6.869 MiniPlaces Challenge Part I

Ryan Sander

7 November 2019

1 Problem 1: Filter Visualization

A neural network diagram for the state-of-the-art ResNet50 model is given below:

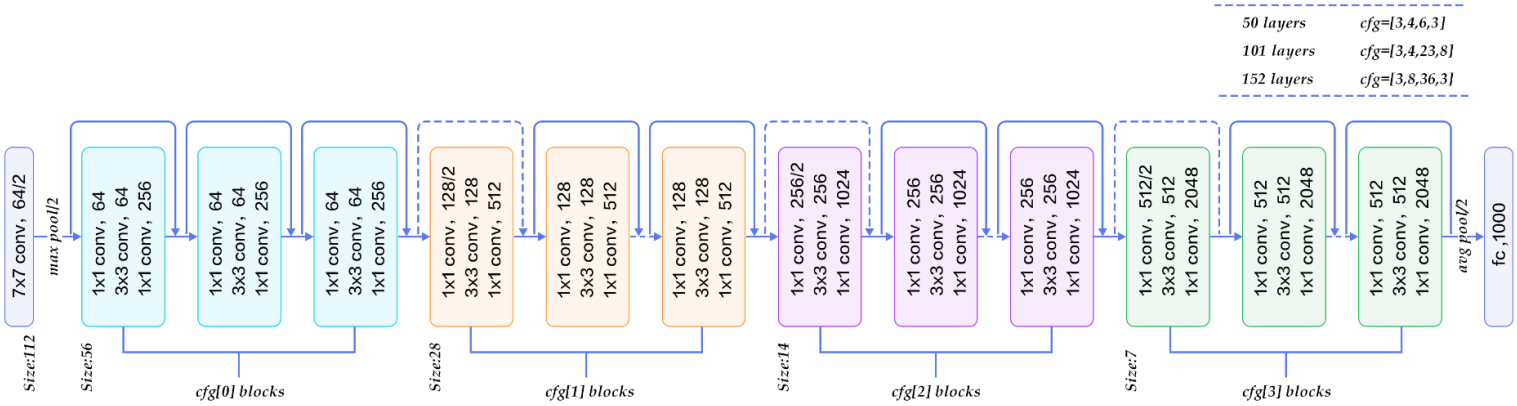

Figure 1: A diagram of our neural network model of choice for this work: ResNet50.

1.1 (a) Normalize Input Kernel

The normalization scheme I choose to use for normalizing the weights between [0,1] is the following: $w^{(i)}_{\text{norm}} = \frac{w^{(i)} - w_{min}}{w_{max} - w_{min}}$.

My code for normalization is given here:

```
def visualize_filters(conv_w, output_size = None):
    wmin, wmax = torch.min(conv_w), torch.max(conv_w)
    print(wmin, wmax)
    w_normalized = (conv_w-wmin)/(wmax-wmin)
    map_t = 255*w_normalized
    map_t = map_t.numpy()
    map_t = map_t.astype(np.uint8)
    if output_size is not None:
        map_t = cv2.resize(map_t,(output_size,output_size))
    return map_t
```

Some of the filters from the first convolutional layer (conv1) are shown below:

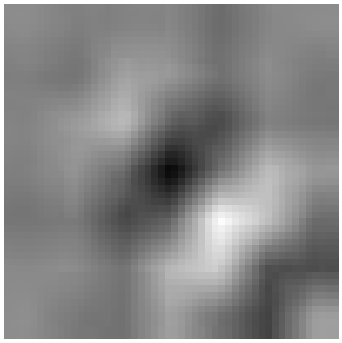

Figure 2: A pretrained filter 2 of the first convolutional layer of this network.

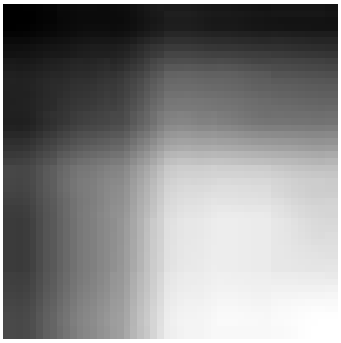

Figure 3: A pretrained filter 11 of the first convolutional layer of this network.

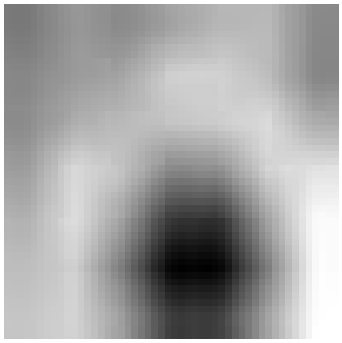

Figure 4: A pretrained filter 11 of the first convolutional layer of this network.

