rock-paper-scissor

April 25, 2022

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import os
from datetime import datetime

%matplotlib inline

start_time = datetime.now()
```

[2]:
```
image_dir = '/kaggle/input/rockpaperscissors'
labels = ['paper','scissors','rock']
nb = len(labels)
```

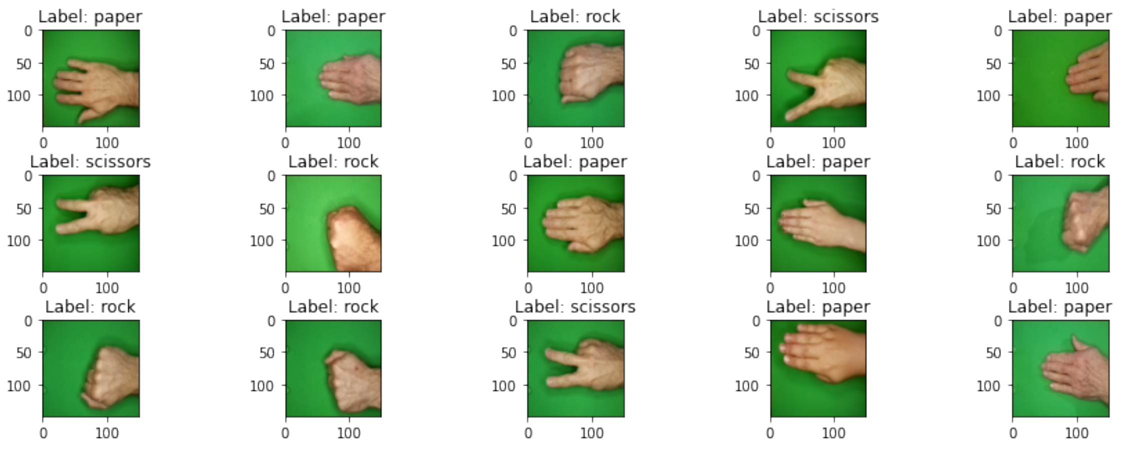
[3]:
```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.preprocessing.image import load_img, img_to_array
from tensorflow.keras.layers import Dense, Flatten, GlobalAveragePooling2D,␣
  ↪Conv2D, MaxPooling2D
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping
```

[4]:
```
#ref  https://www.kaggle.com/code/quadeer15sh/tf-keras-cnn-99-accuracy
def get_XandY(train_dir,labels):
    dataset = []
    count = 0
    for label in labels:
        folder = os.path.join(train_dir,label)
        for image in os.listdir(folder):
            img=load_img(os.path.join(folder,image), target_size=(150,150))
            img=img_to_array(img)
            img=img/255.0
            dataset.append((img,count))
        print(">>> ",label)
        count+=1
    np.random.shuffle(dataset)
    X, y = zip(*dataset)
    return np.array(X),np.array(y)
```

[5]:
```
images,label = get_XandY(image_dir,labels)
```

```
>>>  paper
>>>  scissors
>>>  rock
```

[6]:
```
plt.figure(figsize = (15 , 9))
n = 0
for i in range(15):
    n+=1
    plt.subplot(5 , 5, n)
    plt.subplots_adjust(hspace = 0.5 , wspace = 0.3)
    plt.imshow(images[i])
    plt.title(f'Label: {labels[label[i]]}')
```



[7]:
```
np.unique(label,return_counts=True)
```

[7]:
```
(array([0, 1, 2]), array([712, 750, 726]))
```

[8]:
```
from sklearn.model_selection import train_test_split

xtrain,xtest,ytrain,ytest =␣
 ↪train_test_split(images,label,stratify=label,random_state=42,test_size=0.25)

print(f"Train length:{len(xtrain)} \n Test length: {len(xtest)}")
```

```
Train length:1641
 Test length: 547
```

[9]:
```python
datagen = ImageDataGenerator(horizontal_flip=True,
                             vertical_flip=True,
                             rotation_range=20,
                             zoom_range=0.2,
                             width_shift_range = 0.2,
                             height_shift_range = 0.2,
                             shear_range=0.1,
                             fill_mode="nearest")


