Project Report

"Explainable AI using Class Activation Mappings(CAM) in Pytorch"

Submitted by

Kashif Riyaz, Nitesh Morem

Supervised by

Prof. Dr.-Ing. Christian Bergler

Ostbayerische Technische Hochschule Amberg-Weiden
Department of Electrical Engineering, Media and Computer Science

June 30, 2024

Abstract

In the technological era of deep learning, neural network architecture has become incredibly well-known for its ability to look for patterns in data and use those patterns. However, the way to discover these patterns is still unknown to most people. Many people are still unable to comprehend the mechanisms underlying these discoveries.

Our project aims to shed light into the decision-making process of the pretrained models showing the transparency and interpretability of AI systems using various class activation mappings such as Grad-CAM, Grad-CAM++, Eigen-CAM, XGrad-CAM. These class activation mappings focus on relevant parts of the input data to capture features in the image which lead to models prediction.

Overall, deep learning models are great at recognizing patterns, but their lack of transparency makes it difficult to trust them completely. We aim to produce a framework for Explainable AI using different class activation mappings and layer embeddings. By these techniques, we can uncover how pre-trained models make decisions, which in turn promotes trust and interpretability.

1 Introduction and Motivation

The deep neural networks have been the hot topic of this century, revolutionizing various fields with their remarkable performance. However, the nature of these neural networks, "black-box" often highlights some concerns about transparency and interpretability of these networks, hindering trust and broader adoption in critical applications. This project aims to improve the interpretability of deep learning models through Class Activation Mapping and techniques associated with Gradient-Weighted Class Activation Mapping. As AI models are deployed everywhere around the globe especially in the high-stake institutions such as Healthcare, automotive industries, Finance and Banking, it becomes crucial to understand the level of trust these AI models can provide in real life scenarios. This research will bridge the existing gap between understanding intricate CNN decision-making, trusting these models, and using them since there is currently a lack of standard methods for their interpretation. In order to understand and address this, our project unveils the inner workings of AI models and explains the decision-making process by visual explanations. By this we aim not just for building Trust and confidence in AI systems but also ensure that they are used ethically and safely for applications through this project.

The basis of this project is to understand the behavior of deep neural networks through two important methods: visualization of layer-specific embeddings and different class activation mappings. Layer-wise embeddings bring out the learned representations from the layers of a network before the output, which has clear insight with respect to how the model processes information in the background. On the other hand, class activation mappings highlight relevant regions in input data for specific class predictions, giving a visual explanation of the model's decisions. The goals of this research involve class implementation of pre-trained models which are trained on Imagenet dataset, testing them on diverse data, and setting up a extendable standardized framework for pre-trained models in Torch.nn.

The following techniques were used for visualization of class activation mappings(CAM):

1.1 Grad-CAM (Gradient-weighted Class Activation Mapping)

The Grad-CAM is a powerful technique for explaining the visualizations by highlighting the specific parts of a given input which are crucial for the prediction of the target class. It uses 2D activations and gradients to highlight key parts of an input for accurate target class prediction[1][2].

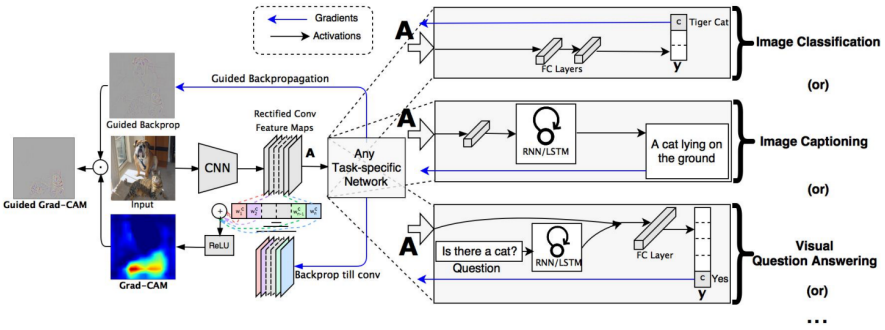

Figure 1: Grad-CAM

Grad-CAM overview: Given an image and a class of interest (e.g., 'tiger cat' or any other type of differentiable output) as input, we forward propagate the image through the CNN part of the model and then through task-specific computations to obtain a raw score for the category. The gradients are set to zero for all classes except the desired class (tiger cat), which is set to 1. This signal is then back propagated to the rectified convolutional feature maps of interest, which we combine to compute the coarse Grad-CAM localization (blue heatmap) which represents where the model has to look to make the particular decision. Finally, we pointwise multiply the heatmap with guided backpropagation to get Guided Grad-CAM visualizations which are both high-resolution and concept-specific[1].

