Final Project

Convolutional Networks for Image classification : Rock, Paper, Scissors

Maddie Lebiedzinski and Ethan Kong

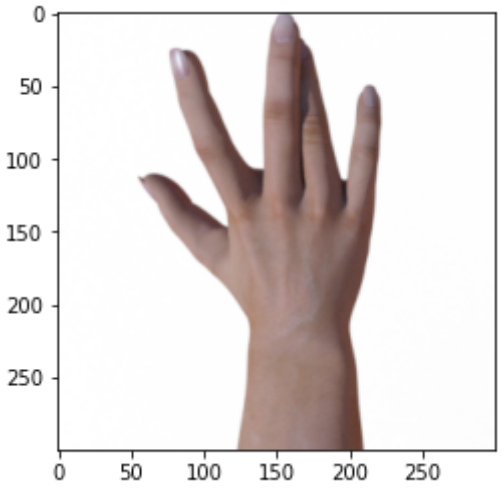
In [1]:
```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.preprocessing import image
from tensorflow.keras.optimizers import RMSprop
import tensorflow as tf
import matplotlib.pyplot as plt
import os
import cv2
import numpy as np
```

```
2023-04-28 16:29:40.273966: I tensorflow/core/platform/cpu_feature_guard.
cc:193] This TensorFlow binary is optimized with oneAPI Deep Neural Netwo
rk Library (oneDNN) to use the following CPU instructions in performance-
critical operations:  SSE4.1 SSE4.2
To enable them in other operations, rebuild TensorFlow with the appropria
te compiler flags.
```

In [2]:
```
# for train images
train_path = "Rock-Paper-Scissors/train/"
# for test images
test_path = "Rock-Paper-Scissors/test/"
# validation
validation_path = "Rock-Paper-Scissors/validation/"
```

Load and visualize data

In [3]:
```
# example of a loaded image
img = image.load_img(train_path + "/paper/paper01-000.png")
plt.imshow(img)
plt.axis("on")
plt.show()
```



In [4]:
```
# shape of this image
cv2.imread(train_path + "/paper/paper01-000.png").shape
# output : 300 pixels height, 300 pixels width, rgb 3 colors
```

Out[4]: (300, 300, 3)

In [5]:
```
# generating training
train = ImageDataGenerator(rescale= 1/255) # because rgb values go up
validation = ImageDataGenerator(rescale= 1/255)
```

In [ ]:
```

```

In [6]:
```
train_dataset = train.flow_from_directory(train_path,
                                          target_size= (300, 300), # in
                                          batch_size = 32, #
                                          class_mode = 'categorical') #

validation_dataset = train.flow_from_directory(validation_path,
                                          target_size= (300, 300), # in
                                          batch_size = 32, #
                                          class_mode = 'categorical') #
```

```
Found 2520 images belonging to 3 classes.
Found 33 images belonging to 3 classes.
```

In [7]:
```
train_dataset.class_indices
```

Out[7]: {'paper': 0, 'rock': 1, 'scissors': 2}

In [8]:
```
train_dataset.classes
```

Out[8]: array([0, 0, 0, ..., 2, 2, 2], dtype=int32)

Construct deep learning model

In [9]:
```
model = tf.keras.models.Sequential([tf.keras.layers.Conv2D(16,(3, 3),
                                    tf.keras.layers.MaxPool2D(2,2),

                                    tf.keras.layers.Conv2D(32,(3, 3),
                                    tf.keras.layers.MaxPool2D(2,2),

                                    tf.keras.layers.Conv2D(64,(3, 3),
                                    tf.keras.layers.MaxPool2D(2,2),

                                    tf.keras.layers.Flatten(),

                                    tf.keras.layers.Dense(2892, activa
                                                          # number of
                                    tf.keras.layers.Dense(3, activatio
])                                                        # 3 categori


# to view all the layers of the network using the Keras Model.summary
model.summary()
```

```
2023-04-28 16:29:45.475462: I tensorflow/core/platform/cpu_feature_guard.
cc:193] This TensorFlow binary is optimized with oneAPI Deep Neural Netwo
rk Library (oneDNN) to use the following CPU instructions in performance-
critical operations:  SSE4.1 SSE4.2
To enable them in other operations, rebuild TensorFlow with the appropria
te compiler flags.

Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 298, 298, 16)      448

 max_pooling2d (MaxPooling2D  (None, 149, 149, 16)     0
 )

 conv2d_1 (Conv2D)           (None, 147, 147, 32)      4640

 max_pooling2d_1 (MaxPooling  (None, 73, 73, 32)       0
 2D)

 conv2d_2 (Conv2D)           (None, 71, 71, 64)        18496

 max_pooling2d_2 (MaxPooling  (None, 35, 35, 64)       0
 2D)

 flatten (Flatten)           (None, 78400)             0

 dense (Dense)               (None, 2892)              226735692

 dense_1 (Dense)             (None, 3)                 8679

=================================================================
Total params: 226,767,955
Trainable params: 226,767,955
Non-trainable params: 0
_________________________________________________________________
```

