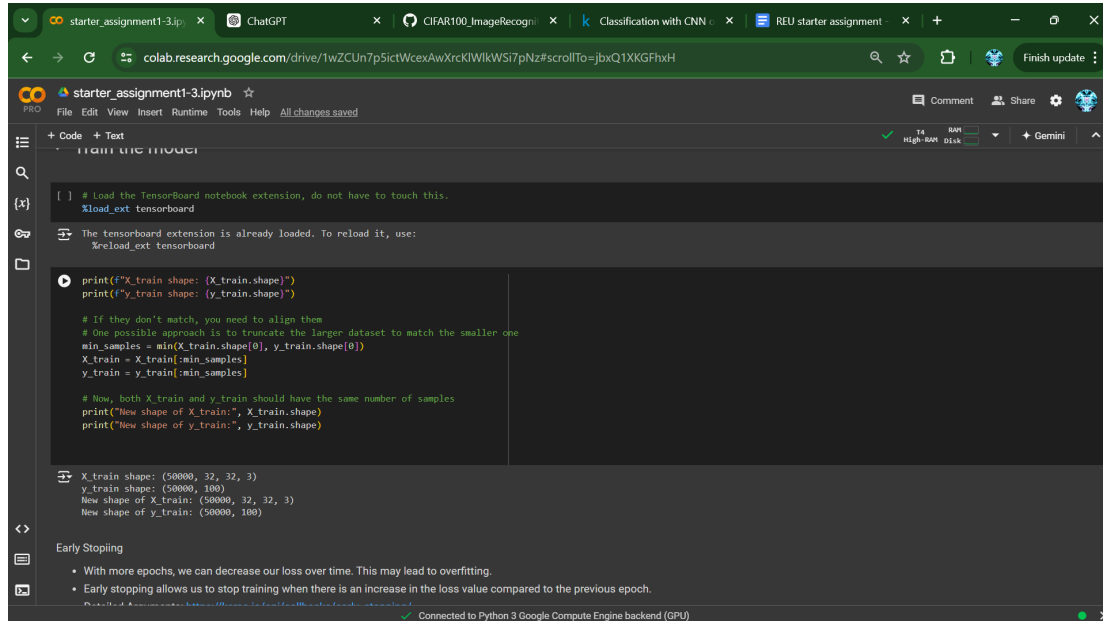


For this assignment, I stacked multiple convolutional layers to classify each image to its corresponding class. The more epochs I used, the higher the validation accuracy, and the lower the loss history. I only used 25 epochs due to lengthy processing times, gpu restrictions, and time constraints. But I was able to see genuine improvement in the accuracy, reaching 50% accuracy with just 25 epochs.



```
[ ] # Load the TensorBoard notebook extension, do not have to touch this.
%load_ext tensorboard

The tensorboard extension is already loaded. To reload it, use:
%reload_ext tensorboard

print("X_train shape: (X_train.shape)")
print("y_train shape: (y_train.shape)")

# If they don't match, you need to align them
# One possible approach is to truncate the larger dataset to match the smaller one
min_samples = min(X_train.shape[0], y_train.shape[0])
X_train = X_train[:min_samples]
y_train = y_train[:min_samples]

# Now, both X_train and y_train should have the same number of samples
print("New shape of X_train:", X_train.shape)
print("New shape of y_train:", y_train.shape)
```

X\_train shape: (50000, 32, 32, 3)  
y\_train shape: (50000, 100)  
New shape of X\_train: (50000, 32, 32, 3)  
New shape of y\_train: (50000, 100)

### Early Stopping

- With more epochs, we can decrease our loss over time. This may lead to overfitting.
- Early stopping allows us to stop training when there is an increase in the loss value compared to the previous epoch.

