6.869 Miniplaces 1

Nada Hussein

April 2021

Problem 1

a) To normalize the input kernel on the range 0 to 1 I added the following code:

```
def visualize_filters(conv_w, output_size = None):
    maxW = torch.max(conv_w)
    minW = torch.min(conv_w)
    w_normalized = (conv_w-minW)/(maxW-minW) #TODO: Normalize conv_w values to 0-1 range
    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
```

Figure 1: Input Kernel Normalization

Below are a few example filters from layer 0:

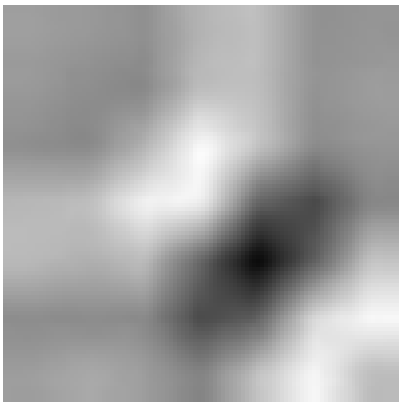

Figure 2: Layer 0 Filter 0

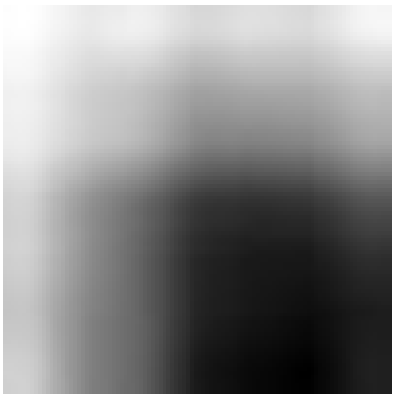

Figure 3: Layer 0 Filter 15

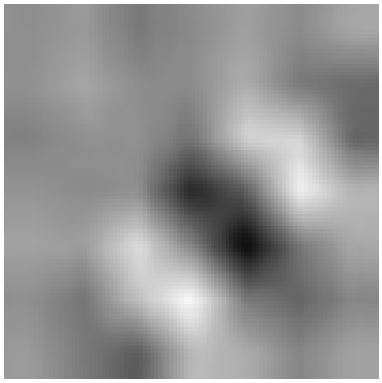

Figure 4: Layer 0 Filter 29

b) I chose to visualize the filters for layer 2 of ResNet. Below are a few example filters:

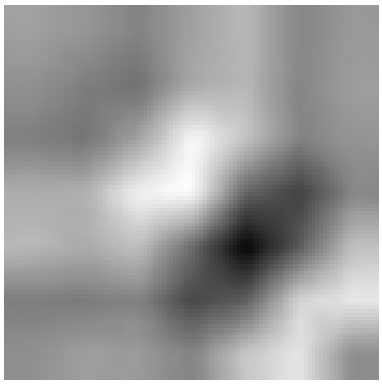

Figure 5: Layer 2 Filter 0

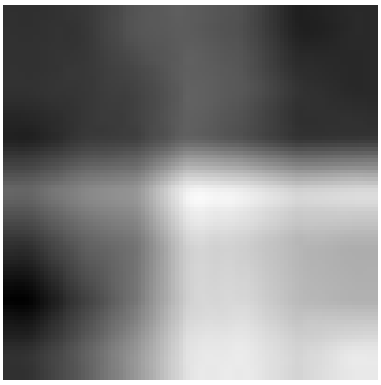

Figure 6: Layer 2 Filter 15

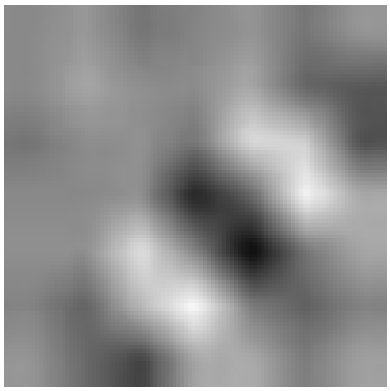

Figure 7: Layer 2 Filter 29

Problem 2

a) Below is the code I wrote to remove the final 2 layers of the model:

```
# TODO: remove the last 2 layers of resnet
# Note: the .children() function of nn.Module (which resn
model_cut = nn.Sequential(*list(resnet.children())[:-2])
```