In [10]:
```
# compile the model
model.compile(loss= 'categorical_crossentropy',
              optimizer = RMSprop(lr=0.001),
              metrics = ['accuracy'])
```

```
WARNING:absl:`lr` is deprecated, please use `learning_rate` instead, or u
se the legacy optimizer, e.g.,tf.keras.optimizers.legacy.RMSprop.
```

Training

In [11]:
```
model_fit= model.fit(train_dataset,
                     steps_per_epoch = 20,
                     epochs= 20,
                     validation_data = validation_dataset)
```

```
Epoch 1/20
20/20 [==============================] - 75s 4s/step - loss: 2.1607 - acc
uracy: 0.5222 - val_loss: 0.8475 - val_accuracy: 0.7576
Epoch 2/20
20/20 [==============================] - 73s 4s/step - loss: 0.9757 - acc
uracy: 0.7281 - val_loss: 0.7622 - val_accuracy: 0.6667
Epoch 3/20
20/20 [==============================] - 71s 4s/step - loss: 0.2048 - acc
uracy: 0.9399 - val_loss: 0.9675 - val_accuracy: 0.7273
Epoch 4/20
20/20 [==============================] - 71s 4s/step - loss: 0.0252 - acc
uracy: 0.9953 - val_loss: 1.1462 - val_accuracy: 0.6667
Epoch 5/20
20/20 [==============================] - 65s 3s/step - loss: 0.0172 - acc
uracy: 0.9968 - val_loss: 1.0417 - val_accuracy: 0.7879
Epoch 6/20
20/20 [==============================] - 75s 4s/step - loss: 0.4563 - acc
uracy: 0.8906 - val_loss: 0.7344 - val_accuracy: 0.7576
Epoch 7/20
20/20 [==============================] - 71s 4s/step - loss: 0.0166 - acc
uracy: 0.9984 - val_loss: 0.8632 - val_accuracy: 0.8182
Epoch 8/20
20/20 [==============================] - 68s 3s/step - loss: 0.0163 - acc
uracy: 0.9953 - val_loss: 1.1737 - val_accuracy: 0.7576
Epoch 9/20
20/20 [==============================] - 77s 4s/step - loss: 8.2154e-04 -
accuracy: 1.0000 - val_loss: 1.3731 - val_accuracy: 0.7576
Epoch 10/20
20/20 [==============================] - 70s 3s/step - loss: 3.9407e-04 -
accuracy: 1.0000 - val_loss: 1.5656 - val_accuracy: 0.7273
Epoch 11/20
20/20 [==============================] - 62s 3s/step - loss: 1.7029 - acc
uracy: 0.9019 - val_loss: 1.0658 - val_accuracy: 0.7576
Epoch 12/20
20/20 [==============================] - 68s 3s/step - loss: 0.0054 - acc
uracy: 1.0000 - val_loss: 1.6269 - val_accuracy: 0.6970
Epoch 13/20
20/20 [==============================] - 101s 5s/step - loss: 0.0013 - ac
curacy: 1.0000 - val_loss: 1.4174 - val_accuracy: 0.7273
Epoch 14/20
20/20 [==============================] - 85s 4s/step - loss: 8.8733e-04 -
accuracy: 1.0000 - val_loss: 1.6309 - val_accuracy: 0.7273
Epoch 15/20
20/20 [==============================] - 83s 4s/step - loss: 1.9818e-04 -
accuracy: 1.0000 - val_loss: 1.6107 - val_accuracy: 0.7273
Epoch 16/20
20/20 [==============================] - 84s 4s/step - loss: 0.9632 - acc
uracy: 0.9225 - val_loss: 0.6992 - val_accuracy: 0.8182
Epoch 17/20
20/20 [==============================] - 88s 4s/step - loss: 0.0051 - acc
uracy: 1.0000 - val_loss: 0.7134 - val_accuracy: 0.8182
Epoch 18/20
```

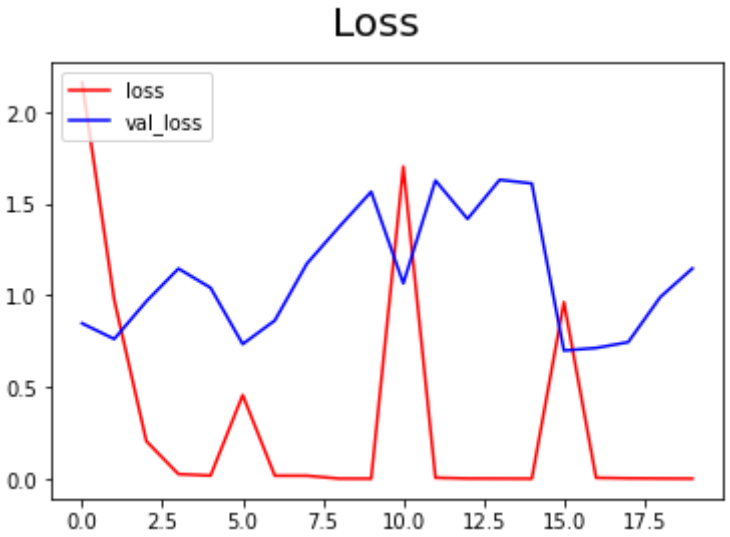
```
20/20 [==============================] - 88s 4s/step - loss: 0.0023 - acc
uracy: 1.0000 - val_loss: 0.7450 - val_accuracy: 0.8182
Epoch 19/20
20/20 [==============================] - 81s 4s/step - loss: 0.0013 - acc
uracy: 1.0000 - val_loss: 0.9910 - val_accuracy: 0.8182
Epoch 20/20
20/20 [==============================] - 70s 3s/step - loss: 4.3057e-04 -
accuracy: 1.0000 - val_loss: 1.1472 - val_accuracy: 0.8182
```

