華中科技大學
计算机科学与技术学院
School of Computer Science &Technology，HUST

COMPUTER VISION

Experimental Report 4

CS2110 U2021XXXXX Gao Lang
Huazhong University of Science and Technology

1 Introduction

Interpretability of neural networks has always been a subject of great importance to researchers. Neural networks are widely used in various aspects of social production due to their powerful feature extraction capabilities and excellent fitting effects. However, neural networks are highly complex, which makes it impossible for us to understand them using the same methods as simple models such as linear regression models. Up until now, neural networks remain a black box model for humans. Nevertheless, scientists have proposed some methods to try to "understand" the behavior of neural networks to some extent, which are known as interpretability techniques. The significance of interpretability techniques for deep learning lies in making the model more easily understood, while enhancing the transparency of the model, allowing humans to have more control over the model, and making the model more suitable for the tasks it is intended to perform.

2 Framework

In this experiment, we use Grad-CAM and Layer-CAM to conduct interpretability analysis on AlexNet (a neural network structure) respectively.

2.1 Grad-CAM

Grad-CAM, which stands for Gradient-weighted Class Activation Mapping, is a visualization technique used in deep neural networks. Its primary goal is to determine the part of the network that contributes the most to the prediction results, thereby enabling the localization of specific regions in an image. This technique helps increase transparency and interpretability of the decision-making process of neural networks.The computational flow of Grad-CAM can be represented by Figure 1.

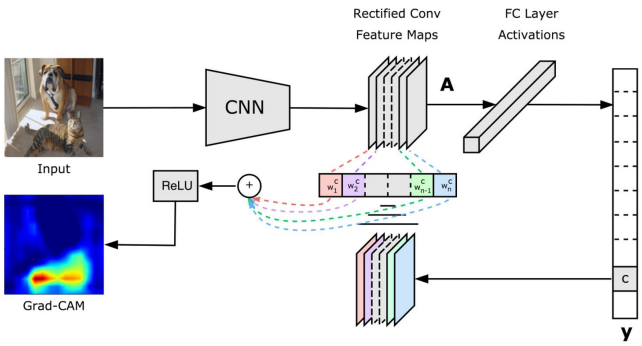

Figure 1:Architecture of Grad-CAM

When applying Grad-CAM to a convolutional layer, the scores of the corresponding class are first backpropagated to obtain the gradient map of that convolutional layer. Then, global average pooling is performed on the gradient map to obtain a weight vector. The corresponding channels of the feature map obtained by forward propagation through that layer and the weight vector are multiplied, and then the corresponding elements of these channels are added and passed through a ReLU activation function layer to obtain the activation mapping. At this point, the activation mapping can numerically reflect which region of the image the model assigns greater importance to. If this activation mapping is upsampled and displayed as a heatmap, it can more intuitively reflect the importance of different regions of the image to the model.

2.1 Layer-CAM

Layer-CAM is an interpretability technique for neural network layers, and its basic flow is very similar to Grad-CAM. However, Layer-CAM has a finer granularity and can more accurately reflect the weights assigned by the model to different regions of the image. Specifically, Layer-CAM also retains the feature map obtained during forward propagation through the convolutional layer and the gradient map obtained by performing backward propagation on the scores of a certain class. The difference is that after passing the gradient map through an activation function layer, Layer-CAM directly multiplies it element-wise with the feature map, sums the results channel-wise, and then activates them again to obtain the activation mapping. It is not difficult to find that Layer-CAM assigns independent weights to all nodes in the feature map, rather than assigning overall weights to channels as in Grad-CAM. Therefore, the activation mapping obtained by Layer-CAM can better reflect the analysis process of the model.

3 Experiments

3.1 Experimental setup

Datasets In this experiment, we conduct interpretability analysis of a deep learning model using image classification task as an example. We use three images of cat and dog classification as the dataset. These three images contain a cat,a dog, and both cat and dog respectively. In subsequent experiments, we will display the weights assigned by the model to different regions of the image in the form of heatmaps superimposed on the image.

