```
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import tensorflow
from tensorflow import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Conv2D , MaxPool2D , Flatten , Dropout , BatchNormalization
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report,confusion_matrix
from tensorflow.keras.callbacks import ReduceLROnPlateau
import cv2 as cv
import h5py
```

```
2024-07-28 13:48:10.490061: E external/local_xla/xla/stream_executor/cuda/cuda_dnn.cc:9261] Unable to register cuDNN factory: Attempting
2024-07-28 13:48:10.490181: E external/local_xla/xla/stream_executor/cuda/cuda_fft.cc:607] Unable to register cuFFT factory: Attempting
2024-07-28 13:48:10.649342: E external/local_xla/xla/stream_executor/cuda/cuda_blas.cc:1515] Unable to register cuBLAS factory: Attempti
```

Dataset link - https://www.kaggle.com/datasets/aryarishabh/hand-gesture-recognition-dataset/code

```
labels = ['call_me','fingers_crossed','up','okay','paper','rock','rock_on','scissor','peace','thumbs']
x_signs = []
y_signs = []
path = '../input/hand-gesture-recognition/HandGesture/images'
for i in os.listdir(path):
    for j in os.listdir(path + "/" + i):
        img =  cv.imread(path + "/" + i + '/' + j,0)
        img =  cv.resize(img,(64,64),interpolation = cv.INTER_AREA)
        img = np.array(img)
        x_signs.append(img)
        y_signs.append(labels.index(i))
```

```
x = np.array(x_signs)
x.shape
```

```
(5243, 64, 64)
```

```
X = np.array(x_signs)
Y = np.array(y_signs)
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(X, Y, train_size=0.75, random_state = 42)
```

```
x_train.shape
```

```
(3932, 64, 64)
```

```
y_train.shape
```

```
(3932,)
```

```
x_train[11].shape
```

```
(64, 64)
```

```
train_x = []
train_y = []
for i in range(len(x_train)):
    _, bw_image = cv.threshold(x_train[i], 120, 255, cv.THRESH_BINARY)
    y = y_train[i]
    # orginal image
    train_x.append(bw_image)
    train_y.append(y)

    #rotate 100 degree
    train_x.append(cv.flip(bw_image,1))
    train_y.append(y)
train_x  = np.array(train_x)
train_y  = np.array(train_y)
```

```
test_x = []
test_y = []
for i in range(len(x_test)):
    _, bw_image = cv.threshold(x_test[i], 120, 255, cv.THRESH_BINARY)
    y = y_test[i]
    # orginal image
    test_x.append(bw_image)
    test_y.append(y)
    #rotate 90 degree
    test_x.append(cv.flip(bw_image,1))
    test_y.append(y)
test_x = np.array(test_x)
test_y = np.array(test_y)
```

```
from sklearn.preprocessing import LabelBinarizer
```

```
label_binarizer = LabelBinarizer()
```

```
y_train = label_binarizer.fit_transform(train_y)
y_test = label_binarizer.fit_transform(test_y)
```

```
y_train[:7]
```

```
array([[0, 0, 0, 1, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 1, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 1],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 1],
       [0, 0, 0, 0, 0, 0, 1, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 1, 0, 0, 0],
       [0, 0, 0, 0, 1, 0, 0, 0, 0, 0]])
```

```
x_train = train_x.reshape(-1,64,64,1)
x_test = test_x.reshape(-1,64,64,1)
```

```
x_train.shape
```

```
(7864, 64, 64, 1)
```

```
data_generator = ImageDataGenerator(
    rotation_range = 0.1,
    zoom_range = 0.1,
    width_shift_range=0.1,
    height_shift_range=0.1
)
```

```
data_generator.fit(x_train)
```

```
len(labels)
```

```
10
```

```
model = Sequential()
#first layer
model.add(Conv2D(75,(3,3),strides=1,padding='same',activation='relu',input_shape = (64,64,1)))
model.add(BatchNormalization())
model.add(MaxPool2D((2,2),strides=2, padding='same'))
#second layer
model.add(Conv2D(50,(3,3),strides=1,padding='same',activation='relu'))
model.add(Dropout(0.2))
model.add(BatchNormalization())
model.add(MaxPool2D((2,2),strides=2, padding='same'))
#third layer
model.add(Conv2D(25,(3,3),strides=1,padding='same',activation='relu'))
model.add(BatchNormalization())
model.add(MaxPool2D((2,2),strides=2, padding='same'))
#flatten
model.add(Flatten())
#First fully connected layer
model.add(Dense(units=512,activation='relu'))
model.add(Dropout(0.3))
#Second fully connected layer
model.add(Dense(units=10,activation='softmax'))
```

```
/opt/conda/lib/python3.10/site-packages/keras/src/layers/convolutional/base_conv.py:107: UserWarning: Do not pass an `input_shape`/`inpu
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

