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, we attempted to classify scenes on the 15-classified Scence dataset using a convolutional neural network. We used two main approaches to accomplish this task. First, we tried to define and implement a convolutional neural network ourselves, and trained it on the Scence dataset from scratch. We achieved a maximum accuracy of **82.04%** on the validation set. In addition, we also tried fine-tuning the scene recognition task on a given Scene dataset using a VGG-F model pre-trained on ImageNet, and achieved a maximum accuracy of **89.88%** on the validation set.

Implementation Detail

Task 1

your_model: below is the *your_model* code snippets. The final CNN structures contains 6 Conv2D layers with number of filters increase from 64 to 256. Every Conv2D layer is followed by one Dropout layer to avoid overfitting, and after every two Conv2D layers, one MaxPool2D layer is added. The final Conv2D layer is followed by a GlobalAverage-Pooling2D layer. This layer will average values each channel and thus no flatten layer is needed. Then one Dense layer is followed by one Dropout layer to avoid overfitting and one BatchNotmalization layer is added before the final prediction Dense layer. The activation functions for each layer are relu except for the final layer, which is softmax to calculate the prediction probability. The total number of parameters in the model is 3,250,255, which is much smaller than the limit of 15 million parameters.

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

self.architecture = [
              Conv2D(filters=64, kernel_size=5, activation="relu",
                                                padding="same"),
              Dropout(rate=0.15),
              Conv2D(filters=64, kernel_size=5, activation="relu",
                                                padding="same"),
              MaxPool2D(pool_size=(3, 3), strides=2, padding="valid"),
              Dropout(rate=0.15),
              Conv2D(filters=128, kernel_size=5, activation="relu",
                                                padding="same"),
              Dropout(rate=0.15),
              Conv2D(filters=128, kernel_size=5, activation="relu",
                                                padding="same"),
              MaxPool2D(pool_size=(3, 3), strides=2, padding="valid"),
              Conv2D(filters=256, kernel_size=5, activation="relu",
                                                padding="same"),
              Dropout(rate=0.15),
              Conv2D(filters=256, kernel_size=5, activation="relu",
                                                padding="same"),
              MaxPool2D(pool_size=(3, 3), strides=2, padding="valid"),
              Dropout(rate=0.15),
              GlobalAveragePooling2D(),
              Dense(256, activation="relu"),
