Project 4 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 I implemented a CNN using Keras.

Implementation Detail

Task 1

For task 1 I first standardized the image. Using numpy functions I found the mean and standard deviation across the images, and then set the image to the (image - mean) / standard deviation. To increase the amount of data through augmentation I used a rotation range of 5 degrees and a horizontal flip. I used RMSprop and SparseCategoricalCrossentropy.

I had the most trouble designing your_model's architecture. After fiddling a little, I was able to get 88% training accuracy but with only a 46% testing accuracy. This signalled that I was overfitting, so I tried adjusting my architecture by adding convolution layers and dropout layers. I made the gap smaller, but my training accuracy only reached 52% with testing accuracy 34%. I have turned in the first architecture, and have commented out the second architecture.

The 1st architecture:

```
self.architecture = [
    Conv2D(32, (3,3), padding="same", activation="relu", name="
                                        block1_conv1"),
    Conv2D(32, (3,3), padding="same", activation="relu", name="
                                        block1_conv2"),
    MaxPooling2D(pool_size=(2,2)),
    Flatten(),
    Dropout(0.2),
    Dense(15),
    tf.keras.layers.Softmax()
]
```

The 2nd architecture:

```
self.architecture = [
    Conv2D(32, 8, padding='same', activation='relu'),
    MaxPooling2D(pool_size=(2,2)),
    Dropout(0.2),
    Conv2D(64, 4, padding='same', activation='relu'),
    Dropout(0.2),
    Flatten(),
    Dropout(0.2),
    Dense(15),
    tf.keras.layers.Softmax()
]
```

Task 2

Lime images:

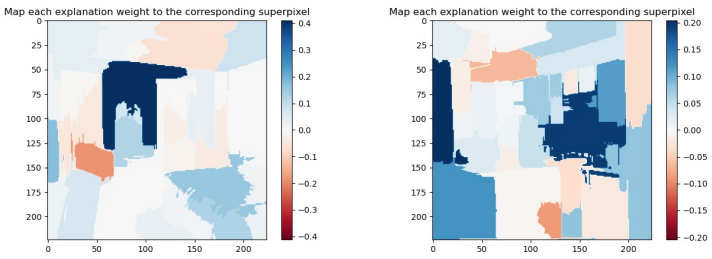

Figure 1: *Left:* Heatmap Bedroom *Right:* Heatmap Kitchen

Task 3

For task 3 I used Adamax and SparseCategoricalCrossentropy, and I made the head:

```
self.head = [
    tf.keras.layers.GlobalAveragePooling2D(),
    Dropout(0.2),
    Dense(15)
]
```

