```
!wget --no-check-certificate \
    https://storage.googleapis.com/laurencemoroney-blog.appspot.com/rps.zip \
    -O /tmp/rps.zip

!wget --no-check-certificate \
    https://storage.googleapis.com/laurencemoroney-blog.appspot.com/rps-test-set.zip \
    -O /tmp/rps-test-set.zip
```

```
--2020-09-17 10:02:01--  https://storage.googleapis.com/laurencemoroney-blog.appspot.com
Resolving storage.googleapis.com (storage.googleapis.com)... 74.125.20.128, 74.125.195.1
Connecting to storage.googleapis.com (storage.googleapis.com)|74.125.20.128|:443... conn
HTTP request sent, awaiting response... 200 OK
Length: 200682221 (191M) [application/zip]
Saving to: '/tmp/rps.zip'

/tmp/rps.zip        100%[===================>] 191.38M   115MB/s    in 1.7s

2020-09-17 10:02:02 (115 MB/s) - '/tmp/rps.zip' saved [200682221/200682221]

--2020-09-17 10:02:03--  https://storage.googleapis.com/laurencemoroney-blog.appspot.com
Resolving storage.googleapis.com (storage.googleapis.com)... 74.125.20.128, 74.125.197.1
Connecting to storage.googleapis.com (storage.googleapis.com)|74.125.20.128|:443... conn
HTTP request sent, awaiting response... 200 OK
                                    tion/zip]
                                    ip'

/tmp/rps-test-set.z 100%[===================>]  28.15M  72.0MB/s    in 0.4s

2020-09-17 10:02:03 (72.0 MB/s) - '/tmp/rps-test-set.zip' saved [29516758/29516758]
```

Saved successfully!

```
import os
import zipfile

local_zip = '/tmp/rps.zip'
zip_ref = zipfile.ZipFile(local_zip, 'r')
zip_ref.extractall('/tmp/')
zip_ref.close()

local_zip = '/tmp/rps-test-set.zip'
zip_ref = zipfile.ZipFile(local_zip, 'r')
zip_ref.extractall('/tmp/')
zip_ref.close()


rock_dir = os.path.join('/tmp/rps/rock')
```

```
rock_dir = os.path.join('/tmp/rps/rock')
paper_dir = os.path.join('/tmp/rps/paper')
scissors_dir = os.path.join('/tmp/rps/scissors')

print('total training rock images:', len(os.listdir(rock_dir)))
print('total training paper images:', len(os.listdir(paper_dir)))
print('total training scissors images:', len(os.listdir(scissors_dir)))

rock_files = os.listdir(rock_dir)
print(rock_files[:10])

paper_files = os.listdir(paper_dir)
print(paper_files[:10])

scissors_files = os.listdir(scissors_dir)
print(scissors_files[:10])
```

```
total training rock images: 840
total training paper images: 840
total training scissors images: 840
['rock04-089.png', 'rock02-028.png', 'rock01-095.png', 'rock01-091.png', 'rock03-078.png
['paper01-055.png', 'paper07-043.png', 'paper01-091.png', 'paper03-040.png', 'paper06-01
['scissors02-039.png', 'scissors03-105.png', 'testscissors03-066.png', 'testscissors03-0
```

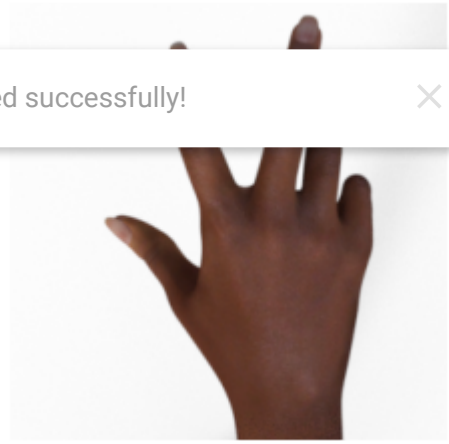
```
%matplotlib inline

import matplotlib.pyplot as plt
import matplotlib.image as mpimg
```



```
next_rock = [os.path.join(rock_dir, fname)
                for fname in rock_files[pic_index-2:pic_index]]
next_paper = [os.path.join(paper_dir, fname)
                for fname in paper_files[pic_index-2:pic_index]]
next_scissors = [os.path.join(scissors_dir, fname)
                for fname in scissors_files[pic_index-2:pic_index]]

for i, img_path in enumerate(next_rock+next_paper+next_scissors):
  #print(img_path)
  img = mpimg.imread(img_path)
  plt.imshow(img)
  plt.axis('Off')
  plt.show()
```



