Lab 7-2: GradCAM final-report

CHI YEONG HEO[1],
[1]School of Computer Science and Engineering, Pusan National University, Busan 46241 Republic of Korea

1. Introduction

This report provides an analysis of the paper Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization [2], with pretrained models (AlexNet[1], VGGNet[3]).

2. Grad-CAM

2.1. Overview

Gradient-weighted Class Activation Mapping (Grad-CAM) is a method developed to provide visual explanations for the decisions of convolutional neural networks (CNNs). Grad-CAM computes the gradients of a specific target output and backpropagates them to the final convolutional layer to generate coarse heatmaps, which localize the regions of the input image that are most relevant to the model's prediction. This technique is applicable to a wide range of CNN architectures without requiring modifications to the network's structure or additional training. By enhancing the interpretability of model predictions, Grad-CAM contributes to improving the transparency and explainability of computer vision systems.

2.2. Method

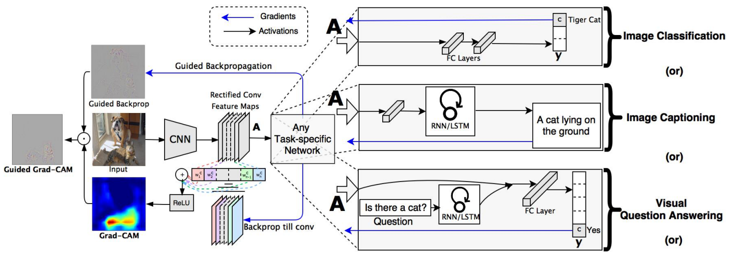

Figure 1: Overview of Grad-CAM: Gradients for a target class are backpropagated to the final convolutional layer, producing a coarse localization map (blue heatmap) that highlights regions relevant to the model's prediction. Combining this map with Guided Backpropagation yields high-resolution, class-specific visualizations.

Grad-CAM generates class-discriminative localization maps by utilizing the gradients of the target class score y^c , with respect to feature map activations A^k of a convolutional layer, $\frac{\partial y^c}{\partial A^k}$. The gradients are globally average-pooled to compute importance weights α_k^c, representing the contribution of each feature map to the target class.

$$\alpha_k^c = \overbrace{\frac{1}{Z}\sum_i\sum_j}^{\text{global average pooling}} \underbrace{\frac{\partial y^c}{\partial A_{ij}^k}}_{\text{gradients via backprop}} \tag{1}$$

A linear combination of the feature maps weighted by α_k^c, followed by ReLU, produces the localization map:

$$L^c_{\text{Grad-CAM}} = \text{ReLU}\left(\sum_k \alpha_k^c A^k\right).$$

ReLU ensures only positive contributions are considered, resulting in a coarse heatmap highlighting key regions influencing the prediction.

Grad-CAM generalizes CAM: Grad-CAM extends CAM to work with a wider range of CNN architectures, including those used for complex tasks like image captioning and VQA.

Guided Grad-CAM: Combining Grad-CAM with Guided Backpropagation provides high-resolution, class-specific visualizations by integrating global and fine-grained details.

Counterfactual explanations: Modifying Grad-CAM approach allows models to identify regions that, if removed, would increase the model's confidence in its current prediction. This is achieved by altering the importance weights:

$$\alpha_k^c = -\frac{1}{Z}\sum_i\sum_j \frac{\partial y^c}{\partial A_{ij}^k}.$$

3. AlexNet

3.1. Overview

AlexNet, introduced in Krizhevsky et al. [1], marked a significant breakthrough in the field of computer vision. It was the winning model of the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2012, achieving top-1 and top-5 error rates of 37.5% and 17.0%, respectively. The model leveraged deep convolutional neural networks (CNNs) and was one of the first to demonstrate that such architectures, when combined with large datasets and GPU acceleration, could significantly outperform traditional machine learning techniques. AlexNet's success is often credited with sparking the deep learning revolution in computer vision, paving the way for more advanced architectures like VGGNet, ResNet, and Inception.

3.2. Implementation Details

In this implementation, a pretrained AlexNet model available at `https://download.pytorch.org/models/alexnet-owt-4df8aa71.pth` was utilized. The detailed network architecture of AlexNet is summarized in Table 1. This architecture consists of five convolutional layers followed by three fully connected layers, utilizing ReLU activations, dropout for regularization, and max pooling for downsampling. These design elements were instrumental in improving both computational efficiency and performance on image classification tasks.

