# 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

This project was to design and train convolutional neural networks (CNNs) for scene recognition using the TensorFlow system.

## Implementation Details

There were three tasks for this project, the first of which was to design a CNN architecture with less than 15 million parameters and train it on a small dataset of 1,500 training examples. I used standardization, data augmentation, and regularization via dropout to compensate for the fact that our dataset is not quite big enough. For example, I used the arguments of ImageDataGenerator() to augment the data as follows:

```
data_gen = tf.keras.preprocessing.image.ImageDataGenerator(zoom_range=
                                    0.1, shear_range=10, rotation_range
                                    =20, horizontal_flip=True,
                                    preprocessing_function=self.
                                    preprocess_fn)
```

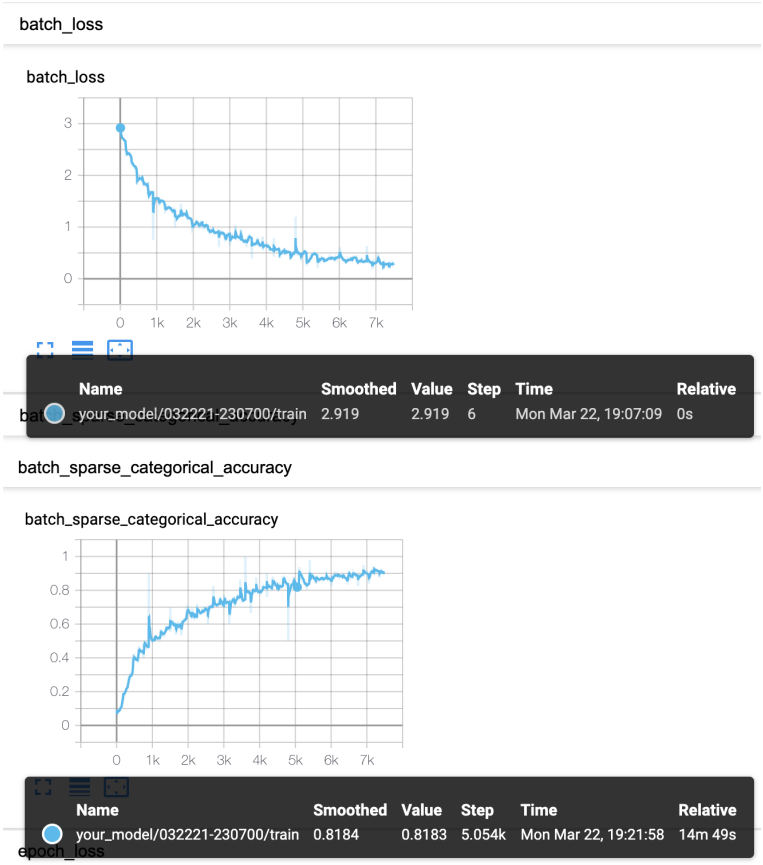
I originally had zoomRange=0.5, and changing this value to 0.1 improved my test accuracy by about 10 percentage points.

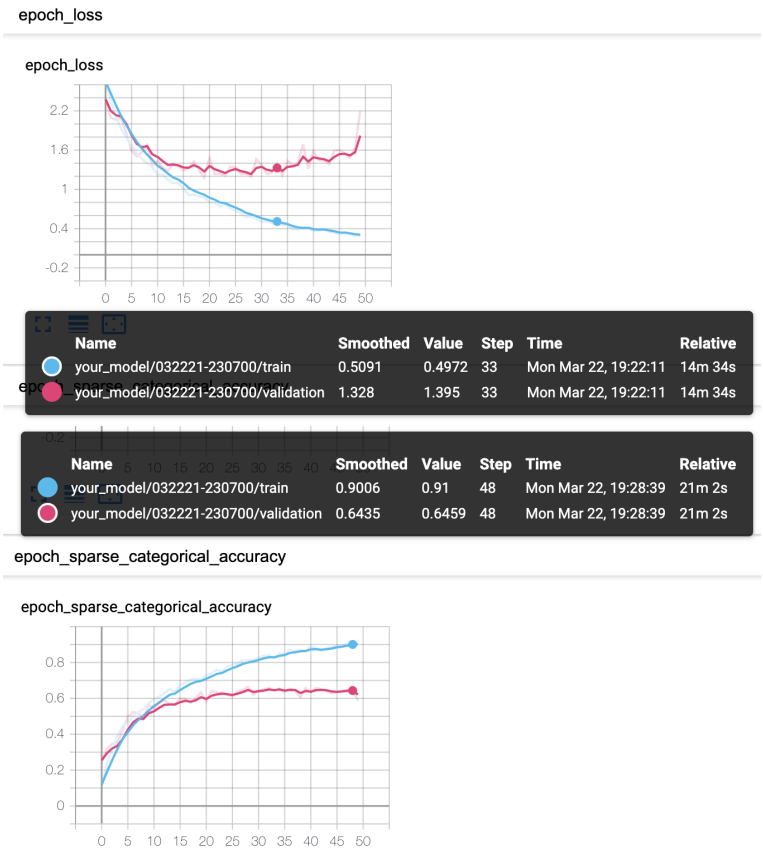
Next I built the following architecture for Task 1, which effectively achieves the desired accuracy while using fewer than 15 million parameters. The two following code snippets display my chosen optimizer and architecture. (I had originally used the SGD optimizer, but achieved significantly better results with the Adam optimizer.)

```
self.optimizer = tf.keras.optimizers.Adam(hp.learning_rate, beta_1=0.9
                                          , beta_2=0.999)
```

```
self.architecture = [
                Conv2D(64, 3, 1, padding="same", activation="relu"),
                MaxPool2D(pool_size=(2, 2)),
                Conv2D(128, 3, 1, padding="same", activation="relu"),
                MaxPool2D(pool_size=(2, 2)),
                Conv2D(256, 3, 1, padding="same", activation="relu"),
                MaxPool2D(pool_size=(4, 4)),
                Flatten(data_format=None),
                Dense(256, activation='relu'),
                Dropout(.2),
                Dense(64, activation='relu'),
                Dropout(.2),
                Dense(15, activation='softmax')]
```

