Image_forgery

April 14, 2020

[0]:
```
!apt-get install -y -qq software-properties-common python-software-properties␣
 ↪module-init-tools
!add-apt-repository -y ppa:alessandro-strada/ppa 2>&1 > /dev/null
!apt-get update -qq 2>&1 > /dev/null
!apt-get -y install -qq google-drive-ocamlfuse fuse
from google.colab import auth
auth.authenticate_user()
from oauth2client.client import GoogleCredentials
creds = GoogleCredentials.get_application_default()
import getpass
!google-drive-ocamlfuse -headless -id={creds.client_id} -secret={creds.
 ↪client_secret} < /dev/null 2>&1 | grep URL
vcode = getpass.getpass()
!echo {vcode} | google-drive-ocamlfuse -headless -id={creds.client_id}␣
 ↪-secret={creds.client_secret}

!mkdir -p drive
!google-drive-ocamlfuse drive

!ls drive
```

[5]:
```
%matplotlib inline
import keras
import numpy as np
from keras.applications.vgg16 import VGG16
from keras.preprocessing.image import load_img
from keras.callbacks import ModelCheckpoint
import matplotlib.pyplot as plt
import itertools
from keras.models import Sequential
from keras.optimizers import Adam
from keras.preprocessing.image import ImageDataGenerator
import tensorflow as tf
from keras.layers import Dense, Flatten, Dropout
from tensorflow.python.client import device_lib
# keras library import  for Saving and loading model and weights
from keras.models import model_from_json
from keras.models import load_model
from sklearn.metrics import confusion_matrix
# import cv2
```

```
Using TensorFlow backend.
```

[6]:
```
model = VGG16()
```

```
Downloading data from https://github.com/fchollet/deep-learning-
models/releases/download/v0.1/vgg16_weights_tf_dim_ordering_tf_kernels.h5
553467904/553467096 [==============================] - 13s 0us/step
```

[7]:
```
model.summary()
```

```
Model: "vgg16"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 224, 224, 3)       0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 28, 28, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
```

```
block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0
_________________________________________________________________
flatten (Flatten)            (None, 25088)             0
_________________________________________________________________
fc1 (Dense)                  (None, 4096)              102764544
_________________________________________________________________
fc2 (Dense)                  (None, 4096)              16781312
_________________________________________________________________
predictions (Dense)          (None, 1000)              4097000
=================================================================
Total params: 138,357,544
Trainable params: 138,357,544
Non-trainable params: 0
_________________________________________________________________
```

[8]:
```
type(model)
```

[8]:
```
keras.engine.training.Model
```

[0]:
```
newModel = Sequential()

for layer in model.layers:
  newModel.add(layer)
```

[10]:
```
newModel.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0
```

```
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 28, 28, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0
_________________________________________________________________
flatten (Flatten)            (None, 25088)             0
_________________________________________________________________
fc1 (Dense)                  (None, 4096)              102764544
_________________________________________________________________
fc2 (Dense)                  (None, 4096)              16781312
_________________________________________________________________
predictions (Dense)          (None, 1000)              4097000
=================================================================
Total params: 138,357,544
Trainable params: 138,357,544
Non-trainable params: 0
_________________________________________________________________
```

[11]:
```
newModel.layers.pop()
```

[11]:
```
<keras.layers.core.Dense at 0x7f44b00d7908>
```

[12]:
```
newModel.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 28, 28, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0
_________________________________________________________________
flatten (Flatten)            (None, 25088)             0
_________________________________________________________________
fc1 (Dense)                  (None, 4096)              102764544
_________________________________________________________________
fc2 (Dense)                  (None, 4096)              16781312
_________________________________________________________________
predictions (Dense)          (None, 1000)              4097000
=================================================================
Total params: 138,357,544
Trainable params: 138,357,544
Non-trainable params: 0
_________________________________________________________________
```

[0]:
```
for layer in newModel.layers:
  layer.trainable = False
```

[0]:
```
newModel.add(Dense(2, activation='softmax'))
```

[15]:
```
newModel.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792      
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928     
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0         
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856     
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584    
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0         
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168    
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080    
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080    
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 28, 28, 256)       0         
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160   
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 28, 28, 512)       2359808   
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808   
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0         
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808   
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808   
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808   
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0         
_________________________________________________________________
flatten (Flatten)            (None, 25088)             0         
```

```
_________________________________________________________________
fc1 (Dense)                  (None, 4096)              102764544
_________________________________________________________________
fc2 (Dense)                  (None, 4096)              16781312
_________________________________________________________________
predictions (Dense)          (None, 1000)              4097000
_________________________________________________________________
dense_1 (Dense)              (None, 2)                 2002
=================================================================
Total params: 138,359,546
Trainable params: 2,002
Non-trainable params: 138,357,544
_________________________________________________________________
```

[0]:
```
#Compiling Model
newModel.compile(Adam(lr=.0001), loss = 'categorical_crossentropy', metrics =␣
 ↪['accuracy'])
```

[0]:
```
train_path = 'drive/dataset/train'
test_path = 'drive/dataset/test'
valid_path = 'drive/dataset/valid'
```