Connected to Python 3 Google Compute Engine backend (GPU)

This was one of the problems that took me much research and trial/error to figure out. I had essentially had my y\_train sample size differ from my X\_train sample size. The error message that it displayed was very 'wonky' and I didn't understand the error at first. So after much research, I set both variables to [:min\_samples] meaning the number up to the minimum number of samples (which happened to be 50,000). This solved everything from this block to the future ones!

```
Epoch 9/100
391/391 [=====] - 28s 70ms/step - loss: 2.0507 - accuracy: 0.4631 - val_loss: 2.0540 - val_accuracy: 0.4705
Epoch 10/100
391/391 [=====] - 28s 71ms/step - loss: 1.9406 - accuracy: 0.4859 - val_loss: 2.0283 - val_accuracy: 0.4759
Epoch 11/100
391/391 [=====] - 28s 71ms/step - loss: 1.8366 - accuracy: 0.5107 - val_loss: 1.9889 - val_accuracy: 0.4851
Epoch 12/100
391/391 [=====] - 28s 70ms/step - loss: 1.7404 - accuracy: 0.5322 - val_loss: 1.9601 - val_accuracy: 0.4950
Epoch 13/100
391/391 [=====] - 28s 70ms/step - loss: 1.6492 - accuracy: 0.5528 - val_loss: 1.9036 - val_accuracy: 0.5040
Epoch 14/100
391/391 [=====] - 28s 71ms/step - loss: 1.5462 - accuracy: 0.5757 - val_loss: 1.8932 - val_accuracy: 0.5075
Epoch 15/100
391/391 [=====] - 27s 70ms/step - loss: 1.4644 - accuracy: 0.5919 - val_loss: 1.8873 - val_accuracy: 0.5131
Epoch 16/100
391/391 [=====] - 27s 70ms/step - loss: 1.3748 - accuracy: 0.6154 - val_loss: 1.8809 - val_accuracy: 0.5204
Epoch 17/100
391/391 [=====] - 28s 70ms/step - loss: 1.2942 - accuracy: 0.6345 - val_loss: 1.8742 - val_accuracy: 0.5194
Epoch 18/100
391/391 [=====] - 27s 70ms/step - loss: 1.2066 - accuracy: 0.6544 - val_loss: 1.8849 - val_accuracy: 0.5234
Epoch 19/100
391/391 [=====] - 27s 70ms/step - loss: 1.1259 - accuracy: 0.6739 - val_loss: 1.9187 - val_accuracy: 0.5240
Epoch 20/100
391/391 [=====] - 27s 70ms/step - loss: 1.0436 - accuracy: 0.6946 - val_loss: 1.9115 - val_accuracy: 0.5249
Epoch 21/100
391/391 [=====] - 27s 70ms/step - loss: 0.9723 - accuracy: 0.7126 - val_loss: 1.9354 - val_accuracy: 0.5241
Epoch 22/100
391/391 [=====] - 27s 70ms/step - loss: 0.8952 - accuracy: 0.7318 - val_loss: 2.0107 - val_accuracy: 0.5200
Epoch 23/100
391/391 [=====] - 27s 70ms/step - loss: 0.8434 - accuracy: 0.7455 - val_loss: 1.9719 - val_accuracy: 0.5281
Epoch 24/100
391/391 [=====] - 27s 70ms/step - loss: 0.7767 - accuracy: 0.7638 - val_loss: 2.0198 - val_accuracy: 0.5212
Epoch 25/100
390/391 [=====>] - ETA: 0s - loss: 0.7168 - accuracy: 0.7796Restoring model weights from the end of the best epoch: 17.
391/391 [=====] - 27s 70ms/step - loss: 0.7169 - accuracy: 0.7796 - val_loss: 2.0585 - val_accuracy: 0.5327
Epoch 25: early stopping
```

