In [ ]:
```
import os
import zipfile

local_zip = 'D:/5-2/rps.zip'
zip_ref = zipfile.ZipFile(local_zip, 'r')
zip_ref.extractall('D:/5-2/')
zip_ref.close()

local_zip = 'D:/5-2/rps-test-set.zip'
zip_ref = zipfile.ZipFile(local_zip, 'r')
zip_ref.extractall('D:/5-2/')
zip_ref.close()
```

打印相关信息¶

In [ ]:
```
rock_dir = os.path.join('D:/5-2/rps/rock')
paper_dir = os.path.join('D:/5-2/rps/paper')
scissors_dir = os.path.join('D:/5-2/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
['rock01-000.png', 'rock01-001.png', 'rock01-002.png', 'rock01-003.png', 'rock01-004.png', 'rock01-005.png', 'rock01-006.png', 'rock01-007.png', 'rock01-008.png', 'rock01-009.png']
['paper01-000.png', 'paper01-001.png', 'paper01-002.png', 'paper01-003.png', 'paper01-004.png', 'paper01-005.png', 'paper01-006.png', 'paper01-007.png', 'paper01-008.png', 'paper01-009.png']
['scissors01-000.png', 'scissors01-001.png', 'scissors01-002.png', 'scissors01-003.png', 'scissors01-004.png', 'scissors01-005.png', 'scissors01-006.png', 'scissors01-007.png', 'scissors01-008.png', 'scissors01-009.png']
```

各打印两张石头剪刀布训练集图片¶

In [ ]:
```
%matplotlib inline

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

pic_index = 2

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()
```

调用 TensorFlow 的 keras 进行数据模型的训练和评估。¶

In [ ]:

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

TRAINING_DIR = "D:/5-2/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,
      horizontal_flip=True,
      fill_mode='nearest')

VALIDATION_DIR = "D:/5-2/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 =
validation_generator, verbose = 1, validation_steps=3)
```

```
model.save("rps.h5")
Found 2520 images belonging to 3 classes.
Found 372 images belonging to 3 classes.
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 148, 148, 64)      1792

 max_pooling2d (MaxPooling2D  (None, 74, 74, 64)       0
 )

 conv2d_1 (Conv2D)           (None, 72, 72, 64)        36928

 max_pooling2d_1 (MaxPooling  (None, 36, 36, 64)       0
 2D)

 conv2d_2 (Conv2D)           (None, 34, 34, 128)       73856

 max_pooling2d_2 (MaxPooling  (None, 17, 17, 128)      0
 2D)

 conv2d_3 (Conv2D)           (None, 15, 15, 128)       147584

 max_pooling2d_3 (MaxPooling  (None, 7, 7, 128)        0
 2D)

 flatten (Flatten)           (None, 6272)              0

 dropout (Dropout)           (None, 6272)              0

 dense (Dense)               (None, 512)               3211776

 dense_1 (Dense)             (None, 3)                 1539

=================================================================
Total params: 3,473,475
Trainable params: 3,473,475
Non-trainable params: 0
_________________________________________________________________
Epoch 1/25
20/20 [==============================] - 81s 4s/step - loss: 1.2043 - accuracy:
0.3615 - val_loss: 1.0957 - val_accuracy: 0.3333
Epoch 2/25
20/20 [==============================] - 64s 3s/step - loss: 1.0875 - accuracy:
0.3909 - val_loss: 1.0616 - val_accuracy: 0.5914
Epoch 3/25
20/20 [==============================] - 75s 4s/step - loss: 1.0220 - accuracy:
0.4647 - val_loss: 0.8552 - val_accuracy: 0.5833
Epoch 4/25
20/20 [==============================] - 73s 4s/step - loss: 0.9733 - accuracy:
0.5175 - val_loss: 1.1185 - val_accuracy: 0.3683
Epoch 5/25
20/20 [==============================] - 73s 4s/step - loss: 0.8419 - accuracy:
0.6004 - val_loss: 0.4914 - val_accuracy: 0.8414
```

