# CAM and Grad-CAM final report

Taehun Kim[1]
[1] Department of Computer Science and Engineering, Pusan National University.

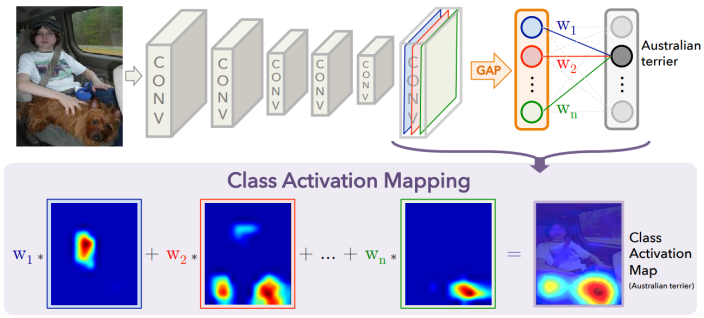


Figure 2. Class Activation Mapping: the predicted class score is mapped back to the previous convolutional layer to generate the class activation maps (CAMs). The CAM highlights the class-specific discriminative regions.

Figure 1: Class Activation Mapping: the predicted class score is mapped back to the previous convolutional layer to generate the class.

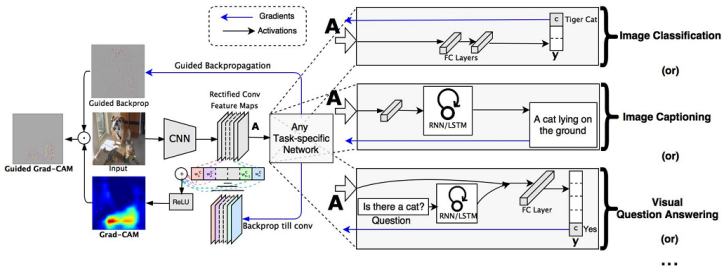


Figure 2: Grad-CAM overview

## 1 Introduction

Class Activation Mapping(CAM)[4] modifies image classification CNN architectures replacing fully connected layers with global average pooling and a $1 \times 1$ convolutional layer for visual explanation.

In contrast, Gradient-weighted Class Activation Mapping(Grad-CAM)[2] is the technique for producing visual explanations, without modifying the CNN architecture or re-training the model.

In this report, We implemented the CAM[4] and GradCAM[2] architecture, and test and analyze the result.

## 2 CAM

We use CIFAR-10 dataset for training, and we use the pretrained AlexNet[1] and pretrained VGG-13[3]. We removed the layers after last maxpooling layer, and we add $3 \times 3$ convolutional layers with 10 output channels, which we call *cam-conv*.

After that, we add a global average pooling layer to each CNN, and we add a fully-connected layer with 10 output. Global average pooling refers to applying average pooling to the last convolutional layer, reducing its shape from $H \times W \times C$ to $1 \times 1 \times C$:

$$F_k = \sum_{x,y} f_k(x,y)$$

After training, we compute a weighted sum of those feature maps to produce CAM:

$$M_c(x,y) = \sum_{x,y} w_k^c f_k(x,y)$$

, where $M_c$ is class activation map for class c, and $w_k^c$ is the fully connected layer's weight parameters corresponding to specific output class c for unit k.

| | fixed feature extractor | fine-tuning |
|---|---|---|
| Test accuracy | 79.92% | 90.02% |

Table 1: The test accuracy of alexnet-cam model

### 2.1 Training

As mentioned earlier, we use the CIFAR-10 dataset. The images are resized to $32 \times 32$ to $224 \times 224$ because $32 \times 32$ resolution of CIFAR-10 is too small for multiple max-pooling layers to be effectively applied. Additionally, we applied random horizontal flipping to the training images with a 50% probability for data augmentation. Finally, all images were normalized with a mean of 0.5 and a standard deviation of 0.5.

We trained AlexNet using two different approaches: as a fixed feature extractor and with fine-tuning. In the first method, the convolutional layers are initialized with pretrained ImageNet weights to extract features, while the final classifier is trained for the new dataset. In the second method, not only is the classifier replaced or retrained, but the weights of the pretrained layers are also fine-tuned.

The learning rate for the first method is set to 0.001, while for the second method, it is 0.0001. Both methods were trained for 150 epochs. The results are shown in Table 1.