[18]:
```
train_batches = ImageDataGenerator().flow_from_directory(train_path,␣
 ↪target_size=(224,224), classes=['original', 'spliced'], batch_size=10,␣
 ↪shuffle=True)
test_batches =  ImageDataGenerator().flow_from_directory(test_path ,␣
 ↪target_size=(224,224), classes=['original', 'spliced'], batch_size=4)
valid_batches = ImageDataGenerator().flow_from_directory(valid_path,␣
 ↪target_size=(224,224), classes=['original', 'spliced'], batch_size=10,␣
 ↪shuffle=True)
```

```
Found 18 images belonging to 2 classes.
Found 13 images belonging to 2 classes.
Found 11 images belonging to 2 classes.
```

[25]:
```
newModel.fit_generator(train_batches, steps_per_epoch=182,␣
 ↪validation_data=valid_batches, validation_steps=40, epochs=100,␣
 ↪callbacks=callbacks_list, verbose=2)
```

```
Epoch 1/100
 - 26s - loss: 0.6865 - accuracy: 0.7363 - val_loss: 0.6968 - val_accuracy:
0.8182
Epoch 2/100

/usr/local/lib/python3.6/dist-packages/keras/callbacks/callbacks.py:707:
RuntimeWarning: Can save best model only with val_acc available, skipping.
  'skipping.' % (self.monitor), RuntimeWarning)
 - 11s - loss: 0.6765 - accuracy: 0.8694 - val_loss: 0.6797 - val_accuracy:
0.8182
Epoch 3/100
 - 11s - loss: 0.6666 - accuracy: 0.9408 - val_loss: 0.7059 - val_accuracy:
0.6364
Epoch 4/100
 - 11s - loss: 0.6568 - accuracy: 0.9444 - val_loss: 0.6736 - val_accuracy:
0.6364
Epoch 5/100
 - 11s - loss: 0.6474 - accuracy: 0.9463 - val_loss: 0.7272 - val_accuracy:
0.6364
Epoch 6/100
 - 11s - loss: 0.6384 - accuracy: 1.0000 - val_loss: 0.6719 - val_accuracy:
0.6364
Epoch 7/100
 - 11s - loss: 0.6293 - accuracy: 1.0000 - val_loss: 0.6687 - val_accuracy:
0.5455
Epoch 8/100
 - 11s - loss: 0.6205 - accuracy: 1.0000 - val_loss: 0.6612 - val_accuracy:
0.5455
Epoch 9/100
 - 11s - loss: 0.6116 - accuracy: 1.0000 - val_loss: 0.6514 - val_accuracy:
0.5455
Epoch 10/100
 - 12s - loss: 0.6032 - accuracy: 1.0000 - val_loss: 0.7615 - val_accuracy:
0.5455
Epoch 11/100
 - 11s - loss: 0.5945 - accuracy: 1.0000 - val_loss: 0.6547 - val_accuracy:
0.5455
Epoch 12/100
 - 11s - loss: 0.5862 - accuracy: 1.0000 - val_loss: 0.6401 - val_accuracy:
0.5455
Epoch 13/100
 - 11s - loss: 0.5780 - accuracy: 1.0000 - val_loss: 0.6439 - val_accuracy:
0.5455
Epoch 14/100
 - 11s - loss: 0.5700 - accuracy: 1.0000 - val_loss: 0.7593 - val_accuracy:
0.5455
Epoch 15/100
 - 11s - loss: 0.5620 - accuracy: 1.0000 - val_loss: 0.7136 - val_accuracy:
0.5455
Epoch 16/100
 - 11s - loss: 0.5541 - accuracy: 1.0000 - val_loss: 0.7742 - val_accuracy:
0.5455
Epoch 17/100
 - 11s - loss: 0.5469 - accuracy: 1.0000 - val_loss: 0.6203 - val_accuracy:
0.5455
Epoch 18/100
```

```
 - 11s - loss: 0.5391 - accuracy: 1.0000 - val_loss: 0.7912 - val_accuracy:
0.4545
Epoch 19/100
 - 11s - loss: 0.5314 - accuracy: 1.0000 - val_loss: 0.6077 - val_accuracy:
0.4545
Epoch 20/100
 - 11s - loss: 0.5240 - accuracy: 1.0000 - val_loss: 0.5865 - val_accuracy:
0.4545
Epoch 21/100
 - 12s - loss: 0.5171 - accuracy: 1.0000 - val_loss: 0.8150 - val_accuracy:
0.4545
Epoch 22/100
 - 11s - loss: 0.5099 - accuracy: 1.0000 - val_loss: 0.8273 - val_accuracy:
0.4545
Epoch 23/100
 - 11s - loss: 0.5027 - accuracy: 1.0000 - val_loss: 0.7176 - val_accuracy:
0.4545
Epoch 24/100