```
Epoch 6/25
20/20 [==============================] - 76s 4s/step - loss: 0.7870 - accuracy:
0.6480 - val_loss: 0.4908 - val_accuracy: 0.7823
Epoch 7/25
20/20 [==============================] - 70s 3s/step - loss: 0.6982 - accuracy:
0.6821 - val_loss: 0.3660 - val_accuracy: 0.8495
Epoch 8/25
20/20 [==============================] - 69s 3s/step - loss: 0.5575 - accuracy:
0.7702 - val_loss: 0.3571 - val_accuracy: 0.8629
Epoch 9/25
20/20 [==============================] - 69s 3s/step - loss: 0.4714 - accuracy:
0.8040 - val_loss: 0.2953 - val_accuracy: 0.8898
Epoch 10/25
20/20 [==============================] - 70s 3s/step - loss: 0.3988 - accuracy:
0.8389 - val_loss: 0.0931 - val_accuracy: 0.9919
Epoch 11/25
20/20 [==============================] - 70s 3s/step - loss: 0.3510 - accuracy:
0.8627 - val_loss: 0.1194 - val_accuracy: 0.9624
Epoch 12/25
20/20 [==============================] - 68s 3s/step - loss: 0.3073 - accuracy:
0.8786 - val_loss: 0.1383 - val_accuracy: 0.9489
Epoch 13/25
20/20 [==============================] - 68s 3s/step - loss: 0.2422 - accuracy:
0.9099 - val_loss: 0.1776 - val_accuracy: 0.9462
Epoch 14/25
20/20 [==============================] - 71s 4s/step - loss: 0.2667 - accuracy:
0.9032 - val_loss: 0.0841 - val_accuracy: 0.9651
Epoch 15/25
20/20 [==============================] - 68s 3s/step - loss: 0.1961 - accuracy:
0.9266 - val_loss: 0.4319 - val_accuracy: 0.8387
Epoch 16/25
20/20 [==============================] - 69s 3s/step - loss: 0.1687 - accuracy:
0.9337 - val_loss: 0.2307 - val_accuracy: 0.8978
Epoch 17/25
20/20 [==============================] - 68s 3s/step - loss: 0.1763 - accuracy:
0.9306 - val_loss: 0.1666 - val_accuracy: 0.9382
Epoch 18/25
20/20 [==============================] - 68s 3s/step - loss: 0.1293 - accuracy:
0.9603 - val_loss: 0.0571 - val_accuracy: 0.9758
Epoch 19/25
20/20 [==============================] - 68s 3s/step - loss: 0.1761 - accuracy:
0.9294 - val_loss: 0.1159 - val_accuracy: 0.9597
Epoch 20/25
20/20 [==============================] - 68s 3s/step - loss: 0.1077 - accuracy:
0.9643 - val_loss: 0.1117 - val_accuracy: 0.9462
Epoch 21/25
20/20 [==============================] - 68s 3s/step - loss: 0.2010 - accuracy:
0.9282 - val_loss: 0.1393 - val_accuracy: 0.9516
Epoch 22/25
20/20 [==============================] - 68s 3s/step - loss: 0.0764 - accuracy:
0.9730 - val_loss: 0.0977 - val_accuracy: 0.9543
Epoch 23/25
20/20 [==============================] - 69s 3s/step - loss: 0.1123 - accuracy:
0.9599 - val_loss: 0.0245 - val_accuracy: 0.9812
Epoch 24/25
```

```
20/20 [==============================] - 70s 3s/step - loss: 0.1244 - accuracy:
0.9595 - val_loss: 0.1766 - val_accuracy: 0.9462
Epoch 25/25
20/20 [==============================] - 67s 3s/step - loss: 0.0781 - accuracy:
0.9718 - val_loss: 0.2643 - val_accuracy: 0.9032
```

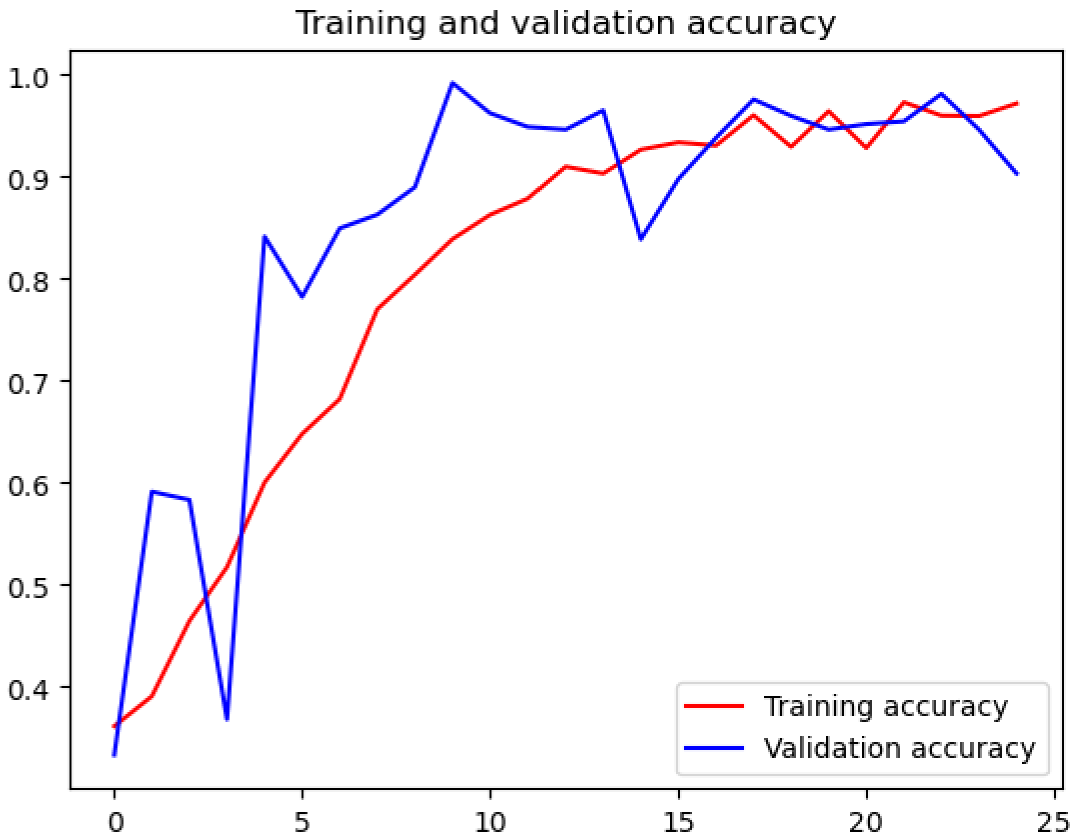
完成模型训练之后，我们绘制训练和验证结果的相关信息

In [ ]:

```
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 640x480 with 0 Axes>
```