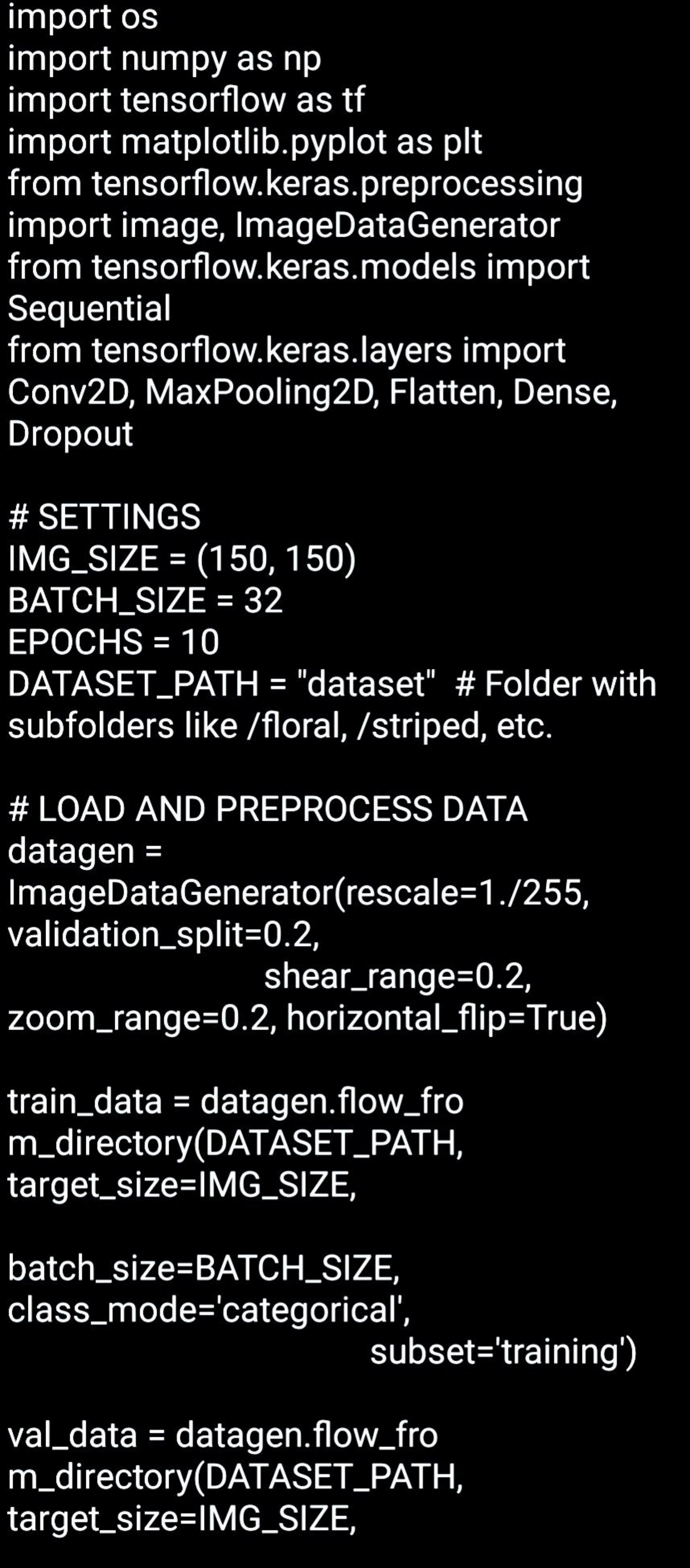
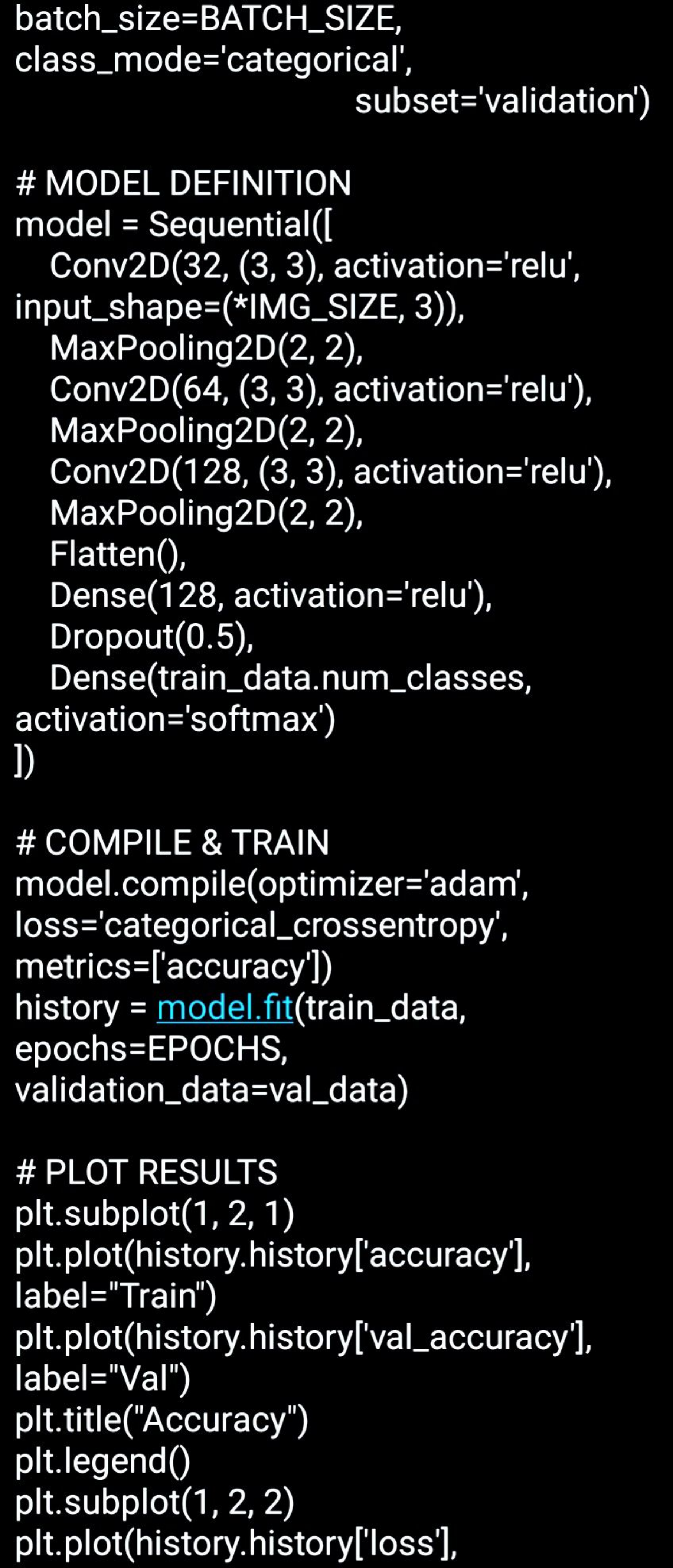
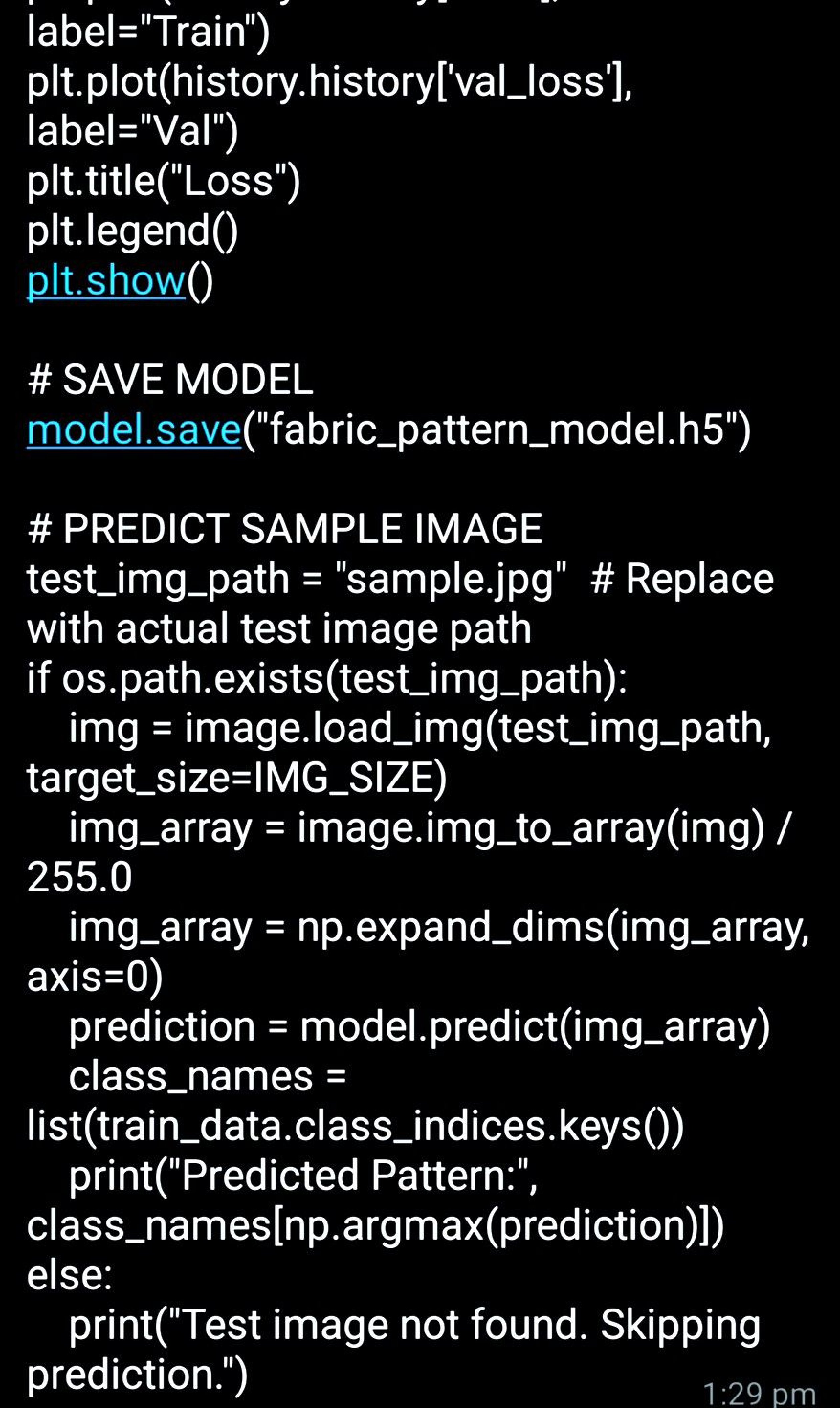


```
import os
import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt
from tensorflow.keras.preprocessing
import image, ImageDataGenerator
from tensorflow.keras.models import
Sequential
from tensorflow.keras.layers import
Conv2D, MaxPooling2D, Flatten, Dense,
Dropout

# SETTINGS
IMG_SIZE = (150, 150)
BATCH_SIZE = 32
EPOCHS = 10
DATASET_PATH = "dataset"  # Folder with
subfolders like /floral, /striped, etc.

# LOAD AND PREPROCESS DATA
datagen =
ImageDataGenerator(rescale=1./255,
validation_split=0.2,
                              shear_range=0.2,
zoom_range=0.2, horizontal_flip=True)

train_data = datagen.flow_fro
m_directory(DATASET_PATH,
target_size=IMG_SIZE,

batch_size=BATCH_SIZE,
class_mode='categorical',
