

Homework 4: Use ChatGPT to get help with your Final Project

Hyunsung Oh (UT EID: ho3626)

1. A statement of your goal; what you want the model to accomplish or what you want it to help you do.

My goal is to implement and analyze feature map visualization techniques, such as Grad-CAM or activation maps, on a ResNet-18 model trained on CIFAR-10. I want GPT to help me understand how to extract and visualize intermediate feature representations, compare outputs across different layers and classes, and interpret what these visualizations reveal about the network's behavior.

2. Copy/pasted prompts and model responses. Everything you sent the model and all the responses you got back.

ME: My goal is to implement and analyze feature map visualization techniques, such as Grad-CAM or activation maps, on a ResNet-18 model trained on CIFAR-10. I want GPT to help me understand how to extract and visualize intermediate feature representations, compare outputs across different layers and classes, and interpret what these visualizations reveal about the network's behavior.
I want you to help with this work.

GPT: Great — that's a solid and meaningful project goal. Here's how we can break this down step by step, and I'll guide you through **each part** with explanations, code examples, and suggestions on what to explore or ask next.

Overview of the Plan

1. **Model & Dataset Setup**
 - Load pretrained ResNet-18 on CIFAR-10.
 - Prepare a sample image and get predictions.
2. **Activation Map Extraction**
 - Use *forward hooks* to visualize intermediate layer outputs.
3. **Grad-CAM Implementation**
 - Apply Grad-CAM to the model and visualize class-discriminative regions.
4. **Layer-wise & Class-wise Comparison**
 - Visualize multiple layers for the same image.
 - Visualize one layer for different classes.

5. Interpretation & Analysis

- Discuss what the visualizations mean.

ME: Can you explain how Grad-CAM works, starting from the concept?

GPT: Sure! Grad-CAM (Gradient-weighted Class Activation Mapping) is a technique used to visualize **which parts of an image a CNN focuses on when making a prediction**. It's especially helpful for interpreting image classification results.

How Grad-CAM Works:

1. You pass an image through the model and identify the target class (e.g., "this is a cat").
2. You capture the **feature maps from the final convolutional layer** of the model.
 - These layers retain spatial information and are suitable for visualization.
3. You compute the **gradient of the output class score** with respect to the feature maps.
 - This tells you how important each feature channel was to the prediction.
4. You **average the gradients** across the spatial dimensions to get a weight for each channel.
5. You **multiply each channel in the feature map by its corresponding weight** and sum them up.
6. The resulting heatmap is **overlaid on the original image** to show where the model "looked."

This highlights the **regions most responsible** for a given classification.

ME: Can you also explain the forward hook method?

GPT: Absolutely!

What is a Forward Hook?

In PyTorch, a **forward hook lets you intercept and store the output of any layer** during the model's forward pass. It's useful to **see what each intermediate layer is computing**, without modifying the model itself.

How it works:

1. You **register a hook** on a specific layer in the model.
2. When the model processes an input, the hook is **automatically called** and saves the output of that layer.

3. You can then **visualize the saved feature map**.

✓ What can you learn?

- You can see how early layers respond to simple features (like edges), and later layers capture more complex patterns (like faces or objects).
 - You can trace how the representation evolves across the network.
-

🔍 Grad-CAM vs Forward Hook

Feature	Grad-CAM	Forward Hook
Purpose	Highlights what and where the model focuses on for a specific class	Visualizes what features each layer detects
Output	Class-specific heatmap	Generic feature map (not tied to class)
Complexity	More complex (needs gradients)	Simpler to use
Use Case	Model interpretability	Layer inspection , debugging

ME: Can you show example PyTorch code to implement both of these methods?

GPT: Absolutely! Here's how you can implement both Grad-CAM and forward hook visualizations using PyTorch and ResNet-18.

✓ 1. Forward Hook Implementation in PyTorch

python

CopyEdit

```
import torch

import torchvision.models as models

import torchvision.transforms as transforms

import torchvision.datasets as datasets

import matplotlib.pyplot as plt


# Load ResNet18

model = models.resnet18(pretrained=True)

model.eval()


# Dictionary to store feature maps

feature_maps = {}


# Hook function

def get_activation(name):

    def hook(model, input, output):

        feature_maps[name] = output.detach()

    return hook


# Register hook to a layer (e.g., layer1[0])

model.layer1[0].register_forward_hook(get_activation('layer1'))
```

```
# Dummy image input (resize to 224x224)

transform = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ToTensor()
])

dataset = datasets.FakeData(transform=transform)

image, _ = dataset[0]

image = image.unsqueeze(0)


# Run image through model

with torch.no_grad():
    model(image)


# Visualize feature maps

activation = feature_maps['layer1'].squeeze()

fig, axs = plt.subplots(1, 5, figsize=(15, 4))

for i in range(5):
    axs[i].imshow(activation[i].cpu(), cmap='viridis')
    axs[i].set_title(f'Channel {i}')
    axs[i].axis('off')
```

```
plt.tight_layout()
```

```
plt.show()
```

✓ 2. Grad-CAM Implementation (using a library)

bash

CopyEdit

```
pip install grad-cam
```

python

CopyEdit

```
from torchvision.models import resnet18
```

```
from pytorch_grad_cam import GradCAM
```

```
from pytorch_grad_cam.utils.image import show_cam_on_image,  
preprocess_image
```

```
from pytorch_grad_cam.utils.model_targets import  
ClassifierOutputTarget
```

```
import numpy as np
```

```
import matplotlib.pyplot as plt
```

```
# Load model
```

```
model = resnet18(pretrained=True)
```

```
model.eval()
```

```
# Define target layer (usually the last conv layer)

target_layers = [model.layer4[-1]]

cam = GradCAM(model=model, target_layers=target_layers,
use_cuda=False)


# Prepare dummy image

dummy_img = np.random.rand(224, 224, 3).astype(np.float32)

input_tensor = preprocess_image(dummy_img, mean=[0.485, 0.456,
0.406], std=[0.229, 0.224, 0.225])


# Set class index (e.g., 207 = goldfish in ImageNet)

targets = [ClassifierOutputTarget(207)]


# Run Grad-CAM

grayscale_cam = cam(input_tensor=input_tensor,
targets=targets)[0]


# Overlay CAM on image

visualization = show_cam_on_image(dummy_img, grayscale_cam,
use_rgb=True)

plt.imshow(visualization)

plt.title("Grad-CAM Visualization")
```

```
plt.axis('off')
```

```
plt.show()
```

This code shows a class-specific Grad-CAM heatmap over a dummy image. For your real experiment, you can load CIFAR-10 images instead.

3. A brief reflection (<100 words) on how this process went. What did the model do well? Which kinds of prompts worked, and which gave you useless answers?

Using ChatGPT helped me clearly understand the differences between Grad-CAM and forward hook visualization techniques, which I had not fully grasped before. The model was especially helpful in explaining concepts and providing PyTorch code that could be directly applied to my experiments. By asking about different methods and their applications across layers and classes, I was able to gain a better intuition for how CNNs work internally. Specific and well-defined prompts yielded the most helpful responses, while vague questions sometimes resulted in abstract answers. Overall, GPT was an extremely useful partner for this interpretability experiment. That said, I realized I still need to compile and test the generated code myself to catch any potential errors. Even so, the fact that GPT can generate meaningful code from scratch is an incredible capability.