This is displaying the number of epochs. I couldn't do much as it kept stopping early (as shown). But I did buy colab pro to get better/faster results!

```
[ ] score = model.evaluate(X_test, y_test, batch_size=50,
steps=X_test.shape[0] // 50)

print('Test loss: %.2f' % score[0])
print('Test accuracy: %.2f' % score[1])

200/200 [=====] - 2s 10ms/step - loss: 1.8742 - accuracy: 0.5194
Test loss: 1.87
Test accuracy: 0.52

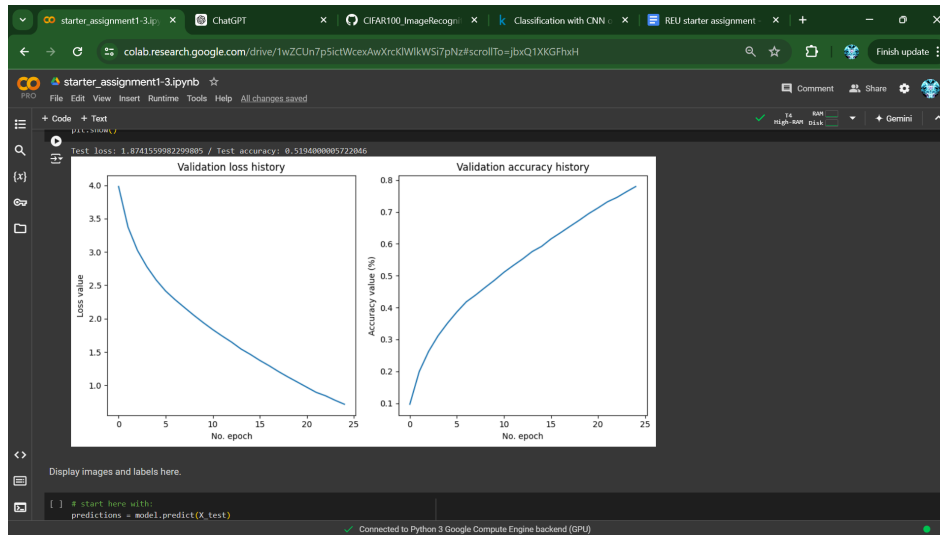
Results

Use matplotlib to draw your results.

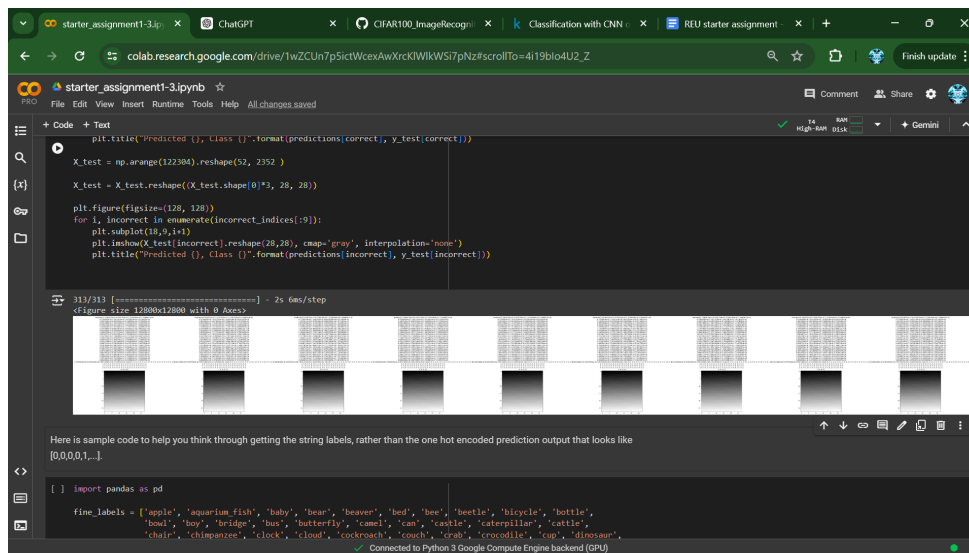
[ ] import matplotlib.pyplot as plt

score = model.evaluate(X_test, y_test, verbose=0)
print('Test loss: (score[0]) // Test accuracy: (score[1])')

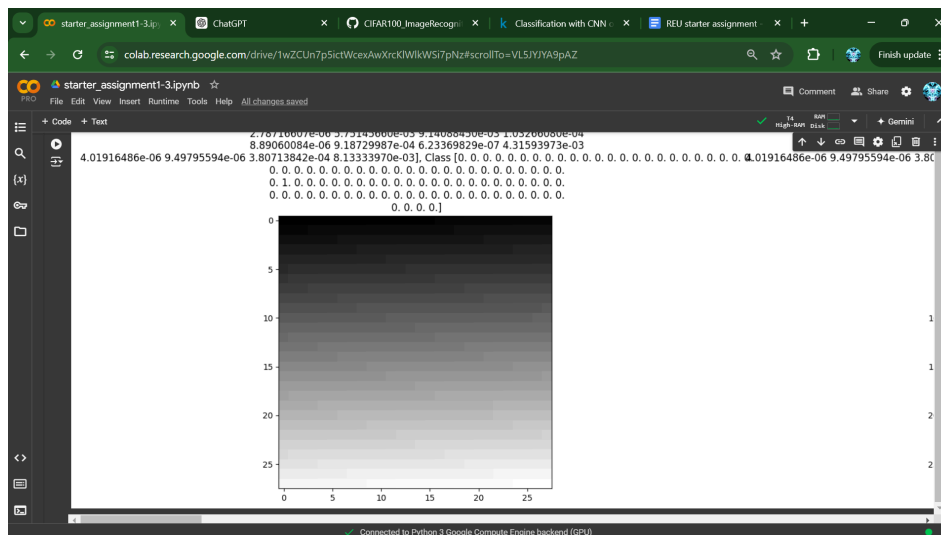