The following graphs were generated by Tensorboard and depict the batch loss, batch sparse categorical accuracy, epoch loss, and epoch sparse categorical accuracy for this model.






Task 3 consisted of completing a VGG model by writing a classification head for our 15-scene classification task. I chose the following optimizer and head architecture for this task:

```
self.optimizer = tf.keras.optimizers.SGD(hp.learning_rate)
```

```
self.head = [
            Flatten(),
            Dense(15, activation='softmax')
]
```

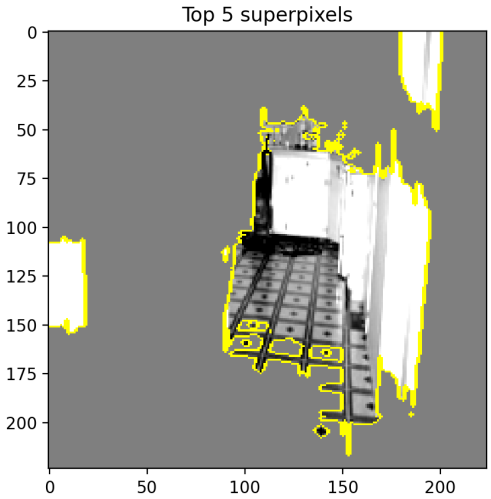
For both of these tasks, I chose to use the following loss function:

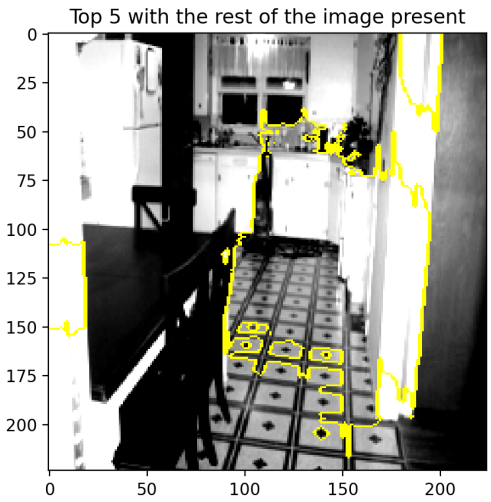
```
loss = tf.keras.losses.sparse_categorical_crossentropy(labels,
                                    predictions, from_logits=False)
```

## Results

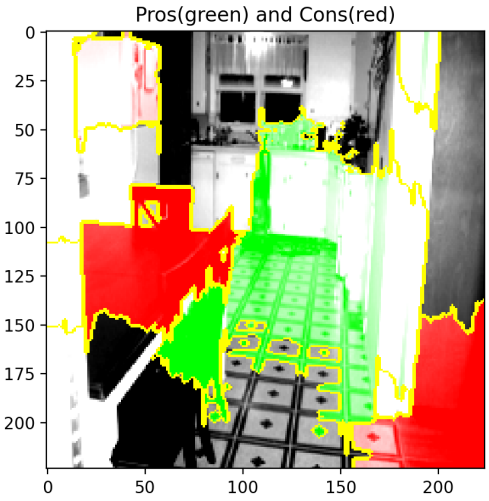
My model achieves 66.50% accuracy on the test set, and my VGG model achieves 86.53% accuracy. The image below from Tensorboard shows several pictures from the dataset that were incorrectly classified. For Task 2 I used the LIME package to to examine two of these images' visual explanations of the model prediction. The LIME explanation figures and my reasoning for two of the falsely-classified images are included below.