1.1.1 Working of Grad-CAM

- **Forward and Backward Pass:**A user inputs an image through a URL, passes it through the model where the activation of the specific convolutional layer gives a spatial representation. Now, in the Backward pass the gradient of target class is calculated with respect to the last convolutional layer[1][2].
- **Gradient Aggregation:**The gradients are globally averaged to determine the weight of each feature map; these weights indicate the significance for each feature map for the target class[1].
- **Weighted sum:**The final localization map is obtained by calculating the weighted sum of feature maps and then it undergoing ReLU activation to identify the most relevant regions and also the ReLU activation ensures that the only positive values are considered, which focuses on features that support predicted class[1].

1.1.2 Advantages

- Grad-CAM generates localization maps specifically for a particular class, which helps in identifying crucial parts of images for class prediction.
- Easy to implement.
- It is computationally efficient
- Suitable for real time applications.

1.1.3 Limitations

- It may struggle to provide accurate localization for complex structures or multiple occurrences [1].

1.2 Grad-CAM++

Grad-CAM++ is an improved version of Grad-CAM, enhancing visual explanations by improving localization of key regions in input images, particularly when multiple target object instances are present, and generating more precise class-discriminative maps. It also addresses the limitations of Grad-CAM by integrating higher-order derivatives, which helps provide more precise and detailed visual explanations [2][3].

1.2.1 Working of Grad-CAM++

- **Forward and Backward Pass:**This is almost similar to Grad-Cam, but it has additional higher-order derivatives that enable it to give a fine-tuned understanding of the interactions between class scores and activations [3].
- **Gradient Aggregation:**Instead of just globally averaging the gradients, Grad-CAM++ uses a weighted combination of the gradients and higher-order derivatives to calculate the important weights for each feature map[3].
- **Weighted sum:**The process of combining these weighted feature maps is more sophisticated, leading to improved localization, especially for overlapping objects[3].

1.2.2 Advantages

- Improved Localization: Unlike Grad-CAM it uses second-order gradients for more accurate and detailed maps[1].
- Improved Precision: More accurate representation of the areas contributing to the model's prediction [1].

1.2.3 Limitations

- It is more computationally complex as it includes second-order gradients and requires careful handling of second-order derivatives, which make it computationally intensive [3].

1.3 Eigen-CAM

Eigen-Cam differs from methods like Grad-CAM and Grad-CAM++ as it uses Principal Component Analysis (PCA) to identify the most significant components influencing the model's decision. It focuses on capturing major patterns within the activation maps of convolutional layers [1][4].

1.3.1 Working of Eigen-CAM

- **Feature Map Extraction:**The model takes the input image and breaks it down into smaller parts called feature maps.These feature maps hold important information about the image which is given to the model[4].
- **PCA Application:**Eigen-CAM applies a technique called PCA that is responsible for the analysis of feature maps. It uses a unique way of analyzing and capturing patterns and provides insights to gradient based methods[4].
- **Component Analysis:**Here, the components obtained from PCA will be assessed to indicate their contribution towards the predicted class.[4].
- **Localization map:**It combines all the significant components which are important for the model's decision, which creates a localization map that focuses on important regions[4].

1.3.2 Advantages

- Identifies and captures the most important trends in the feature maps.[4].
- This method offers a unique perspective on important regions compared to gradient-based approaches.[4].

1.3.3 Limitations

- Because of the use of PCA it becomes Computationally expensive[4].
- The interpretation of Principal Components can be less insightful[4].

1.4 XGrad-CAM

XGrad-CAM, also known as Axiom-based Grad-CAM, or Extended Grad-CAM which combines the advantages of Grad-CAM for more accurate visual explanations. It is similar to Grad-CAM but it takes additional information from the activation maps and gradients[1][5].