                                          subset='training')

val_data = datagen.flow_fro
m_directory(DATASET_PATH,
target_size=IMG_SIZE,
```



```
batch_size=BATCH_SIZE,
class_mode='categorical',
                                        subset='validation')

# MODEL DEFINITION
model = Sequential([
    Conv2D(32, (3, 3), activation='relu',
input_shape=(*IMG_SIZE, 3)),
    MaxPooling2D(2, 2),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D(2, 2),
    Conv2D(128, (3, 3), activation='relu'),
    MaxPooling2D(2, 2),
    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(train_data.num_classes,
activation='softmax')
])

# COMPILE & TRAIN
model.compile(optimizer='adam',
loss='categorical_crossentropy',
metrics=['accuracy'])
history = model.fit(train_data,
epochs=EPOCHS,
validation_data=val_data)

# PLOT RESULTS
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'],
label="Train")
plt.plot(history.history['val_accuracy'],
label="Val")
plt.title("Accuracy")
plt.legend()
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'],
```



```
label="Train")
plt.plot(history.history['val_loss'],
label="Val")
plt.title("Loss")
plt.legend()
plt.show()

# SAVE MODEL
model.save("fabric_pattern_model.h5")

# PREDICT SAMPLE IMAGE
test_img_path = "sample.jpg"  # Replace
with actual test image path
if os.path.exists(test_img_path):
    img = image.load_img(test_img_path,
target_size=IMG_SIZE)
    img_array = image.img_to_array(img) /
255.0
    img_array = np.expand_dims(img_array,
axis=0)
    prediction = model.predict(img_array)
    class_names =
list(train_data.class_indices.keys())
    print("Predicted Pattern:",
class_names[np.argmax(prediction)])
else:
    print("Test image not found. Skipping
prediction.")
```