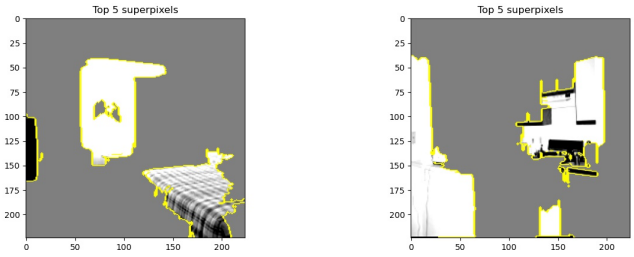

Figure 2: *Left:* Bedroom *Right:* Kitchen

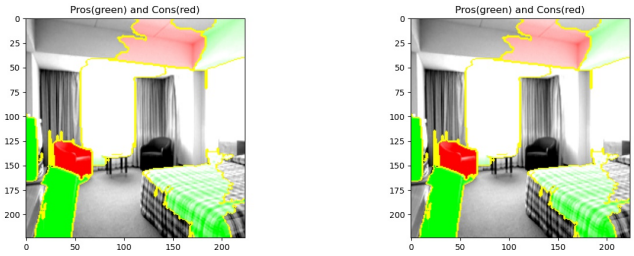

Figure 3: *Left:* Bedroom *Right:* Kitchen

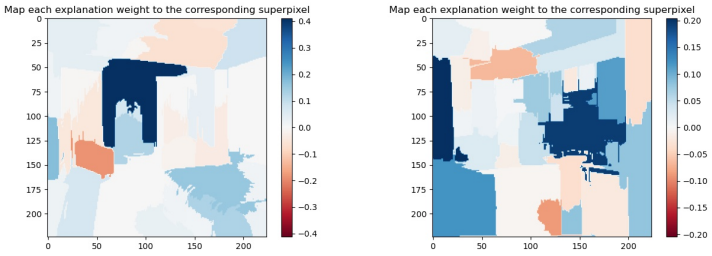

Figure 4: *Left:* Bedroom *Right:* Kitchen

Result

Task 1

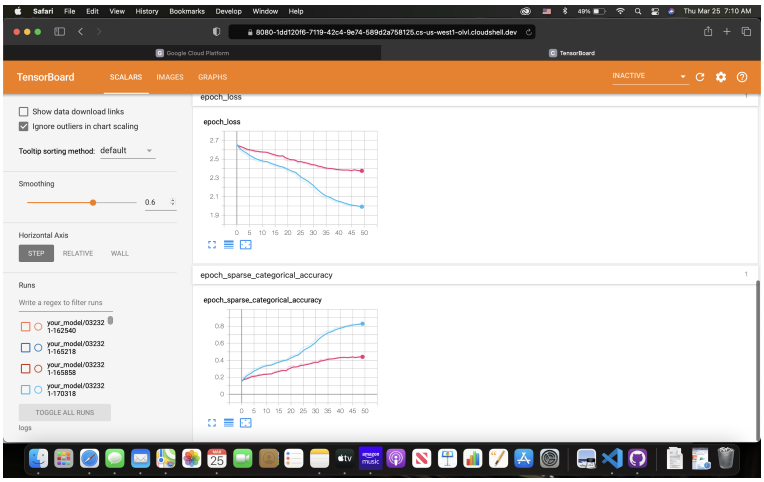

Figure 5: Task 1 epoch accuracy graphs: greatest test accuracy 44%

```
Dataset mean: [0.4596, 0.4596, 0.4596]
Dataset std: [0.2487, 0.2487, 0.2487]
Found 1500 images belonging to 15 classes.
Found 2985 images belonging to 15 classes.
Model: "your_model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
block1_conv1 (Conv2D)        (None, 224, 224, 32)      896
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 224, 224, 32)      9248
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 112, 112, 32)      0
_________________________________________________________________
flatten (Flatten)            (None, 401408)            0
_________________________________________________________________
dropout (Dropout)            (None, 401408)            0
_________________________________________________________________
dense (Dense)                (None, 15)                6021135
_________________________________________________________________
softmax (Softmax)            (None, 15)                0
=================================================================
Total params: 6,031,279
Trainable params: 6,031,279
Non-trainable params: 0
_________________________________________________________________
```

Figure 6: Task 1 Architecture

Task 3

```
Model: "vgg_head"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
global_average_pooling2d (Gl (None, 512)               0
_________________________________________________________________
dropout (Dropout)            (None, 512)               0
_________________________________________________________________
dense (Dense)                (None, 15)                7695
=================================================================
Total params: 7,695
Trainable params: 7,695
Non-trainable params: 0
```

Figure 7: Task 3 head architecture

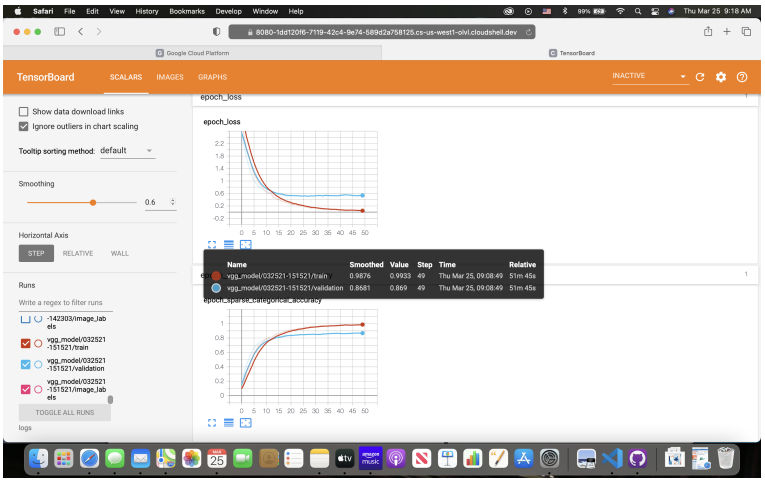

Figure 8: Task 3 epoch accuracy graphs: greatest test accuracy 87%