

M3d-CAM: A PyTorch library to generate 3D attention maps for medical deep learning

Karol Gotkowski¹, Camila Gonzalez², Andreas Bucher³,
Anirban Mukhopadhyay⁴

¹Graphisch-Interaktive Systeme, Technical university of Darmstadt

²Graphisch-Interaktive Systeme, Technical university of Darmstadt

³Institut für Diagnostische und Interventionelle Radiologie, Universitätsklinikum
Frankfurt

⁴Graphisch-Interaktive Systeme, Technical university of Darmstadt
`karolgotkowski@gmx.de`

Abstract. Deep learning models achieve state-of-the-art results in a wide array of medical imaging problems. Yet the lack of interpretability of deep neural networks is a primary concern for medical practitioners and poses a considerable barrier before the deployment of such models in clinical practice. Several techniques have been developed for visualizing the decision process of DNNs. However, few implementations are openly available for the popular PyTorch library, and existing implementations are often limited to two-dimensional data and classification models. We present M3d-CAM, an easy to use library for generating attention maps of CNN-based PyTorch models for both 2D and 3D data, and applicable to both classification and segmentation models. The attention maps can be generated with multiple methods: Guided Backpropagation, Grad-CAM, Guided Grad-CAM and Grad-CAM++. The maps visualize the regions in the input data that most heavily influence the model prediction at a certain layer. Only a single line of code is sufficient for generating attention maps for a model, making M3d-CAM a plug-and-play solution that requires minimal previous knowledge.

1 Introduction

Deep learning has become a central focus of medical image computing. Yet the lack of interpretability and *black-box* internal workings of deep neural networks, believed by many to be a primary drawback of deep learning methods [1,2], is of particular concern in the medical field. Within the clinical routine, deep learning models would most likely act as *decision support systems* for practitioners. Such a system would only be used insofar it is trusted by the practitioner and he or she can follow the decision process. Fortunately, significant progress has been made in the development of interpretability techniques [3]. For image data, the generation of *attention maps*, that show which input regions network layers consider when making a prediction, are highly popular.

Despite this, few implementations are available that are suited for the particularities of medical data. In this work we aim to close this gap by presenting *M3d-CAM*, an easy-to-use library for generating attention maps that can be seamlessly integrated into any medical imaging project. The library is suited for any CNN-based *PyTorch* [4] classification or segmentation model and is compatible with both two- and three-dimensional data.

With this work we aim to simplify the generation of attention maps, thus increasing the interpretability of classification and segmentation models. We hope that this in turn raises the trust of medical practitioners in deep learning.

2 Materials and methods

M3d-CAM works through injection into a given model, appending certain functions to it. The model itself will work as usual and its predictions remain untouched. *M3d-CAM* itself works behind the scenes and generates attention maps every time `model.forward` is called. The most important functions of *M3d-CAM* are explained in the following sections.

2.1 Injection

To inject a model with *M3d-CAM* one simply needs to insert the single line `model = medcam.inject(model)` after model initialization, as shown in code example 1. This will add all the necessary functionality to the model. Additionally, `inject` offers multiple parameters that can be adjusted. As an example one can define an `output_dir` and set `save_maps=True` to save every generated attention map. One can also set a desired `backend` which is used for generating the attention maps such as Grad-CAM. These backends are explained in more detail in section 2.4. Furthermore, it is possible to choose the layer of interest with `layer`. Hereby one can specifically define a single layer, a set of layers, every layer with *full* or the highest CNN-layer with *auto* for the most comfort.

2.2 Layer retrieval

As the layer names of a model are often unknown to the user, *M3d-CAM* offers the method `medcam.get_layers(model)` for quickly acquiring every layer name of a model.

It should be noted that attention maps cannot be generated for every type of layer. This is the case for fully-connected, bounding box and other special kinds of layers. While the attention for theses layers can be computed, it is not possible to project them back to the original input data.

```

# Import M3d-CAM
from medcam import medcam

# Init your model and dataloader
model = MyCNN()
data_loader = DataLoader(dataset, batch_size=1, shuffle=False)

# Inject model with M3d-CAM
model = medcam.inject(model, output_dir="attention_maps", save_maps=True)

# Continue to do what you're doing...
# In this case inference on some new data
model.eval()
for i, batch in enumerate(data_loader):
    # Every time forward is called, attention maps are generated and
    # saved
    output = model(batch)
    # more of your code...

```

Fig. 1. Example of injecting a model with M3d-CAM.

