图标

描述已自动生成

**计算机视觉**

**上机实验三报告**

|  |  |  |  |
| --- | --- | --- | --- |
| **专业:** | **：** | **计算机电脑** |  |
| **班级:** | **：** | **CS2002** |  |
| **学号:** | **：** | **I201920029** |  |
| **姓名:** | **：** | **冯就康** |  |
| **电话:** | **：** | **15623031879** |  |
| **邮箱:** | **：** | **sokhorng526@gmail.com** |  |

Contents

[I. Introduction 3](#_Toc155712375)

[1. Task Introduction 3](#_Toc155712376)

[2. Grad-CAM Introduction 3](#_Toc155712377)

[3. LayerCAM Introduction 4](#_Toc155712378)

[II. Task Design 5](#_Toc155712379)

[1. Model Loading 5](#_Toc155712380)

[2. Grad-CAM and LayerCAM Analysis 7](#_Toc155712381)

[III. Experiment Result and Analysis 8](#_Toc155712382)

[1. Experiment Result and Analysis 8](#_Toc155712383)

[IV. Experiment Summary 10](#_Toc155712384)

[1. Experiment Summary 10](#_Toc155712385)

[2. Experiment Conclusion 10](#_Toc155712386)

# Introduction

## Task Introduction

**Task** **requirements**: For the trained convolutional neural network, given an input image, generate the interpretability analysis results of the image for a specific category.

The experiment will provide two different versions of two-classification models based on **PyTorch** and **TensorFlow**, which can be used for the classification of cats and dogs.

**Note**: The network architecture used by **PyTorch** is **AlexNet**, and **TensorFlow** uses **VGG16**. The two are slightly different. Please choose any model for the experiment.

The experiment will provide three input images at the same time. For each image, we will try to do the interpretability analysis using **Grad**-**CAM** and **LayerCAM** for all the input images.

## Grad-CAM Introduction

**Grad**-**CAM**, short for Gradient-weighted Class Activation Mapping, is a technique used in deep learning, particularly with Convolutional Neural Networks (CNNs), to understand which regions of an input image are important for the network’s prediction of a particular class.

It uses the gradients of any target concept, flowing into the final convolutional layer to produce a coarse localization map highlighting important regions in the image for predicting the concept.

Grad-CAM is applicable to a wide variety of CNN model families:

* CNNs with fully connected layers (e.g., VGG)
* CNNs used for structured outputs (e.g., captioning)
* CNNs used in tasks with multimodal inputs or reinforcement learning

The use of Grad-CAM in computer vision is quite extensive. It provides ‘visual explanations’ for decisions from a large class of CNN-based models, making them more transparent and explainable. It helps in understanding why a deep learning network makes its classification decisions.

In the context of image classification models, Grad-CAM visualizations lend insights into their failure modes, are robust to adversarial images, outperform previous methods on localization, are more faithful to the underlying model, and help achieve generalization by identifying dataset bias. For tasks like image captioning and visual question answering (VQA), Grad-CAM shows that even non-attention-based models can localize inputs. It can also be applied to non-classification examples such as regression or semantic segmentation.

In summary, Grad-CAM is a powerful tool for interpreting and understanding the decisions made by convolutional neural networks in computer vision tasks.

