Saliency Slider Using Grad-CAM

Kian Ambrose, Matheus Kunzler Maldaner, Lexie Certo

April 2024

1 Introduction

For our project we developed an AI system that employs a convolutional neural network (CNN) to classify images and determine which parts of those images are most important for its decision. Upon determining the important features, we created and implemented a "saliency slider" that reconstructs the image piece by piece, starting with the most important features and gradually adding in each feature until the original image is displayed again. Our goal was to highlight the AI's decision-making process, working to make the decisions of complex AI models more understandable and transparent.

In its final form, the web app is deployed and accessible to all users. It allows users to upload an image, after which they can use a saliency slider to dynamically view how different portions of the image contribute to the neural network's classification decision.

Visit SaliencySlider to try it out the web app yourself with your own images. For those interested in experimenting directly with the code or making modifications, we have also made the project available on Google Colab, allowing for real-time code changes and testing without any local setup. Please note, currently, moving the slider rapidly can send excessive requests to the server, potentially causing the web app to crash. We are aware of this issue and are working on optimizing this in the upcoming weeks to handle requests more efficiently and prevent downtime.

2 Explaining the Code

Our first step was deciding what to use for our neural network. We determined that the VGG19 model would fit our needs well. This is a pre-trained CNN from the Keras library which is trained on the ImageNet dataset, a large-scale database of images used for machine learning. The model starts by taking in an image, then resizing and pre-processing the image to fit the model. We then have the model process the images using Grad-CAM to determine what parts of the image were most influential for the model's decision. We use Grad-CAM to create a saliency overlap, similar to a heatmap, which highlights the regions

of the image most relevant for the models decision. We will dive deeper into Grad-CAM in the next section.

After this model was implemented, we worked on our saliency slider. As described above, the saliency slider allows the user to construct the image piece by piece, starting with the most important aspects, ultimately ending up with the full image. As the user moves the slider, the opacity of the saliency overlap changes. When the user moves towards the maximum, the overlap becomes less visible, showing more of the original image. Along with the slider, we also have a predicted class variable which shows what part of the image was most important for the model's classification for each percentage of saliency.

Finally, we created a Web App for this demonstration using Django. We then deployed the web app using *pythonanywhere*. Users can interact with the system by uploading images, viewing them, and receiving feedback on the images in real time. Additionally, source code used for the website creation can be found in our GitHub repository.

3 Grad-CAM

To determine the most important parts of the image, we used Grad-CAM. Gradient-weighted Class Activation Mapping (Grad-CAM) is a technique focused on the interpretability of CNNs by highlighting which parts of an image are used for the model's predictions. It is used widely in image classification within the field of computer vision. For our purposes, we were interested in the Grad-CAM heatmap, which shows which parts of the images were used in the prediction making process of the CNN.

Here is the process for Grad-CAM:

1. The first step in the process is forward propagating an image through the CNN to produce class scores at the output layer. In other words, the image is passed through the layers of the neural network. Within each layer, the image undergoes a mathematical transformation. After the image has passed through all of the layers it reaches an output layer. This output layer produces output values, which represent the confidence level that the image is classified correctly.

2. After the forward propagating, we select the class of interest for which the heatmap is to be generated. The class of interest can be any label within the dataset that we want to focus on.

3. Grad-CAM then computes the gradients of the class score with respect to the feature maps of a chosen convolutional layer. The feature maps are the outputs of convolutional layers (more on this later) within the CNN. They are grids of values representing the aspects of the image that have been learned. This is represented as:

$$\frac{\partial y^c}{\partial A^k}$$

where y^c is the score for class c and A^k represents the feature maps at layer k.

4. Next, Grad-CAM takes the average of these gradients across the entire area of each feature map, turning the gradients into a single average value per feature map. This steps makes the complex data more manageable. This is shown as:
$$\alpha_k^c = \frac{1}{Z} \sum_i \sum_j \frac{\partial y^c}{\partial A_{ij}^k}$$
where Z is the total number of elements in feature map A^k, and i, j index the spatial dimensions.

5. Finally, using these scalars as weights, it applies a weighted combination of the original feature maps, followed by a rectified linear unit (ReLU) function that ensures there are no negative values in the visualization. Here is this representation:
$$L_{\text{Grad-CAM}}^c = \text{ReLU}\left(\sum_k \alpha_k^c A^k\right)$$

Through these operations, Grad-CAM converts a complex CNN into a visual representation (heatmap) emphasizing the key areas of the input image that influence the model's decisions.