datagen.fit(xtrain)
```

[10]:
```python
model = Sequential()
model.add(Conv2D(32, (3,3), input_shape=(150,150,3), activation='relu'))
model.add(MaxPooling2D(2,2))
model.add(Conv2D(32, (3, 3), activation = 'relu'))
model.add(MaxPooling2D(2, 2))
model.add(Flatten())
model.add(Dense(units=512, activation='relu'))
model.add(Dense(units=3, activation='softmax'))
```

```
2022-04-25 03:56:29.621593: I
tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] successful NUMA node
read from SysFS had negative value (-1), but there must be at least one NUMA
node, so returning NUMA node zero
2022-04-25 03:56:29.710558: I
tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] successful NUMA node
read from SysFS had negative value (-1), but there must be at least one NUMA
node, so returning NUMA node zero
2022-04-25 03:56:29.711329: I
tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] successful NUMA node
read from SysFS had negative value (-1), but there must be at least one NUMA
node, so returning NUMA node zero
2022-04-25 03:56:29.712451: I tensorflow/core/platform/cpu_feature_guard.cc:142]
This TensorFlow binary is optimized with oneAPI Deep Neural Network Library
(oneDNN) to use the following CPU instructions in performance-critical
operations:  AVX2 AVX512F FMA
To enable them in other operations, rebuild TensorFlow with the appropriate
compiler flags.
2022-04-25 03:56:29.713495: I
tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] successful NUMA node
read from SysFS had negative value (-1), but there must be at least one NUMA
node, so returning NUMA node zero
2022-04-25 03:56:29.714190: I
tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] successful NUMA node
read from SysFS had negative value (-1), but there must be at least one NUMA
node, so returning NUMA node zero
```

```
2022-04-25 03:56:29.714795: I
tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] successful NUMA node
read from SysFS had negative value (-1), but there must be at least one NUMA
node, so returning NUMA node zero
2022-04-25 03:56:31.454056: I
tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] successful NUMA node
read from SysFS had negative value (-1), but there must be at least one NUMA
node, so returning NUMA node zero
2022-04-25 03:56:31.454901: I
tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] successful NUMA node
read from SysFS had negative value (-1), but there must be at least one NUMA
node, so returning NUMA node zero
2022-04-25 03:56:31.455572: I
tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] successful NUMA node
read from SysFS had negative value (-1), but there must be at least one NUMA
node, so returning NUMA node zero
2022-04-25 03:56:31.456157: I
tensorflow/core/common_runtime/gpu/gpu_device.cc:1510] Created device
/job:localhost/replica:0/task:0/device:GPU:0 with 15403 MB memory:  -> device:
0, name: Tesla P100-PCIE-16GB, pci bus id: 0000:00:04.0, compute capability: 6.0
```

[11]:
```
model.compile(optimizer = tf.keras.optimizers.Adam(lr = 0.001), loss =␣
 ↪'sparse_categorical_crossentropy', metrics=['accuracy'])
```

```
/opt/conda/lib/python3.7/site-packages/keras/optimizer_v2/optimizer_v2.py:356:
UserWarning: The `lr` argument is deprecated, use `learning_rate` instead.
  "The `lr` argument is deprecated, use `learning_rate` instead.")
```

[12]:
```
print("No fo Layers: ",len(model.layers))
```

```
No fo Layers:  7
```

[13]:
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 148, 148, 32)      896
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 74, 74, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 72, 72, 32)        9248
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 36, 36, 32)        0
_________________________________________________________________
flatten (Flatten)            (None, 41472)             0
_________________________________________________________________
```

```
dense (Dense)                (None, 512)               21234176
_________________________________________________________________
dense_1 (Dense)              (None, 3)                 1539
=================================================================
Total params: 21,245,859
Trainable params: 21,245,859
Non-trainable params: 0
_________________________________________________________________
```