Saved successfully!

```
import tensorflow as tf
import keras_preprocessing
from keras_preprocessing import image
from keras_preprocessing.image import ImageDataGenerator

TRAINING_DIR = "/tmp/rps/"
training_datagen = ImageDataGenerator(
      rescale = 1./255,
      rotation_range=40,
      width_shift_range=0.2,
      height_shift_range=0.2,
      shear_range=0.2,
      zoom_range=0.2
```

Saved successfully!

```
VALIDATION_DIR = "/tmp/rps-test-set/"
validation_datagen = ImageDataGenerator(rescale = 1./255)

train_generator = training_datagen.flow_from_directory(
  TRAINING_DIR,
  target_size=(150,150),
  class_mode='categorical',
  batch_size=126
)

validation_generator = validation_datagen.flow_from_directory(
  VALIDATION_DIR,
  target_size=(150,150),
  class_mode='categorical',
  batch_size=126
)

model = tf.keras.models.Sequential([
    # Note the input shape is the desired size of the image 150x150 with 3 bytes color
```

```
    # This is the first convolution
    tf.keras.layers.Conv2D(64, (3,3), activation='relu', input_shape=(150, 150, 3)),
    tf.keras.layers.MaxPooling2D(2, 2),
    # The second convolution
    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    # The third convolution
    tf.keras.layers.Conv2D(128, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    # The fourth convolution
    tf.keras.layers.Conv2D(128, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    # Flatten the results to feed into a DNN
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dropout(0.5),
    # 512 neuron hidden layer
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dense(3, activation='softmax')
])


model.summary()

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

history = model.fit(train_generator, epochs=25, steps_per_epoch=20, validation_data = validat

model.save("rps.h5")
```