              BatchNormalization(),
              Dropout(rate=0.5),
              Dense(hp.num_classes, activation="softmax")]
### loss definition
loss = tf.keras.losses.sparse_categorical_crossentropy(labels,
                                    predictions)
```

Task 2

Local Interpretable Model-agnostic Explanation (LIME) can explain how the input features of a machine learning model affect its predictions. For image classification tasks, LIME finds the region of an image (set of superpixels) with the strongst association with a prediction label.

Example 1: The LIME plots are shown in Fig. 1. The image is falsely classified as *OpenCountry* while the ground truth label is *Coast*. From the LIME plots, we can see that the super pixels focus mainly on the sky and the mountains, ignoring the widespread ocean and sandy beaches that surround them, and that the waves are calmer and closer to the fields from a greyscale perspective. That is why the model would classify this image into *OpenCountry* rather than *Coast*. *Example 2*: The LIME plots are shown in Fig. 2. The image is falsely classified as *Tall building* while the ground truth label is *Industrial*. From the LIME plots, we could see that the superpixels are mainly focused on a shelf in the middle that is taller and that the exterior of this shelf resembles, to some extent, a building in the city. It ignores the environment of the surrounding factory. That is why the model would classify this image into *Tall building* rather than *Industrial*.

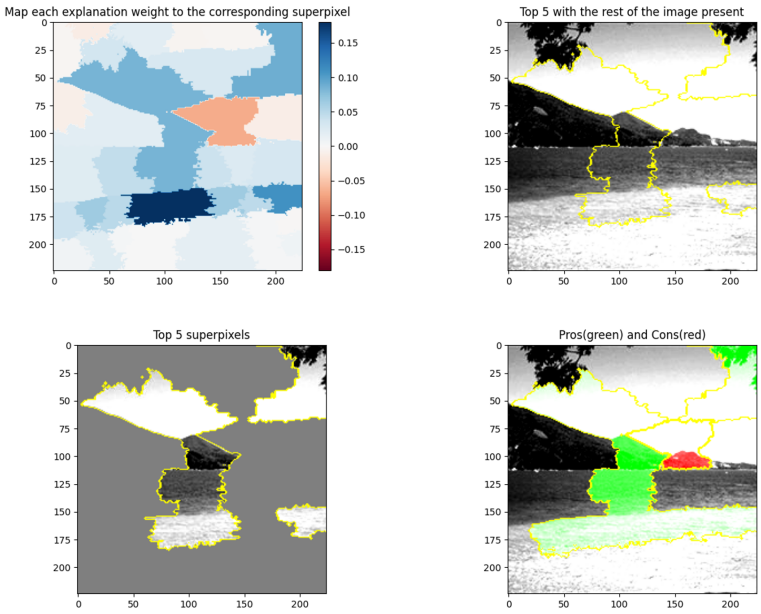


Figure 1: LIME plots for the first falsely categorized image, predicted as *OpenCountry* but ground truth is *Coast*

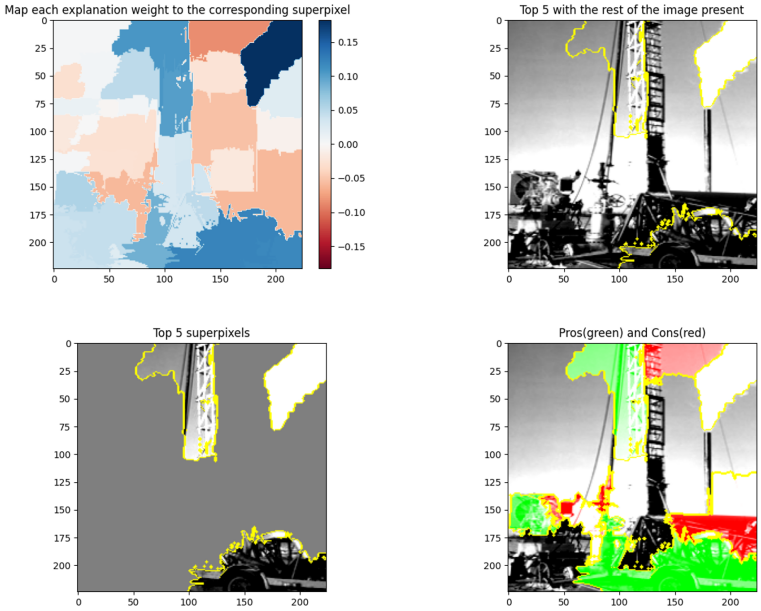


Figure 2: LIME plots for the second falsely categorized image, predicted as *tallbuilding* but ground truth is *Industrial*

Task 3

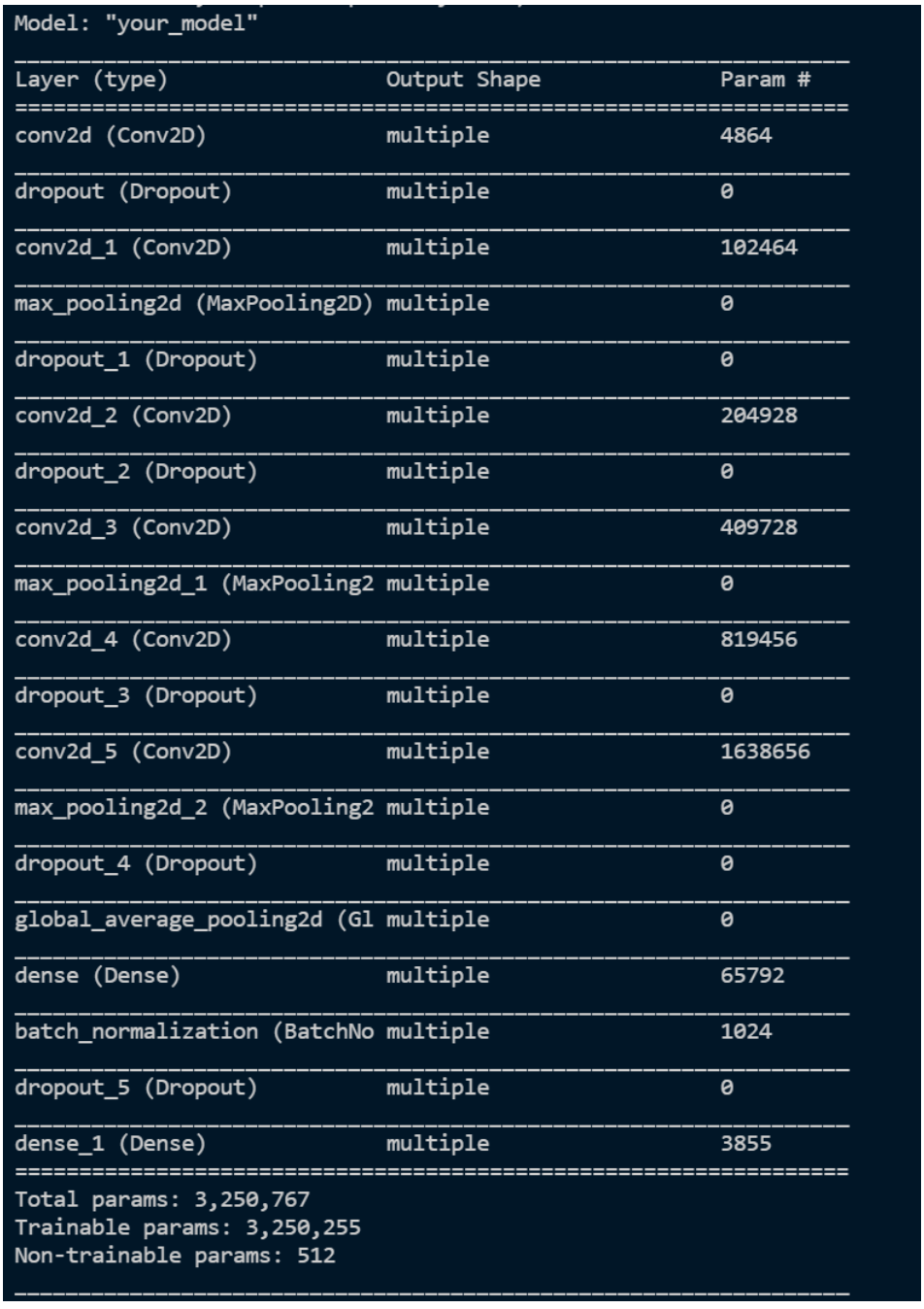
Vgg_head: First, after we defined the basic structure of VGG16, in order to freeze the pretrained VGG16 weights into place so that only the classificaiton head is trained, we make all layers in vgg16 non-trainable. After the VGG model, we first add one Conv2D layer with 1*1 filter to correlate values among channels. Then a GlobalAveragePooling2D layer is added and then one Dropout layer. After this, a Dense layer is added for the prediction. The activation functions for each layer are relu except for the final layer, which is softmax to calculate the prediction probability.

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

self.head = [
    Conv2D(256, 1, 1, padding='same', activation='relu'),
    GlobalAveragePooling2D(),
    Dropout(rate=0.5),
    Dense(hp.num_classes, activation="softmax"),
  ]

### loss definition
loss = tf.keras.losses.sparse_categorical_crossentropy(labels,
                                    predictions)
```

