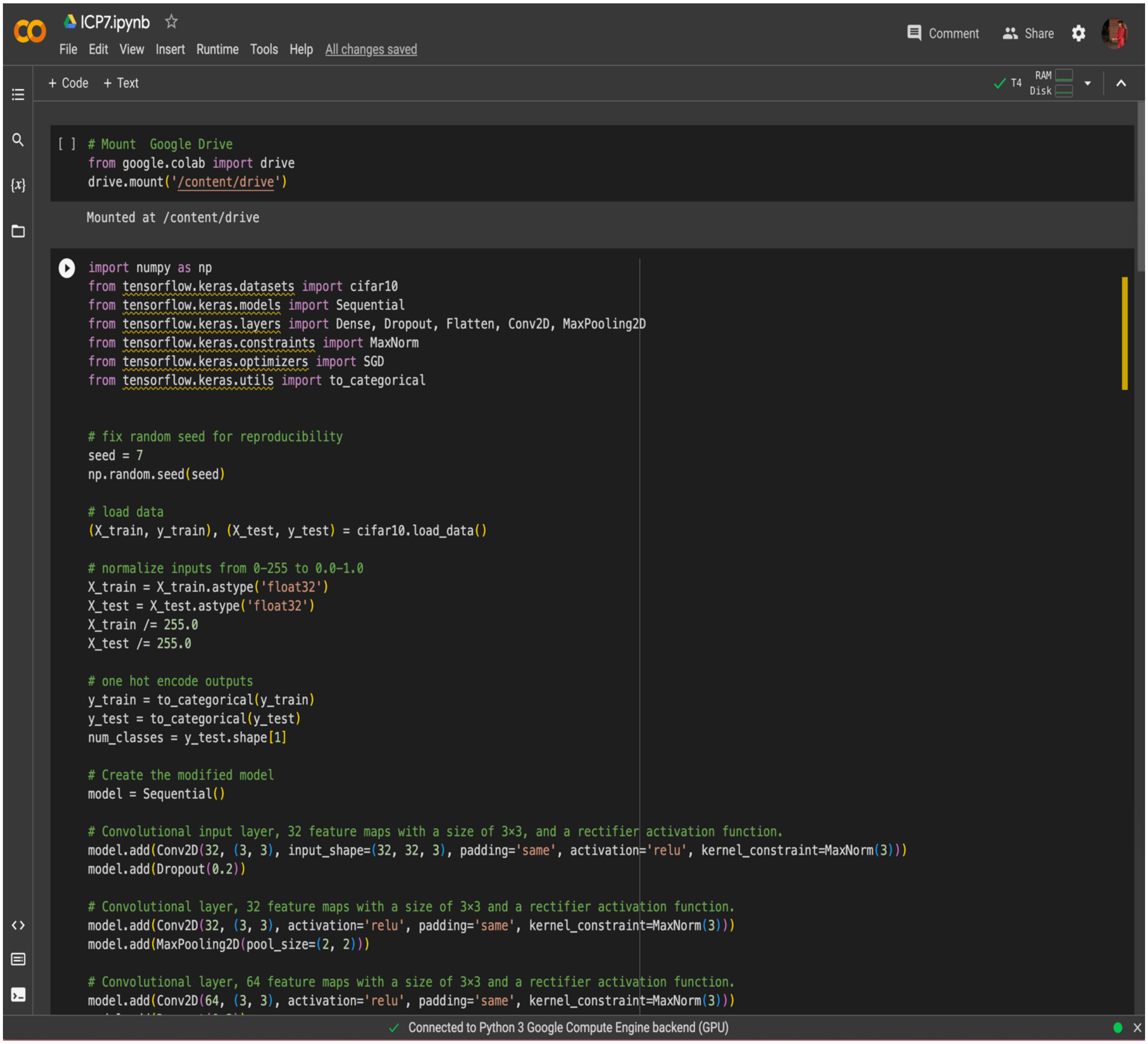
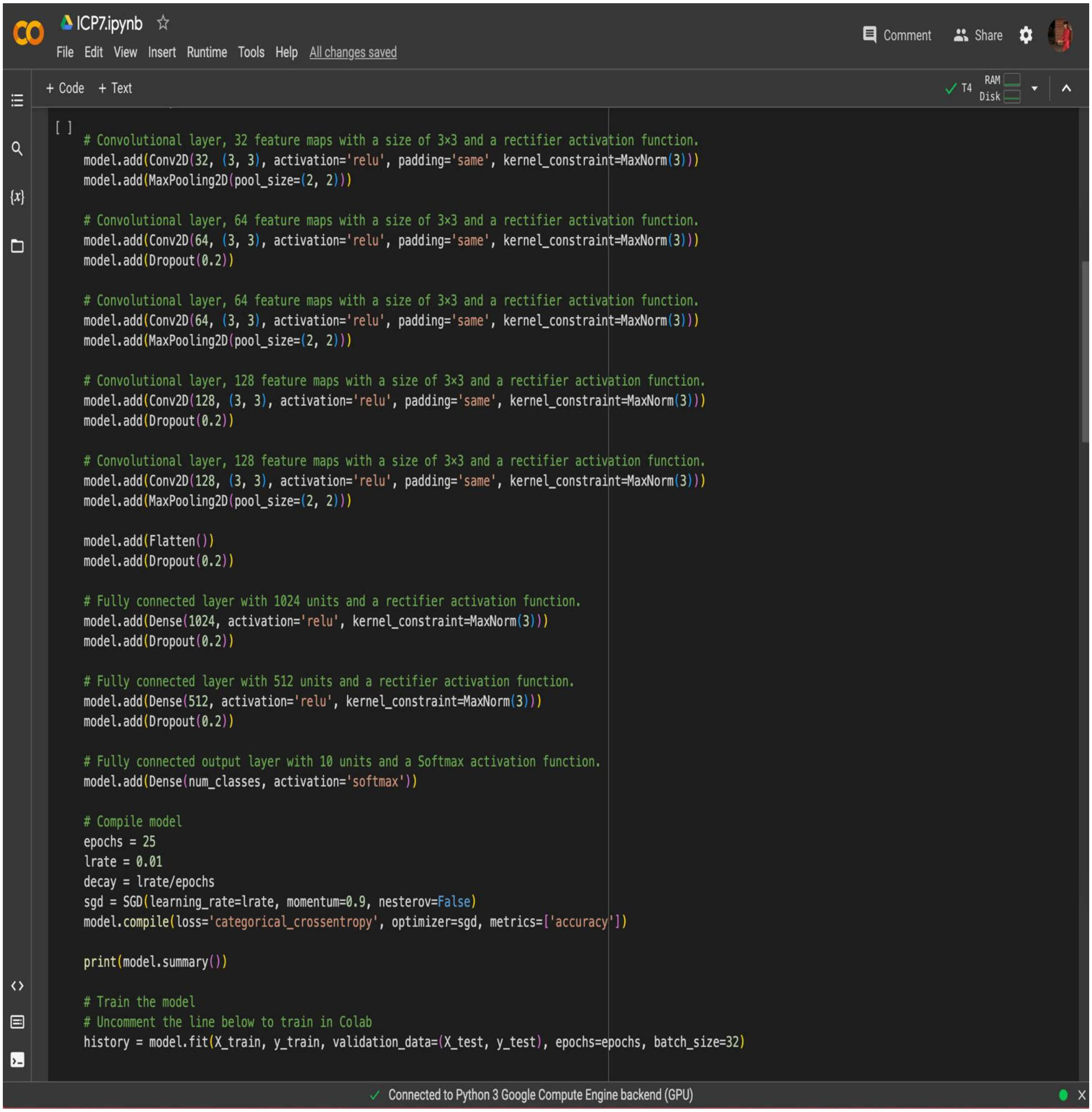


```
# Mount  Google Drive
from google.colab import drive
drive.mount('/content/drive')
```

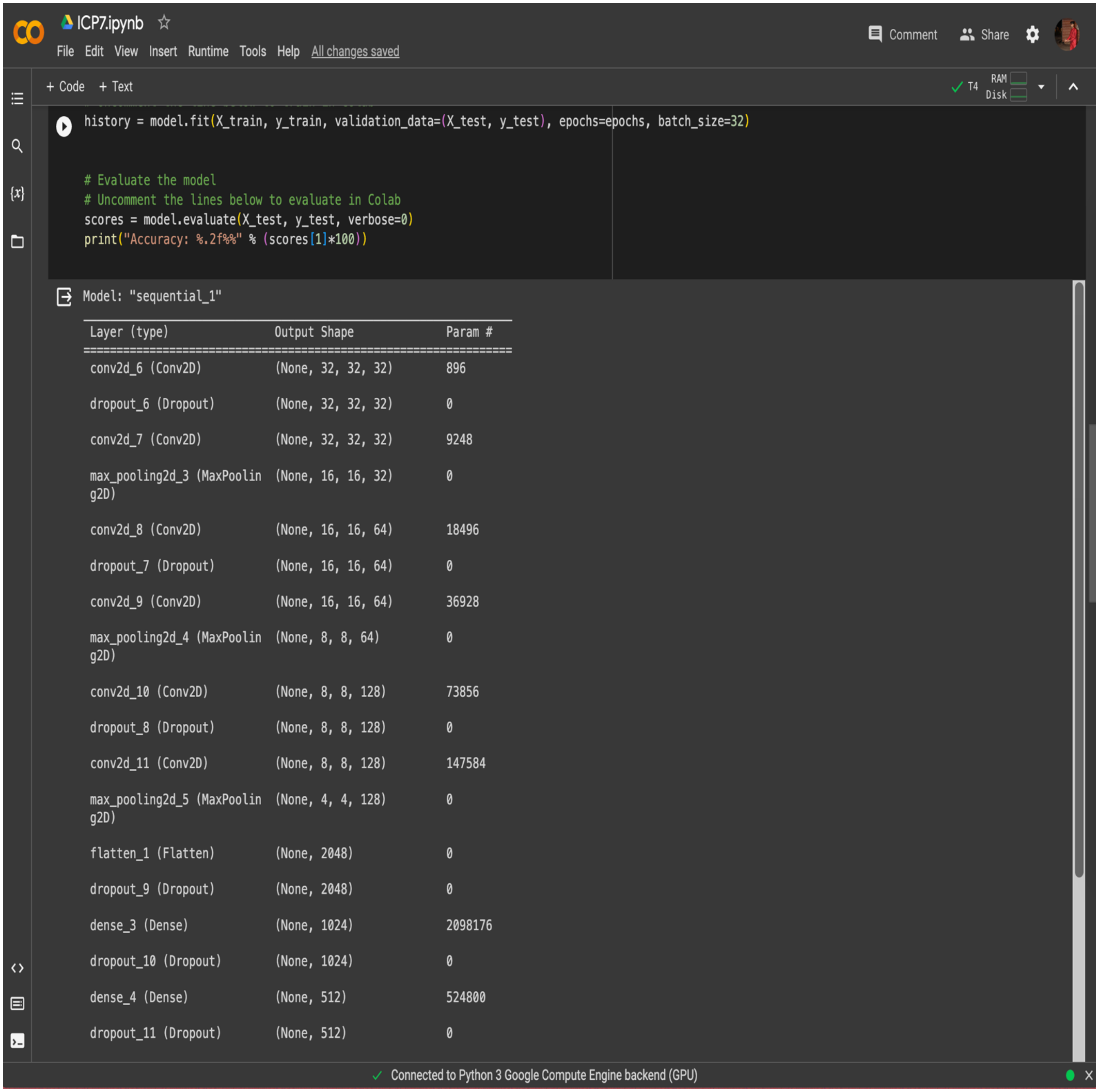
```
Mounted at /content/drive
```

```
import numpy as np
from tensorflow.keras.datasets import cifar10
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPooling2D
from tensorflow.keras.constraints import MaxNorm
from tensorflow.keras.optimizers import SGD
from tensorflow.keras.utils import to_categorical


# fix random seed for reproducibility
seed = 7
np.random.seed(seed)

# load data
(X_train, y_train), (X_test, y_test) = cifar10.load_data()

# normalize inputs from 0-255 to 0.0-1.0
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
X_train /= 255.0
X_test /= 255.0

# one hot encode outputs
y_train = to_categorical(y_train)
y_test = to_categorical(y_test)
num_classes = y_test.shape[1]

# Create the modified model
model = Sequential()

