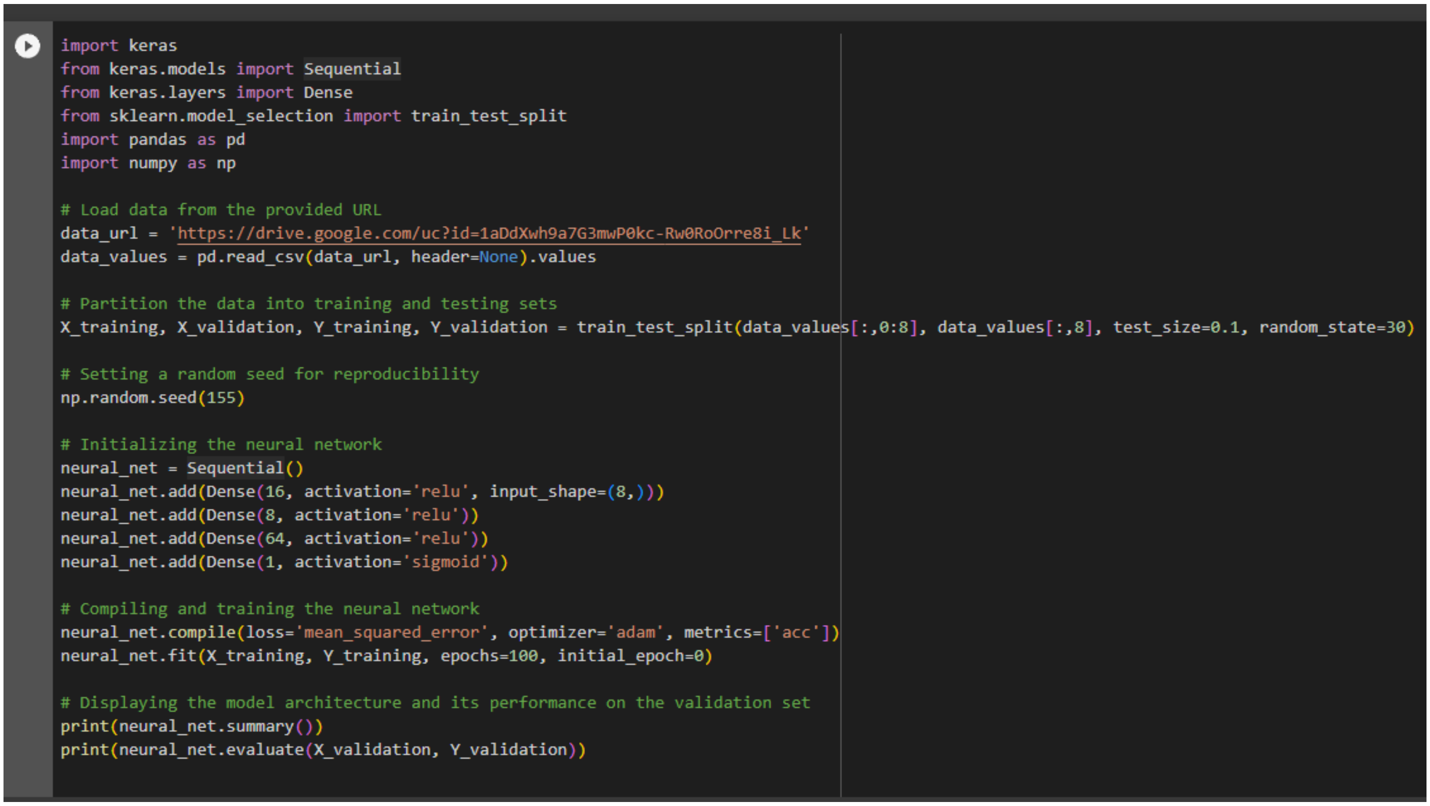
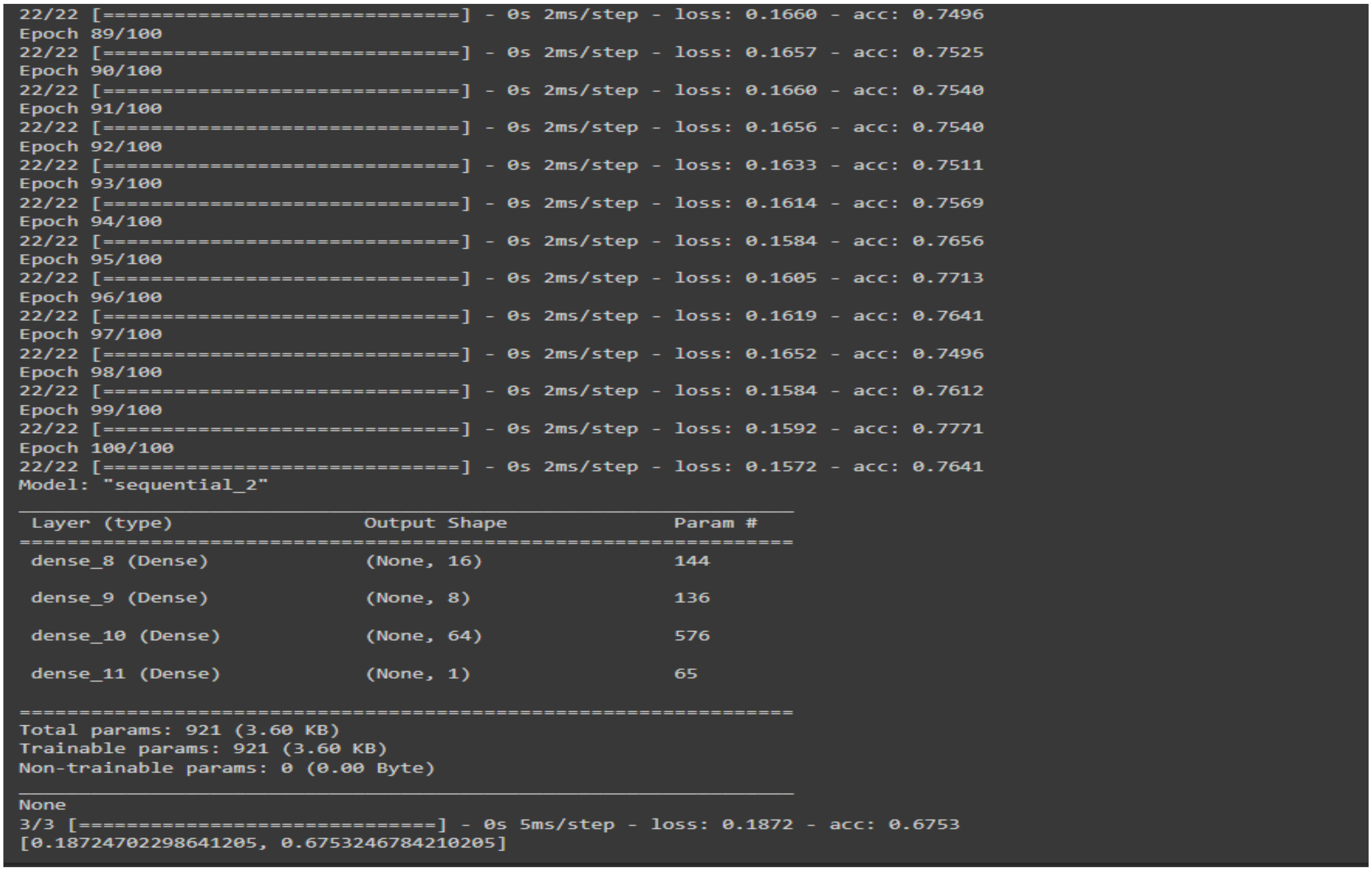
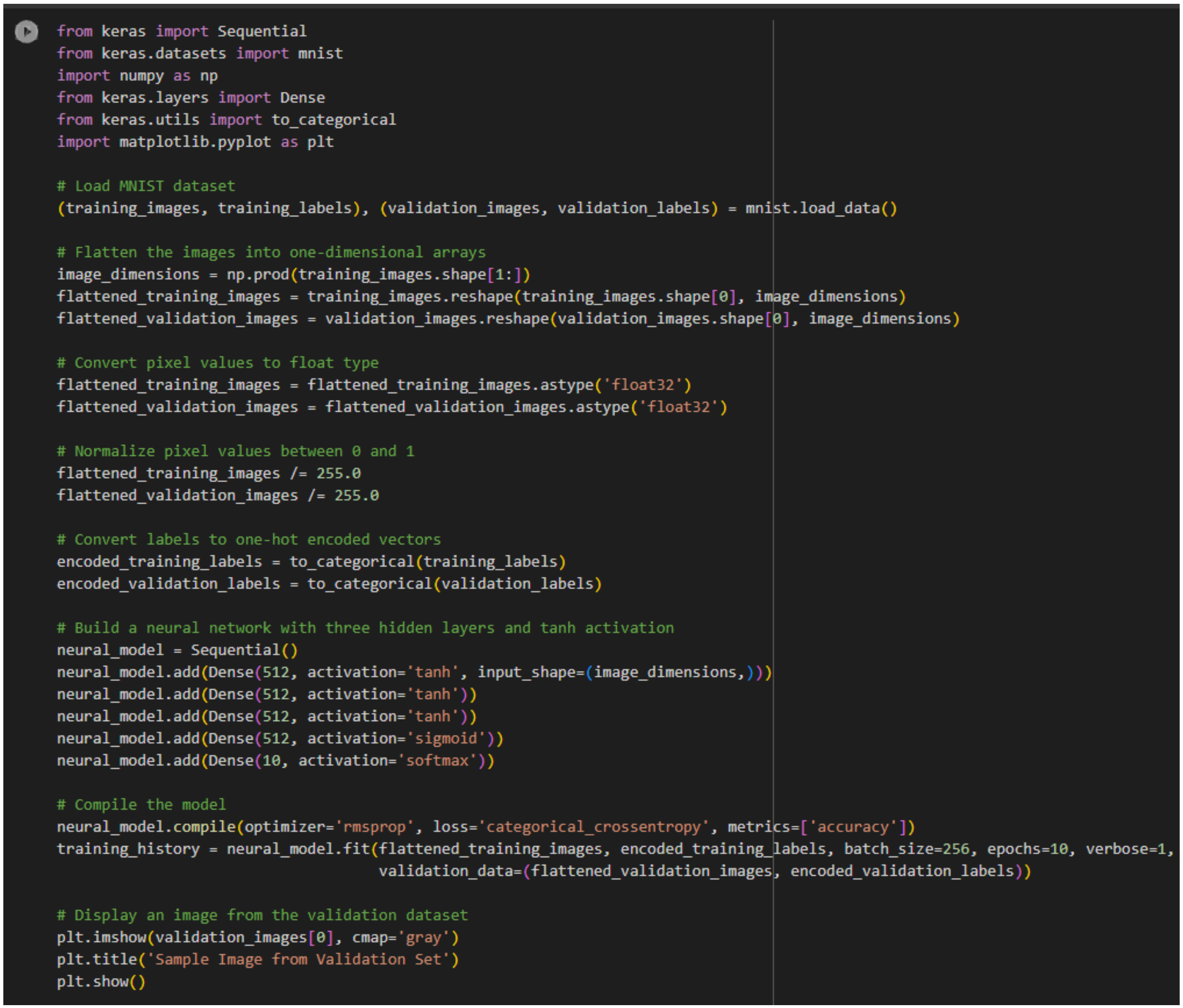


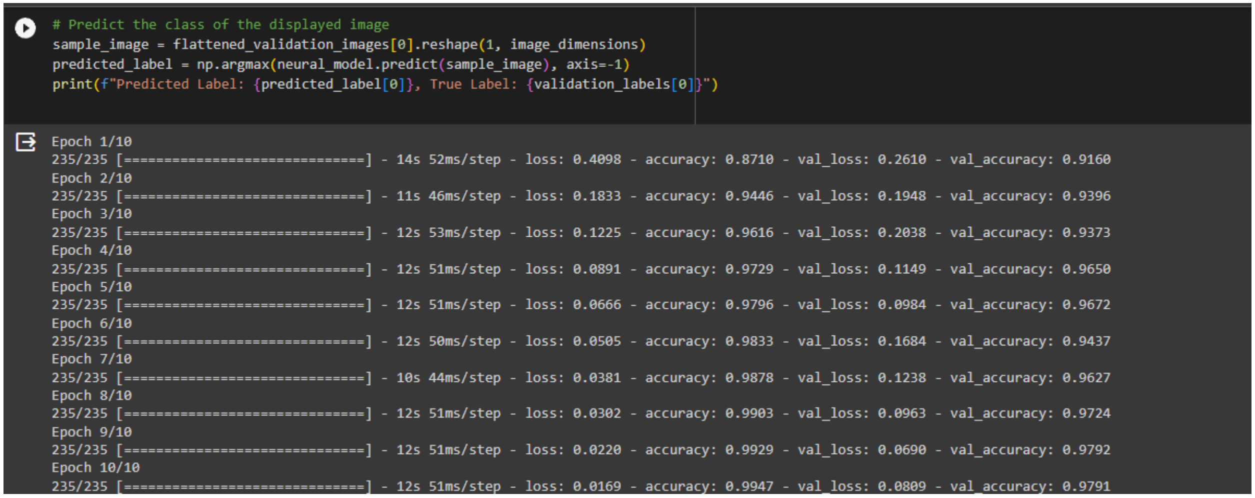
```
import keras
from keras.models import Sequential
from keras.layers import Dense
from sklearn.model_selection import train_test_split
import pandas as pd
import numpy as np

# Load data from the provided URL
data_url = 'https://drive.google.com/uc?id=1aDdXwh9a7G3mwP0kc-Rw0RoOrre8i_Lk'
data_values = pd.read_csv(data_url, header=None).values

# Partition the data into training and testing sets
X_training, X_validation, Y_training, Y_validation = train_test_split(data_values[:,0:8], data_values[:,8], test_size=0.1, random_state=30)

# Setting a random seed for reproducibility
np.random.seed(155)

# Initializing the neural network
neural_net = Sequential()
neural_net.add(Dense(16, activation='relu', input_shape=(8,)))
neural_net.add(Dense(8, activation='relu'))
neural_net.add(Dense(64, activation='relu'))
neural_net.add(Dense(1, activation='sigmoid'))

# Compiling and training the neural network
neural_net.compile(loss='mean_squared_error', optimizer='adam', metrics=['acc'])
neural_net.fit(X_training, Y_training, epochs=100, initial_epoch=0)