4. VGGNet

4.1. Overview

VGGNet, introduced in Simonyan and Zisserman [3], is a seminal convolutional neural network architecture that significantly advanced the field of computer vision. It is renowned for demonstrating the impact of increasing network depth while using small convolutional filters (e.g., 3×3), which were uncommon at the time. This architectural principle not only enhanced performance on large-scale visual recognition tasks but also influenced numerous subsequent designs. By achieving second place in the ILSVRC-2014 classification task with an impressive top-5 error rate of 7.3%, VGGNet set a benchmark for simplicity and scalability, thereby cementing its role as a cornerstone model in deep learning research.

4.2. Implementation Details

In this implementation, the pretrained VGG-16 model provided by the PyTorch library is utilized. Specifically, the 'vgg16' configuration correspond-

Layer (Type)	Output Shape	Param #	Cumulative Param #
Conv2d-1	[-1, 64, 55, 55]	23,296	23,296
ReLU-2	[-1, 64, 55, 55]	0	23,296
MaxPool2d-3	[-1, 64, 27, 27]	0	23,296
Conv2d-4	[-1, 192, 27, 27]	307,392	330,688
ReLU-5	[-1, 192, 27, 27]	0	330,688
MaxPool2d-6	[-1, 192, 13, 13]	0	330,688
AlexNet_Block-7	[-1, 192, 13, 13]	0	330,688
Conv2d-8	[-1, 384, 13, 13]	663,936	994,624
ReLU-9	[-1, 384, 13, 13]	0	994,624
AlexNet_Block-10	[-1, 384, 13, 13]	0	994,624
Conv2d-11	[-1, 256, 13, 13]	884,992	1,879,616
ReLU-12	[-1, 256, 13, 13]	0	1,879,616
AlexNet_Block-13	[-1, 256, 13, 13]	0	1,879,616
Conv2d-14	[-1, 256, 13, 13]	590,080	2,469,696
ReLU-15	[-1, 256, 13, 13]	0	2,469,696
MaxPool2d-16	[-1, 256, 6, 6]	0	2,469,696
Dropout-17	[-1, 9216]	0	2,469,696
Linear-18	[-1, 4096]	37,752,832	40,222,528
ReLU-19	[-1, 4096]	0	40,222,528
Dropout-20	[-1, 4096]	0	40,222,528
Linear-21	[-1, 4096]	16,781,312	57,003,840
ReLU-22	[-1, 4096]	0	57,003,840
Linear-23	[-1, 1000]	4,097,000	61,100,840
Total Params:		61,100,840	
Trainable Params:		61,100,840	
Non-trainable Params:		0	
Input Size:		0.57 MB	
Forward/Backward Pass Size:		9.38 MB	
Params Size:		233.08 MB	
Estimated Total Size:		243.04 MB	

Table 1: Pretrained AlexNet Network Architecture Summary

ing to the 'D' variant is adopted. This model is initialized with the pretrained weights available at `https://download.pytorch.org/models/vgg16-397923af.pth`. The architectural summary of VGG-16 is outlined in Table 2.

