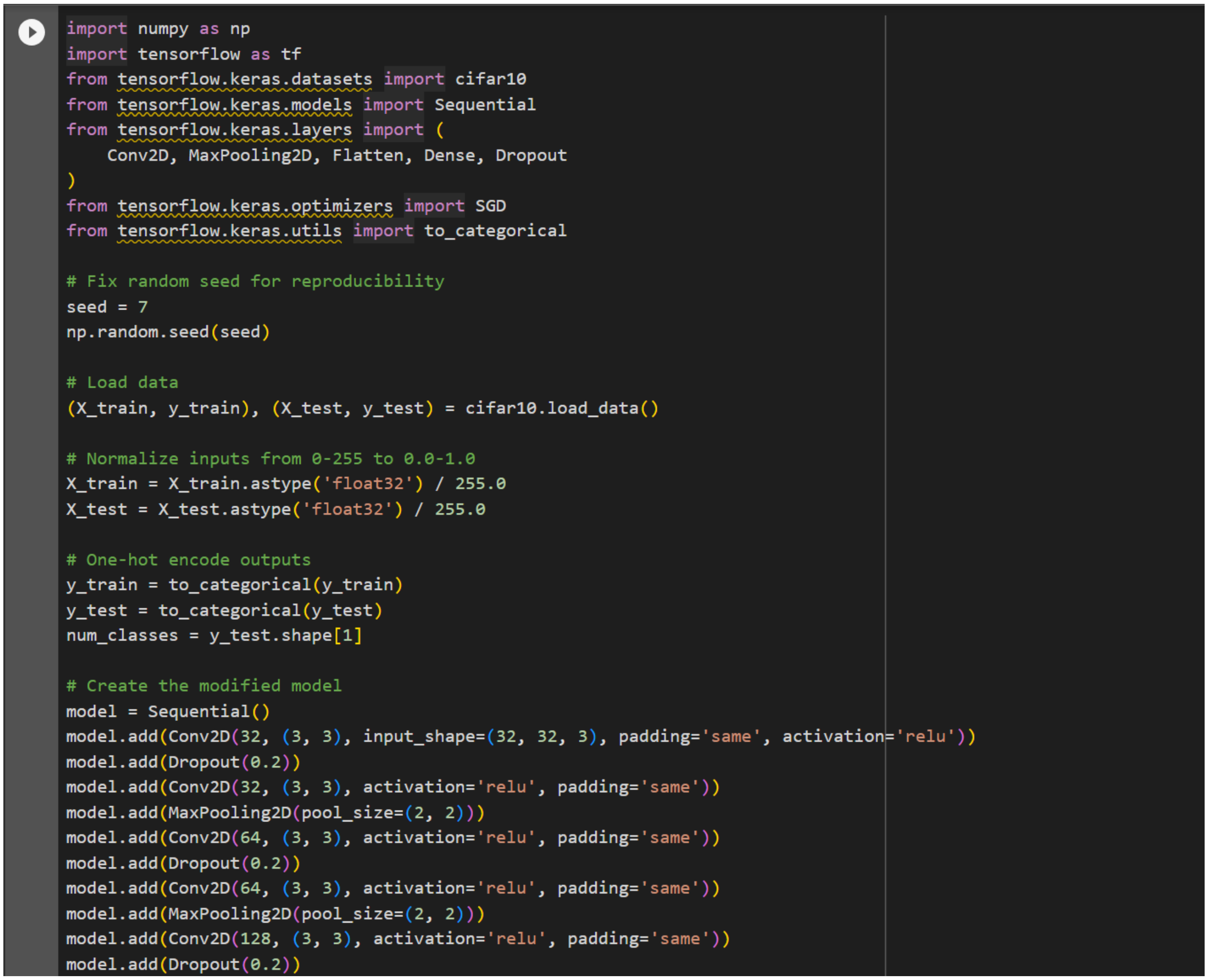
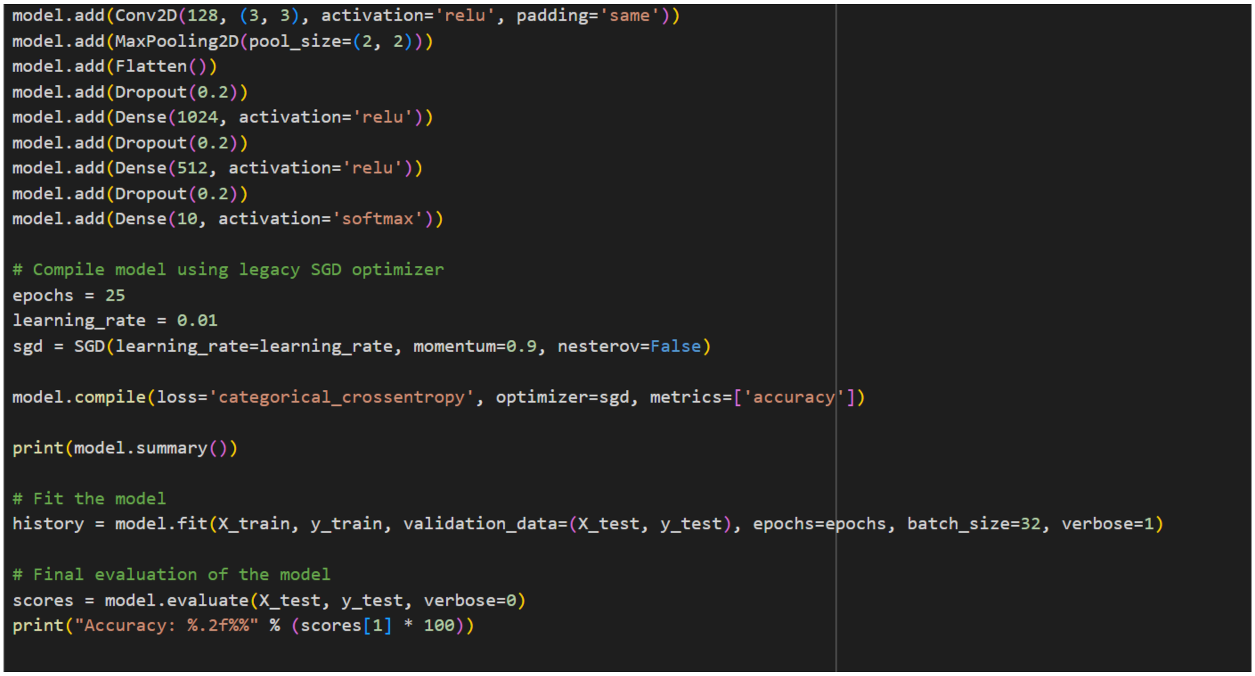
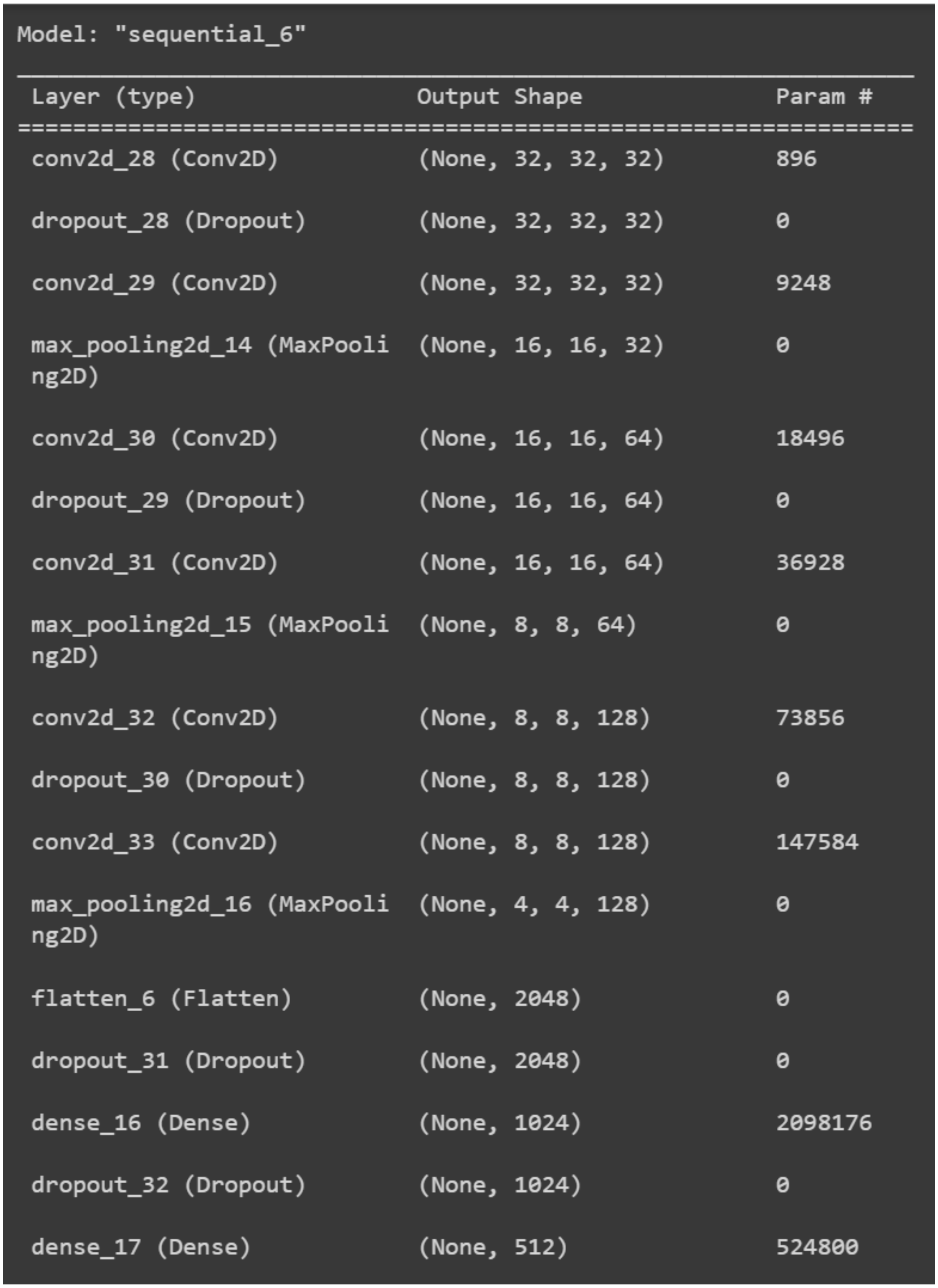


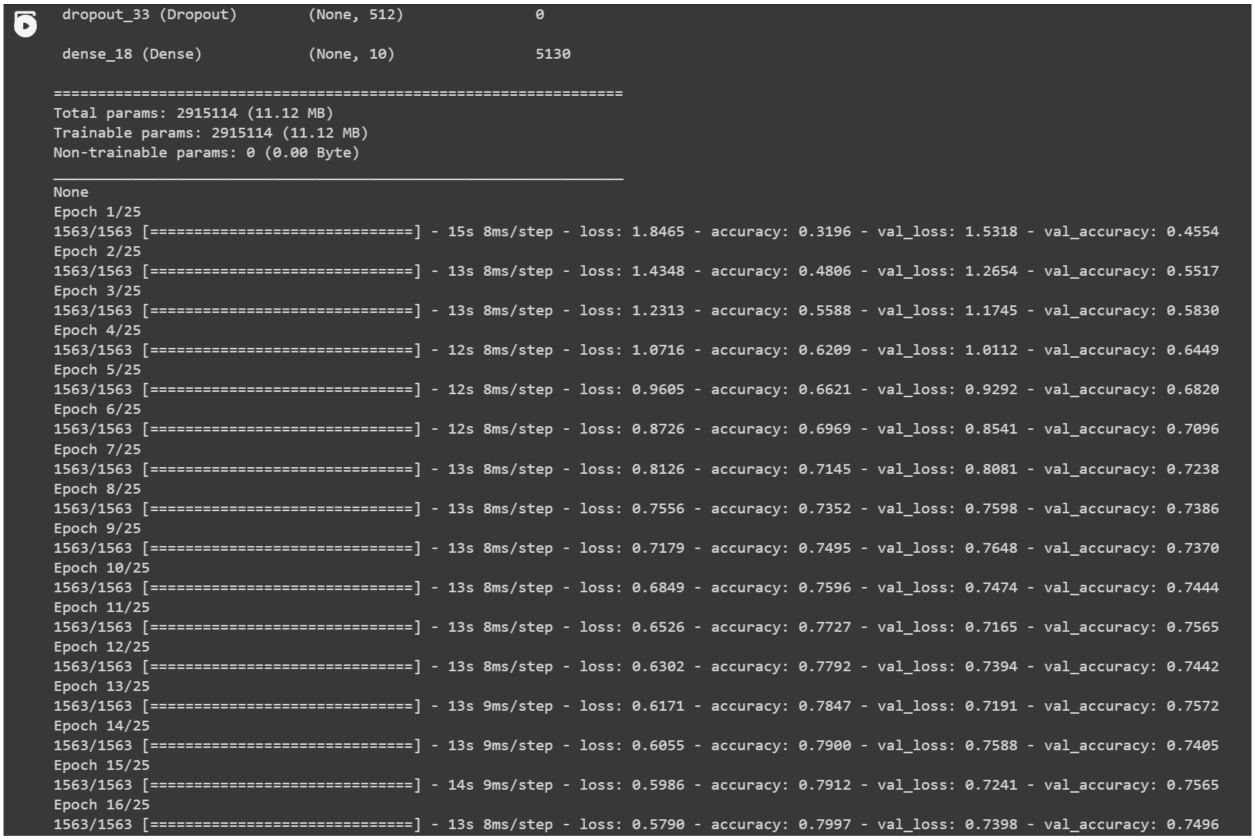
```
import numpy as np
import tensorflow as tf
from tensorflow.keras.datasets import cifar10
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import (
    Conv2D, MaxPooling2D, Flatten, Dense, Dropout
)
from tensorflow.keras.optimizers import SGD
from tensorflow.keras.utils 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') / 255.0
X_test = X_test.astype('float32') / 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()
model.add(Conv2D(32, (3, 3), input_shape=(32, 32, 3), padding='same', activation='relu'))
model.add(Dropout(0.2))
model.add(Conv2D(32, (3, 3), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu', padding='same'))
model.add(Dropout(0.2))
model.add(Conv2D(64, (3, 3), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(128, (3, 3), activation='relu', padding='same'))
model.add(Dropout(0.2))