[14]:
```
filepath= "best_model.h5"
checkpoint = ModelCheckpoint(filepath, monitor='val_accuracy', verbose=1,␣
 ↪save_best_only=True, mode='max', save_weights_only=False)

early_stopping = EarlyStopping(monitor='val_loss',min_delta = 0, patience = 5,␣
 ↪verbose = 1, restore_best_weights=True)

# learning_rate_reduction = tf.keras.callbacks.
 ↪ReduceLROnPlateau(monitor='val_loss',
#                                                patience=3,
#                                                verbose=1,
#                                                factor=0.2,
#                                                min_lr=0.00001)

callbacks_list = [
        checkpoint,
        early_stopping,
#         learning_rate_reduction
    ]
```

[15]:
```
%%time
h = model.fit_generator(datagen.flow(xtrain,ytrain,batch_size=32),
                                        validation_data=(xtest,ytest),
                                        epochs=50,
                                        callbacks=callbacks_list)
```

```
/opt/conda/lib/python3.7/site-packages/keras/engine/training.py:1972:
UserWarning: `Model.fit_generator` is deprecated and will be removed in a future
version. Please use `Model.fit`, which supports generators.
  warnings.warn('`Model.fit_generator` is deprecated and '
2022-04-25 03:56:32.543154: I
tensorflow/compiler/mlir/mlir_graph_optimization_pass.cc:185] None of the MLIR
Optimization Passes are enabled (registered 2)

Epoch 1/50

2022-04-25 03:56:33.980120: I tensorflow/stream_executor/cuda/cuda_dnn.cc:369]
Loaded cuDNN version 8005

52/52 [==============================] - 15s 160ms/step - loss: 1.3528 -
```

```
accuracy: 0.4071 - val_loss: 0.9656 - val_accuracy: 0.4570

Epoch 00001: val_accuracy improved from -inf to 0.45704, saving model to
best_model.h5
Epoch 2/50
52/52 [==============================] - 8s 151ms/step - loss: 0.7566 -
accuracy: 0.6825 - val_loss: 0.3256 - val_accuracy: 0.9141

Epoch 00002: val_accuracy improved from 0.45704 to 0.91408, saving model to
best_model.h5
Epoch 3/50
52/52 [==============================] - 7s 142ms/step - loss: 0.4938 -
accuracy: 0.8178 - val_loss: 0.2136 - val_accuracy: 0.9433

Epoch 00003: val_accuracy improved from 0.91408 to 0.94333, saving model to
best_model.h5
Epoch 4/50
52/52 [==============================] - 7s 140ms/step - loss: 0.3209 -
accuracy: 0.8982 - val_loss: 0.1335 - val_accuracy: 0.9726

Epoch 00004: val_accuracy improved from 0.94333 to 0.97258, saving model to
best_model.h5
Epoch 5/50
52/52 [==============================] - 8s 148ms/step - loss: 0.2562 -
accuracy: 0.9202 - val_loss: 0.1216 - val_accuracy: 0.9726

Epoch 00005: val_accuracy did not improve from 0.97258
Epoch 6/50
52/52 [==============================] - 7s 143ms/step - loss: 0.2294 -
accuracy: 0.9220 - val_loss: 0.1091 - val_accuracy: 0.9726