Figure 8: ResNet Layer Removal

b) After trying different unit numbers, I found the following units detecting the mountain, sky, and buildings, respectively:

Figure 9: Mountain: Unit 300

Figure 10: Mountain: Unit 60

Figure 11: Sky: Unit 410

Figure 12: Sky: Unit 510

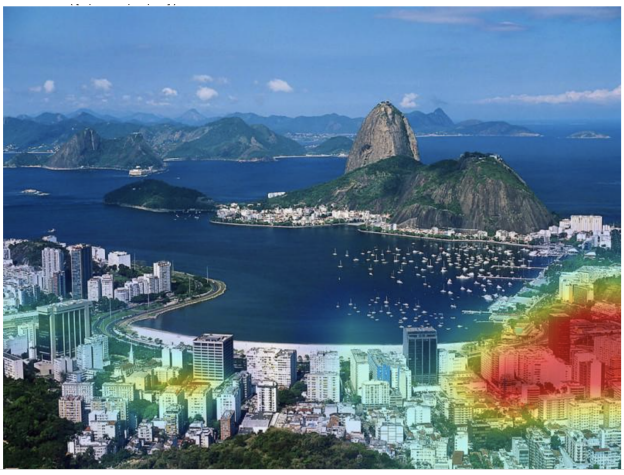

Figure 13: Buildings: Unit 250

Figure 14: Buildings: Unit 70

c) Here you can see the code I used for this section:

```
# Get the output features for this model
features = model_cut(img_tensor.unsqueeze(0))
# Shape is now (1, # units, H, W)
# TODO: deactivate the unit index that has the maximum weights (Set all values for that unit to 0)
features[:,index,:,:] = torch.zeros(features[:,index,:,:].shape)

# TODO: run the modified features through the last two layers of the original network
avg_pool_out = resnet.avgpool(features).squeeze().unsqueeze(0)
out_modified = resnet.fc(avg_pool_out).squeeze()
```

Figure 15: Deactivate Top5 Units

When we deactivate the top-5 units, I ended up with the following predictions before and after: Below you can see the feature map for one of these indices:

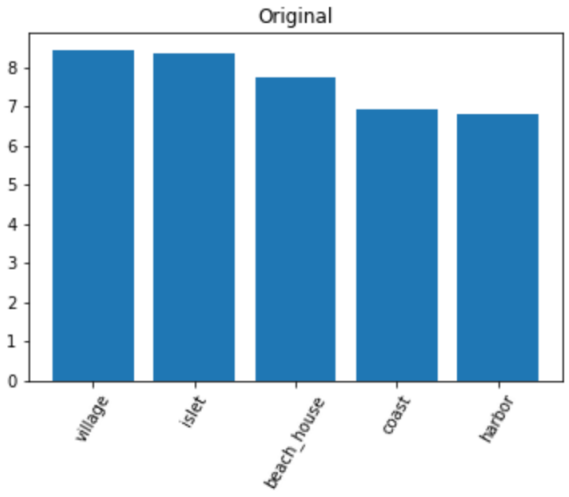

Figure 16: Original: Village

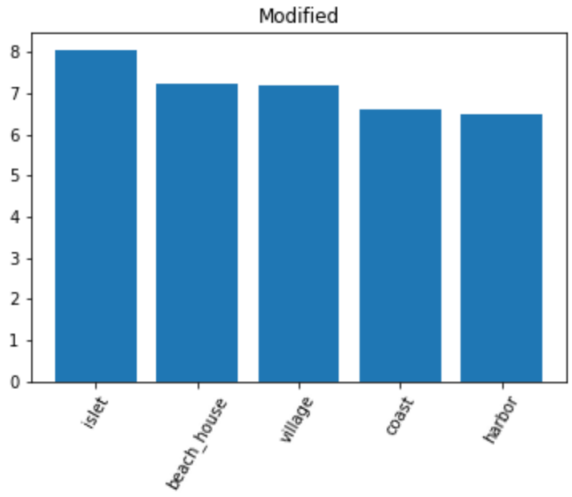

Figure 17: Modified: Islet

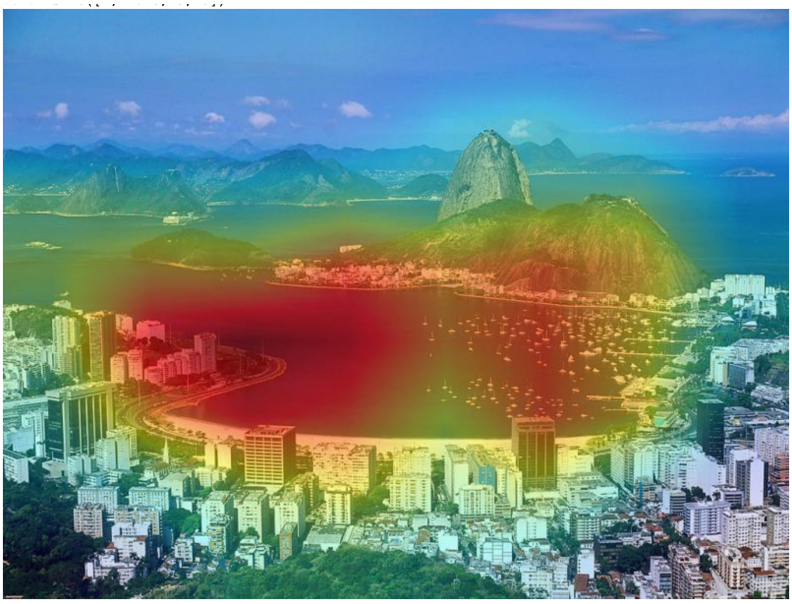

Figure 18: Feature Map for index[0]

d) After playing with the amount of units, I discovered that we only need to set top k to 1 to successfully change the predictions. Below you can see the original vs. modified plot of the category:

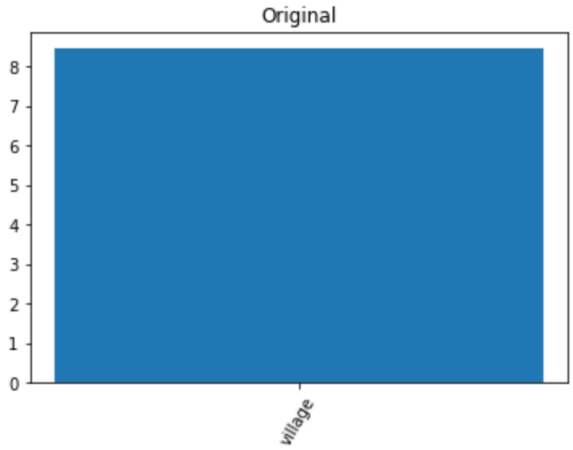

Figure 19: Original: Village

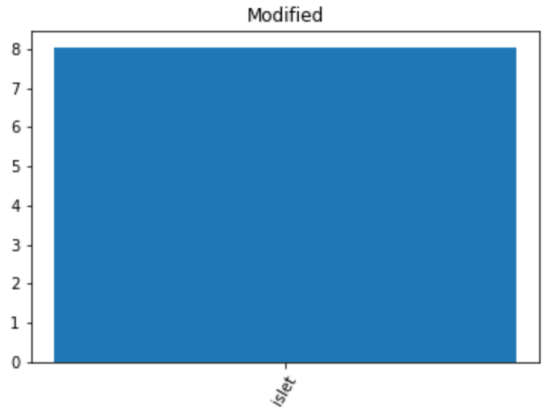

Figure 20: Modified: Islet

Problem 3

a) Below is the code I wrote to reshape the tensor to shape (hw x c):

```
# TODO: convert the shape of the output (out variable) to (h*w) x c
# The .view() function and .transpose() functions will help
activations_reshaped = torch.transpose(activations.view(num_channels, height*width), 0, 1)
```