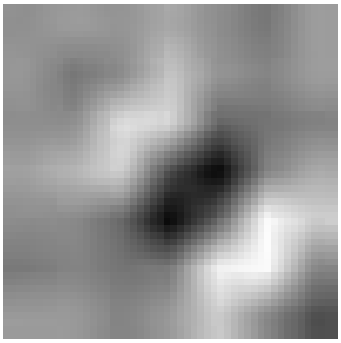

Figure 5: A pretrained filter 2 of the first convolutional layer of this network.

1.2 (b) Visualizing ResNet Filters

We can also visualize the convolutional filters from the second convolutional layer of the ResNet50 network:

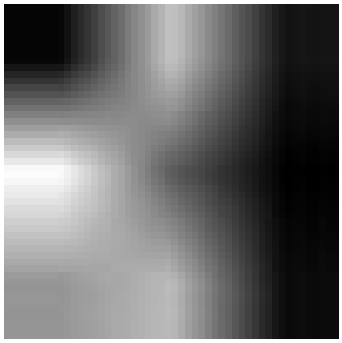

Figure 6: A pretrained filter 14 of the second convolutional layer of this network.

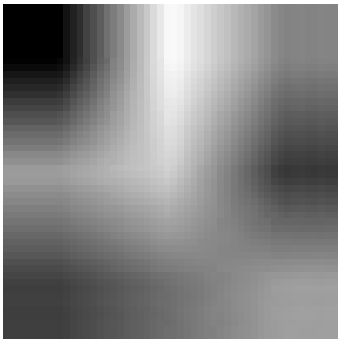

Figure 7: A pretrained filter 23 of the second convolutional layer of this network.

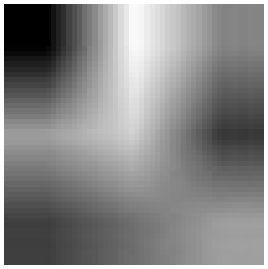

Figure 8: A pretrained filter 25 of the second convolutional layer of this network (conv2).

2 Problem 2: Internal Activation Visualization

2.1 (a) Creating a Chopped ResNet 50 Network

In order to visualize the filter activations for the different filters, we can remove the fully connected layer of the network. This is accomplished through the following line of code in PyTorch:

```
model_cut = nn.Sequential(*list(resnet.children())[:-2]) # Removes last two layers of ResNet50
```

This "ResNet48" model directly exposes the final convolution layers of the network, which enables us to gain valuable insights as to what features of an image activate which unit(s) of the network.

2.2 (b) Visualizing What Activates Different Units

As aforementioned, removing the fully connected layers of this network enables us to gleam valuable insight into the behavior of the network, such as what feature(s) of an image activate which units in the convolution layers. Some of these activation heat maps (each corresponding to a convolutional filter in the final convolution layer) are displayed below (our original unit/filter 300 is shown first

Figure 9: An activation "heat map" for convolution filter 100 in our final convolution layer. This filter is activated primarily by instances of the "mountain/islet" class, or something similar to it.

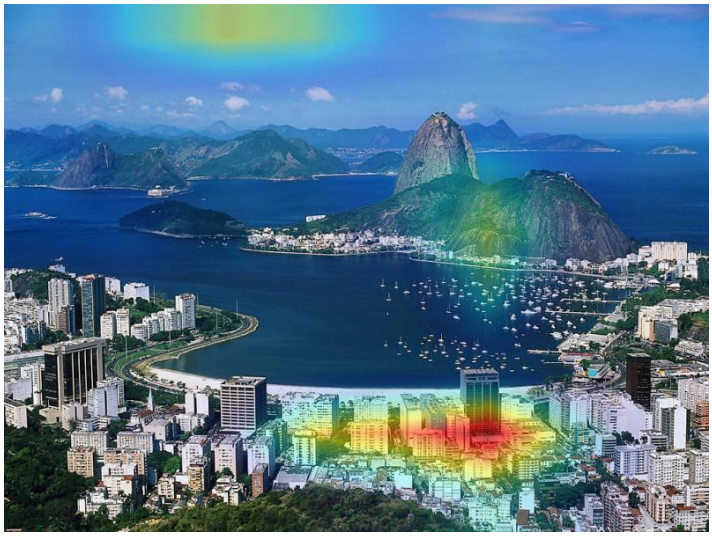

Figure 10: An activation "heat map" for convolution filter 100 in our final convolution layer. This filter is activated primarily by instances of the "village" class, or something similar to it.

Figure 11: An activation "heat map" for convolution filter 77 in our final convolution layer. This filter is activated primarily by instances of the "coast" class, or something similar to it.