2.3 Evaluation

M3d-CAM supports the evaluation of attention maps with respect to ground truth masks. This can be done by calling `model.forward(input, mask)`, including the mask in the forward call. The attention map is then internally evaluated by the `medcam.Evaluator` class with a predefined metric by the user. Alternatively, one can call the `medcam.Evaluator` class directly. By calling `model.dump()` or respectively `medcam.Evaluator.dump()` the evaluation results are saved as an excel table.

2.4 Backends

M3d-CAM supports multiple methods for generating attention maps. For simplicity we will refer to them as *backends*. We show examples of how the different backends perform in section 3.

Grad-CAM [5] works by first propagating the input through the entire model. In a second step a desired class in the output is isolated by setting every other class to zero. The output of this isolated class is then backpropagated through the model up to the desired layer. Here, the layer gradients are extracted and

together with the feature maps of the same layer the attention map is computed. The approach of generating an attention map from a specific preferably high layer gives a good compromise between high-level semantics and detailed spatial information. Furthermore, by isolating a specific class Grad-CAM becomes class-discriminant.

Guided Backpropagation [6] works by first propagating the input through the entire model similar to Grad-CAM. In a second step the output is backpropagated through the entire model. However, only the non-negative gradients are passed to the next layer as negative gradients correspond to suppressed pixels deemed not relevant by the authors. The advantage of Guided Backpropagation is that the attention is pixel-precise. The downsides are that it is neither class- nor layer-discriminant.

Guided Grad-CAM is another backend presented by Selvaraju et al. [5], which is a combination of Guided Backpropagation and Grad-CAM in an effort to combine the best of both approaches. When generating attention maps with both backends the resulting attention maps can be combined through simply multiplying them element-wise. The only downside of Guided Grad-CAM is the need of performing backpropagation two times.

Grad-CAM++ is an extension of Grad-CAM introduced by Chattopadhyay et al. [7]. It differs from the naive Grad-CAM in that it weights the gradients before combining them with the feature maps. This results in more precise attention maps, especially when dealing with multiple instances of the same class in an image.

3 Results

In this section, we qualitatively illustrate the attention maps extracted by *M3d-CAM* for three examples. These are found in Fig. 1. The first row shows the original images. The first displays a chest X-Ray used on the task of classification by employing a CovidNet [8], the second a lung CT slice on the task of 2D segmentation with an Inf-Net [9] and the third a 3D prostate CT image on the task of 3D segmentation by employing a nnUNet [10].

The result of applying the Grad-CAM backend is a heatmap-like image of the attention at the desired layer, as shown in Table 2. For Guided Backpropagation, the result is a noise-like image depicting the model attention (see Table 3). The result of combining both approaches in the Guided Grad-CAM approach is a noise-like class and layer discriminant pixel-precise attention map, found in Table 4. Finally, Grad-CAM++ produces heatmaps similar the original Grad-CAM but that more precisely depict the gradient weighting. Examples of these attention maps are shown in Table 5.

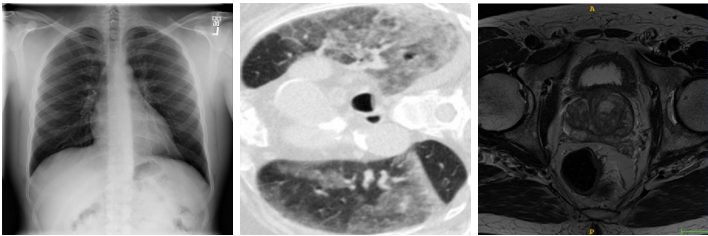


Table 1. From left to right: A chest X-Ray from the COVID-19 image data collection [11], a lung CT slice also from [11] and 3D prostate CT image from the Medical Decathlon dataset [12].

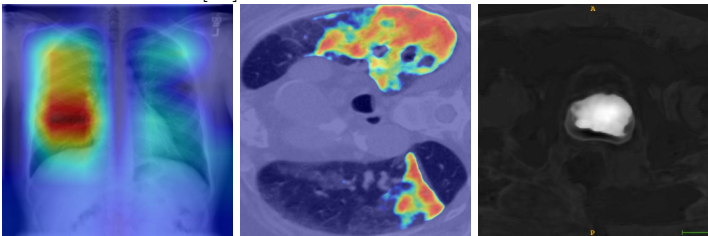


Table 2. The resulting Grad-CAM attention maps.