1.4.1 Working of XGrad-CAM

- **Forward and Backward Pass:**An input image is passed through a convolutional neural network like Grad-CAM. During the forward pass, the activations of different layers are recorded by the network. In the backward pass, gradients of the predicted class score are computed with respect to these activations[5].
- **Extended Gradient Aggregation:**In the case of gradient and activation map integration, XGrad-CAM resorts to the most comprehensive technique. It could be done through higher-order gradient analysis or by using any other statistical method which yields an importance of each feature map with a very high accuracy[5].
- **Localization Map:**It generates a better localization map by fusing information from multiple sources. This map sows the specific regions within the input image that most affect the result of the CNN[5].

1.4.2 Advantages

- Enhanced Accuracy:XGrad-CAM is a technique that combines various techniques to enhance the accuracy and dependability of class activation maps[5].
- Comprehensive Explanation:It provides comprehensive visual explanation by incorporating knowledge from both gradients and activation maps[5].
- Versatility:It is a flexible approach that can be used with different model architectures and datasets, making it useful for a range of deep learning applications[5].

1.4.3 Limitations

- Computational Complexity:Merging a number of steps and computations involved together may increase the computational cost for XGrad-CAM[5].
- Implementation Challenges:It requires good tuning backed by adequate validation to make sure that gradient and activation map information are effectively integrated for the intended estimate[5].
- Interpretability:Like most complex methods, interpreting the results from XGrad-CAM requires domain knowledge or expertise[5].

2 Related Work

The next big breakthrough in terms of how to make deep neural networks more interpretable came with Grad-CAM: Gradient-weighted Class Activation Mapping, by Selvaraju et al. (2017) [1]. It uses gradient information coming into the final convolution layer to generate a coarse localization map, highlighting key regions in the input image. Thus, the approach enables visual explanations from deep networks and gives insights into how CNN models function.

Grad-CAM was generalized into Grad-CAM++ by Chattopadhay et al. [3], which refines the localization map by using second-order gradients. Grad-CAM++ refines the fidelity of visual explanation of deep convolutional networks, giving fine details and more informative insight into the model prediction.

Muhammad et al. (2020) [4] proposed Eigen-CAM, which employed PCA to obtain class activation maps. With principal components, Eigen-CAM improves the accuracy of localization maps and thus contributes to better interpretability and performance in applications involving neural networks.

Fu et al. (2020) [5] proposed XGrad-CAM for improving the accuracy and explainability of AI models using advanced gradient-based visual explanation methods. In essence, this is an extension of Grad-CAM that uses cross-gradient information to assist in localizing, more accurately, several locations in which the model could be making specific decisions.

These methodologies, all together, are major steps toward the interpretability of deep learning techniques, providing different ways of visualization and understanding of the mechanisms of decisions taken by a convolutional neural network. They underline the necessity of scrutable, interpretable AI systems to build trust in their inner functioning and realize bigger applications in various domains.

3 Data Material and Preprocessing

The dataset used in this project is open source. The images used for classification tasks are from Google. The following images were fetched via google search and used to show explainability techniques. The URL of the image was copied, and the following code is used to load an image into a NumPy array, first by using the "requests" library to fetch the image data, then opening the image with "Image.open" method from the PIL library, and finally turning it into a NumPy array.

```
tensor([[[[ 0.7933, -0.2856, -1.5185,  ..., -0.4226, -1.7240, -2.0152],
          [ 0.5193, -1.3130, -1.3987,  ..., -0.8849, -0.9192, -1.6042],
          [-0.5424, -0.9534, -1.2959,  ..., -1.3644, -1.6555, -1.7754],
          ...,
          [ 0.0912, -1.0048, -0.6794,  ..., -1.1418, -1.2445, -1.5357],
          [-0.9705, -0.7479, -0.1314,  ..., -1.6213, -1.6384, -1.5870],
          [-0.4226, -0.7822, -0.8164,  ..., -1.9124, -1.7754, -1.4843]],

         [[ 1.2206,  0.4153, -1.0378,  ...,  0.0301, -1.0028, -1.4055],
          [ 1.1331, -0.7052, -0.9678,  ..., -0.5826, -0.4251, -0.5476],
          [ 0.1527, -0.4951, -0.9153,  ..., -0.9153, -1.4405, -1.1078],
          ...,
          [ 0.8004, -0.3550,  0.0476,  ..., -1.5455, -1.8606, -1.6856],
          [-0.3200, -0.0574,  0.3803,  ..., -1.6681, -1.8957, -1.6331],
          [ 0.1176, -0.0224, -0.4251,  ..., -1.9657, -2.0007, -1.7731]],

         [[-0.0615, -1.1944, -1.6999,  ..., -0.7238, -1.7347, -1.6824],
          [-0.6715, -1.7347, -1.5430,  ..., -1.4559, -0.4275, -1.4210],
          [-1.2293, -1.4384, -1.4907,  ..., -1.5081, -1.7173, -1.4559],
          ...,
          [-0.5495, -1.6824, -1.1596,  ..., -1.6302, -1.5604, -1.6824],
          [-1.3339, -1.3687, -0.9678,  ..., -1.6650, -1.7696, -1.5430],
          [-1.1770, -1.3687, -1.3339,  ..., -1.8044, -1.8044, -1.6824]]]])
```

