SignalGrad-CAM: beyond image explanation

Samuele Pe[1,*], Tommaso Buonocore1[1], Giovanna Nicora[1] and Enea Parimbelli[1,2]

[1] *Department of Electrical, Computer, and Biomedical Engineering, University of Pavia, Via Adolfo Ferrata 5, Pavia (Italy)*
[2] *Telfer School of Management, University of Ottawa, 55 Laurier Avenue East, Ottawa, Ontario K1N 6N5 (Canada)*

Abstract

Deep learning models have demonstrated remarkable performance across various domains; however, their black-box nature hinders interpretability and trust. As a result, the demand for explanation algorithms has grown, driving advancements in the field of eXplainable AI (XAI). However, relatively few efforts have been dedicated to developing interpretability methods for signal-based models. In this work, we introduce SignalGrad-CAM , a versatile and efficient interpretability tool that extends the principles of Grad-CAM to both 1D- and 2D-convolutional neural networks for signal processing. SGrad-CAM is designed to interpret models for either image or signal elaboration, supporting both PyTorch and TensorFlow/Keras frameworks, and provides diagnostic and visualization tools to enhance model transparency. The package is also designed for batch processing, ensuring efficiency even for large-scale applications, while maintaining a simple and user-friendly structure. We validated SGrad-CAM on multiple open-source models and datasets, including speech emotion recognition, cardiovascular disease classification, and human activity recognition (HAR). Results suggest that SGrad-CAM consistently highlights salient features in the inputs, aiding model understanding and bias detection.

Keywords

Grad-CAM, XAI, time series, CNN, HAR

1. Introduction

1.1. Background

The rapid evolution of deep learning (DL) architectures has outpaced the development of tools to interpret their increasingly complex decision-making processes. The opaque nature of these systems suffers from the critical flaw of alienating users, leading to detrimental consequences in human-AI decision-making. This phenomenon, often termed "algorithm aversion" [1], fosters distrust in artificial intelligence (AI), exacerbating the already contentious human-AI relationship. The need for interpretability tools to demystify the *black box* and ensure trustworthiness has grown increasingly urgent, forming the core objective of eXplainable AI (XAI) [2], an innovative research trend. XAI offers diverse solutions emphasizing interpretability, especially in high-stakes decision-making contexts, such as in healthcare.

In image processing, the visual, post-hoc algorithm Gradient-weighted Class Activation Mapping (Grad-CAM) [3] remains a gold standard for visualizing spatial importance in convolutional networks. By leveraging layer activations and partial derivatives of the output with respect to layer parameters, Grad-CAM efficiently constructs saliency maps that highlight image regions that most influence the model's decision. Numerous extensions of Grad-CAM have emerged to refine or complement it, such as HiResCAM [4] and Grad-CAM++ [5].

While interpretability research in image processing has received significant attention, other input modalities – i.e. signals [6] and 3D volumetric/video data [7] – remain underexplored in the context

Late-breaking work, Demos and Doctoral Consortium, colocated with the 3rd World Conference on eXplainable Artificial Intelligence: July 09–11, 2025, Istanbul, Turkey

*Corresponding author.

samuele.pe01@universitadipavia.it (S. Pe); tommaso.buonocore@unipv.it (T. Buonocore1); giovanna.nicora@unipv.it (G. Nicora); enea.parimbelli@unipv.it (E. Parimbelli)
https://github.com/samuelepe11/ (S. Pe); https://github.com/detsutut/ (T. Buonocore1);
https://github.com/GiovannaNicora/ (G. Nicora); https://github.com/eparimbelli/ (E. Parimbelli)
0009-0002-7877-0294 (S. Pe); 0000-0002-2887-088X (T. Buonocore1); 0000-0001-7007-0862 (G. Nicora);
0000-0003-0679-828X (E. Parimbelli)

of XAI. For signals, basic interpretability can be achieved through example-based methods like those leveraging Shapelets [8], while more sophisticated techniques often adapt existing algorithms originally designed for images or tabular data. Examples include perturbation-based approaches like SHAP [9] and gradient-driven methods like the "gradient × input" technique [10] or Class Activation Mapping (CAM) [11].