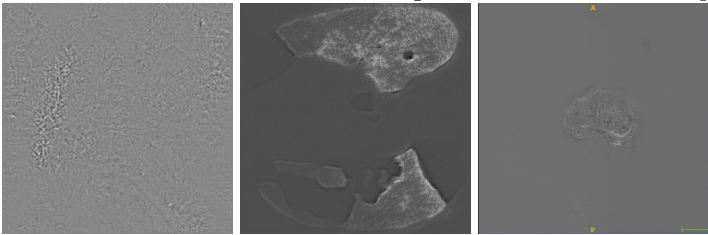


Table 3. The resulting Guided Backpropagation attention maps.

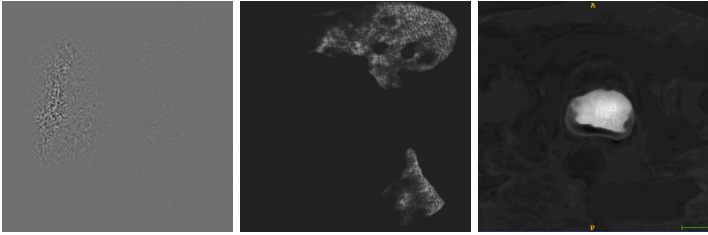


Table 4. The resulting Guided Grad-CAM attention maps.

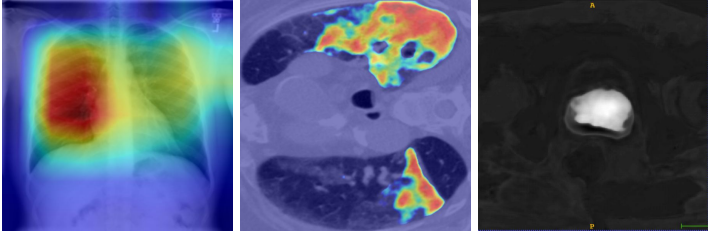


Table 5. The resulting Grad-CAM++ attention maps.

4 Discussion

In this work we present *M3d-CAM*, a plug-and-play interpretability suite suited for *PyTorch* classification and segmentation models. The library implements a number of backend methods to extract attention maps, and supports both two- and three-dimensional data. With this contribution we hope to simplify the application of interpretability techniques to accelerate trust in medical deep learning models.

References

1. Hooker S, Erhan D, Kindermans PJ, et al. A benchmark for interpretability methods in deep neural networks. In: *Advances in Neural Information Processing Systems*; 2019. p. 9737–9748.
2. Huang X, Kroening D, Ruan W, et al. A survey of safety and trustworthiness of deep neural networks: Verification, testing, adversarial attack and defence, and interpretability. *Computer Science Review*. 2020;37:100270.
3. Xu F, Uszkoreit H, Du Y, et al. Explainable AI: A brief survey on history, research areas, approaches and challenges. In: *CCF International Conference on Natural Language Processing and Chinese Computing*. Springer; 2019. p. 563–574.
4. Paszke A, Gross S, Massa F, et al. PyTorch: An imperative style, high-performance deep learning library. In: *Advances in Neural Information Processing Systems*; 2019. p. 8024–8035.
5. Selvaraju RR, Cogswell M, Das A, et al. Grad-cam: Visual explanations from deep networks via gradient-based localization. In: *Proceedings of the IEEE international conference on computer vision*; 2017. p. 618–626.
6. Springenberg JT, Dosovitskiy A, Brox T, et al. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:14126806*. 2014;.
7. Chattopadhyay A, Sarkar A, Howlader P, et al. Grad-cam++: Generalized gradient-based visual explanations for deep convolutional networks. In: *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE; 2018. p. 839–847.
8. Linda Wang ZQL, Wong A. COVID-Net: A Tailored Deep Convolutional Neural Network Design for Detection of COVID-19 Cases from Chest Radiography Images; 2020.
9. Fan DP, Zhou T, Ji GP, et al. Inf-Net: Automatic COVID-19 Lung Infection Segmentation from CT Images. *IEEE TMI*. 2020;.
10. Isensee F, Petersen J, Klein A, et al. nnu-net: Self-adapting framework for u-net-based medical image segmentation. *arXiv preprint arXiv:180910486*. 2018;.
11. Cohen JP, Morrison P, Dao L. COVID-19 image data collection. *arXiv 200311597*. 2020; Available from: <https://github.com/ieee8023/covid-chestxray-dataset>.
12. Simpson AL, Antonelli M, Bakas S, et al. A large annotated medical image dataset for the development and evaluation of segmentation algorithms. *arXiv preprint arXiv:190209063*. 2019;.