```



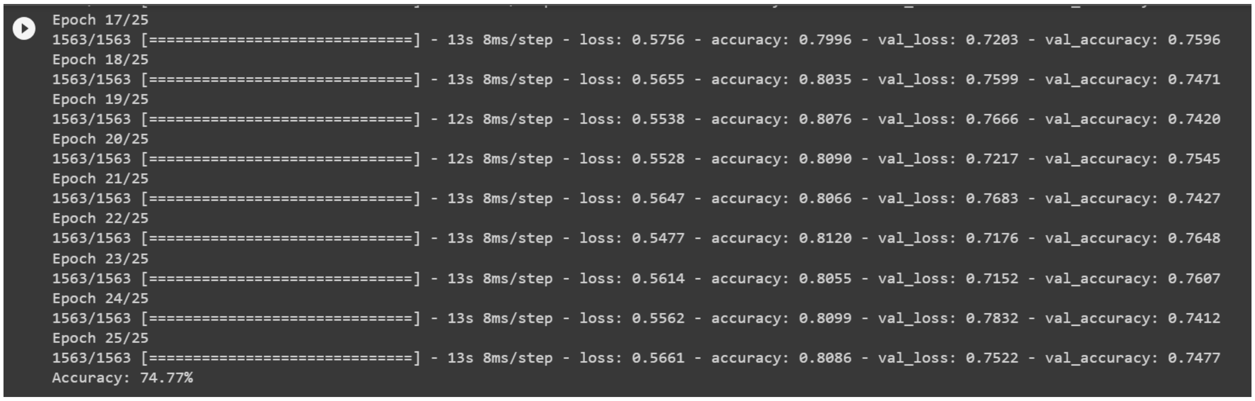
```
model.add(Conv2D(128, (3, 3), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dropout(0.2))
model.add(Dense(1024, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(10, activation='softmax'))

# Compile model using legacy SGD optimizer
epochs = 25
learning_rate = 0.01
sgd = SGD(learning_rate=learning_rate, momentum=0.9, nesterov=False)

model.compile(loss='categorical_crossentropy', optimizer=sgd, metrics=['accuracy'])

print(model.summary())

# Fit the model
history = model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=epochs, batch_size=32, verbose=1)

# Final evaluation of the model
scores = model.evaluate(X_test, y_test, verbose=0)
print("Accuracy: %.2f%%" % (scores[1] * 100))
```



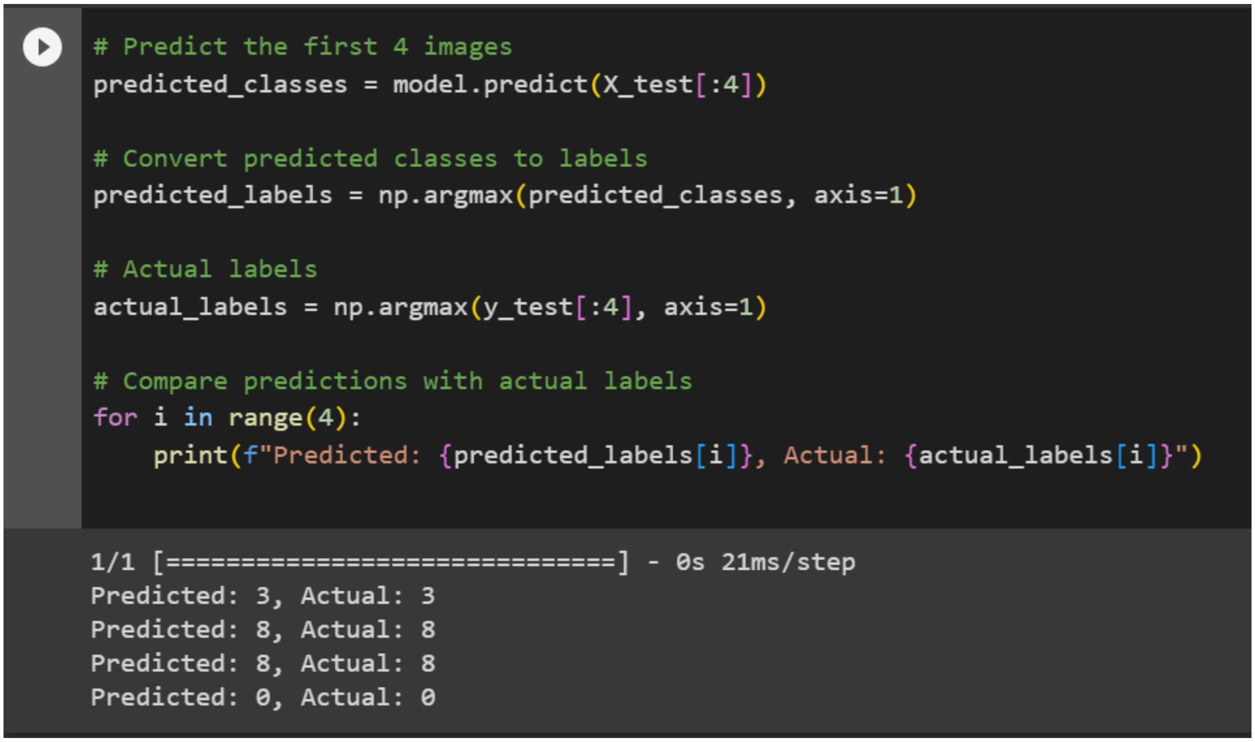
```
Model: "sequential_6"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_28 (Conv2D)          (None, 32, 32, 32)        896

 dropout_28 (Dropout)        (None, 32, 32, 32)        0