Result

The model summary are presented in Fig. 3 and Fig. 4. The graphs of loss function over time during training are presented in Fig. 5, Fig. 6 and Fig. 7, Fig. 8.

After implementing the model structures described above and hyperparameters optimization, the best achieved performance of *your_model* is 82.04% and VGG is 89.88%. The classification performance for each step of two models are plotted in Fig. 5, Fig. 6 and Fig. 7, Fig. 8.



```
Model: "your_model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              multiple                  4864
_________________________________________________________________
dropout (Dropout)            multiple                  0
_________________________________________________________________
conv2d_1 (Conv2D)            multiple                  102464
_________________________________________________________________
max_pooling2d (MaxPooling2D) multiple                  0
_________________________________________________________________
dropout_1 (Dropout)          multiple                  0
_________________________________________________________________
conv2d_2 (Conv2D)            multiple                  204928
_________________________________________________________________
dropout_2 (Dropout)          multiple                  0
_________________________________________________________________
conv2d_3 (Conv2D)            multiple                  409728
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 multiple                  0
_________________________________________________________________
conv2d_4 (Conv2D)            multiple                  819456
_________________________________________________________________
dropout_3 (Dropout)          multiple                  0
_________________________________________________________________
conv2d_5 (Conv2D)            multiple                  1638656
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 multiple                  0
_________________________________________________________________
dropout_4 (Dropout)          multiple                  0
_________________________________________________________________
global_average_pooling2d (Gl multiple                  0
_________________________________________________________________
dense (Dense)                multiple                  65792
_________________________________________________________________
batch_normalization (BatchNo multiple                  1024
_________________________________________________________________
dropout_5 (Dropout)          multiple                  0
_________________________________________________________________
dense_1 (Dense)              multiple                  3855
=================================================================
Total params: 3,250,767
Trainable params: 3,250,255
Non-trainable params: 512
_________________________________________________________________
```