# Displaying the model architecture and its performance on the validation set
print(neural_net.summary())
print(neural_net.evaluate(X_validation, Y_validation))
```



```
22/22 [==============================] - 0s 2ms/step - loss: 0.1660 - acc: 0.7496
Epoch 89/100
22/22 [==============================] - 0s 2ms/step - loss: 0.1657 - acc: 0.7525
Epoch 90/100
22/22 [==============================] - 0s 2ms/step - loss: 0.1660 - acc: 0.7540
Epoch 91/100
22/22 [==============================] - 0s 2ms/step - loss: 0.1656 - acc: 0.7540
Epoch 92/100
22/22 [==============================] - 0s 2ms/step - loss: 0.1633 - acc: 0.7511
Epoch 93/100
22/22 [==============================] - 0s 2ms/step - loss: 0.1614 - acc: 0.7569
Epoch 94/100
22/22 [==============================] - 0s 2ms/step - loss: 0.1584 - acc: 0.7656
Epoch 95/100
22/22 [==============================] - 0s 2ms/step - loss: 0.1605 - acc: 0.7713
Epoch 96/100
22/22 [==============================] - 0s 2ms/step - loss: 0.1619 - acc: 0.7641
Epoch 97/100
22/22 [==============================] - 0s 2ms/step - loss: 0.1652 - acc: 0.7496
Epoch 98/100