(a) input tensor after preprocessing

(b) input image after preprocessing

Figure 2: Processing the raw image as tensors

The image is resized to 224x224 pixels using the "cv2.resize" function from the OpenCV library. It is then normalized by turning the image into a float32 NumPy array and scaling the pixel values into the range [0,1]. The image is then preprocessed using the "preprocess_image" function from Jacobs gills Github repository "pytorch_grad_cam.utils.image", standardizing the image into a tensor of mean [0.485, 0.456, 0.406] and standard deviation [0.229, 0.224, 0.225] to be fed into the pre-trained model, as shown in Figure (a). These values of the mean and standard deviation are used while training every model on the ImageNet dataset. So, we processed the raw URL of an image into the input tensor, which is fed into the model to make predictions and retrieve the label only for Grad-cam visualization, in addition with, the converted image as shown in Figure (b).

4 Methodology

The Framework of this project uses the CNN models which are trained on ImageNet dataset and applies CAM visualization techniques such as Grad-CAM, Grad-CAM++, Eigen-CAM, and XGrad-CAM to visualize important regions in the input image.

4.1 Framework overview

After installing all the necessary modules for this project, the framework consists of five sections:

4.1.1 Importing Modules

In this section all the necessary modules are imported which includes Pytorch, numpy, requests, warnings, PIL and matplotlib. Apart from these modules we also imported the GitHub repository by Jacob gill named "pytorch_grad_cam"

Using this repository all the CAM visualization techniques such as Grad-CAM, Grad-CAM++, Eigen-CAM, and XGrad-CAM are also imported [2].

4.1.2 Defining Functions:

In this section three necessary functions which are used in the whole framework are defined which include:

- **image_transform function:** This function takes an image URL as input. It opens the image from the provided URL, resizes it, and converts it into an input tensor according to data preprocessing which is suitable for model prediction.
- **Predict_and_get_target_class function:** It makes predictions with a model input, taking a pre-trained model and an input image tensor. It uses the input tensor to generate a prediction from the model and saves the label index number of the highest-scoring prediction. The function will look up the corresponding label name in ImageNet using this label index number. Finally, it returns the class index with its name.
- **Get_conv_layer function:** The function takes a pre-trained model as input and counts the number of convolutional layers in the model. It then asks for user input on the layer he wants to visualize. This returns the layer chosen by the user for further processing.

4.1.3 Importing image and model:

In this section, a pre-trained CNN model (e.g. densenet121 or Resnet18) is selected for its performance on image classification tasks. Image URL is defined, and all the functions used to get the image and input tensor form the image URL, model prediction, and layer number from the model are also stored in respective variables.

4.1.4 CAM visualization techniques:

This section uses visualization techniques such as Grad-CAM, Grad-CAM++, Eigen-CAM, and XGrad-CAM. When applied each technique takes the model and layer number as input and generates a heat map which highlights the regions in the image that contribute most to the target class prediction. These CAM techniques have been imported from jacob gill's github repository and his github Tech Blog called "Advanced AI Explainability with pytorch-gradcam ".