Similar to CAM, Grad-CAM and its descendants were initially designed for images and 2D-CNNs, but their core principles are readily applicable to signal processing [12] since many state-of-the-art signal processing networks rely on convolutional layers – typically 1D-CNNs that perform temporal convolutions, but 2D models (which treat signals as pseudo-images) are also common. Extending Grad-CAM to these domains represents a simple yet understudied approach to generating efficient, intuitive explanations for signal-based models.

Building on this foundation, we present *SignalGrad-CAM* (SGrad-CAM, https://github.com/bmi-labmedinfo/signal_grad_cam), an easy-to-use, versatile Python package for generating class activation maps. SGrad-CAM supports both 1D- and 2D-CNNs, accommodates image and signal data, and efficiently processes batched inputs. Beyond producing saliency maps, the package includes diagnostic tools for quantitative and qualitative evaluation, such as customizable visualization pipelines that enable users to validate the model's behavior effectively.

1.2. Related works

Over time, the open-source community has developed numerous Python packages that implement Grad-CAM and its variants. Building on PyTorch, we can consider, for example, the Grad-CAM official repository (https://github.com/ramprs/grad-cam) [3], *pytorch-grad-cam* (https://github.com/jacobgil/pytorch-grad-cam), *TorchCAM* (https://github.com/frgfm/torch-cam), *GradCam-Pytorch-Implementation* (https://github.com/irfanbykara/GradCam-Pytorch-Implementation), and *OmniXAI* (https://github.com/salesforce/OmniXAI). When considering TensorFlow or Keras as a framework, we find packages such as *keras-grad-cam* (https://github.com/jacobgil/keras-grad-cam), *OmniXAI*, and *Xplique* (https://github.com/deel-ai/xplique).

Many of these tools excel in efficiency – leveraging GPUs and batching acceleration to generate saliency maps in real time – and in flexibility, allowing users to inspect arbitrary convolutional layers or compare multiple attribution methods (e.g., HiResCAM [4], Grad-CAM++ [5]). Furthermore, modern libraries like Xplique and pytorch-grad-cam integrate quantitative evaluation metrics to assess explanation faithfulness, such as RemOve And Debias (ROAD) [13]. Some of them also provide baselines (e.g., RandomCAM) for comparisons and additional tools for sanity checks [14].

However, despite these advancements, critical limitations persist. First, modality constraints limit existing implementations: most tools focus exclusively on 2D image data, and their support for other data modalities (like 1D signals or 3D data) is limited, non-intuitive, or requires manual adjustments. These shortcoming hinder the adoption of interpretability tools in emerging domains such as computational pathology (multi-scale 3D imagery), autonomous systems (multi-modal sensor fusion), and real-time signal processing. Second, framework lock-in hinders reproducibility, as packages like TorchCAM (PyTorch) and tf-keras-vis (TensorFlow) enforce ecosystem-specific workflows, complicating cross-framework comparisons. Third, only a few tools are optimized for distributed computation or large batches, limiting their utility in large-scale applications. Finally, the Grad-CAM algorithm envisions the computation of gradients with respect to the output scores (logits) [3] – i.e., before the application of Softmax/Sigmoid – but these packages compute gradients directly on the output, regardless of its form.

To address these challenges, we present SignalGrad-CAM, a Python package designed for efficiency, flexibility, and cross-modality compatibility. SGrad-CAM extends Grad-CAM's core principles with four key innovations. First, our approach has a modality-agnostic design, natively supporting 1D (time-series) and 2D (images) through a unified API. Second, it is compatible with both PyTorch and TensorFlow/Keras. Third, SignalGrad-CAM can be applied to any custom architecture with at least one convolutional layer. Fourth and last, by performing back-transformation of probabilistic outputs into their original logit form before gradient computation, our implementation faithfully adheres to the