22/22 [==============================] - 0s 2ms/step - loss: 0.1584 - acc: 0.7612
Epoch 99/100
22/22 [==============================] - 0s 2ms/step - loss: 0.1592 - acc: 0.7771
Epoch 100/100
22/22 [==============================] - 0s 2ms/step - loss: 0.1572 - acc: 0.7641
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_8 (Dense)             (None, 16)                144

 dense_9 (Dense)             (None, 8)                 136

 dense_10 (Dense)            (None, 64)                576

 dense_11 (Dense)            (None, 1)                 65

=================================================================
Total params: 921 (3.60 KB)
Trainable params: 921 (3.60 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
None
3/3 [==============================] - 0s 5ms/step - loss: 0.1872 - acc: 0.6753
[0.18724702298641205, 0.6753246784210205]
```

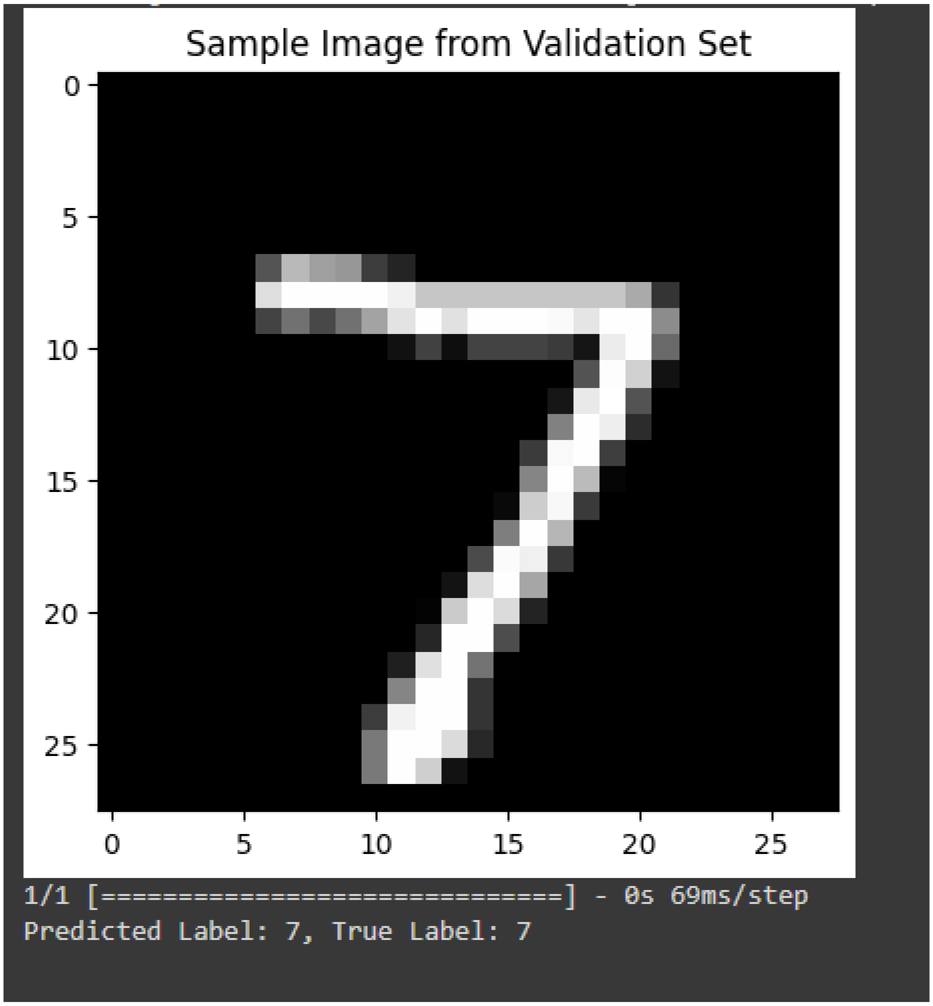


```python
from keras import Sequential
from keras.datasets import mnist
import numpy as np
from keras.layers import Dense
from keras.utils import to_categorical
import matplotlib.pyplot as plt

# Load MNIST dataset
(training_images, training_labels), (validation_images, validation_labels) = mnist.load_data()