Figure 21: Reshaping to (hw x c)

b)Below is the code I wrote to run the tensor through the fully connected layer and reshaping the output:

```
# TODO: Run the fully connected layer from resnet to compute the weighted average with activations as the input variable
# out_final should be a h x w x 365 tensor.
out_final = list(resnet.children())[-1](activations_reshaped)

# TODO: obtain the class activation map for the corresponding unit_id
# class_activation_maps should be a 365 x height x width tensor.
class_activation_maps = out_final.transpose(0, 1).view(-1, height, width)
return class_activation_maps[unit_id]
```

Figure 22: Output of fully connected layer

c) Below are all the feature maps for the top 5 predicted categories:

Figure 23: Village

Figure 24: Islet

Figure 25: Beachhouse

Figure 26: Coast

Figure 27: Harbor

Here you can see the village category tends to light up on the buildings and land, the islet lights up on the island in the middle of the image, and beachhouse, coast, and harbor light up at various elements of the water. For example the coast is more on the outsides, the harbor is the further in sections of water, and the beachhouse lights up on both these sections. **d)** For my own image, I used a photo of the coast of Santorini. Below is the original image:

Figure 28: Original Image

And here you can see how the visualizations worked on my image:

Figure 29: Village

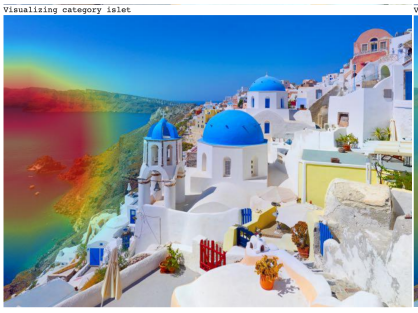

Figure 30: Islet

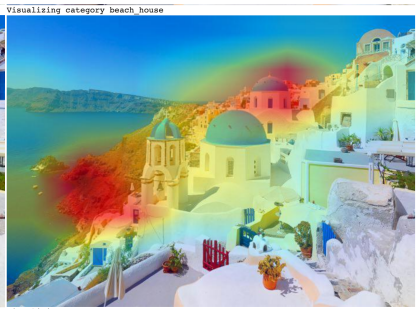

Figure 31: Beachhouse

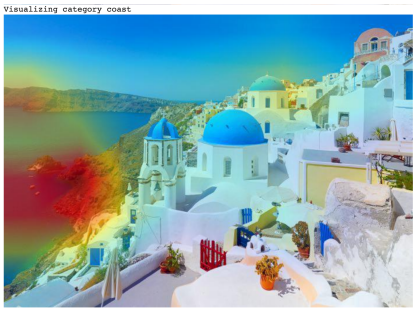

Figure 32: Coast

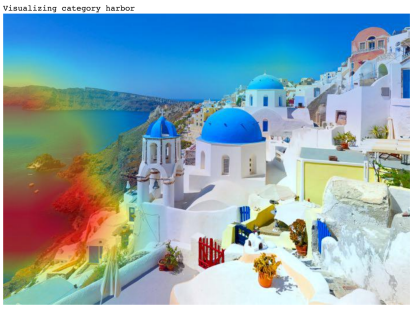

Figure 33: Harbor

Since I chose a scene that was still similar to the original, the heatmaps still look reasonable. The village and beachhouse categories light up on the buildings in the image, while islet focuses on the land in the background, and coast and harbor focus on the water. There is a bit more discrepancy just because the image is not the same structure, but the fact that it contains very similar features helps the visualization still work pretty well.