 conv2d_29 (Conv2D)          (None, 32, 32, 32)        9248

 max_pooling2d_14 (MaxPooli  (None, 16, 16, 32)        0
 ng2D)

 conv2d_30 (Conv2D)          (None, 16, 16, 64)        18496

 dropout_29 (Dropout)        (None, 16, 16, 64)        0

 conv2d_31 (Conv2D)          (None, 16, 16, 64)        36928

 max_pooling2d_15 (MaxPooli  (None, 8, 8, 64)          0
 ng2D)

 conv2d_32 (Conv2D)          (None, 8, 8, 128)         73856

 dropout_30 (Dropout)        (None, 8, 8, 128)         0

 conv2d_33 (Conv2D)          (None, 8, 8, 128)         147584

 max_pooling2d_16 (MaxPooli  (None, 4, 4, 128)         0
 ng2D)

 flatten_6 (Flatten)         (None, 2048)              0

 dropout_31 (Dropout)        (None, 2048)              0

 dense_16 (Dense)            (None, 1024)              2098176

 dropout_32 (Dropout)        (None, 1024)              0

 dense_17 (Dense)            (None, 512)               524800
```



```
 dropout_33 (Dropout)        (None, 512)               0

 dense_18 (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 [==============================] - 15s 8ms/step - loss: 1.8465 - accuracy: 0.3196 - val_loss: 1.5318 - val_accuracy: 0.4554
Epoch 2/25
1563/1563 [==============================] - 13s 8ms/step - loss: 1.4348 - accuracy: 0.4806 - val_loss: 1.2654 - val_accuracy: 0.5517
Epoch 3/25
1563/1563 [==============================] - 13s 8ms/step - loss: 1.2313 - accuracy: 0.5588 - val_loss: 1.1745 - val_accuracy: 0.5830
Epoch 4/25
1563/1563 [==============================] - 12s 8ms/step - loss: 1.0716 - accuracy: 0.6209 - val_loss: 1.0112 - val_accuracy: 0.6449
Epoch 5/25
1563/1563 [==============================] - 12s 8ms/step - loss: 0.9605 - accuracy: 0.6621 - val_loss: 0.9292 - val_accuracy: 0.6820
Epoch 6/25
1563/1563 [==============================] - 12s 8ms/step - loss: 0.8726 - accuracy: 0.6969 - val_loss: 0.8541 - val_accuracy: 0.7096
Epoch 7/25