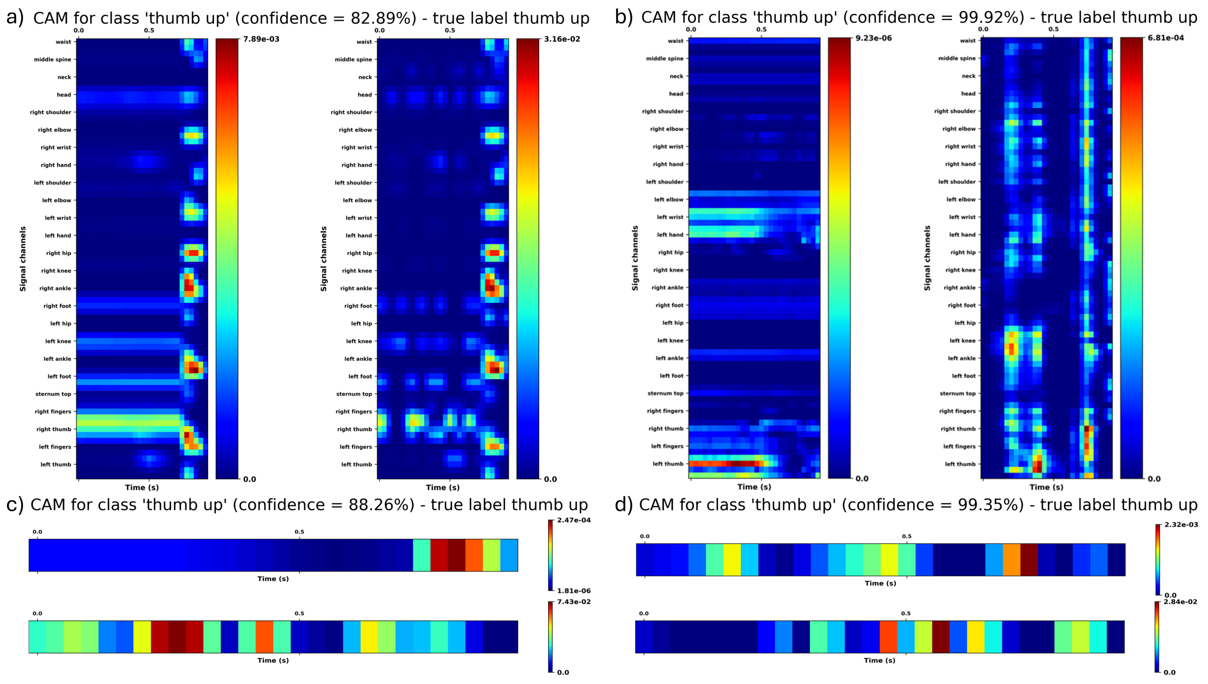

Figure 1: SGrad-CAM explaining four models for Human Activity Recognition (HAR). We considered a 2D-CNN (a), a hybrid 2D CNN-LSTM network (b), a 1D-CNN (c), and a hybrid 1D CNN-LSTM network (d). We show the CAMs produced for a data instance using Grad-CAM (on the left/top) and HiResCAM (on the right/bottom).

original Grad-CAM algorithm [3].

The package is designed to be easy-to-use and provide self-explanatory class methods, and capable of computing multiple CAMs per time – e.g., for an entire batch of data, across multiple target classes and layers, and with several algorithm variants (vanilla Grad-CAM and HiResCAM, for the moment). Moreover, SGrad-CAM is designed to guarantee efficiency by leveraging GPU and batching acceleration. Finally, the package is equipped with methods for CAM visualization, either standalone or overlaid with the input data.

2. SignalGrad-CAM: overview and implementation details

SignalGrad-CAM is designed to provide a versatile implementation of Grad-CAM, enabling users to interpret 2D-CNNs for image analysis and 1D- or 2D-CNNs for signal processing. Our package allows users to explain models built with either PyTorch or TensorFlow/Keras, offering two specific post-hoc visual XAI algorithms: Grad-CAM and HighResCAM. Moreover, the code is optimized to ensure efficient execution, even when generating explanations for multiple batched inputs.

When applied to 2D networks, SGrad-CAM's output is a bidimensional matrix, identifying the most relevant regions (features) of the input that influence the classification of the selected class. In the case of images, as demonstrated by Selvaraju et al. [3], the highlighted features typically correspond to highly informative objects, structures, or patterns in the image, visually indicating the factors influencing the classification.