# Convolutional input layer, 32 feature maps with a size of 3x3, and a rectifier activation function.
model.add(Conv2D(32, (3, 3), input_shape=(32, 32, 3), padding='same', activation='relu', kernel_constraint=MaxNorm(3)))
model.add(Dropout(0.2))

# Convolutional layer, 32 feature maps with a size of 3x3 and a rectifier activation function.
model.add(Conv2D(32, (3, 3), activation='relu', padding='same', kernel_constraint=MaxNorm(3)))
model.add(MaxPooling2D(pool_size=(2, 2)))

# Convolutional layer, 64 feature maps with a size of 3x3 and a rectifier activation function.
model.add(Conv2D(64, (3, 3), activation='relu', padding='same', kernel_constraint=MaxNorm(3)))
```



```
# Convolutional layer, 32 feature maps with a size of 3x3 and a rectifier activation function.
model.add(Conv2D(32, (3, 3), activation='relu', padding='same', kernel_constraint=MaxNorm(3)))
model.add(MaxPooling2D(pool_size=(2, 2)))

# Convolutional layer, 64 feature maps with a size of 3x3 and a rectifier activation function.
model.add(Conv2D(64, (3, 3), activation='relu', padding='same', kernel_constraint=MaxNorm(3)))
model.add(Dropout(0.2))

# Convolutional layer, 64 feature maps with a size of 3x3 and a rectifier activation function.
model.add(Conv2D(64, (3, 3), activation='relu', padding='same', kernel_constraint=MaxNorm(3)))
model.add(MaxPooling2D(pool_size=(2, 2)))

# Convolutional layer, 128 feature maps with a size of 3x3 and a rectifier activation function.
model.add(Conv2D(128, (3, 3), activation='relu', padding='same', kernel_constraint=MaxNorm(3)))
model.add(Dropout(0.2))

# Convolutional layer, 128 feature maps with a size of 3x3 and a rectifier activation function.
model.add(Conv2D(128, (3, 3), activation='relu', padding='same', kernel_constraint=MaxNorm(3)))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Flatten())
model.add(Dropout(0.2))

# Fully connected layer with 1024 units and a rectifier activation function.
model.add(Dense(1024, activation='relu', kernel_constraint=MaxNorm(3)))
model.add(Dropout(0.2))

# Fully connected layer with 512 units and a rectifier activation function.
model.add(Dense(512, activation='relu', kernel_constraint=MaxNorm(3)))
model.add(Dropout(0.2))

# Fully connected output layer with 10 units and a Softmax activation function.
model.add(Dense(num_classes, activation='softmax'))

