Katya Reichert (kpr2120)
Artificial Intelligence

README – HW 5

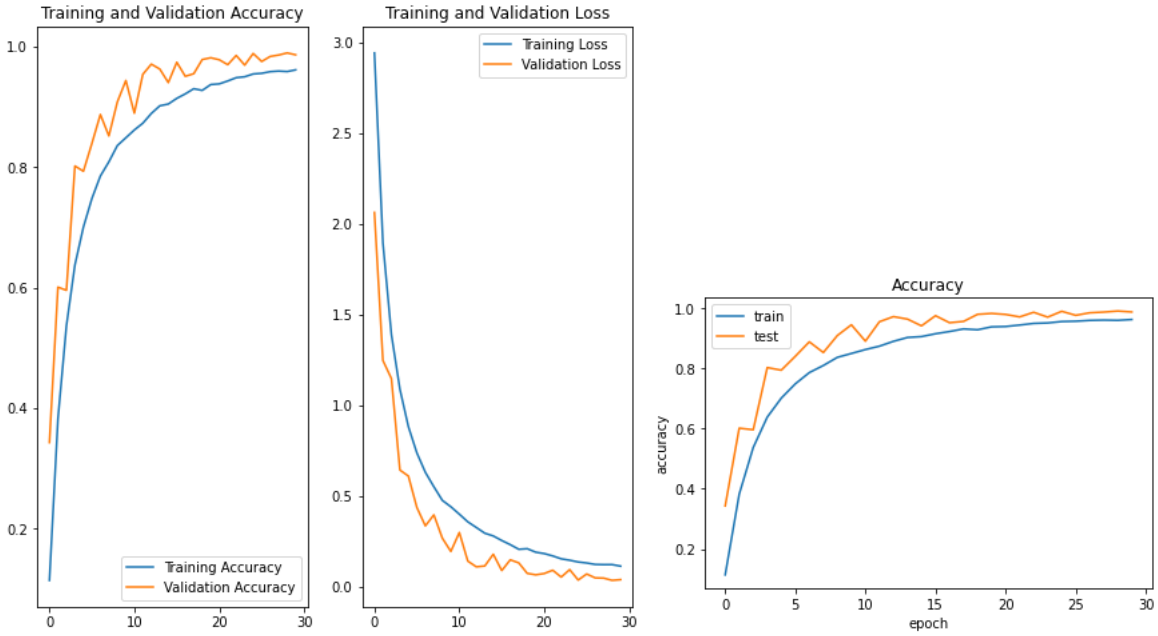
Questions and Answers

1. One-hot encoding is a way to turn categorical values into numerical values, where each category becomes a new feature with a value of 0 or 1, depending on if the example fits in said category or not. In Keras, you can easily one-hot encode categorical values by first assigning each a numerical value [0 - length of category), then using the built in method of `to_categorical()`
2. Dropout is a method of regularization that zeroes out the activation for a randomly selected portion of output units in the network for a single step gradient. This helps solve overfitting by acting like an ensemble of less complex networks and changing how the model responds to the input every time.
3. ReLU differs from sigmoid activation because its gradient has a constant value (since it just picks max(0,x), whereas the sigmoid has to perform complex calculations where the gradient becomes very small as the input gets bigger. This makes ReLU much faster to compute. ReLU also creates more spare representations than sigmoid. ReLU is also just generally preferred by the ML community, as far as I can tell.
4. Softmax is necessary in the output layer so that the output is a vector of probabilities for each category rather than just a vector of numbers.
5. The output dimensions for the convolution layer would be (100, 100, 16) and the output for the max pooling layer would be (50, 50, 16)

Architecture Choices

I chose to format my model very closely to what the above questions were leading to, using successive convolution layers and max pooling layers, but I wanted it to get increasingly more complex as the layers progressed. I found that this would help in increasing the accuracy of the model. I also took inspiration from the TensorFlow documentation and tutorials themselves in the strategies to reduce overfitting, such as adding in layers to the model that add transformed images to the training set for improved performance.

Here are some graphs from my training progress:



The orange ‘test’ line in the second graph refers to the validation set.

The training resulted in the following stats:
train_loss: 0.1136 train_accuracy: 0.9615
val_loss: 0.0394 val_accuracy: 0.9865

In my testing, I was very proud to achieve 92% test accuracy!

```
if __name__=="__main__":
    y_pred = my_model.predict(test_images)
    accuracy = accuracy_score(test_labels, y_pred)
    print(accuracy)
```

```
225/225 [==============================] - 1s 2ms/step
0.9274958170663692
```