1563/1563 [==============================] - 13s 8ms/step - loss: 0.8126 - accuracy: 0.7145 - val_loss: 0.8081 - val_accuracy: 0.7238
Epoch 8/25
1563/1563 [==============================] - 13s 8ms/step - loss: 0.7556 - accuracy: 0.7352 - val_loss: 0.7598 - val_accuracy: 0.7386
Epoch 9/25
1563/1563 [==============================] - 13s 8ms/step - loss: 0.7179 - accuracy: 0.7495 - val_loss: 0.7648 - val_accuracy: 0.7370
Epoch 10/25
1563/1563 [==============================] - 13s 8ms/step - loss: 0.6849 - accuracy: 0.7596 - val_loss: 0.7474 - val_accuracy: 0.7444
Epoch 11/25
1563/1563 [==============================] - 13s 8ms/step - loss: 0.6526 - accuracy: 0.7727 - val_loss: 0.7165 - val_accuracy: 0.7565
Epoch 12/25
1563/1563 [==============================] - 13s 8ms/step - loss: 0.6302 - accuracy: 0.7792 - val_loss: 0.7394 - val_accuracy: 0.7442
Epoch 13/25
1563/1563 [==============================] - 13s 9ms/step - loss: 0.6171 - accuracy: 0.7847 - val_loss: 0.7191 - val_accuracy: 0.7572
Epoch 14/25
