import tensorflow as tf

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout

import matplotlib.pyplot as plt

dataset/

train/

gesture1/

gesture2/

...

validation/

gesture1/

gesture2/

...

# Define paths to the dataset directories

train\_dir = 'dataset/train'

validation\_dir = 'dataset/validation'

# ImageDataGenerator for data augmentation and normalization

train\_datagen = ImageDataGenerator(

rescale=1./255,

rotation\_range=20,

width\_shift\_range=0.2,

height\_shift\_range=0.2,

shear\_range=0.2,

zoom\_range=0.2,

horizontal\_flip=True,

fill\_mode='nearest'

)

validation\_datagen = ImageDataGenerator(rescale=1./255)

# Create generators

train\_generator = train\_datagen.flow\_from\_directory(

train\_dir,

target\_size=(64, 64),

batch\_size=32,

class\_mode='categorical'

)

validation\_generator = validation\_datagen.flow\_from\_directory(

validation\_dir,

target\_size=(64, 64),

batch\_size=32,

class\_mode='categorical'

)

model = Sequential([

Conv2D(32, (3, 3), activation='relu', input\_shape=(64, 64, 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\_generator.num\_classes, activation='softmax')

])

model.compile(optimizer='adam', loss='categorical\_crossentropy', metrics=['accuracy'])

model.summary()

history = model.fit(

train\_generator,

epochs=25,

validation\_data=validation\_generator

)

# Plot training and validation accuracy and loss

acc = history.history['accuracy']

val\_acc = history.history['val\_accuracy']

loss = history.history['loss']

val\_loss = history.history['val\_loss']

epochs\_range = range(25)

plt.figure(figsize=(12, 4))

plt.subplot(1, 2, 1)

plt.plot(epochs\_range, acc, label='Training Accuracy')

plt.plot(epochs\_range, val\_acc, label='Validation Accuracy')

plt.legend(loc='lower right')

plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)

plt.plot(epochs\_range, loss, label='Training Loss')

plt.plot(epochs\_range, val\_loss, label='Validation Loss')

plt.legend(loc='upper right')

plt.title('Training and Validation Loss')

plt.show()

model.save('hand\_gesture\_model.h5')