For signal processing using 2D-CNNs, the general approach treats the input as a single-channel image (length × N_{channels}). The resulting CAM remains bidimensional, and one axis represents time (or sample indices) while the other axis corresponds to signal channels [Fig. 1(a-b) and 2(b)]. Since these algorithms are designed to capture spatial information, their application to signals translates to capturing temporal information along the first axis and channel correlation along the second one. The output can thus be interpreted as a visualization of each channel's importance over time.

Finally, when applying Grad-CAM to 1D-CNNs, the focus shifts from spatial to purely temporal information. The result is a one-dimensional array of importance scores, allowing users to assess the average relevance of different time steps in the signal [Fig. 1(c-d)]. This provides insights into which time segments contribute most significantly to the model's predictions.

2.1. Grad-CAM for signals

In the context of images, the idea behind Grad-CAM (GC) is to retrieve spatial information captured by convolutional layers, which is typically lost in the final fully connected layers of the network. The assumption is that the last convolutional layers of a network extract semantic and class-specific information from the input, making their outputs (i.e., layer activations) good candidates for highlighting the most valuable spatial information in the image. Each layer produces multiple activation maps (one per neuron), which need to be combined. The simplest and most straightforward solution proposed by Selvaraju et al. [3] is to compute a weighted average, using the mean gradient of the output score with respect to the activation map as a weight for the activation map itself.

Concerning 2D-CNNs for either image or signal classification, we followed the algorithm's instructions precisely, obtaining the class activation map for a single, unbatched input as follows:

$$\mathbf{CAM_{GC}}(x,y) = ReLU\left(\sum_{k=1}^{K} w_k \mathbf{A}_k(x,y)\right), \quad w_k = \frac{1}{|\mathbf{A}_k|}\sum_{(x,y)\in\mathbf{A}_k} \frac{\partial s}{\partial \mathbf{A}_k(x,y)} \tag{1}$$

where $\boldsymbol{CAM_{GC}}$ is a bidimensional matrix, $\boldsymbol{A}$ represents the activation tensor (K × height × width, with K the number of neurons in the layer) of the selected layer, and s is the output score for the desired class.

We extended the algorithm to the case of 1D convolutions, where the output activations can be considered 1D vectors instead of 2D feature maps:

$$\mathbf{cam_{GC}}(t) = \text{ReLU}\left(\sum_{k=1}^{K} w_k \mathbf{a}_k(t)\right), \quad w_k = \frac{1}{|\mathbf{a}_k|}\sum_{t\in\mathbf{a}_k} \frac{\partial s}{\partial \mathbf{a}_k(t)} \tag{2}$$

where $\boldsymbol{cam_{GC}}$ is a one-dimensional vector, $\boldsymbol{a}$ represents the activation tensor (K × length, with K the number of neurons in the layer) of the chosen layer, and s is the output score for the selected class.

Many open-source networks do not provide raw score outputs but instead normalize them using Softmax or Sigmoid functions to convert them into class probabilities. According to the original paper, Grad-CAM requires the computation of partial derivatives directly on these scores, which may not always be available. Nevertheless, retrieving the original logits from output probabilities is impossible since the Softmax and Sigmoid functions cannot be mathematically inverted. To partially solve this issue, we defined an approximate inversion formula:

$$\mathbf{s} \sim \ln(\mathbf{p}) + \text{constant} \tag{3}$$

where p is the vector representing the output class probability and s contains the approximated logit scores. The second term in the formula is an unknown constant, but this does not pose a problem during gradient computation, as it cancels out. Note that, even though logits can often be stored during the forward pass, this approach was chosen because the model is externally provided so reconstructing or modifying it can be complex, especially when the architecture is not fully exposed or customizable.

2.2. HiResCAM for signals

HiResCAM (HRC) [4] was developed to address a key limitation of Grad-CAM. Grad-CAM relies on averaging gradients over spatial dimensions to obtain importance weights, but this approach is overly simplistic. Gradients carry critical information, such as whether certain features require a sign change or a rescaling. Averaging these values results in significant information loss. Moreover, HRC was proved to perform better compared to Grad-CAM in various applications, including CT diagnostic imaging. However, as noted by its authors, HRC's results are often equivalent to Grad-CAM, depending on the network selection.