# Flatten the images into one-dimensional arrays
image_dimensions = np.prod(training_images.shape[1:])
flattened_training_images = training_images.reshape(training_images.shape[0], image_dimensions)
flattened_validation_images = validation_images.reshape(validation_images.shape[0], image_dimensions)

# Convert pixel values to float type
flattened_training_images = flattened_training_images.astype('float32')
flattened_validation_images = flattened_validation_images.astype('float32')

# Normalize pixel values between 0 and 1
flattened_training_images /= 255.0
flattened_validation_images /= 255.0

# Convert labels to one-hot encoded vectors
encoded_training_labels = to_categorical(training_labels)
encoded_validation_labels = to_categorical(validation_labels)

# Build a neural network with three hidden layers and tanh activation
neural_model = Sequential()
neural_model.add(Dense(512, activation='tanh', input_shape=(image_dimensions,)))
neural_model.add(Dense(512, activation='tanh'))
neural_model.add(Dense(512, activation='tanh'))
neural_model.add(Dense(512, activation='sigmoid'))
neural_model.add(Dense(10, activation='softmax'))

# Compile the model
neural_model.compile(optimizer='rmsprop', loss='categorical_crossentropy', metrics=['accuracy'])
training_history = neural_model.fit(flattened_training_images, encoded_training_labels, batch_size=256, epochs=10, verbose=1,
                                    validation_data=(flattened_validation_images, encoded_validation_labels))

# Display an image from the validation dataset
plt.imshow(validation_images[0], cmap='gray')
plt.title('Sample Image from Validation Set')
plt.show()
```



```python
# Predict the class of the displayed image
sample_image = flattened_validation_images[0].reshape(1, image_dimensions)
predicted_label = np.argmax(neural_model.predict(sample_image), axis=-1)
print(f"Predicted Label: {predicted_label[0]}, True Label: {validation_labels[0]}")
```

```
Epoch 1/10
235/235 [==============================] - 14s 52ms/step - loss: 0.4098 - accuracy: 0.8710 - val_loss: 0.2610 - val_accuracy: 0.9160
Epoch 2/10
235/235 [==============================] - 11s 46ms/step - loss: 0.1833 - accuracy: 0.9446 - val_loss: 0.1948 - val_accuracy: 0.9396
Epoch 3/10
235/235 [==============================] - 12s 53ms/step - loss: 0.1225 - accuracy: 0.9616 - val_loss: 0.2038 - val_accuracy: 0.9373