1563/1563 [==============================] - 13s 9ms/step - loss: 0.6055 - accuracy: 0.7900 - val_loss: 0.7588 - val_accuracy: 0.7405
Epoch 15/25
1563/1563 [==============================] - 14s 9ms/step - loss: 0.5986 - accuracy: 0.7912 - val_loss: 0.7241 - val_accuracy: 0.7565
Epoch 16/25
1563/1563 [==============================] - 13s 8ms/step - loss: 0.5790 - accuracy: 0.7997 - val_loss: 0.7398 - val_accuracy: 0.7496
```

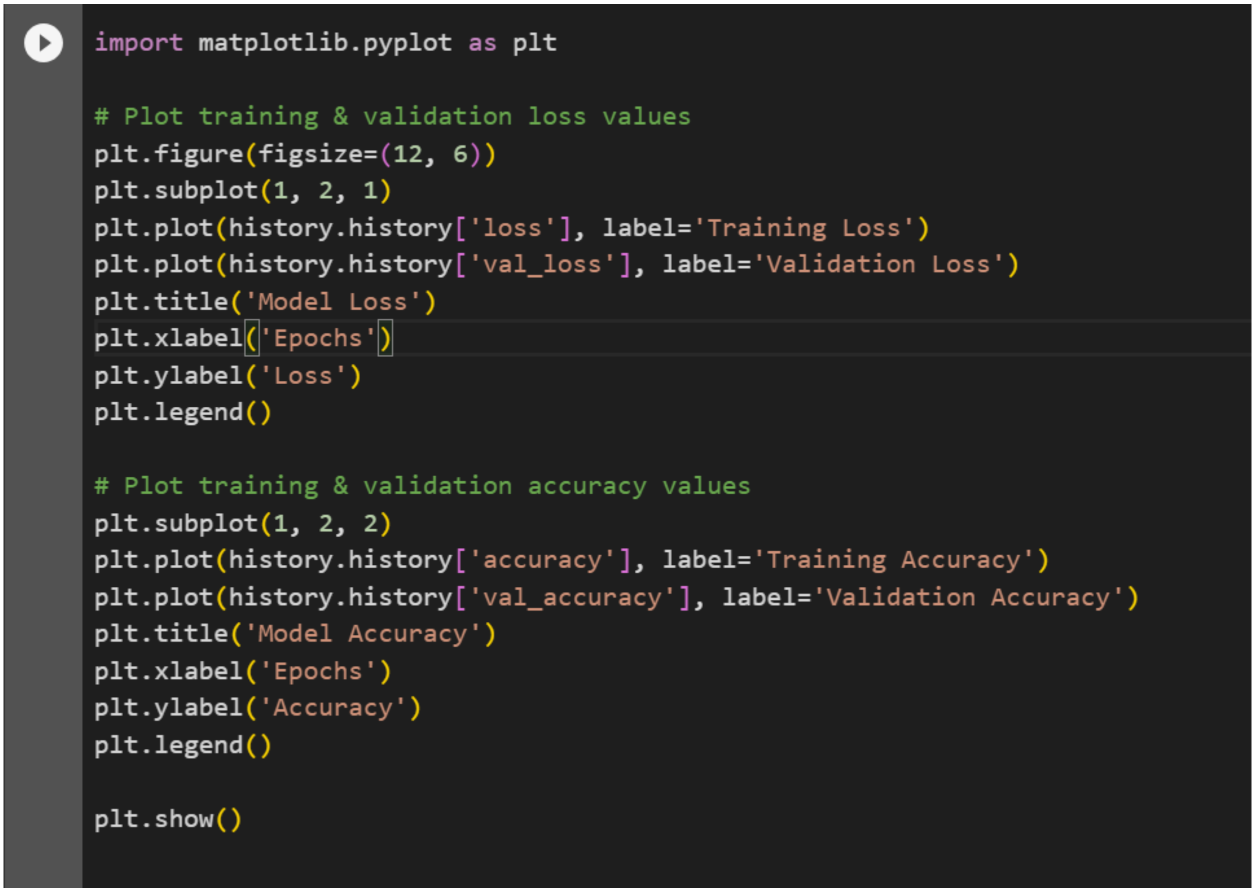


```
Epoch 17/25
1563/1563 [==============================] - 13s 8ms/step - loss: 0.5756 - accuracy: 0.7996 - val_loss: 0.7203 - val_accuracy: 0.7596
Epoch 18/25
1563/1563 [==============================] - 13s 8ms/step - loss: 0.5655 - accuracy: 0.8035 - val_loss: 0.7599 - val_accuracy: 0.7471
Epoch 19/25
1563/1563 [==============================] - 12s 8ms/step - loss: 0.5538 - accuracy: 0.8076 - val_loss: 0.7666 - val_accuracy: 0.7420
Epoch 20/25
1563/1563 [==============================] - 12s 8ms/step - loss: 0.5528 - accuracy: 0.8090 - val_loss: 0.7217 - val_accuracy: 0.7545