4.1.5 Visualization:

In this section the heat maps from the each CAM technique are masked on the image as shown in Figure 3:

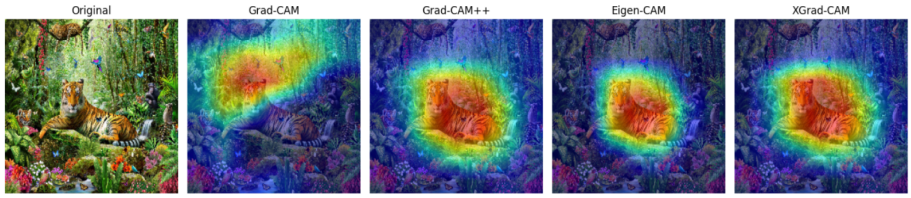

Figure 3: Visualization of model prediction

5 Experiments

We initially wondered whether a single framework could offer visual explanations using various Class Activation Mapping (CAM) techniques and effectively compare them. After conducting several experiments, we discovered that the enhanced techniques—such as Grad-CAM++, Eigen-CAM, and XGrad-CAM—yield more precise and interpretable visual explanations than the original Grad-CAM.

5.1 Experimental Setup and procedures:

- Firstly, we created a basic framework which takes a pre-trained model and an input image URL. torch.nn library, models such as VGG16, ResNet50, and InceptionV3 were imported.
- After Which we started to select specific layers where we can understand the specific regions of images being focused by the model.
- Basically, the framework highlights the regions and important features of the data or images on which the model has made certain predictions.
- We Improved the framework by researching the architecture of these models and looked into the structure of Convolutional layers of these architectures which helped us understand how to get a specific layer from a model.
- Our framework prompts us to choose a specific convolutional layer across all the different models.
- It also enables the use of Grad-CAM, Grad-CAM++, Eigen-CAM, and XGrad-CAM to view the prediction and select various models in addition to a particular layer. Additionally, for easier comparison and explanation of these models, the Framework provides parallel visualization of these various techniques next to one another.
- Initially, we worked with a single pre-trained model and advanced further to many other models and extended them into our framework.

6 Results and Discussions

The class activation mapping (CAM) techniques were implemented on various pre-trained models, including DenseNet121, ResNet152, and inception_v3, each with a different number of convolutional layers. For analysis, the mappings were examined with different layer numbers: the first, middle, and last convolutional layers of each model. The CAM techniques were applied to a diverse set of images, including both single-label and multi-label images, to evaluate the consistency and robustness of the explanations provided by these models.

In this report we have only shown the heatmaps of only three models which are as follows,

- Densnet121 with 120th convolutional layers.
- Resnet152 with 155th convolutional layers.
- Inception_v3 with 96th convolutional layers.

6.1 Heat map for first Layer

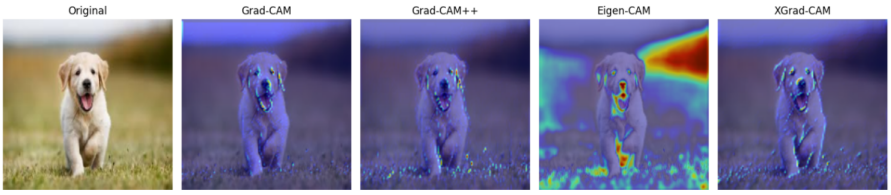

Figure 4: Heat map of the first layer of resnet152

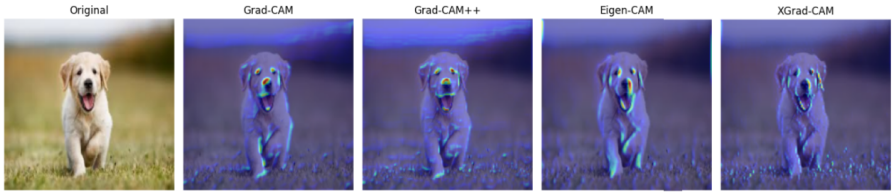

Figure 5: Heat map of the first layer of Densnet121

The similarity in the heat maps for Resnet152 and Densnet121 is due to their similarity in the structure of convolutional layers both the models have

