpropagation. In: Asian Conference on Computer Vision. pp. 119–134. Springer (2018)
11. Guidoni, P.: On natural thinking. The European Journal of Science Education **7**(2), 133–140 (1985)
12. Koh, P.W., Liang, P.: Understanding black-box predictions via influence functions. In: Proceedings of the 34th International Conference on Machine Learning-Volume 70. pp. 1885–1894. JMLR. org (2017)
13. Li, O., Liu, H., Chen, C., Rudin, C.: Deep learning for case-based reasoning through prototypes: A neural network that explains its predictions. In: Thirty-Second AAAI Conference on Artificial Intelligence (2018)
14. Lipton, Z.C.: The mythos of model interpretability. arXiv preprint arXiv:1606.03490 (2016)
15. Palacio, S., Folz, J., Hees, J., Raue, F., Borth, D., Dengel, A.: What do deep networks like to see? In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2018)
16. Samek, W., Wiegand, T., Müller, K.R.: Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models. arXiv preprint arXiv:1708.08296 (2017)
17. Schlegel, U., Arnout, H., El-Assady, M., Oelke, D., Keim, D.A.: Towards a rigorous evaluation of xai methods on time series. arXiv preprint arXiv:1909.07082 (2019)
18. 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)
19. Siddiqui, S.A., Mercier, D., Dengel, A., Ahmed, S.: Tsinsight: A local-global attribution framework for interpretability in time-series data. arXiv preprint arXiv:2004.02958 (2020)
20. Siddiqui, S.A., Mercier, D., Munir, M., Dengel, A., Ahmed, S.: Tsviz: Demystification of deep learning models for time-series analysis. IEEE Access **7**, 67027–67040 (2019)
21. Simonyan, K., Vedaldi, A., Zisserman, A.: Deep inside convolutional networks: Visualising image classification models and saliency maps. arXiv preprint arXiv:1312.6034 (2013)
22. Tomsett, R., Harborne, D., Chakraborty, S., Gurram, P., Preece, A.: Sanity checks for saliency metrics. arXiv preprint arXiv:1912.01451 (2019)
23. Yeh, C.K., Kim, J., Yen, I.E.H., Ravikumar, P.K.: Representer point selection for explaining deep neural networks. In: Advances in Neural Information Processing Systems. pp. 9291–9301 (2018)
24. Yosinski, J., Clune, J., Nguyen, A., Fuchs, T., Lipson, H.: Understanding neural networks through deep visualization. arXiv preprint arXiv:1506.06579 (2015)
25. Zhang, Q.s., Zhu, S.C.: Visual interpretability for deep learning: a survey. Frontiers of Information Technology & Electronic Engineering **19**(1), 27–39 (2018)
26. Zhang, Q., Nian Wu, Y., Zhu, S.C.: Interpretable convolutional neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 8827–8836 (2018)
27. Zintgraf, L.M., Cohen, T.S., Adel, T., Welling, M.: Visualizing deep neural network decisions: Prediction difference analysis. arXiv preprint arXiv:1702.04595 (2017)