Homework 5 Writeup

Instructions

- Provide an overview about how your project functions.
- Describe any interesting decisions you made to write your algorithm.
- Show and discuss the results of your algorithm.
- Feel free to include code snippets, images, and equations.
- List any extra credit implementation and result (optional).
- Use as many pages as you need, but err on the short side.
- **Please make this document anonymous.**

Project Overview

In this project, we start out with a basic CNN, and we slowly add standardization, augmentation, regularization, and change up the architecture to increase the accuracy. We also train a classification head appended to the pretrained model to achieve better accuracy.

Implementation Detail

My model consists of two iterations of Conv2D layer, MaxPool2D layer, and Dropout layer with a rate of 0.2. After that, I flatten the result and feed it through two dense layers, the former producing an output of 32 and the latter outputting the number of classes of images.

For my classification head for my pre trained model, I used a Dropout of 0.2, flattened the output, used a 256 neuron dense layer and then a last layer outputting the number of classes of images.

For both models, I used Sparse Categorical Cross Entropy for the loss and the Adam optimizer for the optimizer.

Results (Tasks 1 and 3)

1. My initial validation accuracy for the naive model after 5 epochs was a 16.38 percent, and the train accuracy was a 17.93 percent. When I trained the model fully through all 50 epochs, the validation accuracy was a 39.53 percent (not pictured). (Figure 1)

2. After I added standardization, the validation accuracy went down to a 34.30 percent, most likely because the model was less fitted. I consulted others on this and they did not do anything special in particular to the standardization, so I figured that this was an interesting result. (not pictured)

3. Next, I added augmentation (changes in rotation, translation, zoom, horizontal flip) and changed the optimizer to the Adam optimizer. This improved my validation accuracy by leaps and bounds, bringing it to a 51.26 percent. Training accuracy was a 48.93 percent. (Figure 2)

4. Lastly, I tweaked the architecture of the model. This also seemed to make a very great difference, and I did not even need to tweak any hyperparameters to bring myself over the 65 percent mark (65.09). However, considering that the train accuracy is much higher than the validation, something I could perhaps consider in the future is to add a few more regularization layers.

5. In terms of fine-tuning the pre-trained model, I was able to pass the 85 percent threshold fairly easily by stacking a dropout layer and 2 dense layers. My fine-tuned validation accuracy peaked at 88.44 percent. (Figure 5)

Results (Task 2)

1. Provided below are two images that were misclassified and their LIME explanations.

2. The first image is a bedroom that was mistakenly classified as a coast. The second image is an industrial image that was mistakenly classified as a coast. The bedroom and industrial image seemed to be classified as a coast because the original image was quite low in quality. The black and white quality of the image allowed LIME to draw the wrong segments in the image because there were many areas in the image that had similar pixels due to bad lighting/bad quality. As a result, the model may have seen this and interpreted the shape detected to be that of a coast.

Figure 1: *Top:* Accuracy after 5 epochs. *Bottom:* Naive model accuracy.

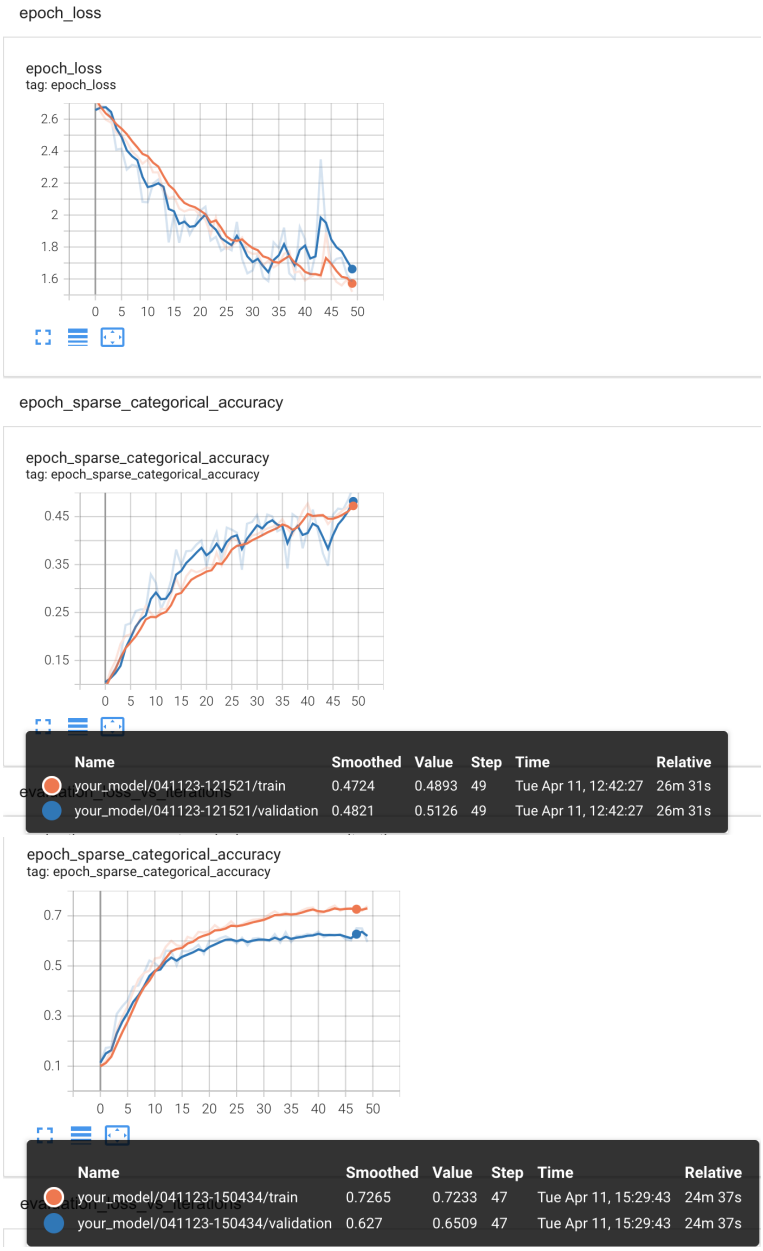

Figure 2: *Top:* Accuracy after augmentation. *Bottom:* Accuracy after architecture change.