Figure 12: An activation "heat map" for convolution filter 175 in our final convolution layer. This filter is activated primarily by instances of the "harbor" class, or something similar to it.

Figure 13: An activation "heat map" for convolution filter 34 in our final convolution layer. This filter is activated primarily by instances of the "coast" or "harbor" classes, or something similar to them.

Figure 14: An activation "heat map" for convolution filter 52 in our final convolution layer. This filter is activated primarily by instances of the "sky" or "coast" classes, or something similar to them.

Figure 15: An activation "heat map" for convolution filter 128 in our final convolution layer. This filter is activated primarily by instances of the "coast" or "harbor" classes, or something similar to them.

2.3 (c) Deactivating the Maximum-Weight Convolution Unit

We also investigate the effect of turning off the maximally-weighted unit for the final convolution layer that serves as input to the fully connected layers. This was accomplished through the following code (note: index refers to the index of the maximally-weighted unit in the final convolution layer):

```
# Set unit values to zero for unit with maximum weights
unit = out1[:,index,:,:] # Get maximum-weighted unit
unit = torch.zeros(unit.shape) # Make array of zeros of unit's size
out1[:,index,:,:] = unit # Set unit kernel to be all zeros
```

The results from this modification are given below:

Top 5 Predictions Before:

1. village (8.458)
2. islet (8.361)
3. beach_house (7.759)
4. coast (6.947)
5. harbor (6.794)

Top 5 Predictions After:

1. islet (8.058)
2. village (7.816)
3. beach_house (7.278)
4. coast (6.650)
5. harbor (6.539)

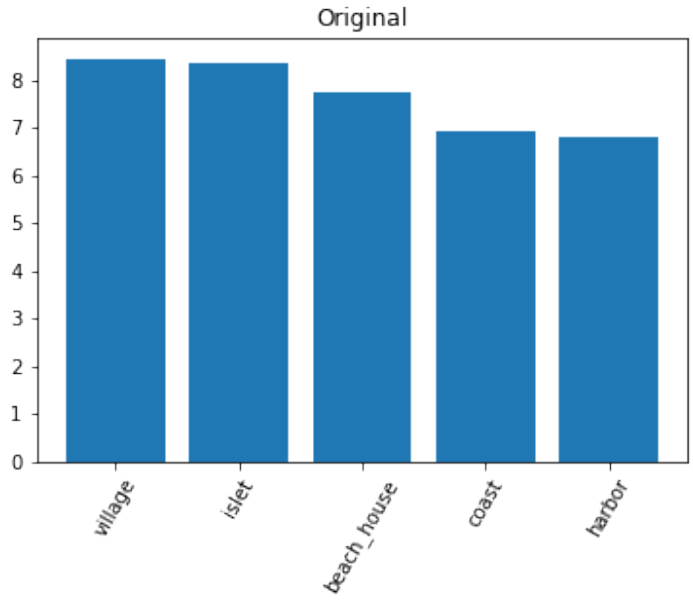

Figure 16: Plot showing the confidence levels (un-normalized) for the top 5 predicted classes prior to the unit modification being made.

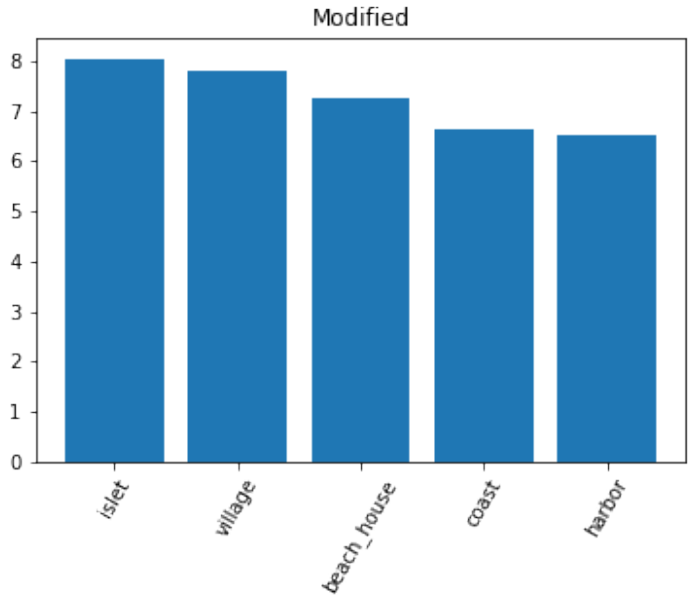

Figure 17: Plot showing the confidence levels (un-normalized) for the top 5 predicted classes after to the unit modification being made.