### 2.2 Analyze and Discuss

AlexNet with fine-tuning performs better than when used as a fixed feature extractor. This is because the pretrained weights are initialized with ImageNet weights, while CIFAR-10 images have a lower resolution compared to ImageNet. As a result, the fixed feature extractor may struggle to extract features effectively from CIFAR-10 images.

Class Activation Maps of several CIFAR-10 images using a weighted sum of those feature maps are shown in Figure 3

## 3 Grad-CAM

The last convolutional layer in a CNN has the best compromise between high-level semantics and spatial information. Grad-CAM uses the gradient information flowing into the last convolutional layer of the CNN. This method identifies the regions that contributed to the CNN's decision-making process.

Let activation maps of last convolutional layer is $A \in \mathbb{R}^{H \times W \times K}$, which $H \times W$ is resolution of the activation map and $K$ is the number of the activation maps. Let $y^c$denote the class score for class c before the softmax activation. First, The paper computes gradient of class score $y^c$ with respect to the $k$th map, $i$th column, $j$th row of $A$. Second, we apply global average pooling to the gradient to get the gradient of class score $y^c$ with respect $k$th $A$, $\alpha_k^c$:

$$\alpha_k^c = \frac{1}{HW} \sum_{i=1}^{W} \sum_{j=1}^{H} \frac{\partial y^c}{\partial A_{ij}^k}$$

, where the $A_{ij}^k$ is the $k$th map, $i$th column, $j$th row of activation maps of the last convolutional layer. Since $A$ is the last convolutional layer, the gradient calculation involves successive matrix multiplications of the weight matrices and the gradient with respect to activation from class score to last convolutional layer through Backpropagation(chain rule). $\alpha_k^c$ captures the importance of $k$th feature map for a target class c.

Additionally, we calculate localization map for class c using $\alpha_k^c, A^k$, and ReLU:

$$L_{Grad-CAM}^c = ReLU(\sum_k \alpha_k^c A^k)$$

The product of $\alpha_k^c$ and $A^k$ highlights the significant regions within the activation map $A^k$. ReLU is applied to retain only features with positive contributions, as negative values likely correspond to other classes.

$$\text{ReLU}(x) = \begin{cases} 0 & \text{if } x \leq 0, \\ x & \text{if } x > 0. \end{cases}$$

### 3.1 Implement and Result

We use AlexNet[1] and VGG-16[3] with initalized with pretrained weight. To preserve the last convolutional layers'gradient($\frac{\partial y^c}{\partial A_{ij}^k}$) during backpropagation, we use register-hook function:

**features = self.featureLayers(x)**

**features.registerHook(self.saveGradient)**

, where featureLayers extract the feature from an image, and saveGradient function store the gradient to self.gradient and return the gradient.

To generate Grad-CAM, we compute the gradient with respect to the specified class probability ($y^c$). The gradient is stored in self.gradient and processed using global average pooling to obtain the averaged weights, $\alpha_k^c$.

These weights are then used to compute a weighted sum of the activation maps($A^k$) to produce the class activation map. Finally, weighted sum of the activation maps is resized to the size of the original image for visualization. The visualization of Grad-CAM is shown in 4.

## 4 Conclusion

We implemented CAM[4] and Grad-CAM[2]. Through CAM and Grad-CAM, we were able to identify which parts of the image the CNN model focuses on(see Figure 3 and Figure 4). However, Grad-CAM is better because it does not require modifying the CNN model architecture or additional training.