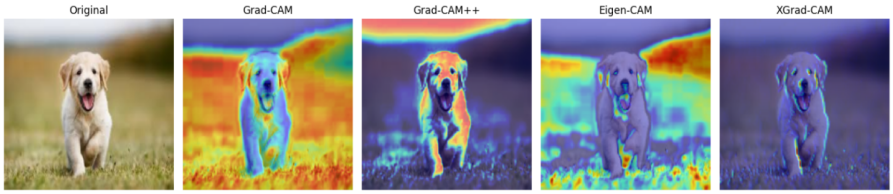

Figure 6: Heat map of the first layer of inception_v3

Conv2d(3, 64, kernel_size=(7, 7), stride=(2, 2), padding=(3, 3), bias=False) Layer due to which the model focuses on capturing low-level features such as edges,texture and colors.

However the inception_v3 has a different structure for the first convolutional layer Conv2d(3, 32, kernel_size=(3, 3), stride=(2, 2), bias=False) which focuses on capturing multi scale features simultaneously by using parallel convolutions of different size (e.g., 1x1, 3x3, 5x5) and hence leads to different activation patterns.The localization for XGrad-CAM is similar because it is enhanced as it focuses on most relevant features.

6.2 Heat map for Middle Layer

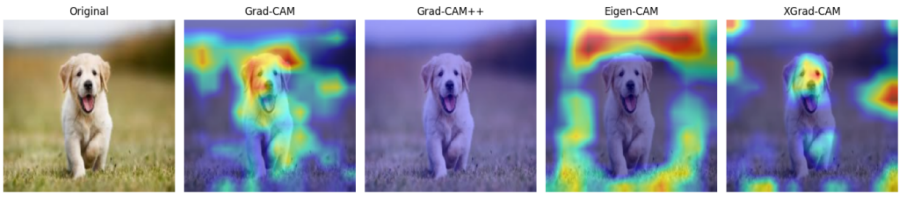

Figure 7: Heat map of the middle layer of resnet152

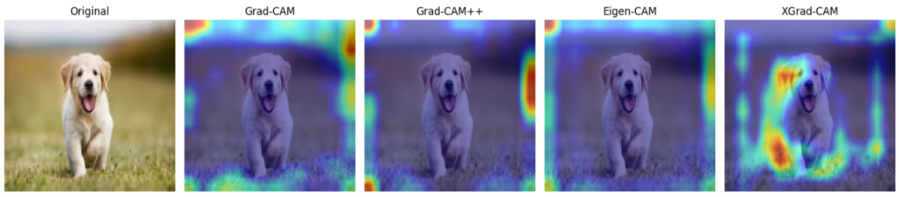

Figure 8: Heat map of the middle layer of Densnet121

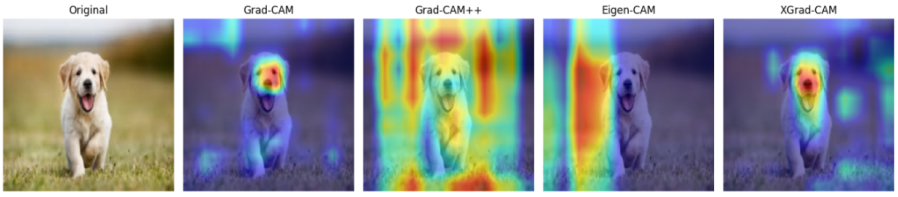

Figure 9: Heat map of the middle layer of inception_v3

Among the three models inception_v3 shows the best performance in highlighting important features in the image due to its multi scale feature extraction technique. Resnet152 also refines effectively to some extent due to its dense architecture but densenet faces challenges in blending the features effectively

6.3 Heat map for Last Layer

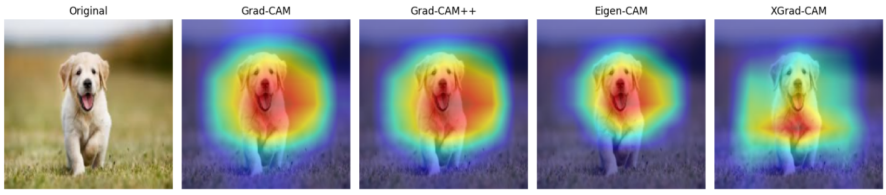

Figure 10: Heat map of the Last layer of resnet152

Figure 11: Heat map of the Last layer of Densnet121

In the last convolutional layer, all models perform well with CAM techniques. They create heat maps that accurately show features important for classifying images. This shows how effective CAM techniques are at revealing these crucial features that help the models make accurate predictions.