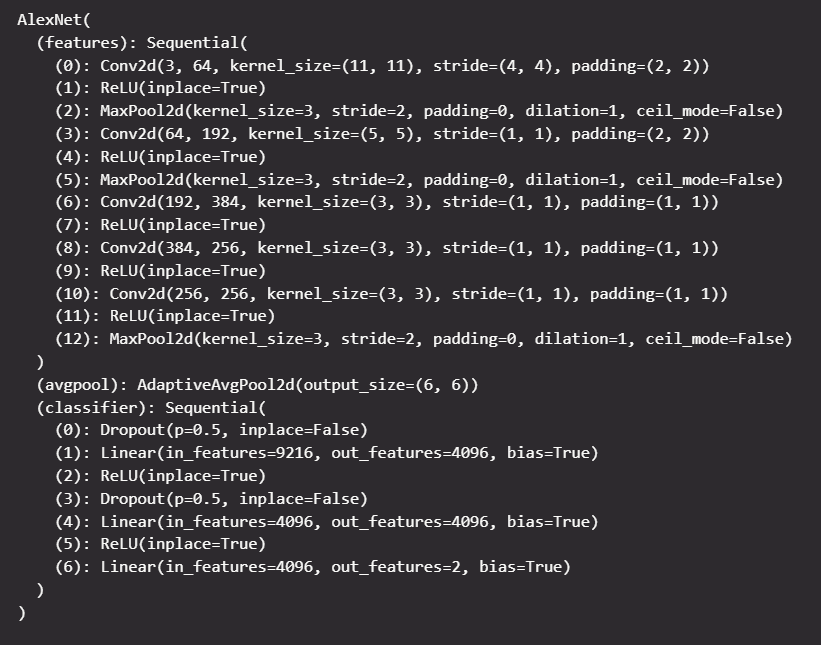
## LayerCAM Introduction

**LayerCAM**, short for Layer-wise Class Activation Mapping, is a technique used in deep learning to generate reliable class activation maps from different layers of a Convolutional Neural Network (CNN). Unlike traditional Class Activation Mapping (CAM) methods that only use the final convolutional layer of a CNN, **LayerCAM** can produce class activation maps for different layers of a CNN. This allows **LayerCAM** to capture both coarse and fine-grained object localization information.

In the field of computer vision, **LayerCAM** is used to highlight discriminative object regions for the class of interest. These highlighted regions can be used for various weakly-supervised tasks. For example, LayerCAM can be applied to weakly-supervised object localization and semantic segmentation. By generating more fine-grained object localization information from the class activation maps, LayerCAM can locate the target objects more accurately. This makes LayerCAM a valuable tool for understanding and interpreting the decisions made by convolutional neural networks in computer vision tasks

# Task Design

## Model Loading

We can load the model by using torch.load(). The AlexNet Model is like below:

Let’s breakdown of the architecture:

**Features** **Layers**: This is a sequential container of the main convolutional layers of the model.

* **Conv2d(3, 64, kernel\_size=(11, 11), stride=(4, 4), padding=(2, 2))**: The first convolutional layer with 64 filters of size 11x11, stride of 4, and padding of 2. The input has 3 channels (RGB image).
* **ReLU(inplace=True):** The activation function used is ReLU (Rectified Linear Unit).
* **MaxPool2d(kernel\_size=3, stride=2, padding=0, dilation=1, ceil\_mode=False):** Max pooling is performed with a filter of size 3x3 and stride of 2.
* The above pattern of **Conv2d** -> **ReLU** -> **MaxPool2d** is repeated with varying numbers of filters and filter sizes.
* The final layer in the features block is another **MaxPool2d** layer.
* **avgpool** **(AdaptiveAvgPool2d(output\_size=(6, 6))):** This layer applies adaptive average pooling to the output of the previous layer, resulting in a fixed output size of 6x6 regardless of the input size.

**Classifier** **Layers**: This is a sequential container of the fully connected layers of the model.

* **Dropout(p=0.5, inplace=False):** Dropout is a regularization technique that randomly sets a fraction p of the input units to 0 at each update during training time. Here, p=0.5, meaning roughly half of the inputs will be dropped out during training.
* **Linear(in\_features=9216, out\_features=4096, bias=True):** A fully connected layer that takes 9216 input features and outputs 4096 features.
* **ReLU(inplace=True):** The activation function used is ReLU.
* The pattern of **Dropout** -> **Linear** -> **ReLU** is repeated.
* **Linear(in\_features=4096, out\_features=2, bias=True):** The final fully connected layer that takes 4096 input features and outputs 2 features, corresponding to the 2 output classes.

This architecture is quite standard for a convolutional neural network, with alternating convolutional and pooling layers, followed by fully connected layers. The use of ReLU activation functions and dropout for regularization are also common in such networks. The final output is a tensor with 2 elements, each representing the score for a class (cat or dog in this case). The class with the higher score is the model’s prediction.

**Note**: In **Grad**-**CAM** and **LayerCAM**, the target layer is typically the last convolutional layer. This is because convolutional layers have feature maps that capture the spatial information in the image, which is what we want to visualize with Grad-CAM and LayerCAM. These layers capture high-level features in the image. In the **AlexNet** model, the last convolutional layer is (10): **Conv2d(256, 256, kernel\_size=(3, 3), stride=(1, 1), padding=(1, 1))**. So, we will use this layer as the target layer for both **Grad**-**CAM** and **LayerCAM**.

## Grad-CAM and LayerCAM Analysis

We input the images (cat.jpg, dog.jpg, both.jpg) into our pre-trained model AlexNet. However, by putting the images into the model, it is just a black box. We don’t know any things about the model decision, or what are the features that make model to predict the output. We can use the Grad-CAM and LayerCAM to see why the model behaves like that.