Figure 3: screenshot of *your_model* summary

```
Model: "vgg_base"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 28, 28, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0
=================================================================
Total params: 14,714,688
Trainable params: 0
Non-trainable params: 14,714,688
_________________________________________________________________
Model: "vgg_head"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 7, 7, 256)         131328
_________________________________________________________________
global_average_pooling2d (Gl (None, 256)               0
_________________________________________________________________
dropout (Dropout)            (None, 256)               0
_________________________________________________________________
dense (Dense)                (None, 15)                3855
=================================================================
Total params: 135,183
Trainable params: 135,183
Non-trainable params: 0
_________________________________________________________________
```


Figure 4: screenshot of VGG model summary

Figure 5: *Above*:classification performance for each step in your_model. *Below*: your_model loss function over time during training

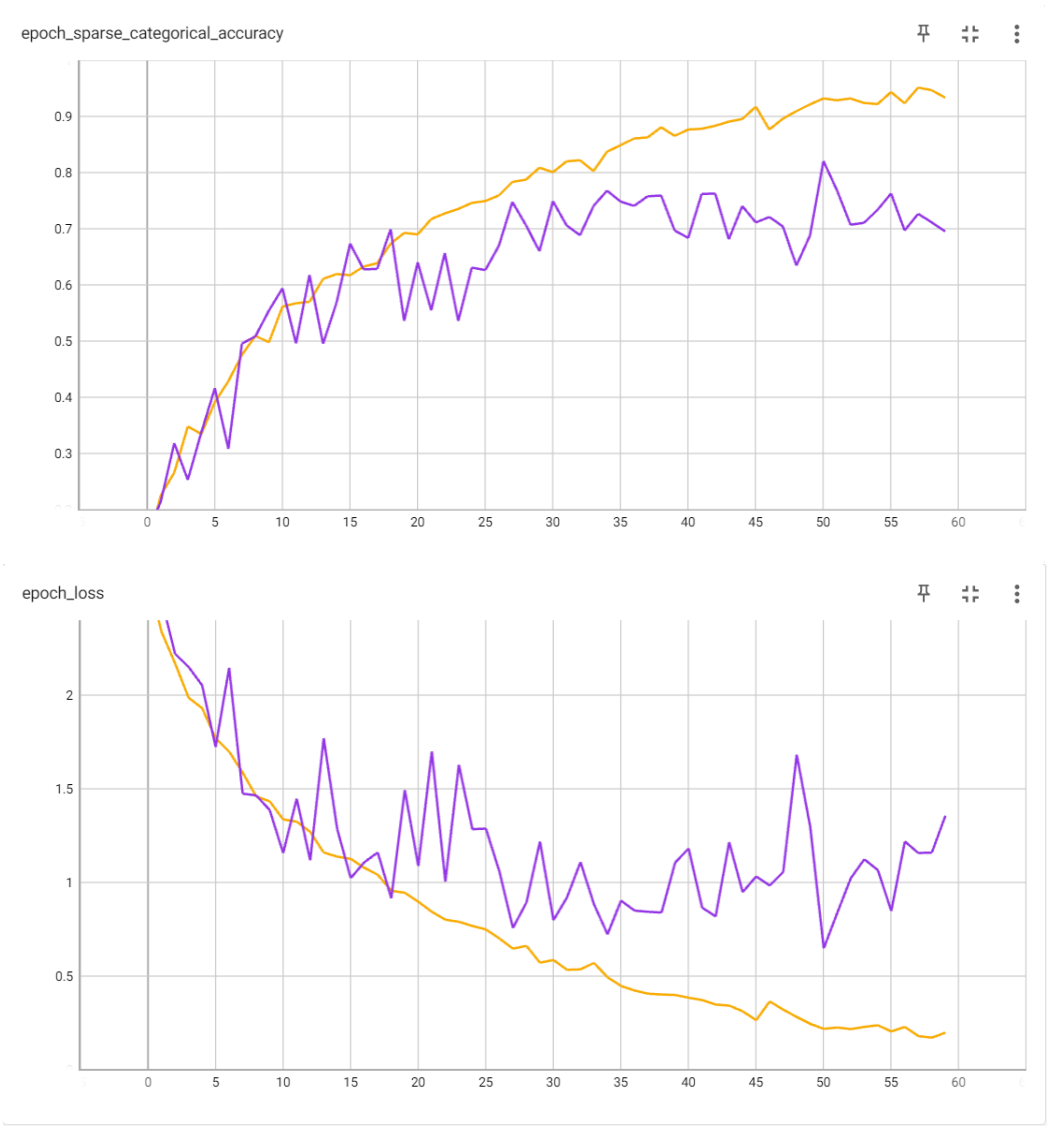


Figure 6: *Above*:classification performance for each epoch in your_model. *Below*: your_model loss function over time during training

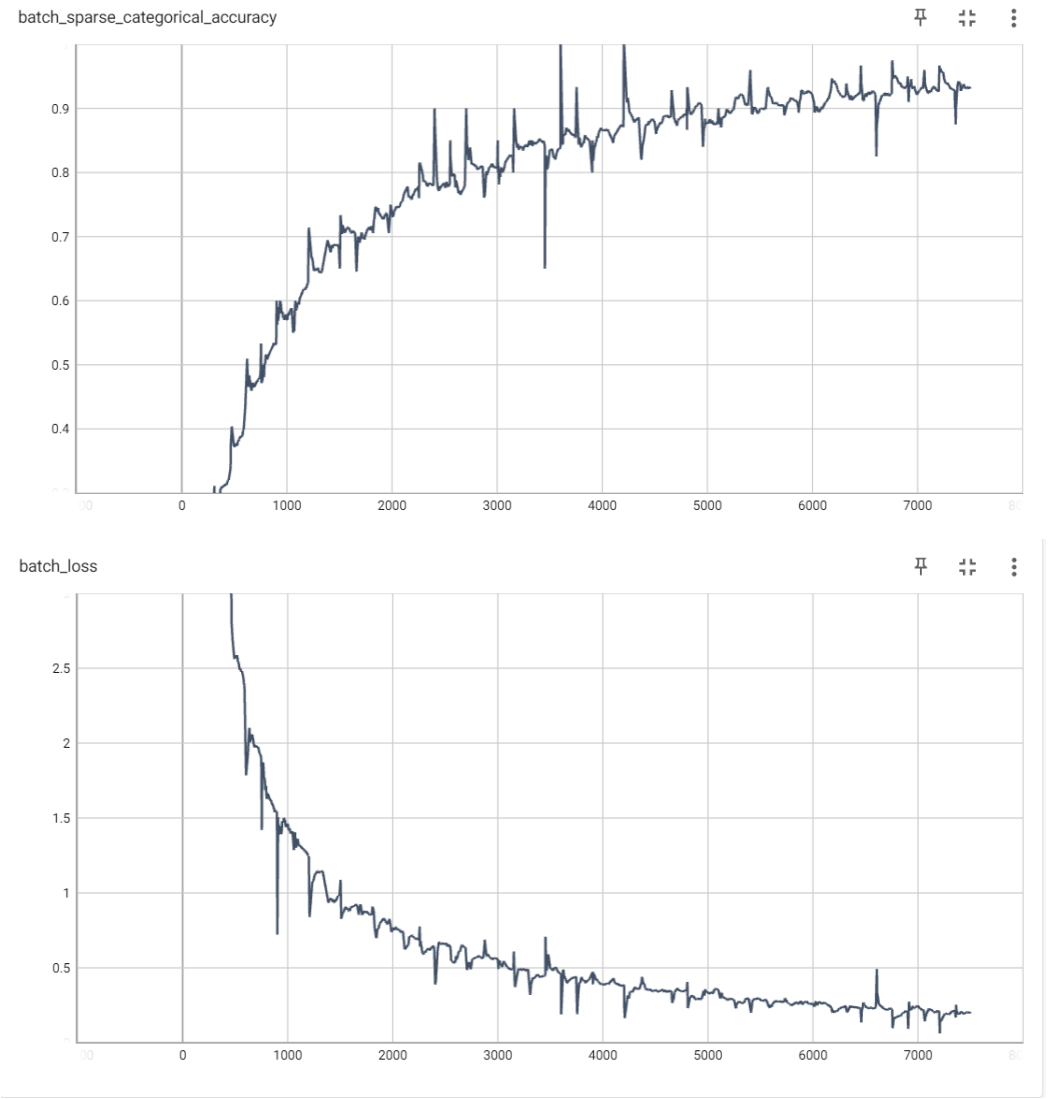


Figure 7: *Above*:classification performance for each step in VGG. *Below*: VGG loss function over time during training