Saved successfully!

```
_________________________________________________________________
dense_1 (Dense)              (None, 3)                 1539      
=================================================================
Total params: 3,473,475
Trainable params: 3,473,475
Non-trainable params: 0
_________________________________________________________________
Epoch 1/25
 2/20 [==>...........................] - ETA: 1s - loss: 1.7413 - accuracy: 0.3452WARNIN
20/20 [==============================] - 19s 933ms/step - loss: 1.1734 - accuracy: 0.368
Epoch 2/25
20/20 [==============================] - 18s 921ms/step - loss: 1.0703 - accuracy: 0.422
Epoch 3/25
20/20 [==============================] - 18s 924ms/step - loss: 1.0805 - accuracy: 0.457
Epoch 4/25
20/20 [==============================] - 18s 908ms/step - loss: 0.8862 - accuracy: 0.590
Epoch 5/25
20/20 [==============================] - 18s 915ms/step - loss: 0.7594 - accuracy: 0.659
Epoch 6/25
20/20 [==============================] - 18s 917ms/step - loss: 0.6283 - accuracy: 0.726
Epoch 7/25
20/20 [==============================] - 18s 912ms/step - loss: 0.5632 - accuracy: 0.767
Epoch 8/25
20/20 [==============================] - 18s 913ms/step - loss: 0.4709 - accuracy: 0.817
Epoch 9/25
20/20 [==============================] - 18s 915ms/step - loss: 0.3937 - accuracy: 0.838
Epoch 10/25
20/20 [==============================] - 18s 911ms/step - loss: 0.5946 - accuracy: 0.852
Epoch 11/25
20/20 [==============================] - 18s 905ms/step - loss: 0.2764 - accuracy: 0.890
Epoch 12/25
20/20 [==============================] - 18s 906ms/step - loss: 0.2508 - accuracy: 0.904
Epoch 13/25
20/20 [==============================] - 18s 904ms/step - loss: 0.2676 - accuracy: 0.901
20/20 [==============================] - 18s 914ms/step - loss: 0.1511 - accuracy: 0.947
Epoch 15/25
20/20 [==============================] - 18s 906ms/step - loss: 0.2289 - accuracy: 0.911
Epoch 16/25
20/20 [==============================] - 18s 909ms/step - loss: 0.2015 - accuracy: 0.927
Epoch 17/25
20/20 [==============================] - 18s 915ms/step - loss: 0.0991 - accuracy: 0.965
Epoch 18/25
20/20 [==============================] - 18s 912ms/step - loss: 0.1711 - accuracy: 0.938
Epoch 19/25
20/20 [==============================] - 18s 921ms/step - loss: 0.1478 - accuracy: 0.950
Epoch 20/25
20/20 [==============================] - 18s 907ms/step - loss: 0.0995 - accuracy: 0.964
Epoch 21/25
20/20 [==============================] - 18s 909ms/step - loss: 0.2278 - accuracy: 0.923
Epoch 22/25
20/20 [==============================] - 18s 915ms/step - loss: 0.0844 - accuracy: 0.973
Epoch 23/25
20/20 [==============================] - 18s 904ms/step - loss: 0.1350 - accuracy: 0.949
Epoch 24/25
20/20 [==============================] - 18s 907ms/step - loss: 0.0862 - accuracy: 0.974
Epoch 25/25
```

Saved successfully!

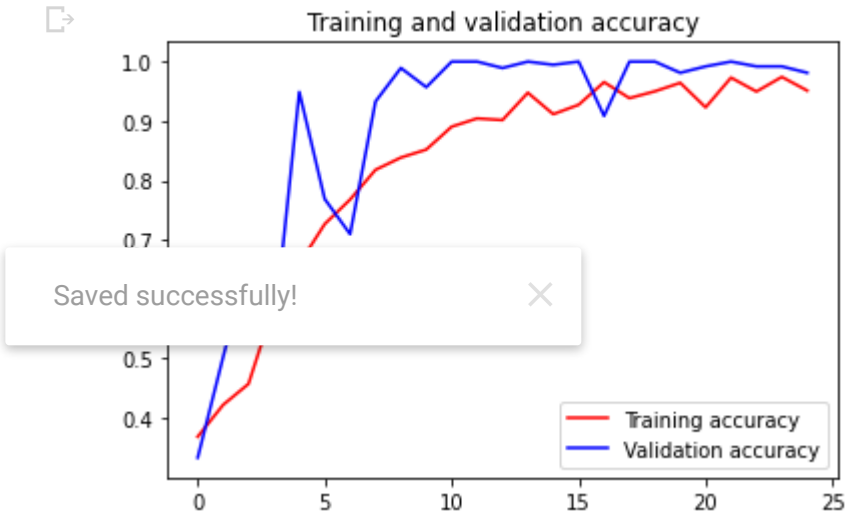
```
20/20 [==============================] - 18s 906ms/step - loss: 0.1238 - accuracy: 0.951
```

```
import matplotlib.pyplot as plt
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(len(acc))

plt.plot(epochs, acc, 'r', label='Training accuracy')
plt.plot(epochs, val_acc, 'b', label='Validation accuracy')
plt.title('Training and validation accuracy')
plt.legend(loc=0)
plt.figure()


plt.show()
```




```
<Figure size 432x288 with 0 Axes>
```

```
import numpy as np
from google.colab import files
from keras.preprocessing import image

uploaded = files.upload()

for fn in uploaded.keys():

  # predicting images
  path = fn
  img = image.load_img(path, target_size=(150, 150))
  x = image.img_to_array(img)
```