For the 2D-CNN case, we obtained the CAM using the following formula:

$$\mathbf{CAM_{HRC}}(x,y) = ReLU\left(\sum_{k=1}^{K} \frac{\partial s}{\partial \mathbf{A}_k(x,y)} \mathbf{A}_k(x,y)\right) \tag{4}$$

where $\boldsymbol{CAM_{HRC}}$ is a two-dimensional matrix, $\boldsymbol{A}$ represents the activation tensor (K × height × width, with K being the number of neurons in the layer), and s is the output score for the target class. For the 1D case, the formula is changed as:

$$\mathbf{cam_{HRC}}(t) = ReLU\left(\sum_{k=1}^{K} \frac{\partial s}{\partial \mathbf{a}_k(t)} \mathbf{a}_k(t)\right) \tag{5}$$

where $\boldsymbol{cam_{HRC}}$ is a one-dimensional vector, $\boldsymbol{a}$ represents the activation tensor (K × length, with K being the number of neurons in the selected layer), and s is the output score for the target class. In cases where outputs are probabilities, the inverse Softmax/Sigmoid solution described in the previous section is applied.

[Fig. 1] includes examples comparing Grad-CAM and HiResCAM, illustrating how the two algorithms either produce almost identical CAMs (a and c) or highlight similar information, even when the maps differ (b and d).

2.3. Visualization tools

The framework offers a range of visualization tools to facilitate a detailed validation of the model and its outputs. First, each CAM can be stored alongside relevant information about the examined item: true class, predicted class, and prediction confidence for the selected class [Fig. 1]. The adopted coloring scheme is *jet*, a rainbow-like gradient that smoothly transitions between multiple colors – blue representing low-importance features (with saliency scores near zero) and red highlighting the most salient ones. To enhance interpretability, a color bar is included, mapping colors to their corresponding importance values assigned by the algirithm. For signal-related outputs, the horizontal axis can be adjusted to convert sample indices into time values based on the sampling frequency, while channel names can be displayed along the vertical axis. SGrad-CAM methods provide additional insights by allowing users to retrieve raw CAM data, class probabilities, and importance score ranges, enabling them to customize the display of the results.

For validating 2D outputs, the package allows users to generate an image by superimposing the CAM onto the original output [Fig. 2(c)]. This approach enhances interpretability by incorporating key details such as prediction confidence for the selected class. While particularly useful for image-based applications, this functionality also benefits the interpretability of signal processing tasks, especially when dealing with numerous channels (e.g., spectrum signals, where the channel axis is continuous).

For signals, both 1D- and 2D-CAMs can be used to enrich data by coloring (in a plot) each raw signal point according to its corresponding importance score [Fig. 2(a, d)]. This output visualization modality can be displayed for multiple channels: note that 1D-CAMs assign a uniform color per timestep across all channels, whereas 2D-CAMs generate a unique color sequence for each channel.

3. Package validation

To assess the functionality and versatility of SGrad-CAM, we applied the package to evaluate multiple models. The selected models and tasks span different framework combinations (PyTorch or TensorFlow/Keras), data types (image or signal), and model architectures (1D- or 2D-CNN). Most of these models are open-source and publicly available along with their training and test datasets. The following sections focus on validating models for signal classification. Each experiment provided insights into the model's functioning and helped validate SGrad-CAM's efficacy: CAMs were always self-consistent, highlighting meaningful areas within a signal – for example, in HAR, the body joints corresponded to the parts actually involved in the movement.