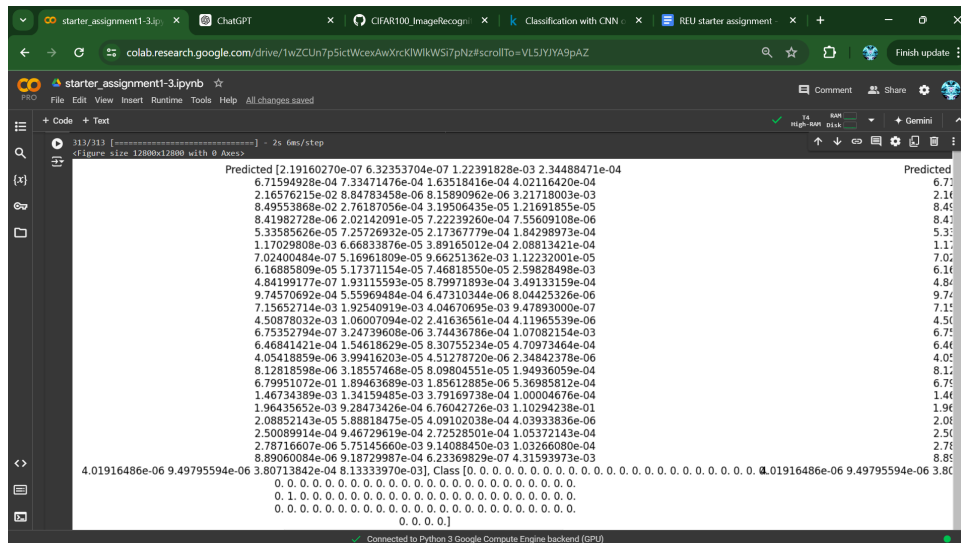
plt.figure(figsize=(10, 5))
```



This is just showing the graphs of my validation loss and accuracy history. As you can see, as the number of epochs grows, my loss decreases while my accuracy increases (reaching 0.8 on the graph).



This is showing the images and labels.



These two are a close up of the first one. I took a bit of formatting for it to not be all jumbled up together.

I think I was able to grasp the understanding of models and displaying results of those models. There were many errors along the way, but solving them helped me understand everything much better. My knowledge of building models and loading in data has definitely improved (as shown by the pictures above) and I can't wait to start my research!!