4 Linear Algebra in Convolutional Layers

Similar to the labs in class, we wanted to showcase the role that linear algebra plays throughout our project. As we know, linear algebra is used in many places throughout neural networks, but for this section we will focus specifically on the linear algebra in convolutional layers, as we used on a CNN for this project.

Convolutional layers, which are crucial components of CNNs, perform operations that are essentially specialized linear transformations. This convolution operation involves sliding a smaller matrix over the input image or feature map and computing the dot product at each position.

This math is represented as follows: Suppose $\mathbf{X}$ is an $m \times n$ matrix representing an input image (or feature map) and $\mathbf{K}$ is a $p \times q$ matrix representing a filter. The convolution operation is a sum of the dot products, producing a new matrix $\mathbf{Y}$:

$$Y_{ij} = (\mathbf{K} * \mathbf{X})_{ij} = \sum_{a=1}^{p} \sum_{b=1}^{q} K_{ab} \cdot X_{i+a-1, j+b-1}$$

Here, $*$ denotes the convolution operation, not matrix multiplication. The convolution can be seen as applying a linear transformation to the patches of the image, where the filter defines the linear transformation.

Convolution layers tie into our project in the processing and analyzing of image data. These layers use many different filters to perform convolutions across input images, effectively capturing important features such as edges, textures, and shapes. By applying multiple filters, each designed to detect different aspects of the image, these layers create a series of feature maps. These maps are then used as inputs for the following layers in the network, allowing the model to build an understanding of the image.

5 Individual Contributions

We took a collaborative approach to creating our project and tried to work on everything together as opposed to dividing up tasks. With that being said, there were some areas that each individual focused more heavily on.

Matheus :
Matheus' primary focus was on writing the code to implement Grad-CAM. He had the most experience with machine learning algorithms and background knowledge on CNNs. He also worked on the deployment side, and made sure our web app was deployed successfully. Beyond this, he helped Kian and Lexie with other tasks such as writing the paper.

Lexie :
Lexie's primary focus was on understanding the linear algebra behind CNNs. She researched convlutional layers to understand how they tied in closely to the information we learned in the course. She also helped write the code to implement Grad-CAM and edit the paper.

Kian :
Kian's primary focus was on writing the paper. He wrote the paper using Latex so that the formulas would be more effectively represented. He also researched Grad-CAM and the process that Grad-CAM takes to create a heatmap. Finally, he helped Matheus with writing the code.

6 Bibliography

6.0.1 For Grad-CAM

1) Selvaraju, Ramprasaath R., et al. "Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization." International Journal of Computer Vision, vol. 128, no. 2, Oct. 2019, pp. 336–59. Crossref, https://doi.org/10.1007/s11263-019-01228-7.

2) Chattopadhay, Aditya, et al. "Grad-CAM++: Generalized Gradient-Based Visual Explanations for Deep Convolutional Networks." 2018 IEEE Winter Conference on Applications of Computer Vision (WACV), IEEE, 2018. Crossref,

https://doi.org/10.1109/wacv.2018.00097.

6.0.2 For Keras

3) Team, Keras. "Keras Documentation: VGG16 and VGG19." Keras, keras.io/-api/applications/vgg/. Accessed 28 Apr. 2024.

6.0.3 For Django

4) "Writing Your First Django App, Part 1: Django Documentation." Django Project, docs.djangoproject.com/en/5.0/intro/tutorial01/. Accessed 28 Apr. 2024.

6.0.4 For Deploying

5) PythonAnywere. "Deploying an Existing Django Project on Pythonanywhere." PythonAnywhere Help, 10 Feb. 2016, help.pythonanywhere.com/pages/-DeployExistingDjangoProject/.

6.0.5 CNNs

6) Skalski, Piotr. "Gentle Dive into Math behind Convolutional Neural Networks." Medium, Towards Data Science, 14 Apr. 2019, towardsdatascience.com/-gentle-dive-into-math-behind-convolutional-neural-networks-79a07dd44cf9.
7) "What Are Convolutional Neural Networks?" IBM, 22 Apr. 2024, www.ibm.c-om/topics/convolutional-neural-networks.