Layer (Type)	Output Shape	Param #	Cumulative Param #
Conv2d-1	[-1, 64, 224, 224]	1,792	1,792
ReLU-2	[-1, 64, 224, 224]	0	1,792
Conv2d-3	[-1, 64, 224, 224]	36,928	38,720
ReLU-4	[-1, 64, 224, 224]	0	38,720
MaxPool2d-5	[-1, 64, 112, 112]	0	38,720
Conv2d-6	[-1, 128, 112, 112]	73,856	112,576
ReLU-7	[-1, 128, 112, 112]	0	112,576
Conv2d-8	[-1, 128, 112, 112]	147,584	260,160
ReLU-9	[-1, 128, 112, 112]	0	260,160
MaxPool2d-10	[-1, 128, 56, 56]	0	260,160
Conv2d-11	[-1, 256, 56, 56]	295,168	555,328
ReLU-12	[-1, 256, 56, 56]	0	555,328
Conv2d-13	[-1, 256, 56, 56]	590,080	1,145,408
ReLU-14	[-1, 256, 56, 56]	0	1,145,408
Conv2d-15	[-1, 256, 56, 56]	590,080	1,735,488
ReLU-16	[-1, 256, 56, 56]	0	1,735,488
MaxPool2d-17	[-1, 256, 28, 28]	0	1,735,488
Conv2d-18	[-1, 512, 28, 28]	1,180,160	2,915,648
ReLU-19	[-1, 512, 28, 28]	0	2,915,648
Conv2d-20	[-1, 512, 28, 28]	2,359,808	5,275,456
ReLU-21	[-1, 512, 28, 28]	0	5,275,456
Conv2d-22	[-1, 512, 28, 28]	2,359,808	7,635,264
ReLU-23	[-1, 512, 28, 28]	0	7,635,264
MaxPool2d-24	[-1, 512, 14, 14]	0	7,635,264
Conv2d-25	[-1, 512, 14, 14]	2,359,808	9,995,072
ReLU-26	[-1, 512, 14, 14]	0	9,995,072
Conv2d-27	[-1, 512, 14, 14]	2,359,808	12,354,880
ReLU-28	[-1, 512, 14, 14]	0	12,354,880
Conv2d-29	[-1, 512, 14, 14]	2,359,808	14,714,688
ReLU-30	[-1, 512, 14, 14]	0	14,714,688
MaxPool2d-31	[-1, 512, 7, 7]	0	14,714,688
Linear-32	[-1, 4096]	102,764,544	117,479,232
ReLU-33	[-1, 4096]	0	117,479,232
Dropout-34	[-1, 4096]	0	117,479,232
Linear-35	[-1, 4096]	16,781,312	134,260,544
ReLU-36	[-1, 4096]	0	134,260,544
Dropout-37	[-1, 4096]	0	134,260,544
Linear-38	[-1, 1000]	4,097,000	138,357,544
Total Params:		138,357,544	
Trainable Params:		138,357,544	
Non-trainable Params:		0	
Input Size:		0.57 MB	
Forward/Backward Pass Size:		218.59 MB	
Params Size:		527.79 MB	
Estimated Total Size:		746.96 MB	

Table 2: Pretrained VGGNet Network Architecture Summary

5. How to Generate Grad-CAM

The following steps outline the process of generating Grad-CAM.

5.1. Define the Grad-CAM Class:

A custom class, `GradCAM`, is implemented to handle all necessary operations for Grad-CAM generation. During the initialization phase, the model layers are separated into two segments: convolutional feature layers and fully connected classification layers. A backward hook function, `save_gradient`, is defined within the class to store the gradients of the target convolutional layer during backpropagation, which are crucial for calculating the class activation map.

5.2. Forward Pass Through the Model:

The input image is first passed through the convolutional layers of the model to extract feature maps. Once the feature maps are extracted, the forward pass continues through the remaining layers of the model to compute the final output predictions.

5.3. Backpropagation for Target Class:

To focus on a specific class, a one-hot encoded vector is created for the desired target class. Backpropagation is then performed using this vector to compute the gradients of the model's output with respect to the selected class. These gradients are stored by the backward hook for later use.

5.4. Generate the Activation Map:

The stored gradients are processed using Global Average Pooling across their spatial dimensions to calculate the weights for the class-specific activation map. These weights are applied to the feature maps by performing a weighted sum, resulting in a class-specific activation map that highlights the most relevant regions for the target class.

5.5. Post-process the Activation Map:

The raw activation map is resized to match the dimensions of the input image using bilinear interpolation. To ensure the activation map only reflects positive contributions, a ReLU operation is applied to remove any negative values. Finally, the map is normalized to the range $[0, 1]$ to enhance visualization clarity.

5.6. Overlay the Activation Map:

The processed Grad-CAM is combined with the original image by applying a colormap such as `jet`, producing a visually interpretable overlay. This highlights the areas of the image that the model deemed most influential for its decision-making, providing valuable insights into the neural network's behavior.

By following these steps, Grad-CAM enables an interpretable analysis of the decision-making process of the model.