# Compile model
epochs = 25
lrate = 0.01
decay = lrate/epochs
sgd = SGD(learning_rate=lrate, momentum=0.9, nesterov=False)
model.compile(loss='categorical_crossentropy', optimizer=sgd, metrics=['accuracy'])

print(model.summary())

# Train the model
# Uncomment the line below to train in Colab
history = model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=epochs, batch_size=32)
```

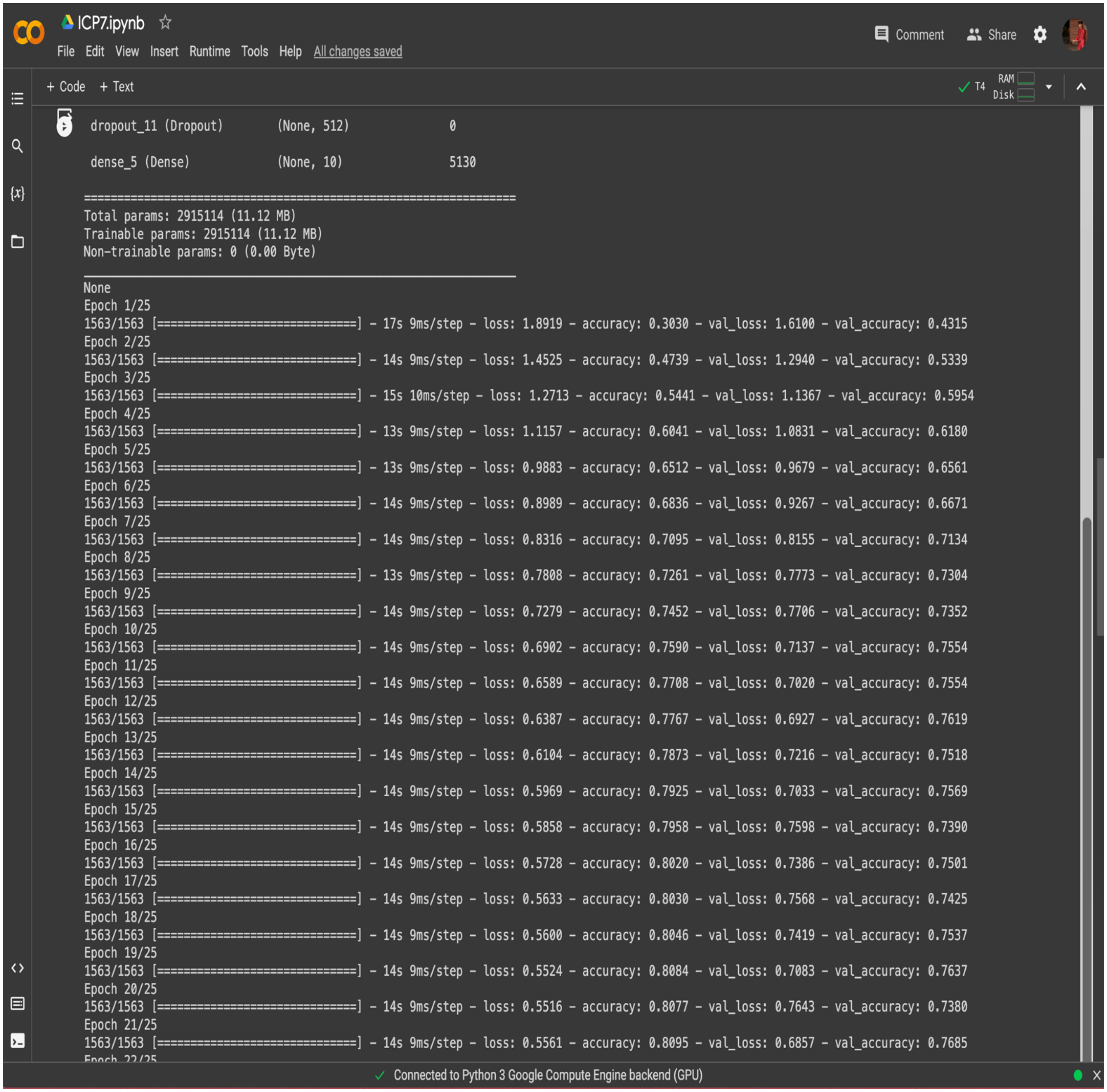
✓ Connected to Python 3 Google Compute Engine backend (GPU)



```
history = model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=epochs, batch_size=32)


# Evaluate the model
# Uncomment the lines below to evaluate in Colab
scores = model.evaluate(X_test, y_test, verbose=0)
print("Accuracy: %.2f%%" % (scores[1]*100))
```

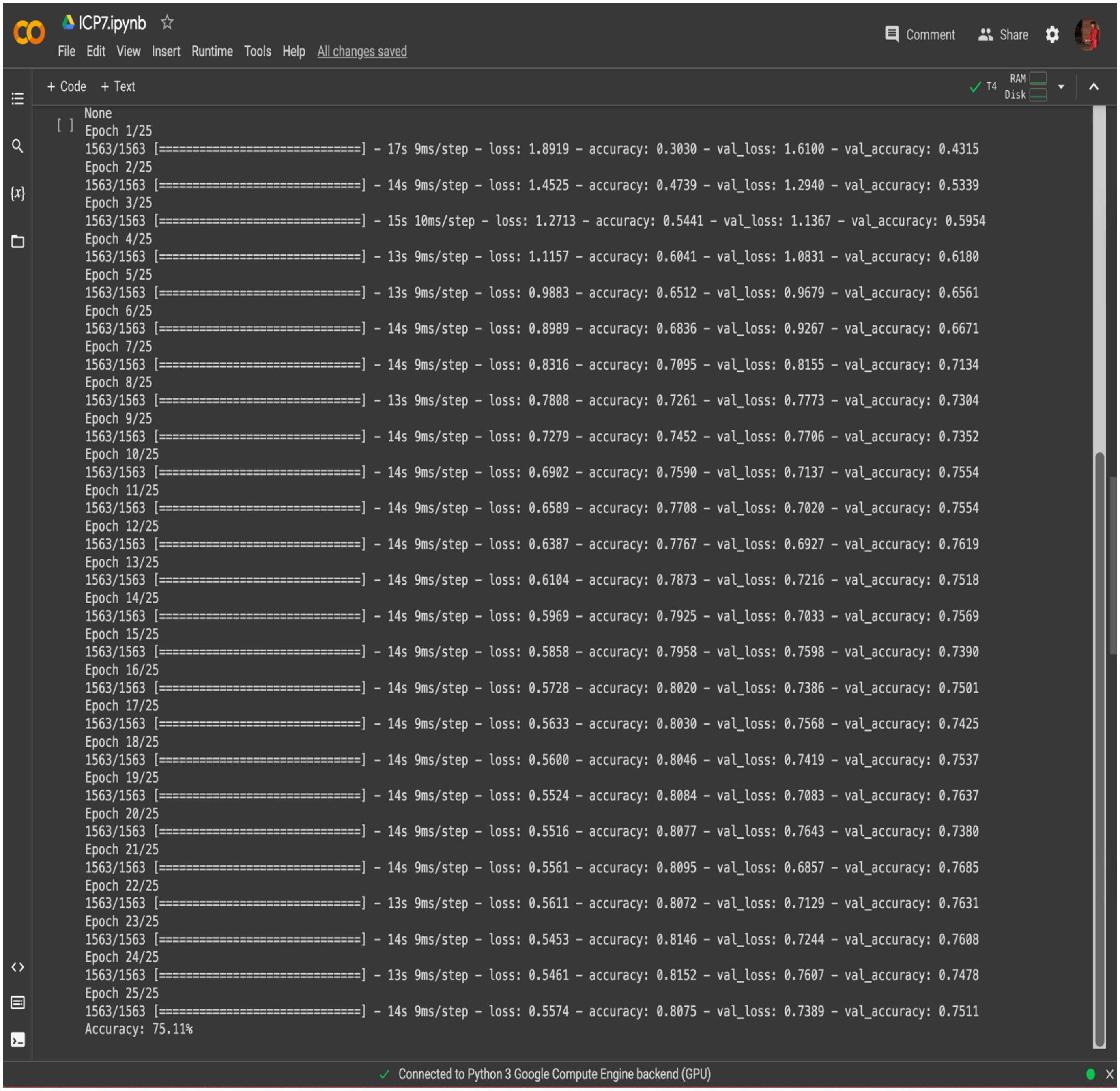
```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_6 (Conv2D)           (None, 32, 32, 32)        896

 dropout_6 (Dropout)         (None, 32, 32, 32)        0

 conv2d_7 (Conv2D)           (None, 32, 32, 32)        9248

 max_pooling2d_3 (MaxPoolin  (None, 16, 16, 32)        0
 g2D)

 conv2d_8 (Conv2D)           (None, 16, 16, 64)        18496

 dropout_7 (Dropout)         (None, 16, 16, 64)        0

 conv2d_9 (Conv2D)           (None, 16, 16, 64)        36928

 max_pooling2d_4 (MaxPoolin  (None, 8, 8, 64)          0
 g2D)

 conv2d_10 (Conv2D)          (None, 8, 8, 128)         73856

 dropout_8 (Dropout)         (None, 8, 8, 128)         0

 conv2d_11 (Conv2D)          (None, 8, 8, 128)         147584

 max_pooling2d_5 (MaxPoolin  (None, 4, 4, 128)         0
 g2D)

 flatten_1 (Flatten)         (None, 2048)              0

 dropout_9 (Dropout)         (None, 2048)              0

 dense_3 (Dense)             (None, 1024)              2098176

 dropout_10 (Dropout)        (None, 1024)              0

 dense_4 (Dense)             (None, 512)               524800

 dropout_11 (Dropout)        (None, 512)               0
```

✓ Connected to Python 3 Google Compute Engine backend (GPU)



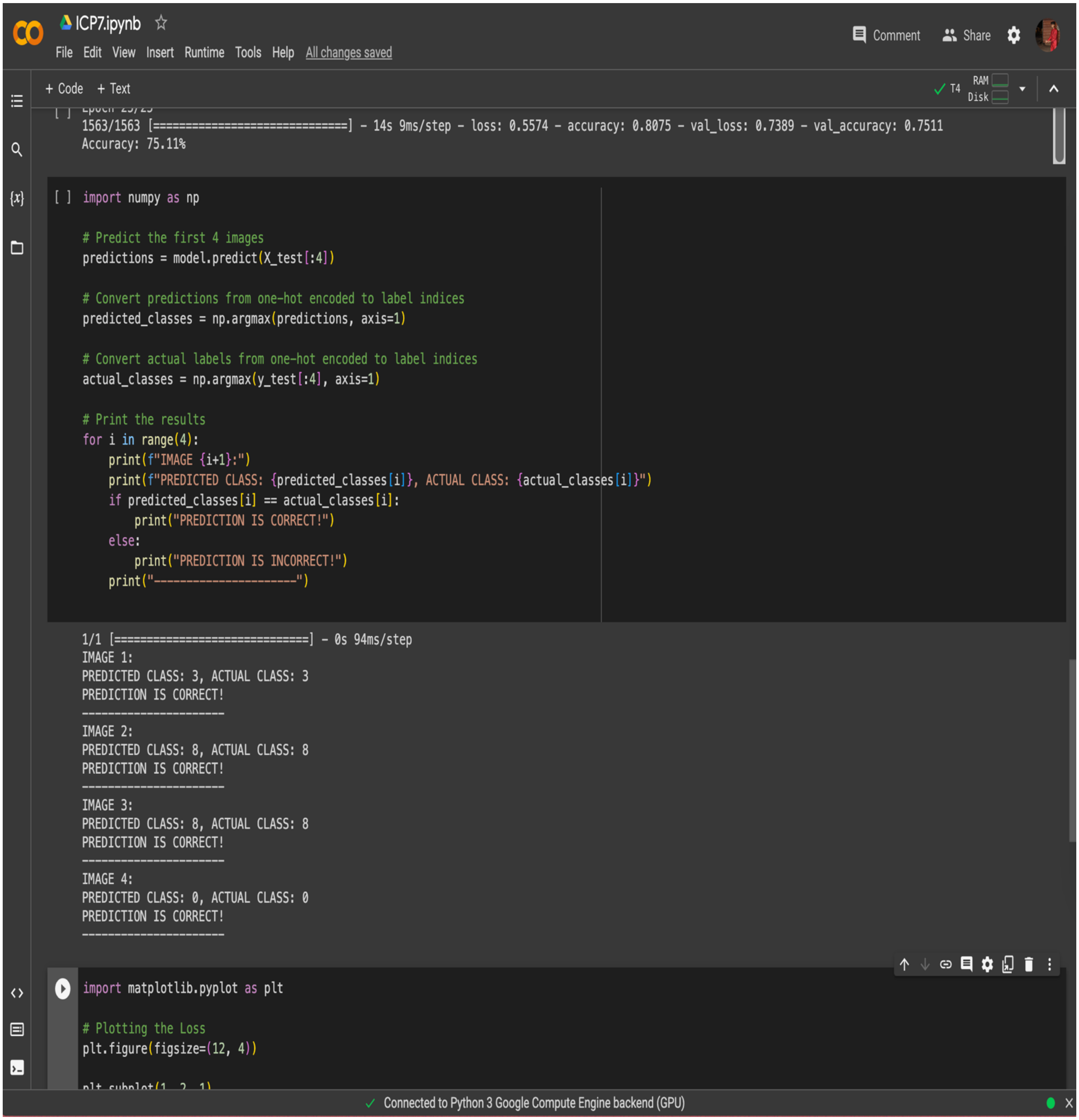
```
 dropout_11 (Dropout)        (None, 512)               0

 dense_5 (Dense)             (None, 10)                5130

=================================================================