Models In this experiment, we will conduct interpretability analysis on AlexNet. As shown in Figure 2,AlexNet is a classic neural network architecture consisting of three modules: features, avgpool, and classifier. The features module contains several convolutional layers and pooling layers, which are used for feature extraction. Avgpool is a global average pooling layer that aggregates the features extracted by the model. The classifier consists of several linear layers that further aggregate the features and output the classification results. The main object of our interpretability analysis is the features module.

Figure 2:Architecture of AlexNet

Implementation details All experiments in this report were completed on a single CPU.

3.2 Experimental Results

The purpose of this experiment is to implement the Layer-CAM and Grad-CAM algorithms and apply them to AlexNet. There is no comparative experiment, so this section only introduces relevant experimental results images without conducting a comparative analysis.

3.2.1 Grad-CAM Performance

Figure 3 shows the Grad-CAM analysis results on three images for different categories.

Figure 3: The model analysis results are presented using Grad-CAM in the form of heatmaps. The three images on the top show the heatmaps for the "cat" category, while the three images on the bottom show the heatmaps for the "dog" category.

During the model inference, the three pairs of images from left to right were classified as: cat, dog, and dog. By conducting Grad-CAM analysis on these three images, we found that in the first image, the category "cat" assigned higher weights to the areas where cats appeared in the image as a whole, and particularly highlighted one of the cat's heads. In the "dog" category, most areas of the image were assigned lower weights, indicating that the model did not find suitable regions to judge as dogs. In the second image, the score for the "dog" category was significantly higher than that for the "cat" category in the model's output. By conducting Grad-CAM analysis on both categories separately, we found that the weight distributions were almost completely opposite. Interestingly, in the "cat" category, the model assigned relatively higher weights to all parts of the image except for the head of the

dog, indicating that the model was able to distinguish between dogs and cats. In the third image, the model classified it as a "dog", and it was also evident from the analysis results that the model focused on the head of the dog. However, in the "cat" category, the model hardly assigned higher weights to certain regions in the image.

3.2.2 Layer-CAM performance

Figure 3 shows the Grad-CAM analysis results on three images for different categories.

Figure 3: The model analysis results are presented using Layer-CAM in the form of heatmaps. The three images on the top show the heatmaps for the "cat" category, while the three images on the bottom show the heatmaps for the "dog" category.

From Figure 3, we can observe that the heatmap generated by Layer-CAM is more detailed and can reflect the model's judgment in smaller regions, which validates the superiority of the Layer-CAM computing mechanism. Through analysis, we can conclude that when predicting the category "cat", the model does pay attention to the head and limbs of the cat and assigns them higher weights. When predicting the category "dog", the model is able to successfully predict the category as "dog" through the dog's head.

3.2.3 Feature map of the last convolution layer

In order to more clearly demonstrate the basis for generating Grad-CAM and Layer-CAM, we also extracted the output of the last convolutional layer in the model's feature module. This is a matrix with 256 channels, each channel being a 13*13 sub-matrix. We concentrated these feature matrices and displayed them in the form of heatmaps.The feature maps are shown in Figure 4.

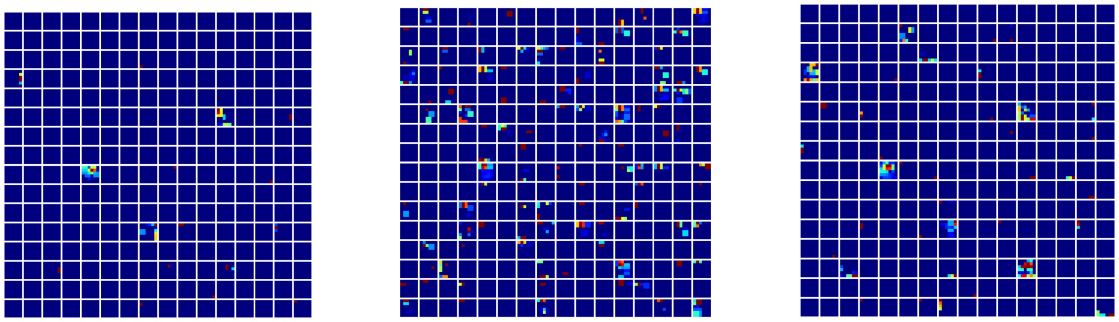

Figure 4:The feature maps output by the last convolutional layer during forward propagation of the three images. From left to right are the feature maps for the cat image, the dog image, and the image containing both cat and dog.