Here is a step-by-step description of the code:

* **Define the Target Layer**: we start by defining the target layer of the model that we want to visualize. In our case, it’s the 10th layer of the model’s features (model.features[10]).
* **Construct the Cam Extractor Object**: Grad-CAM and LayerCAM objects are constructed using model, target layer as parameters. These objects will be used to generate the CAMs (Class Activation Maps).
* **Processing the Input Image**: we do some preprocessing techniques such as normalize, resize, etc. on the input\_image, so that we can feed the input\_image to the model.
* **Get the Output**: We feed the processing input images into the model to get the output.
* **Generate the CAM**: The Grad-CAM and LayerCAM are generated by passing the output into the cam extractor.
* **Visualize the CAM**: We use the matplotlib.pyplot library to show and visualize the final output.

**Note**: We use the class activation map from **torchcam 0.4.0**

**pip install torchcam**

# Experiment Result and Analysis

## Experiment Result and Analysis

|  |  |  |
| --- | --- | --- |
|  | Grad-CAM | LayerCAM |
| Cat Image | The Grad-CAM visualization shows a heatmap on the one of the cat’s head, indicating that the model focused on the head of cat and made a decision or output based on the feature. | The LayerCAM visualization also shows a heatmap on the two cat’s head, also suggesting that the model is considering the overall structure and features of the cat’s head to make its classification prediction. |
| Dog Image | The Grad-CAM visualization shows a heatmap on one of dog’s face. This suggests that these areas were particularly influential in the model’s decision-making process. The model might be recognizing certain features or patterns in these areas that are characteristic of a dog. | The LayerCAM visualization, on the other hand, also covers one of the dog’s faces. Also indicates that the model is considering the overall structure and features of the dog’s face to make its classification prediction. |
| Both Image | The Grad-CAM visualization shows the heatmap on the dog, suggesting that no specific region of the image significantly influenced the model’s decision which show that the model is focusing on the features around the dog to make prediction. | The LayerCAM visualization showed some heatmap dots on the cat’s body instead. This suggests that the model is focusing on these some parts of the cats instead of the dog to make its classification. |

# Experiment Summary

## 1. Experiment Summary

Summary of the Grad-CAM and LayerCam based on the three input images:

* **Image ‘cat.jpg’**: Both the **Grad-CAM** visualization and **LayerCam** visualization showed a heatmap of the cat face. The visualizations shown that the model is focusing right features of a cat, which is the cat tabby that can only be shown on the cat body.
* **Image ‘dog.jpg’**: Both the Grad-CAM and LayerCam show the heatmap on the dog face. This also indicates that the model will make the decision or predictions based on the areas or features which are also resembling a dog.
* **Image ‘both.jpg’**: However, for the picture that contain both dog and cat. The GradCam suggests that the model makes the prediction by focusing on the dog’s body. Meanwhile, the Layer Cam in contrast suggests that the model makes the prediction by focusing on some parts of the cay features instead.

In summary, Grad-CAM and LayerCAM are both techniques used to visualize the areas in an image that a Convolutional Neural Network (CNN) focuses on when making a prediction. However, they use different methods to generate these visualizations, which can result in different heat maps.

## 2. Experiment Conclusion

Grad-CAM uses the gradients of any target concept, flowing into the final convolutional layer to produce a coarse localization map highlighting the important regions in the image for predicting the concept. It uses the gradient information flowing into the last convolutional layer of CNN to understand which features had the most influence on a decision.

LayerCAM is a more recent technique that improves upon Grad-CAM by providing more precise localization maps. Instead of using the gradients flowing into the last convolutional layer, LayerCAM considers all the layers in the network. This can result in more detailed and accurate heat maps, especially for complex models and tasks.

The difference in the heatmaps generated by Grad-CAM and LayerCAM is due to these different methodologies. Depending on the model architecture and the specific task, one method may provide more useful or accurate visualizations than the other.

In our experiment, Grad-CAM is focusing on the dog in the image, while LayerCAM is focusing on the cat. This could be due to a variety of factors, including the specific architecture of the AlexNet model, the weights that the model has learned during training, and the specific implementation of Grad-CAM and LayerCAM that we’re using.

It’s also worth noting that these techniques provide a way to interpret the model’s decisions, but they don’t necessarily provide a perfect representation of how the model is making its decisions. They should be used as tools for understanding and debugging, rather than as definitive explanations of the model’s behavior.