Total params: 2915114 (11.12 MB)
Trainable params: 2915114 (11.12 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
None
Epoch 1/25
1563/1563 [==============================] - 17s 9ms/step - loss: 1.8919 - accuracy: 0.3030 - val_loss: 1.6100 - val_accuracy: 0.4315
Epoch 2/25
1563/1563 [==============================] - 14s 9ms/step - loss: 1.4525 - accuracy: 0.4739 - val_loss: 1.2940 - val_accuracy: 0.5339
Epoch 3/25
1563/1563 [==============================] - 15s 10ms/step - loss: 1.2713 - accuracy: 0.5441 - val_loss: 1.1367 - val_accuracy: 0.5954
Epoch 4/25
1563/1563 [==============================] - 13s 9ms/step - loss: 1.1157 - accuracy: 0.6041 - val_loss: 1.0831 - val_accuracy: 0.6180
Epoch 5/25
1563/1563 [==============================] - 13s 9ms/step - loss: 0.9883 - accuracy: 0.6512 - val_loss: 0.9679 - val_accuracy: 0.6561
Epoch 6/25
1563/1563 [==============================] - 14s 9ms/step - loss: 0.8989 - accuracy: 0.6836 - val_loss: 0.9267 - val_accuracy: 0.6671
Epoch 7/25
1563/1563 [==============================] - 14s 9ms/step - loss: 0.8316 - accuracy: 0.7095 - val_loss: 0.8155 - val_accuracy: 0.7134
Epoch 8/25
1563/1563 [==============================] - 13s 9ms/step - loss: 0.7808 - accuracy: 0.7261 - val_loss: 0.7773 - val_accuracy: 0.7304
Epoch 9/25
1563/1563 [==============================] - 14s 9ms/step - loss: 0.7279 - accuracy: 0.7452 - val_loss: 0.7706 - val_accuracy: 0.7352
Epoch 10/25
1563/1563 [==============================] - 14s 9ms/step - loss: 0.6902 - accuracy: 0.7590 - val_loss: 0.7137 - val_accuracy: 0.7554
Epoch 11/25
1563/1563 [==============================] - 14s 9ms/step - loss: 0.6589 - accuracy: 0.7708 - val_loss: 0.7020 - val_accuracy: 0.7554
Epoch 12/25
1563/1563 [==============================] - 14s 9ms/step - loss: 0.6387 - accuracy: 0.7767 - val_loss: 0.6927 - val_accuracy: 0.7619
Epoch 13/25
1563/1563 [==============================] - 14s 9ms/step - loss: 0.6104 - accuracy: 0.7873 - val_loss: 0.7216 - val_accuracy: 0.7518
Epoch 14/25
1563/1563 [==============================] - 14s 9ms/step - loss: 0.5969 - accuracy: 0.7925 - val_loss: 0.7033 - val_accuracy: 0.7569
Epoch 15/25
1563/1563 [==============================] - 14s 9ms/step - loss: 0.5858 - accuracy: 0.7958 - val_loss: 0.7598 - val_accuracy: 0.7390