In [17]:
```
1 model_fit.history.keys()
```

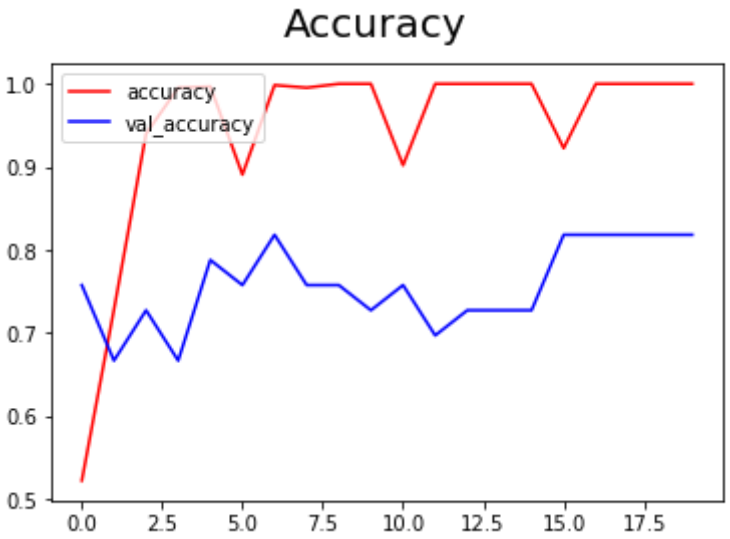
Out[17]:
```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

Performance plot

In [25]:
```
1 fig = plt.figure()
2 loss = model_fit.history['loss']
3 val_loss = model_fit.history['val_loss']
4 plt.plot(loss, color='red', label='loss')
5 plt.plot(val_loss, color='blue', label='val_loss')
6 fig.suptitle('Loss', fontsize=20)
7 plt.legend(loc="upper left")
8 plt.show()
```



In [22]:
```
fig = plt.figure()
acc = model_fit.history['accuracy']
val_acc = model_fit.history['val_accuracy']
plt.plot(acc, color='red', label='accuracy')
plt.plot(val_acc, color='blue', label='val_accuracy')
fig.suptitle('Accuracy', fontsize=20)
plt.legend(loc="upper left")
plt.show()
```



In [23]:
```
validation_dataset.class_indices
```

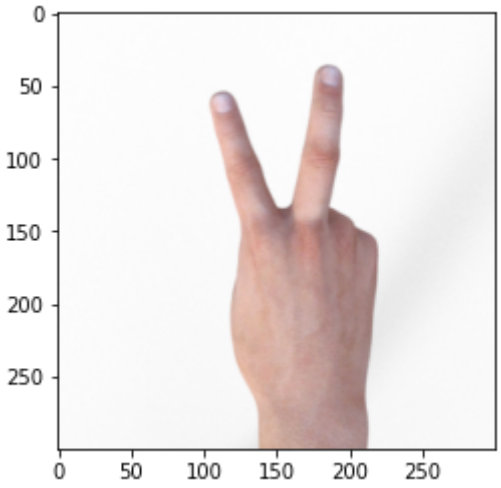
Out[23]:
```
{'paper': 0, 'rock': 1, 'scissors': 2}
```

Test

In [24]:
```
# testset
dir_path = 'Rock-Paper-Scissors/test_all'

for i in os.listdir(dir_path):
    img = image.load_img(dir_path+'//'+i, target_size= (300, 300))
    plt.imshow(img)
    plt.show()

    X = image.img_to_array(img)
    X = np.expand_dims(X, axis =0)
    images = np.vstack([X])
    val = model.predict(images)
    print(val)
    if val[0][1] == 1:
        print("rock")
    elif val[0][0] == 1:
        print ("paper")
    elif val[0][2] == 1:
        print ("scissors")
```




```
1/1 [==============================] - 3s 3s/step
[[0. 0. 1.]]
scissors
```

Link to dataset : https://www.kaggle.com/datasets/sanikamal/rock-paper-scissors-dataset (https://www.kaggle.com/datasets/sanikamal/rock-paper-scissors-dataset)