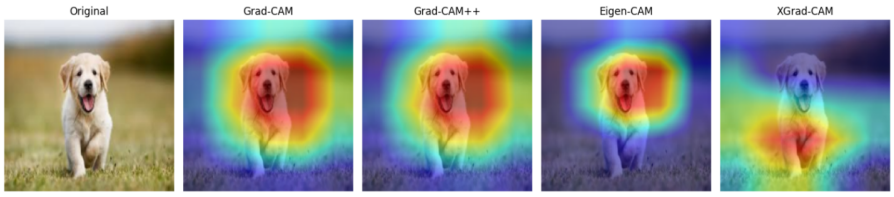

Figure 12: Heat map of the Last layer of inception_v3

6.4 Evaluation Metrics

The effectiveness of the visualizations created by the framework is evaluated by looking at how well they highlight the important areas in the input image that influence the model's prediction. Each visualization is checked to see how clear, easy to understand, and relevant it is. By examining these visualizations, we can see which parts of the image are most important for the model's decisions.

7 Summary, Conclusion and Future Work

This study was focused on class activation mappings (CAM) to show the transparency and interpretability of AI systems involving convolutional neural networks. The key findings in this study are as follows,

- All the visualization methods surpass each other in different criteria such as localization accuracy, clarity and interpretability of the heatmaps.
- Grad- CAM provides a good balance of performance and interpretability but sometime is less precise and produces noisy heatmaps.
- Grad-CAM++ shows good performance involving multiple objects and generally provides better localization and sharper heatmaps than Grad-CAM.
- Eigen-CAM shows those features which the network deems important without constraining itself to a particular class.
- XGrad-CAM provides more precise visualization compared to standard Grad-CAM. However, the results suggest that it might not always outperform Grad-CAM.

In conclusion, if our primary goal is to obtain the most precise and interpretable heatmap, Grad-CAM ++ is often considered the better choice than Grad-CAM. However, our localization is not tied to specific classes. Eigen-CAM is a better choice than the rest.

7.1 Future Scope

In the future the project can be improved in various areas to overcome some of the problems which we faced during our work. Below are the detailed suggestions:

7.1.1 Improved image preprocessing for diverse models

Currently, our framework preprocesses the images according to standards in the ImageNet dataset, which hence significantly reduces our range of choice in models.

This limitation can be addressed by implementing an automated preprocessing adjustment that detects models' input data shape and type required, then adjusts the preprocessing steps accordingly.

7.1.2 Expanding Explainable AI to NLP

The current work focuses on the image classification CNN models, but can be further extended to models for NLP. One can adapt these techniques to models from NLP and use the need to highlight words and phrases that influence the model's predictions.

7.1.3 Framework for multi-layer sequence highlighting

The current framework has been designed to work with one layer at a time; hence, its depth of interpretability is limited, but can be enhanced such that the framework should be in a position to handle a sequence of layers so as to give comprehensive understandings of how the model makes its decisions. These features, added to the framework, make it more robust, versatile, and support a wide range of models; therefore, it is such a powerful tool in enhancing transparency and trustworthiness in AI systems.

References

[1] R. R. Selvaraju *et al.*, "Grad-cam: Visual explanations from deep networks via gradient-based localization," *IEEE International Conference on Computer Vision (ICCV)*, pp. 618–626, 2017.

[2] J. Gildenblat and contributors, *Pytorch library for cam methods*, `https://github.com/jacobgil/pytorch-grad-cam`, 2021.

[3] A. Chattopadhay *et al.*, "Grad-cam++: Generalized gradient-based visual explanations for deep convolutional networks," *IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 839–847, 2018.

[4] A. Muhammad *et al.*, "Eigen-cam: Class activation map using principal components," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–12, 2020.

[5] R. Fu *et al.*, "Xgrad-cam: Towards accurate and explainable ai," *arXiv preprint arXiv:2008.02312*, 2020.