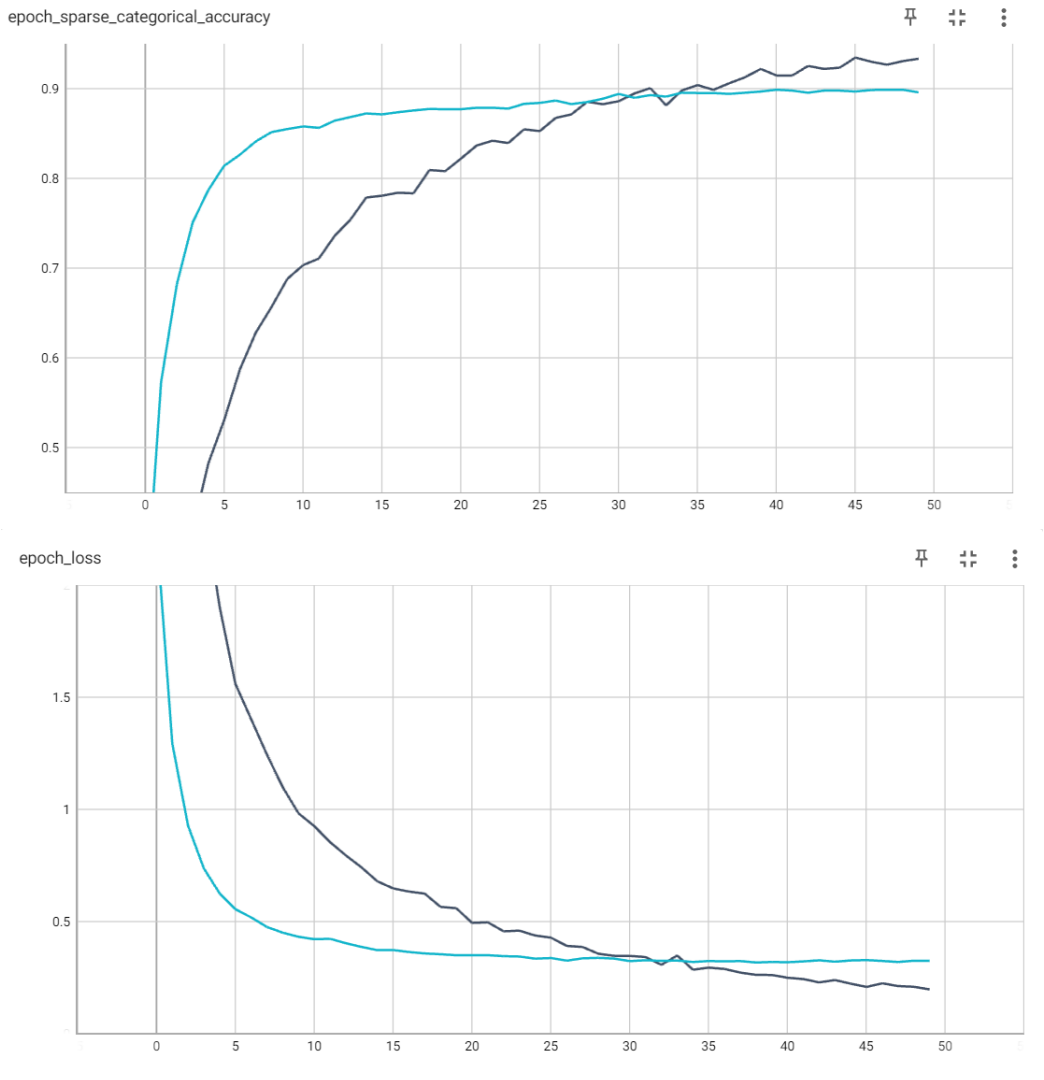


Figure 8: *Above*:classification performance for each epoch in VGG. *Below*: VGG loss function over time during training