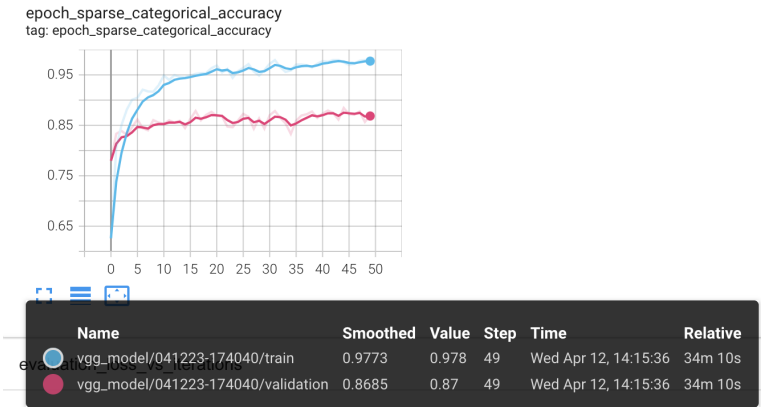

Figure 3: Fine-tuning of pre-trained model

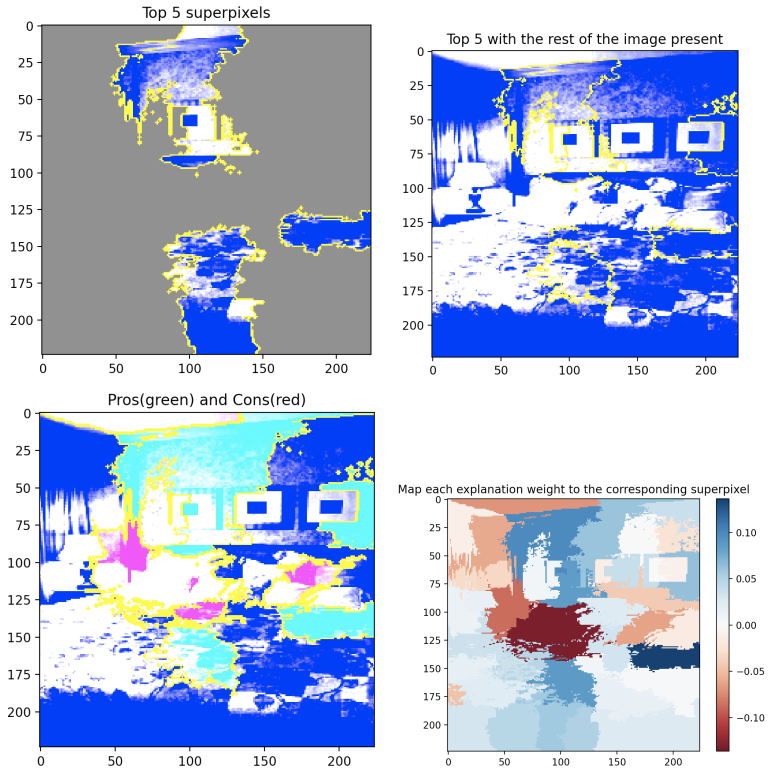

Figure 4: Bedroom mistakenly classified as coast.

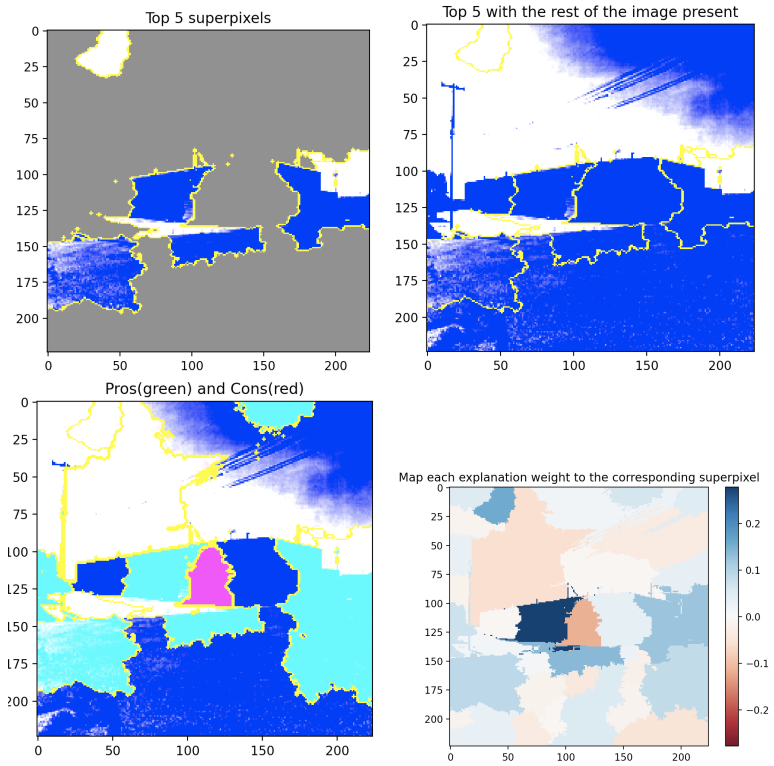

Figure 5: Industrial mistakenly classified as coast.