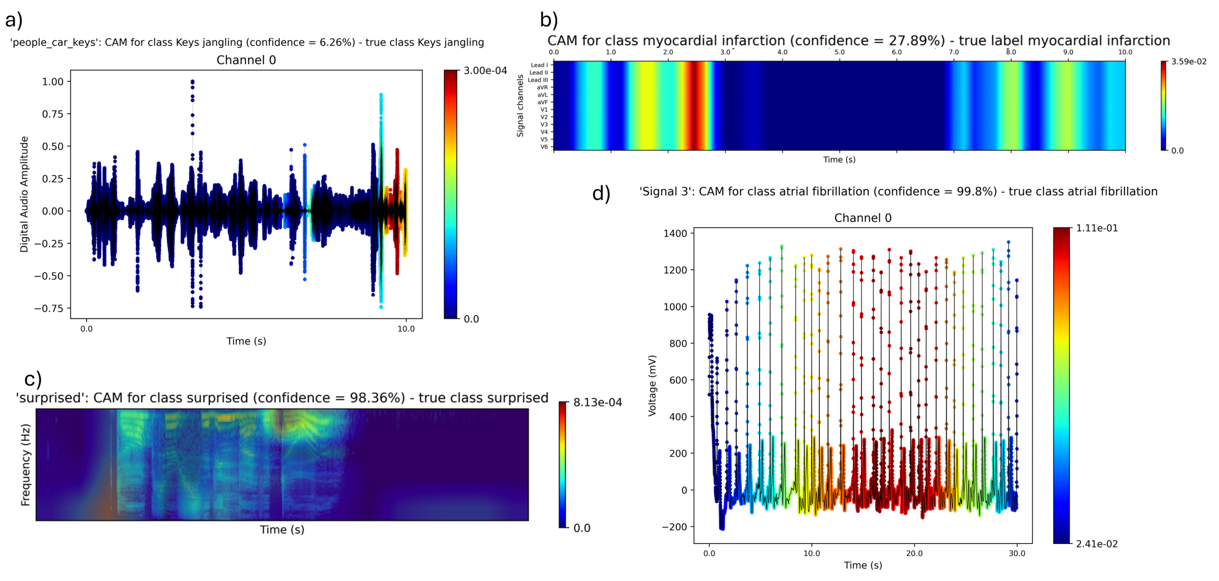

Figure 2: SGrad-CAM applied to a 1D ResNet-18 for audio classification (a), a residual 2D CNN for cardiovascular disease classification in 12-leads ECGs (b), a hybrid 2D CNN-RNN with attention for speech emotion classification (c), and a 1D residual network for ECG classification (d).

3.1. Experiments on open-source models

The first set of tests involved open-source deep learning models. For the 2D case in signal classification, we examined: hybrid convolutional-recurrent network with attention for speech emotion classification (https://github.com/Data-Science-kosta/Speech-Emotion-Classification-with-PyTorch), which leverages the Kaggle *RAVDESS Emotional Speech Audio dataset*, and a TensorFlow-based residual CNN for cardiovascular disease classification (https://github.com/HaneenElyamani/ECG-classification) trained on the *PhysioNet PTB-XL dataset.*

In the context of the former model, [Fig. 2(c)] illustrates an example of a raw spectrum image superimposed with the corresponding CAM, highlighting the most relevant frequency components for detecting the "surprise" emotion in speech. Similarly, for the latter case, [Fig. 2(b)] shows the distribution of importance scores across all channels of a sample ECG, indicating where the model focuses for the classification of "myocardial infarction". Note that, due to the structure of his kernels, this model assigns equal importance scores across all channels even though the network is bidimensional.

For 1D networks in signal classification, we selected a PyTorch deep residual network for ECG classification (https://github.com/hsd1503/resnet1d), originally developed for the *PhysioNet/CinC Challenge 2017*, and a TensorFlow/Keras ResNet-18 for classifying audio recordings from the *AudioSet dataset* (https://github.com/ZFTurbo/classification_models_1D).

The CAM visualization for a sample Physionet/CinC ECG signal [Fig. 2(d)] enables the identification of the time interval where atrial fibrillation is detected by the PyTorch model. Lastly, by analyzing the importance of different time steps for classifying the "Keys jangling" sound in the two signal wavelets of an AudioSet item [Fig. 2(a)], we observe that the model primarily focuses on the final time steps. Upon listening to the recording, we confirm that a jangling sound indeed occurs at the end of the audio.

3.2. Human Activity Recognition: a real-world use-case

To further evaluate the framework's capabilities, we tested four models from an ongoing study briefly introduced in [15]. This project aimed to compare different strategies for splitting datasets into training and test sets in the field of Human Activity Recognition (HAR). The data originate from the open-source *NTU RGB+D 120 dataset* and were collected using Kinect cameras. Each channel in the signal represents one of the three coordinates (x, y, or z) of one of 25 body joints. In this context, we constructed multiple models, including: two TensorFlow-based CNNs (1D and 2D) and two PyTorch hybrid LSTM-CNN networks (1D and 2D).