Epoch 4/10
235/235 [==============================] - 12s 51ms/step - loss: 0.0891 - accuracy: 0.9729 - val_loss: 0.1149 - val_accuracy: 0.9650
Epoch 5/10
235/235 [==============================] - 12s 51ms/step - loss: 0.0666 - accuracy: 0.9796 - val_loss: 0.0984 - val_accuracy: 0.9672
Epoch 6/10
235/235 [==============================] - 12s 50ms/step - loss: 0.0505 - accuracy: 0.9833 - val_loss: 0.1684 - val_accuracy: 0.9437
Epoch 7/10
235/235 [==============================] - 10s 44ms/step - loss: 0.0381 - accuracy: 0.9878 - val_loss: 0.1238 - val_accuracy: 0.9627
Epoch 8/10
235/235 [==============================] - 12s 51ms/step - loss: 0.0302 - accuracy: 0.9903 - val_loss: 0.0963 - val_accuracy: 0.9724
Epoch 9/10
235/235 [==============================] - 12s 51ms/step - loss: 0.0220 - accuracy: 0.9929 - val_loss: 0.0690 - val_accuracy: 0.9792
Epoch 10/10
235/235 [==============================] - 12s 51ms/step - loss: 0.0169 - accuracy: 0.9947 - val_loss: 0.0809 - val_accuracy: 0.9791
```



```
1/1 [==============================] - 0s 69ms/step
Predicted Label: 7, True Label: 7
```

Github: https://github.com/SXP36810/BigData

Youtube: https://youtu.be/SqMOlHevK3M