Epoch 21/25
1563/1563 [==============================] - 13s 8ms/step - loss: 0.5647 - accuracy: 0.8066 - val_loss: 0.7683 - val_accuracy: 0.7427
Epoch 22/25
1563/1563 [==============================] - 13s 8ms/step - loss: 0.5477 - accuracy: 0.8120 - val_loss: 0.7176 - val_accuracy: 0.7648
Epoch 23/25
1563/1563 [==============================] - 13s 8ms/step - loss: 0.5614 - accuracy: 0.8055 - val_loss: 0.7152 - val_accuracy: 0.7607
Epoch 24/25
1563/1563 [==============================] - 13s 8ms/step - loss: 0.5562 - accuracy: 0.8099 - val_loss: 0.7832 - val_accuracy: 0.7412
Epoch 25/25
1563/1563 [==============================] - 13s 8ms/step - loss: 0.5661 - accuracy: 0.8086 - val_loss: 0.7522 - val_accuracy: 0.7477
Accuracy: 74.77%
```

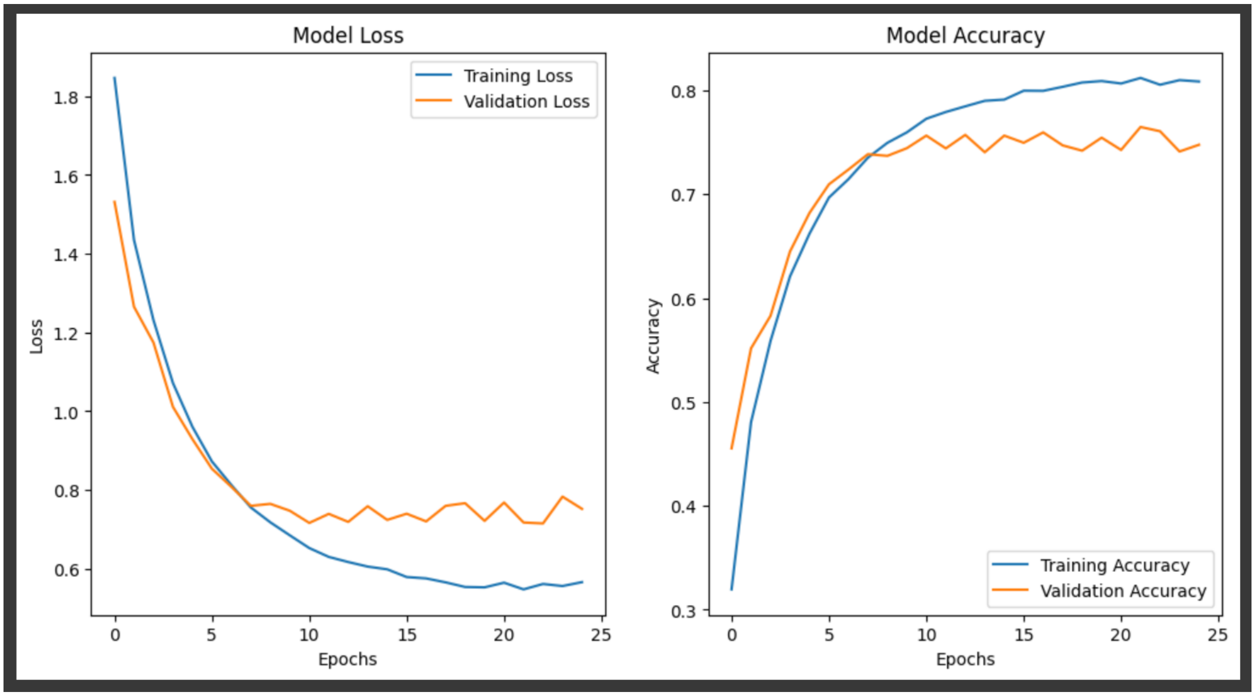


```
# Predict the first 4 images
predicted_classes = model.predict(X_test[:4])

# Convert predicted classes to labels
predicted_labels = np.argmax(predicted_classes, axis=1)

# Actual labels
actual_labels = np.argmax(y_test[:4], axis=1)

# Compare predictions with actual labels
for i in range(4):
    print(f"Predicted: {predicted_labels[i]}, Actual: {actual_labels[i]}")
```

```
1/1 [==============================] - 0s 21ms/step
Predicted: 3, Actual: 3
Predicted: 8, Actual: 8
Predicted: 8, Actual: 8
Predicted: 0, Actual: 0
```



```
import matplotlib.pyplot as plt

# Plot training & validation loss values
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Model Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

# Plot training & validation accuracy values
plt.subplot(1, 2, 2)
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Model Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

plt.show()
```


Model Loss
Training Loss
Validation Loss
Loss
1.8
1.6
1.4
1.2
1.0
0.8
0.6
0
5
10
15
20
25
Epochs
Model Accuracy
Accuracy
0.8
0.7
0.6
0.5
0.4
0.3
Training Accuracy
Validation Accuracy
0
5
10
15
20
25
Epochs