Finally, we can show the feature map of the unit we de-activate:

Figure 18: Feature map of the convolution unit/kernel that contributes most to the top 1 category of the original outputs. Intuitively, we can see that this unit contributes highly to the "village" class (which explains why the activation for "village" drops after removing this unit), since it has its strongest response centered on a combination of smaller buildings, roads, and water.

3 Problem 3: Class Activation Maps (CAM)

We can use the network chopping technique implemented in problem 2 to gain insights into the behavior of different convolution kernels in the final convolution layer:

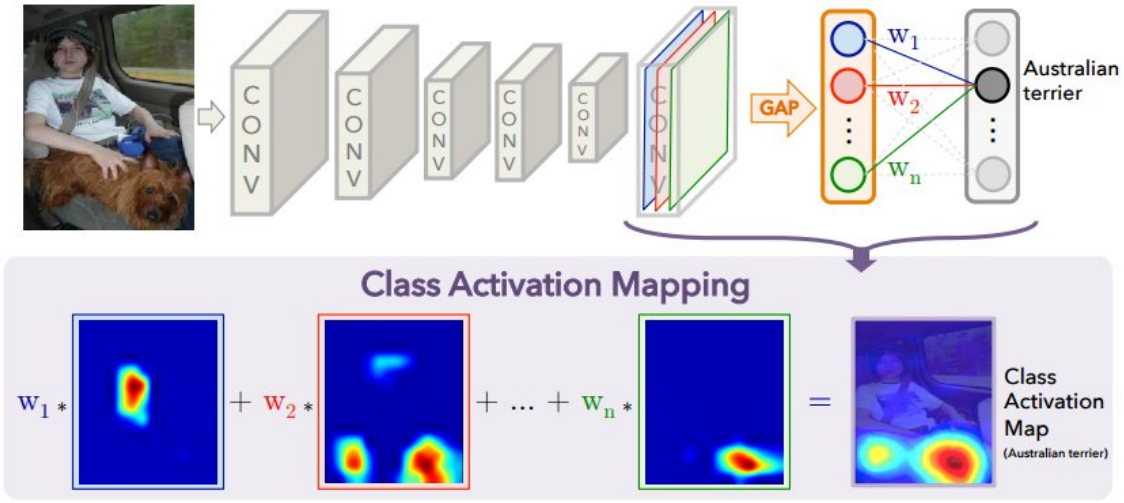

Figure 19: Diagram detailing the procedure of computing a Class Activation Map (CAM).

3.1 (a) Creating a Vectorized Average Output Tensor for Visualizing CAMs

We can reshape our tensor from:

$$(1, c, w, h) \longrightarrow (hw, c) \tag{1}$$

Using the following code in PyTorch:

```
# Get tensor dimensions
w = out.size(3)
h = out.size(2)
b = out.size(0)
c = out.size(1)

# Now reshape tensor and take transpose
out = out.view(c,w*h)
out = torch.t(out)
```

The operation above ensures that we are able to feed this tensor as input into the fully connected layers of the original pre-trained ResNet50 model.

3.2 (b) Using Averaged Vectorized Convolution Tensor as Input to "FC Network"

To use the reshaped tensor above as input to the fully connected layer, we can construct a neural network object using the final pretrained, fully connected layer of the original ResNet50 model. This is achieved with the following PyTorch code:

```
# Create "neural network" consisting only of fully connected layer and activation
FC_layer = nn.Sequential(*list(resnet.children())[-1:])

# Feed output averaged tensor into this "fully connected network"
out_final = FC_layer(out)
```

Note that out_final corresponds to our tensor of class activation maps (one index for each class id).

3.3 (c) Illustrating Connection Between Class Activation Maps and Classes

We are now positioned to illustrate the class activation maps for the top 5 predicted classes for this image. These are given below:

Top 5 predicted classes for input image:

1. village
2. islet
3. beach house

4. coast
5. harbor

Figure 20: Class Activation Map (CAM) for the "village" class. Notice how areas in the image corresponding to buildings yield the highest degree of activation.

Figure 21: Class Activation Map (CAM) for the "islet" class. Notice how areas in the image corresponding to mountains and islands yield the highest degree of activation.

Figure 22: Class Activation Map (CAM) for the "beach_house" class. Notice how areas in the image corresponding to beaches and beach houses yield the highest degree of activation.

Figure 23: Class Activation Map (CAM) for the "coast" class. Notice how areas in the image corresponding to coastline, beaches, and water yield the highest degree of activation.

Figure 24: Class Activation Map (CAM) for the "harbor" class. Notice how areas in the image corresponding to water and boats yield the highest degree of activation.