```
tensorflow.debugging.set_log_device_placement(True)
```

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

```
learning_rate_reduction = ReduceLROnPlateau(monitor='val_accuracy', factor=0.5,verbose=1,patience=2, min_lr=0.00001)
```

```
model.fit(data_generator.flow(x_train,y_train,batch_size = 128),epochs=10,validation_data=(x_test,y_test),callbacks=[learning_rate_reduction
```

```
Epoch 1/10
/opt/conda/lib/python3.10/site-packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:121: UserWarning: Your `PyDataset` class
  self._warn_if_super_not_called()
62/62 ━━━━━━━━━━━━━━━━━━━━ 94s 1s/step - accuracy: 0.4539 - loss: 2.0613 - val_accuracy: 0.2170 - val_loss: 6.3081 - learning_rate: 0.00
Epoch 2/10
62/62 ━━━━━━━━━━━━━━━━━━━━ 91s 1s/step - accuracy: 0.8782 - loss: 0.3791 - val_accuracy: 0.7201 - val_loss: 0.8808 - learning_rate: 0.00
Epoch 3/10
62/62 ━━━━━━━━━━━━━━━━━━━━ 91s 1s/step - accuracy: 0.9311 - loss: 0.2107 - val_accuracy: 0.9741 - val_loss: 0.0890 - learning_rate: 0.00
Epoch 4/10
62/62 ━━━━━━━━━━━━━━━━━━━━ 91s 1s/step - accuracy: 0.9500 - loss: 0.1611 - val_accuracy: 0.9802 - val_loss: 0.0734 - learning_rate: 0.00
Epoch 5/10
62/62 ━━━━━━━━━━━━━━━━━━━━ 90s 1s/step - accuracy: 0.9661 - loss: 0.1063 - val_accuracy: 0.9668 - val_loss: 0.1072 - learning_rate: 0.00
Epoch 6/10
62/62 ━━━━━━━━━━━━━━━━━━━━ 90s 1s/step - accuracy: 0.9729 - loss: 0.0823 - val_accuracy: 0.9859 - val_loss: 0.0529 - learning_rate: 0.00
Epoch 7/10
62/62 ━━━━━━━━━━━━━━━━━━━━ 91s 1s/step - accuracy: 0.9807 - loss: 0.0610 - val_accuracy: 0.9783 - val_loss: 0.0830 - learning_rate: 0.00
Epoch 8/10
62/62 ━━━━━━━━━━━━━━━━━━━━ 91s 1s/step - accuracy: 0.9797 - loss: 0.0609 - val_accuracy: 0.9893 - val_loss: 0.0436 - learning_rate: 0.00
Epoch 9/10
62/62 ━━━━━━━━━━━━━━━━━━━━ 91s 1s/step - accuracy: 0.9865 - loss: 0.0457 - val_accuracy: 0.9889 - val_loss: 0.0380 - learning_rate: 0.00
Epoch 10/10
62/62 ━━━━━━━━━━━━━━━━━━━━ 0s 1s/step - accuracy: 0.9880 - loss: 0.0390
Epoch 10: ReduceLROnPlateau reducing learning rate to 0.0005000000237487257.
62/62 ━━━━━━━━━━━━━━━━━━━━ 92s 1s/step - accuracy: 0.9880 - loss: 0.0390 - val_accuracy: 0.9889 - val_loss: 0.0425 - learning_rate: 0.00
<keras.src.callbacks.history.History at 0x7f99d04f2fe0>
```

```
model.evaluate(x_test,y_test)
```

```
82/82 ━━━━━━━━━━━━━━━━━━━━ 5s 58ms/step - accuracy: 0.9886 - loss: 0.0530
[0.042499661445617676, 0.9889397621154785]
```

```
# Save the model architecture to a JSON file
model_json = model.to_json()
with open("model-bw10.json", "w") as json_file:
    json_file.write(model_json)

# Save the model weights to an H5 file with the correct extension
model.save_weights('model-bw10.weights.h5')
```

Start coding or generate with AI.