Epoch 16/25
1563/1563 [==============================] - 14s 9ms/step - loss: 0.5728 - accuracy: 0.8020 - val_loss: 0.7386 - val_accuracy: 0.7501
Epoch 17/25
1563/1563 [==============================] - 14s 9ms/step - loss: 0.5633 - accuracy: 0.8030 - val_loss: 0.7568 - val_accuracy: 0.7425
Epoch 18/25
1563/1563 [==============================] - 14s 9ms/step - loss: 0.5600 - accuracy: 0.8046 - val_loss: 0.7419 - val_accuracy: 0.7537
Epoch 19/25
1563/1563 [==============================] - 14s 9ms/step - loss: 0.5524 - accuracy: 0.8084 - val_loss: 0.7083 - val_accuracy: 0.7637
Epoch 20/25
1563/1563 [==============================] - 14s 9ms/step - loss: 0.5516 - accuracy: 0.8077 - val_loss: 0.7643 - val_accuracy: 0.7380
Epoch 21/25
1563/1563 [==============================] - 14s 9ms/step - loss: 0.5561 - accuracy: 0.8095 - val_loss: 0.6857 - val_accuracy: 0.7685
Epoch 22/25
```

✓ Connected to Python 3 Google Compute Engine backend (GPU)



```
None
Epoch 1/25
1563/1563 [==============================] - 17s 9ms/step - loss: 1.8919 - accuracy: 0.3030 - val_loss: 1.6100 - val_accuracy: 0.4315
Epoch 2/25
1563/1563 [==============================] - 14s 9ms/step - loss: 1.4525 - accuracy: 0.4739 - val_loss: 1.2940 - val_accuracy: 0.5339
Epoch 3/25
1563/1563 [==============================] - 15s 10ms/step - loss: 1.2713 - accuracy: 0.5441 - val_loss: 1.1367 - val_accuracy: 0.5954
Epoch 4/25
1563/1563 [==============================] - 13s 9ms/step - loss: 1.1157 - accuracy: 0.6041 - val_loss: 1.0831 - val_accuracy: 0.6180
Epoch 5/25
1563/1563 [==============================] - 13s 9ms/step - loss: 0.9883 - accuracy: 0.6512 - val_loss: 0.9679 - val_accuracy: 0.6561
Epoch 6/25
1563/1563 [==============================] - 14s 9ms/step - loss: 0.8989 - accuracy: 0.6836 - val_loss: 0.9267 - val_accuracy: 0.6671
Epoch 7/25
1563/1563 [==============================] - 14s 9ms/step - loss: 0.8316 - accuracy: 0.7095 - val_loss: 0.8155 - val_accuracy: 0.7134
Epoch 8/25
1563/1563 [==============================] - 13s 9ms/step - loss: 0.7808 - accuracy: 0.7261 - val_loss: 0.7773 - val_accuracy: 0.7304
Epoch 9/25
1563/1563 [==============================] - 14s 9ms/step - loss: 0.7279 - accuracy: 0.7452 - val_loss: 0.7706 - val_accuracy: 0.7352