CAM visualizations provided valuable insights into the networks decision-making processes, helping to identify potential biases. For instance, when analyzing specific results for 2D models [Fig. 1(a-b)], we observe that the models focus on genuinely important features for classifying the thumbs-up gesture – such as the positions of the right (a) and left fingers (b) – nevertheless multiple spurious features are also considered, e.g., left knee or head positions. Additionally, we notice a significant shift in the model's decision-making strategy when LSTM layers are introduced [Fig. 1(b)]: for example, CAMs highlight differently organized patterns besides different significant features.

4. Conclusions

In this work, we introduced SignalGrad-CAM, a versatile interpretability tool for CNN-based deep learning models applicable to both image and signal data. By extending the principles of Grad-CAM to support signal-oriented CNNs (1D and 2D) and accommodating multiple frameworks, SGrad-CAM bridges a critical gap in the field of explainable AI for signal processing. Additionally, it offers diagnostic and visualization tools that enhance model interpretability and transparency.

However, some limitations exist. Currently, SGrad-CAM does not support 3D data modalities, which limits its applicability to volumetric medical imaging or video-based applications. The next logical extension of this work will involve adapting Grad-CAM to 3D-CNNs, enabling its use in these more complex domains.

Future work will also focus on further expanding SGrad-CAM's capabilities by integrating support for multi-modal and unconventional input types (e.g., dictionaries or other data collections as input), as well as further optimizing the algorithms for large-scale applications. Additionally, we plan to implement other well-known CAM generation algorithms – such as Grad-CAM++ [5] – to provide more options for model explainability. We also intend to develop additional CAM-based evaluation and visualization tools to deepen the understanding of model behavior. For example, averaging CAMs over the entire dataset or generating a "standard deviation" CAM will provide a broader view of model focus and consistency. Furthermore, we will incorporate metrics to assess the faithfulness of the explanations themselves, such as ROAD [13].

By addressing these challenges, SGrad-CAM aims to further advance trust and transparency in AI-driven applications, contributing to the growing demand for more interpretable and accountable machine learning models.

Acknowledgments

Samuele Pe is a PhD student enrolled in the National PhD program in Artificial Intelligence, XXXIX cycle, course on Health and life sciences, organized by Università Campus Bio-Medico di Roma. This work was supported by the Italian Ministry of Research, under the complementary actions to the NRRP "Fit4MedRob - Fit for Medical Robotics" Grant (# PNC0000007). Enea Parimbelli acknowledge funding support provided by the Italian project PRIN PNRR 2022 InXAID - Interaction with eXplainable Artificial Intelligence in (medical) Decision-making. CUP: H53D23008090001 funded by the European Union - Next Generation EU.

Declaration on Generative AI

The author(s) have not employed any Generative AI tools.

References

[1] B. J. Dietvorst, J. P. Simmons, C. Massey, Algorithm aversion: people erroneously avoid algorithms after seeing them err, Journal of Experimental Psychology. General 144 (2015) 114–126. doi:10.1037/xge0000033.

[2] P. Gohel, P. Singh, M. Mohanty, Explainable AI: current status and future directions, 2021. URL: http://arxiv.org/abs/2107.07045. doi:10.48550/arXiv.2107.07045, arXiv:2107.07045 [cs].

[3] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, D. Batra, Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization, in: 2017 IEEE International Conference on Computer Vision (ICCV), 2017, pp. 618–626. URL: https://ieeexplore.ieee.org/document/8237336. doi:10.1109/ICCV.2017.74, iSSN: 2380-7504.

[4] R. L. Draelos, L. Carin, Use HiResCAM instead of Grad-CAM for faithful explanations of convolutional neural networks, 2021. URL: http://arxiv.org/abs/2011.08891. doi:10.48550/arXiv.2011.08891, arXiv:2011.08891 [cs, eess].