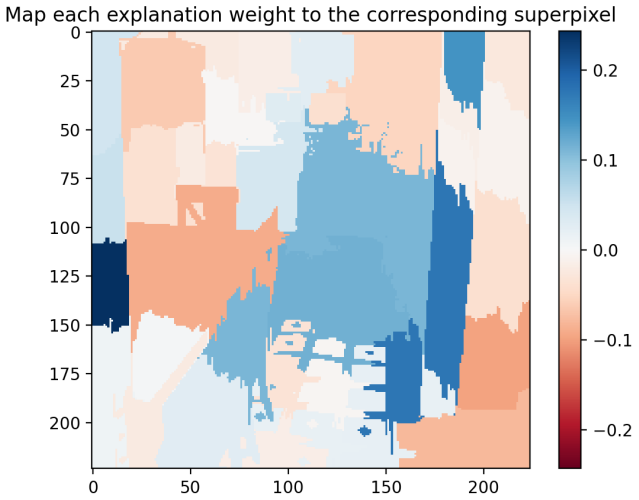
The first image we will examine is of a kitchen, and was falsely classified as industrial. We can see in the first LIME image below the top 5 superpixels that are most positive towards the class (with the rest of the image hidden). The segmentation of this image into superpixels is an integral step in generating explanations for image models. It is important that the segmentation is correct and follows meaningful patterns in the picture, but is also important that the size/number of superpixels is appropriate. A comparison with the second LIME image (the one with the rest of the image present), suggests that the model focused too much on the tiled floor pattern (whose grid-like pattern could have been misinterpreted as an industrial structure) and not enough on the table, chairs, and kitchen cabinets and appliances. This is likely why this image was incorrectly classified as industrial.



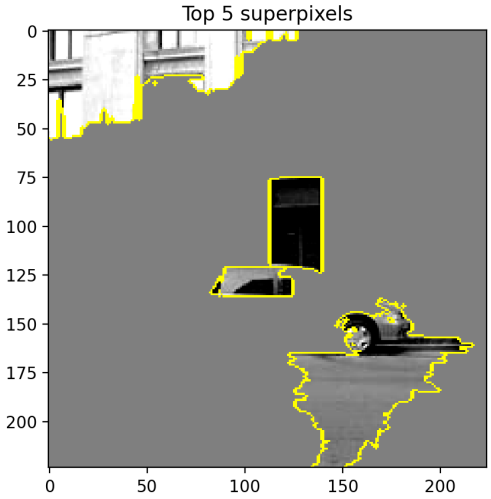



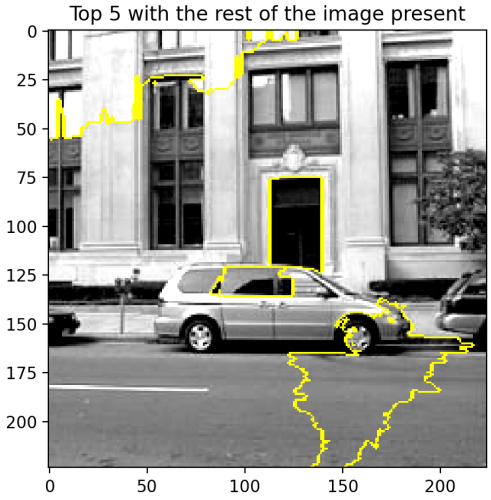
In the following 2 LIME explanation images, we can clearly see that the floor was the area on which the model focused: it disregarded the table, chairs, and surrounding kitchen context. The pros and cons image shows that the model made its classification based almost entirely on the floor pattern, while the mapped image shows how little weight was given to the parts of the image that were not the floor pattern.



The second image we will examine is of type inner city, and was falsely classified as a street. We can see in the first LIME image below the top 5 superpixels that are most positive towards the class (with the rest of the image hidden). The segmentation of this image into superpixels is an integral step in generating explanations for image models. It is important that the segmentation is correct and follows meaningful patterns in the picture, but is also important that the size/number of superpixels is appropriate. A comparison with the second LIME image (the one with the rest of the image present), suggests that the model focused mostly on the the street, the car, and a little bit of the building in the periphery. It thus makes sense that this image was classified as a street, since very little emphasis was placed on the building.



In the following 2 LIME explanation images, we can clearly see that the street, car, and building windows were the areas on which the model focused: it disregarded the original image's emphasis on the street-side building. The pros and cons image shows that the model made its classification based almost entirely on the street, car, and peripheral building windows, while the mapped image shows how little weight was given to the connection between the street and the building, and the majority of the image (which is the building). It therefore is understandable that this image was classified as a street, since most of the building was regarded as relatively unimportant.