Epoch 10/25
1563/1563 [==============================] - 14s 9ms/step - loss: 0.6902 - accuracy: 0.7590 - val_loss: 0.7137 - val_accuracy: 0.7554
Epoch 11/25
1563/1563 [==============================] - 14s 9ms/step - loss: 0.6589 - accuracy: 0.7708 - val_loss: 0.7020 - val_accuracy: 0.7554
Epoch 12/25
1563/1563 [==============================] - 14s 9ms/step - loss: 0.6387 - accuracy: 0.7767 - val_loss: 0.6927 - val_accuracy: 0.7619
Epoch 13/25
1563/1563 [==============================] - 14s 9ms/step - loss: 0.6104 - accuracy: 0.7873 - val_loss: 0.7216 - val_accuracy: 0.7518
Epoch 14/25
1563/1563 [==============================] - 14s 9ms/step - loss: 0.5969 - accuracy: 0.7925 - val_loss: 0.7033 - val_accuracy: 0.7569
Epoch 15/25
1563/1563 [==============================] - 14s 9ms/step - loss: 0.5858 - accuracy: 0.7958 - val_loss: 0.7598 - val_accuracy: 0.7390
Epoch 16/25
1563/1563 [==============================] - 14s 9ms/step - loss: 0.5728 - accuracy: 0.8020 - val_loss: 0.7386 - val_accuracy: 0.7501
Epoch 17/25
1563/1563 [==============================] - 14s 9ms/step - loss: 0.5633 - accuracy: 0.8030 - val_loss: 0.7568 - val_accuracy: 0.7425
Epoch 18/25
1563/1563 [==============================] - 14s 9ms/step - loss: 0.5600 - accuracy: 0.8046 - val_loss: 0.7419 - val_accuracy: 0.7537
Epoch 19/25
1563/1563 [==============================] - 14s 9ms/step - loss: 0.5524 - accuracy: 0.8084 - val_loss: 0.7083 - val_accuracy: 0.7637
Epoch 20/25
1563/1563 [==============================] - 14s 9ms/step - loss: 0.5516 - accuracy: 0.8077 - val_loss: 0.7643 - val_accuracy: 0.7380
Epoch 21/25
1563/1563 [==============================] - 14s 9ms/step - loss: 0.5561 - accuracy: 0.8095 - val_loss: 0.6857 - val_accuracy: 0.7685
Epoch 22/25
1563/1563 [==============================] - 13s 9ms/step - loss: 0.5611 - accuracy: 0.8072 - val_loss: 0.7129 - val_accuracy: 0.7631
Epoch 23/25
1563/1563 [==============================] - 14s 9ms/step - loss: 0.5453 - accuracy: 0.8146 - val_loss: 0.7244 - val_accuracy: 0.7608
Epoch 24/25
1563/1563 [==============================] - 13s 9ms/step - loss: 0.5461 - accuracy: 0.8152 - val_loss: 0.7607 - val_accuracy: 0.7478
Epoch 25/25