In the feature maps, dark blue regions represent values close to 0, while dark blue regions represent values of 0 or very close to 0, while brighter blocks indicate areas with larger values. The more brightly colored blocks there are, the more activated neurons there are. By comparing Figure 4, we can easily observe that the first image (i.e., the feature map for the cat image) has relatively fewer activated neurons, followed by the third image (i.e., the feature map for the image containing both cat and dog), with the second image (i.e., the feature map for the dog image) having the most activated neurons.

4 Discussion

What do the experimental results above indicate? Through a comprehensive evaluation of the feature maps, Grad-CAM analysis results, and Layer-CAM analysis results, it is evident that the model is less sensitive to cat features than to dog features. In the results of both interpretability techniques, we can see that when the model attempts to classify an image as a cat, it tends to assign higher weights to most areas of the image, but it always struggles to accurately locate the cat's face. The three images on the top of Figure 3 show the weight distribution of the model for the cat category, while the three images on the bottom show the weight distribution for the dog category. If an image cannot identify a dog, the model will assign low weights to most areas since these areas did not activate a significant number of neurons. On the other hand, if an image cannot identify a cat, the model will assign high weights to most areas, indicating that the model has not fully learned cat features. This is particularly evident in the middle pair of images in Figure 3, where the model assigned lower weights to the head of the dog than to other areas when trying to classify the image as a cat. Other areas that do not contain any cat features were assigned relatively higher weights, suggesting that the model may have classified the image as a cat by determining that "the image is not a dog." Therefore, the performance of this model suggests that during training, the model did not fully consider cat features. This could be due to insufficient training or issues such as concept shift or confounding variables in the dataset leading to imbalanced class assignments and overfitting.

5 Conclusion

In this experiment, we used Grad-CAM and Layer-CAM as interpretability analysis techniques to analyze a pre-trained AlexNet model. We obtained the feature maps output by the model's feature extraction module and analyzed different samples using both techniques. The results were presented in the form of heatmaps, showing the importance assigned by the model to different regions of the images. The level of importance reflected which parts of the image the model paid more attention to during classification. Through careful analysis of the experimental data, we found that the model had relatively mature and clear criteria for determining whether an image was a dog,

but lacked criteria for determining whether an image was a cat. This could be due to factors such as unbalanced training data distribution and insufficient model training.

Appendix A

Implementation of the Grad-CAM using *Pytorch* structure.

```
feature_map=model.features(sample_img)

output=model.avgpool(feature_map)

output = output.view(output.size(0), -1)

output=model.classifier(output)

cat_score=output[0][0]

dog_score=output[0][1]

model.zero_grad()

dog_score.backward()

global_avg_pool = nn.AdaptiveAvgPool2d((1, 1))

weight = global_avg_pool(layer_gradients[0])

print(weight.shape)

mult = feature_map * weight

mult=torch.sub(mult,torch.min(mult))

summed = torch.sum(mult, dim=1, keepdim=True)

activated = F.relu(summed)

n = normed_imgs[0][0].shape[2:]

upsampled = F.interpolate(activated, size=n, mode='bilinear', align_corners=False)
```

Appendix B

Implementation of the Layer-CAM using Pytorch structure.

```
layer_gradients=[]

def get_output_gradients(module, grad_input, grad_output):

    layer_gradients.append(grad_input[0])

model.features.register_backward_hook(get_output_gradients)

forward=[]

forward.append(sample_img)
```

```
for module in model.features:
    forward.append(module(forward[-1]))
output=model(sample_img)
cat_score=output[0][0]
dog_score=output[0][1]
model.zero_grad()
cat_score.backward()
feature_map=forward[-3]
weight=layer_gradients[0]
weight=F.relu(weight)
mult = torch.mul(feature_map,weight)
summed = torch.sum(mult, dim=1, keepdim=True)
activated = F.relu(summed)
n = normed_imgs[0][0].shape[2:]
upsampled = F.upsample(activated, size=n, mode='bilinear', align_corners=False)
```