Epoch 00006: val_accuracy did not improve from 0.97258
Epoch 7/50
52/52 [==============================] - 7s 143ms/step - loss: 0.1935 -
accuracy: 0.9287 - val_loss: 0.1016 - val_accuracy: 0.9744

Epoch 00007: val_accuracy improved from 0.97258 to 0.97441, saving model to
best_model.h5
Epoch 8/50
52/52 [==============================] - 8s 153ms/step - loss: 0.1832 -
accuracy: 0.9397 - val_loss: 0.1048 - val_accuracy: 0.9707

Epoch 00008: val_accuracy did not improve from 0.97441
Epoch 9/50
52/52 [==============================] - 7s 143ms/step - loss: 0.2049 -
accuracy: 0.9287 - val_loss: 0.1037 - val_accuracy: 0.9744

Epoch 00009: val_accuracy did not improve from 0.97441
Epoch 10/50
52/52 [==============================] - 8s 146ms/step - loss: 0.1801 -
accuracy: 0.9470 - val_loss: 0.2991 - val_accuracy: 0.9031

Epoch 00010: val_accuracy did not improve from 0.97441
Epoch 11/50
52/52 [==============================] - 7s 141ms/step - loss: 0.1509 -
accuracy: 0.9439 - val_loss: 0.0882 - val_accuracy: 0.9854

Epoch 00011: val_accuracy improved from 0.97441 to 0.98537, saving model to
best_model.h5
Epoch 12/50
52/52 [==============================] - 8s 144ms/step - loss: 0.1458 -
accuracy: 0.9482 - val_loss: 0.0924 - val_accuracy: 0.9781

Epoch 00012: val_accuracy did not improve from 0.98537
Epoch 13/50
52/52 [==============================] - 8s 153ms/step - loss: 0.1281 -
accuracy: 0.9598 - val_loss: 0.0776 - val_accuracy: 0.9817

Epoch 00013: val_accuracy did not improve from 0.98537
Epoch 14/50
52/52 [==============================] - 7s 143ms/step - loss: 0.1140 -
accuracy: 0.9573 - val_loss: 0.0764 - val_accuracy: 0.9835

Epoch 00014: val_accuracy did not improve from 0.98537
Epoch 15/50
52/52 [==============================] - 7s 138ms/step - loss: 0.1481 -
accuracy: 0.9512 - val_loss: 0.0914 - val_accuracy: 0.9835

Epoch 00015: val_accuracy did not improve from 0.98537
Epoch 16/50
52/52 [==============================] - 8s 146ms/step - loss: 0.1035 -
accuracy: 0.9677 - val_loss: 0.0852 - val_accuracy: 0.9762

Epoch 00016: val_accuracy did not improve from 0.98537
Epoch 17/50
52/52 [==============================] - 8s 156ms/step - loss: 0.1200 -
accuracy: 0.9604 - val_loss: 0.0913 - val_accuracy: 0.9799

Epoch 00017: val_accuracy did not improve from 0.98537
Epoch 18/50
52/52 [==============================] - 7s 137ms/step - loss: 0.1207 -
accuracy: 0.9580 - val_loss: 0.1223 - val_accuracy: 0.9634

Epoch 00018: val_accuracy did not improve from 0.98537
Epoch 19/50
52/52 [==============================] - 8s 150ms/step - loss: 0.1370 -
```

```
accuracy: 0.9531 - val_loss: 0.0954 - val_accuracy: 0.9835

Epoch 00019: val_accuracy did not improve from 0.98537
Restoring model weights from the end of the best epoch.
Epoch 00019: early stopping
CPU times: user 2min 37s, sys: 6.45 s, total: 2min 44s
Wall time: 2min 39s
```