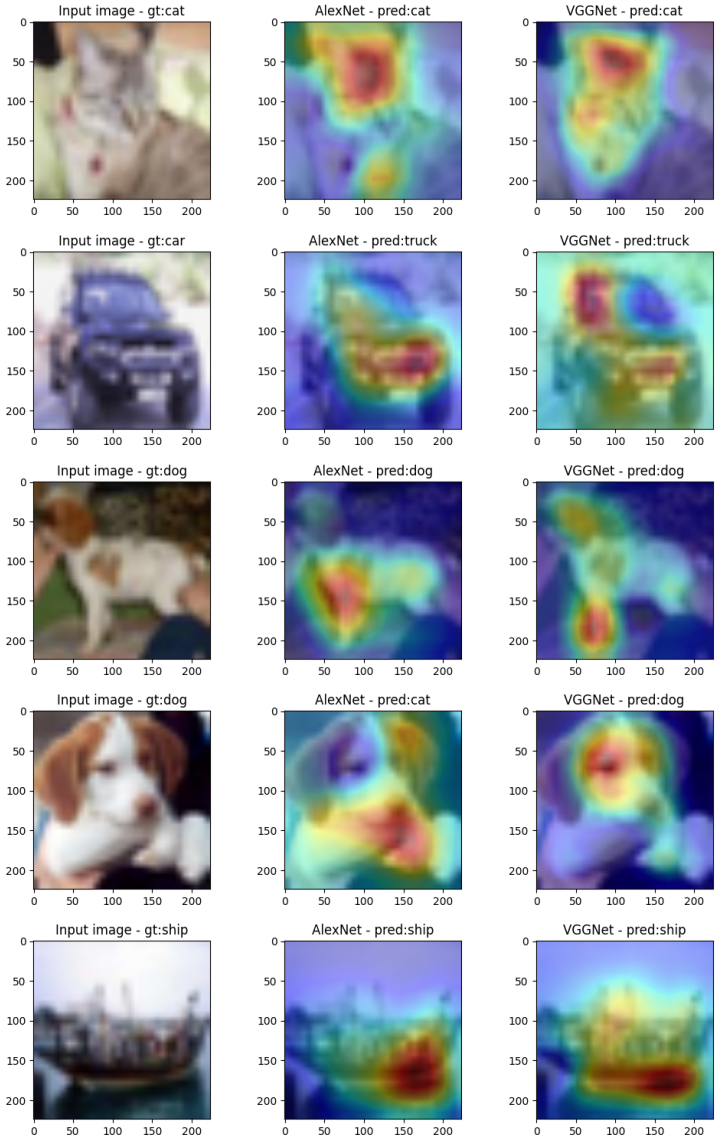


Figure 3: Class Activation Map result of several CIFAR-10 images. The red areas indicate the parts of the image that the model paid more attention to.


[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. URL https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[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, 2015. URL https://arxiv.org/abs/1409.1556.

[4] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization, 2015. URL https://arxiv.org/abs/1512.04150.

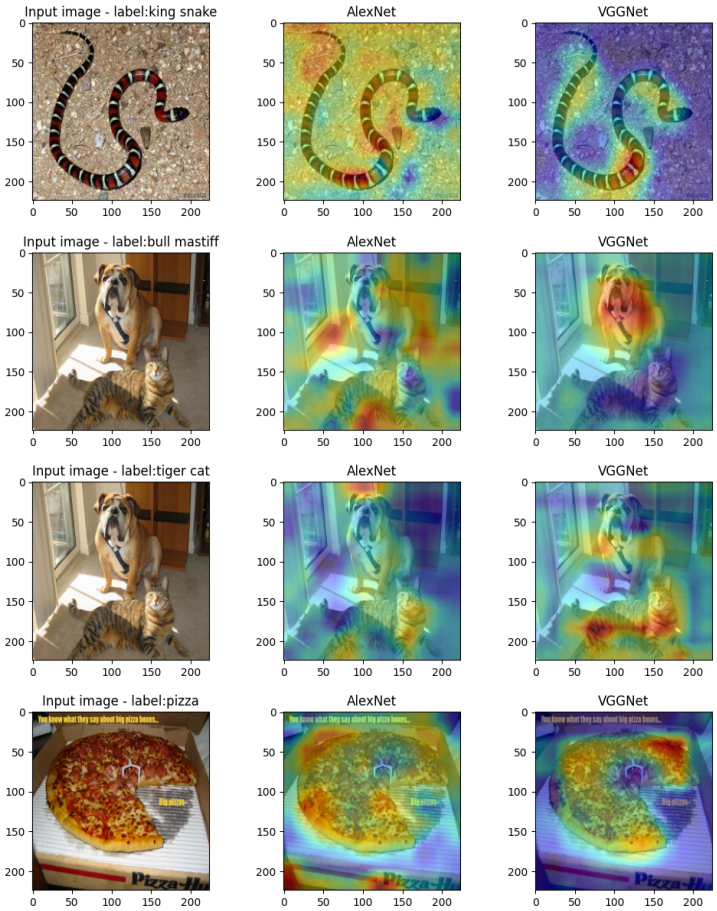


Figure 4: Grad-CAM results of several images. The red areas indicate the parts of the image that the model paid more attention to.