6. Results

6.1. Grad-CAM Visualization

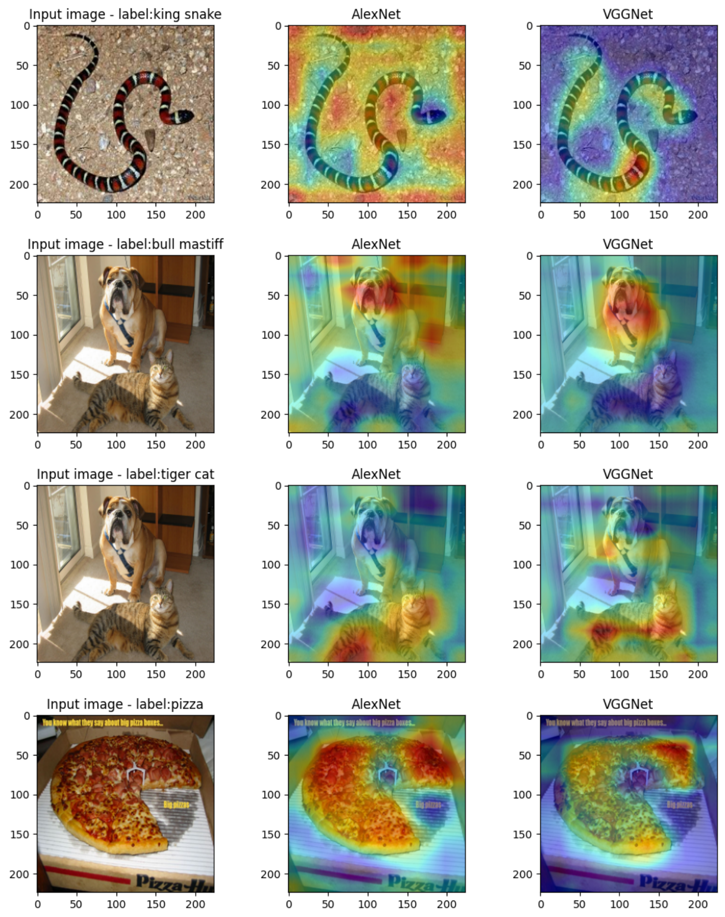
Input image - label:king snake
AlexNet
VGGNet
Input image - label:bull mastiff
AlexNet
VGGNet
Input image - label:tiger cat
AlexNet
VGGNet
Input image - label:pizza
AlexNet
VGGNet
0
50
100
150
200

Figure 2: Grad-CAM visualizations for four test images. Each visualization highlights regions of the input images that significantly contributed to the model's predictions. Warmer colors indicate higher relevance to the prediction.

The Grad-CAM visualizations in Figure 2 provide insights into the decision-making process of the deep learning model for four test images. In each case, the activation maps overlay the input images, with regions of higher relevance highlighted using warm colors such as red and yellow. These regions correspond to the parts of the image that the model found most significant for its predictions.

Notably, in scenarios where two distinct objects corresponding to different classes (e.g., "bull mastiff" and "tiger cat") are present in the same image, Grad-CAM successfully discriminates between the two. The activation maps clearly focus on the relevant features of each object associated with their respective classes, demonstrating the model's ability to localize and differentiate between multiple objects within a complex scene. This capability highlights the utility of Grad-CAM in providing interpretable explanations for multi-class predictions. Additionally, in cases of partial occlusion or complex backgrounds, the Grad-CAM reveals the areas that influenced the model's confidence, even if they extend beyond the target object. This demonstrates the reliability of Grad-CAM in interpreting and validating the model's behavior, particularly in identifying which visual features were most critical in the classification task.

7. Conclusion

This report provided a comprehensive overview of Grad-CAM, a powerful interpretability technique for convolutional neural networks. By highlighting the regions of an input image that contribute most to a model's predictions, Grad-CAM enables users to visualize and understand the decision-making process of deep learning models. The implementation steps, from defining the Grad-CAM class to generating and visualizing activation maps, were detailed to ensure reproducibility and practical applicability. Grad-CAM's ability to produce clear, class-specific visualizations makes it a valuable tool for enhancing model transparency.

Experimental results showed Grad-CAM's effectiveness in various scenarios, including its ability to discriminate between multiple objects in images with distinct classes, such as "bull mastiff" and "tiger cat." These visualizations validated the technique's robustness in complex contexts.

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[2] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. *International Journal of Computer Vision*, 128(2):336–359, October 2019. ISSN 1573-1405. doi: 10.1007/s11263-019-01228-7. URL http://dx.doi.org/10.1007/s11263-019-01228-7.

[3] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.