1563/1563 [==============================] - 14s 9ms/step - loss: 0.5574 - accuracy: 0.8075 - val_loss: 0.7389 - val_accuracy: 0.7511
Accuracy: 75.11%
```

✓ Connected to Python 3 Google Compute Engine backend (GPU)



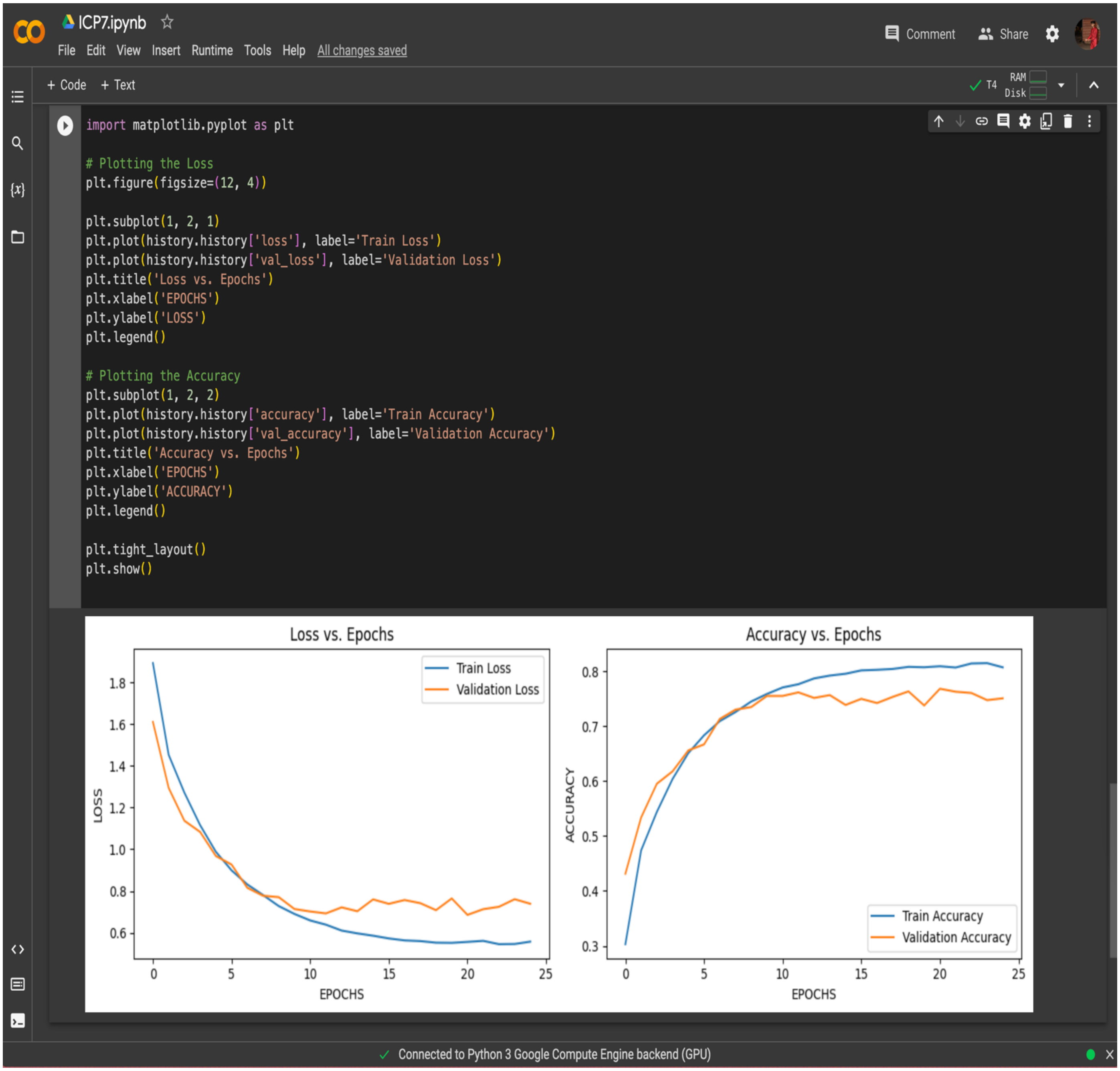
```
Epoch 25/25
1563/1563 [==============================] - 14s 9ms/step - loss: 0.5574 - accuracy: 0.8075 - val_loss: 0.7389 - val_accuracy: 0.7511
Accuracy: 75.11%
```

```
import numpy as np

# Predict the first 4 images
predictions = model.predict(X_test[:4])

# Convert predictions from one-hot encoded to label indices
predicted_classes = np.argmax(predictions, axis=1)

# Convert actual labels from one-hot encoded to label indices
actual_classes = np.argmax(y_test[:4], axis=1)

# Print the results
for i in range(4):
    print(f"IMAGE {i+1}:")
    print(f"PREDICTED CLASS: {predicted_classes[i]}, ACTUAL CLASS: {actual_classes[i]}")
    if predicted_classes[i] == actual_classes[i]:
        print("PREDICTION IS CORRECT!")
    else:
        print("PREDICTION IS INCORRECT!")
    print("----------------------")
```

```
1/1 [==============================] - 0s 94ms/step
IMAGE 1:
PREDICTED CLASS: 3, ACTUAL CLASS: 3
PREDICTION IS CORRECT!
----------------------
IMAGE 2:
PREDICTED CLASS: 8, ACTUAL CLASS: 8
PREDICTION IS CORRECT!
----------------------
IMAGE 3:
PREDICTED CLASS: 8, ACTUAL CLASS: 8
PREDICTION IS CORRECT!
----------------------
IMAGE 4:
PREDICTED CLASS: 0, ACTUAL CLASS: 0
PREDICTION IS CORRECT!
----------------------
```

```
import matplotlib.pyplot as plt

# Plotting the Loss
plt.figure(figsize=(12, 4))

plt.subplot(1, 2, 1)
```

Connected to Python 3 Google Compute Engine backend (GPU)



```
import matplotlib.pyplot as plt

# Plotting the Loss
plt.figure(figsize=(12, 4))

plt.subplot(1, 2, 1)
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Loss vs. Epochs')
plt.xlabel('EPOCHS')
plt.ylabel('LOSS')
plt.legend()

# Plotting the Accuracy
plt.subplot(1, 2, 2)
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Accuracy vs. Epochs')
plt.xlabel('EPOCHS')
plt.ylabel('ACCURACY')
plt.legend()

plt.tight_layout()
plt.show()
```

✓ Connected to Python 3 Google Compute Engine backend (GPU)

YouTube Video Link: https://youtu.be/BsQi0yLABNg
GitHub Repo Link: https://github.com/Krypton0626/Bigdata/tree/main/ICP%207