[5] A. Chattopadhyay, A. Sarkar, P. Howlader, V. N. Balasubramanian, Grad-CAM++: Improved Visual Explanations for Deep Convolutional Networks, in: 2018 IEEE Winter Conference on Applications of Computer Vision (WACV), 2018, pp. 839–847. URL: http://arxiv.org/abs/1710.11063. doi:10.1109/WACV.2018.00097, arXiv:1710.11063 [cs].

[6] T. Rojat, R. Puget, D. Filliat, J. Del Ser, R. Gelin, N. Díaz-Rodríguez, Explainable Artificial Intelligence (XAI) on TimeSeries Data: A Survey, 2021. URL: http://arxiv.org/abs/2104.00950. doi:10.48550/arXiv.2104.00950, arXiv:2104.00950 [cs].

[7] L. Hiley, A. Preece, Y. Hicks, Explainable Deep Learning for Video Recognition Tasks: A Framework & Recommendations, 2019. URL: http://arxiv.org/abs/1909.05667. doi:10.48550/arXiv.1909.05667, arXiv:1909.05667 [cs, eess, stat].

[8] L. Ye, E. Keogh, Time series shapelets: a novel technique that allows accurate, interpretable and fast classification, Data Mining and Knowledge Discovery 22 (2011) 149–182. URL: https://doi.org/10.1007/s10618-010-0179-5. doi:10.1007/s10618-010-0179-5.

[9] A. Nayebi, S. Tipirneni, C. K. Reddy, B. Foreman, V. Subbian, WindowSHAP: An Efficient Framework for Explaining Time-series Classifiers based on Shapley Values, 2023. URL: http://arxiv.org/abs/2211.06507. doi:10.48550/arXiv.2211.06507, arXiv:2211.06507 [cs].

[10] N. Strodthoff, C. Strodthoff, Detecting and interpreting myocardial infarction using fully convolutional neural networks, Physiological Measurement 40 (2019) 015001. URL: https://dx.doi.org/10.1088/1361-6579/aaf34d. doi:10.1088/1361-6579/aaf34d, publisher: IOP Publishing.

[11] F. Oviedo, Z. Ren, S. Sun, C. Settens, Z. Liu, N. T. P. Hartono, S. Ramasamy, B. L. DeCost, S. I. P. Tian, G. Romano, A. Gilad Kusne, T. Buonassisi, Fast and interpretable classification of small X-ray diffraction datasets using data augmentation and deep neural networks, npj Computational Materials 5 (2019) 60. URL: https://www.nature.com/articles/s41524-019-0196-x. doi:10.1038/s41524-019-0196-x, publisher: Nature Publishing Group.

[12] Y. Yan, H. Zhou, L. Huang, X. Cheng, S. Kuang, A Novel Two-Stage Refine Filtering Method for EEG-Based Motor Imagery Classification, Frontiers in Neuroscience 15 (2021) 657540. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8440963/. doi:10.3389/fnins.2021.657540.

[13] Y. Rong, T. Leemann, V. Borisov, G. Kasneci, E. Kasneci, A Consistent and Efficient Evaluation Strategy for Attribution Methods, 2022. URL: http://arxiv.org/abs/2202.00449. doi:10.48550/arXiv.2202.00449, arXiv:2202.00449 [cs].

[14] R. Tomsett, D. Harborne, S. Chakraborty, P. Gurram, A. Preece, Sanity Checks for Saliency Metrics, Proceedings of the AAAI Conference on Artificial Intelligence 34 (2020) 6021–6029. URL: https://ojs.aaai.org/index.php/AAAI/article/view/6064. doi:10.1609/aaai.v34i04.6064.

[15] S. Pe, G. Nicora, B. M. Vittoria Guerra, S. Sozzi, E. Parimbelli, Systematic Comparison of Machine Learning for Activity Recognition in Cross-Subject vs. NonCross-Subject Scenarios: A Preliminary Analysis, in: 2024 IEEE 8th Forum on Research and Technologies for Society and Industry Innovation (RTSI), 2024, pp. 375–379. URL: https://ieeexplore.ieee.org/abstract/document/10761630. doi:10.1109/RTSI61910.2024.10761630, iSSN: 2687-6817.