[16]:
```
best_model =  tf.keras.models.load_model('best_model.h5')
```

[17]:
```
from sklearn.metrics import classification_report

print(classification_report(ytest,np.argmax(best_model.predict(xtest),axis =␣
 ↪1),target_names = labels))
```

```
              precision    recall  f1-score   support

       paper       0.97      0.98      0.98       178
    scissors       0.99      0.98      0.98       188
        rock       0.99      0.99      0.99       181

    accuracy                           0.99       547
   macro avg       0.99      0.99      0.99       547
weighted avg       0.99      0.99      0.99       547
```

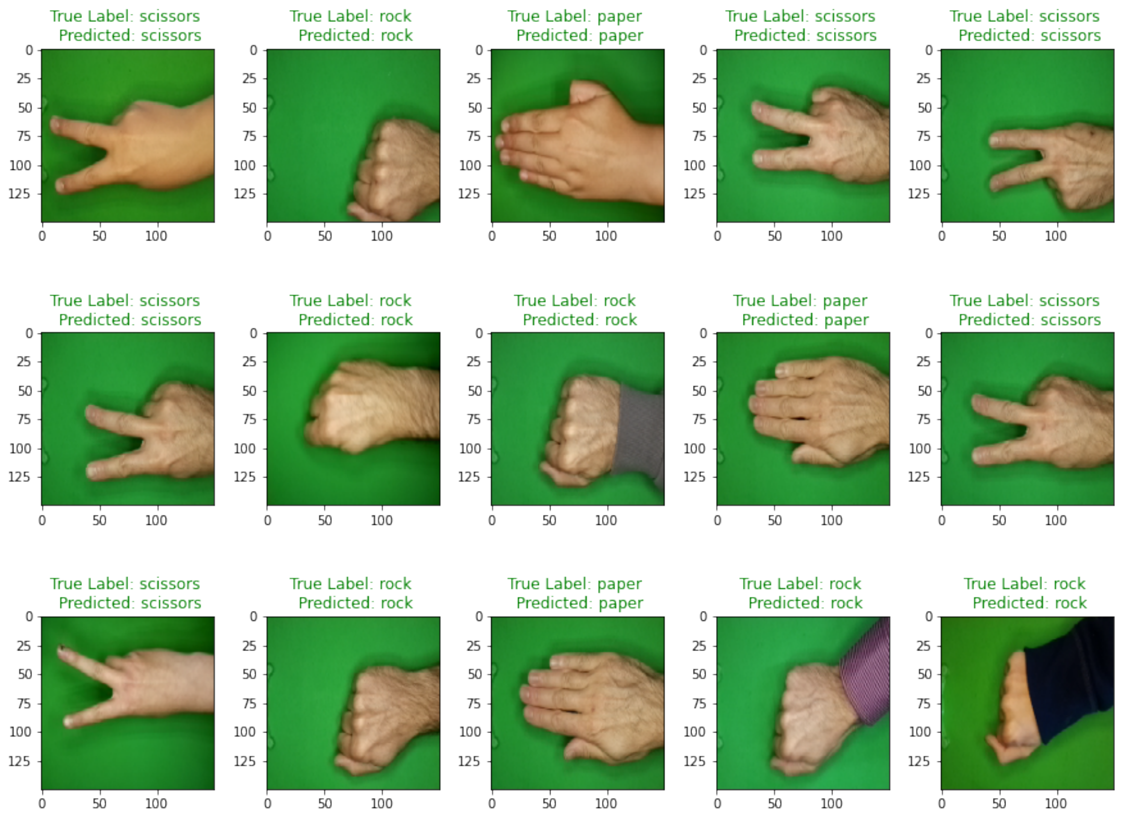
[18]:
```
pred = best_model.predict(xtest)
pred = np.argmax(pred,axis = 1)
```

[19]:
```
plt.figure(figsize = (15 , 19))
n = 0
for i in range(15):
    if pred[i]==ytest[i]:

        n+=1
        plt.subplot(5 , 5, n)
        plt.subplots_adjust(hspace = 0.5 , wspace = 0.3)
        plt.title(f'True Label: {labels[ytest[i]]} \n Predicted:␣
 ↪{labels[pred[i]]}',color = 'green')

        plt.imshow(xtest[i])
#          plt.xlabel(f"",color="green")
    else:
        n+=1
        plt.subplot(5 , 5, n)
        plt.subplots_adjust(hspace = 0.5 , wspace = 0.3)
        plt.title(f'True Label: {labels[ytest[i]]} \n Predicted:␣
↪{labels[pred[i]]}',color = 'red')

        plt.imshow(xtest[i])
```



[20]:
```
print('Time elapsed (hh:mm:ss.ms) {}'.format(datetime.now() - start_time))
```

```
Time elapsed (hh:mm:ss.ms) 0:03:09.378973
```