 - 11s - loss: 0.4956 - accuracy: 1.0000 - val_loss: 0.5485 - val_accuracy:
0.4545
Epoch 25/100
 - 11s - loss: 0.4887 - accuracy: 1.0000 - val_loss: 0.5436 - val_accuracy:
0.4545
Epoch 26/100
 - 11s - loss: 0.4819 - accuracy: 1.0000 - val_loss: 0.7661 - val_accuracy:
0.4545
Epoch 27/100
 - 11s - loss: 0.4749 - accuracy: 1.0000 - val_loss: 0.5359 - val_accuracy:
0.4545
Epoch 28/100
 - 11s - loss: 0.4683 - accuracy: 1.0000 - val_loss: 0.7772 - val_accuracy:
0.4545
Epoch 29/100
 - 11s - loss: 0.4616 - accuracy: 1.0000 - val_loss: 0.5399 - val_accuracy:
0.4545
Epoch 30/100
 - 11s - loss: 0.4550 - accuracy: 1.0000 - val_loss: 0.9671 - val_accuracy:
0.4545
Epoch 31/100
 - 12s - loss: 0.4484 - accuracy: 1.0000 - val_loss: 0.7020 - val_accuracy:
0.4545
Epoch 32/100
 - 11s - loss: 0.4422 - accuracy: 1.0000 - val_loss: 0.4989 - val_accuracy:
0.4545
Epoch 33/100
 - 11s - loss: 0.4358 - accuracy: 1.0000 - val_loss: 0.6975 - val_accuracy:
0.4545
Epoch 34/100
```

```
 - 11s - loss: 0.4295 - accuracy: 1.0000 - val_loss: 0.4781 - val_accuracy:
0.4545
Epoch 35/100
 - 11s - loss: 0.4237 - accuracy: 1.0000 - val_loss: 0.4681 - val_accuracy:
0.5455
Epoch 36/100
 - 11s - loss: 0.4175 - accuracy: 1.0000 - val_loss: 0.4612 - val_accuracy:
0.5455
Epoch 37/100
 - 11s - loss: 0.4113 - accuracy: 1.0000 - val_loss: 0.9849 - val_accuracy:
0.5455
Epoch 38/100
 - 11s - loss: 0.4059 - accuracy: 1.0000 - val_loss: 0.9967 - val_accuracy:
0.5455
Epoch 39/100
 - 11s - loss: 0.3995 - accuracy: 1.0000 - val_loss: 1.0080 - val_accuracy:
0.5455
Epoch 40/100
 - 11s - loss: 0.3942 - accuracy: 1.0000 - val_loss: 0.4689 - val_accuracy:
0.5455
Epoch 41/100
 - 11s - loss: 0.3883 - accuracy: 1.0000 - val_loss: 0.4372 - val_accuracy:
0.5455
Epoch 42/100
 - 12s - loss: 0.3827 - accuracy: 1.0000 - val_loss: 0.4231 - val_accuracy:
0.5455
Epoch 43/100
 - 11s - loss: 0.3772 - accuracy: 1.0000 - val_loss: 0.4139 - val_accuracy:
0.5455
Epoch 44/100
 - 11s - loss: 0.3720 - accuracy: 1.0000 - val_loss: 1.0637 - val_accuracy:
0.5455
Epoch 45/100
 - 11s - loss: 0.3664 - accuracy: 1.0000 - val_loss: 0.4110 - val_accuracy:
0.5455
Epoch 46/100
 - 11s - loss: 0.3611 - accuracy: 1.0000 - val_loss: 0.4318 - val_accuracy:
0.5455
Epoch 47/100
 - 11s - loss: 0.3559 - accuracy: 1.0000 - val_loss: 1.1004 - val_accuracy:
0.5455
Epoch 48/100
 - 12s - loss: 0.3504 - accuracy: 1.0000 - val_loss: 1.1128 - val_accuracy:
0.5455
Epoch 49/100
 - 11s - loss: 0.3458 - accuracy: 1.0000 - val_loss: 1.2251 - val_accuracy:
0.5455
Epoch 50/100
```

```
 - 11s - loss: 0.3403 - accuracy: 1.0000 - val_loss: 0.6532 - val_accuracy:
0.5455
Epoch 51/100
 - 11s - loss: 0.3354 - accuracy: 1.0000 - val_loss: 0.4026 - val_accuracy:
0.5455
Epoch 52/100
 - 11s - loss: 0.3303 - accuracy: 1.0000 - val_loss: 0.3774 - val_accuracy:
0.5455
Epoch 53/100
 - 12s - loss: 0.3256 - accuracy: 1.0000 - val_loss: 0.6458 - val_accuracy:
0.5455
Epoch 54/100
 - 11s - loss: 0.3206 - accuracy: 1.0000 - val_loss: 0.3574 - val_accuracy:
0.5455
Epoch 55/100
 - 11s - loss: 0.3159 - accuracy: 1.0000 - val_loss: 0.8405 - val_accuracy:
0.5455
Epoch 56/100
 - 11s - loss: 0.3114 - accuracy: 1.0000 - val_loss: 0.9633 - val_accuracy:
0.5455
Epoch 57/100
 - 11s - loss: 0.3068 - accuracy: 1.0000 - val_loss: 0.3705 - val_accuracy:
0.5455
Epoch 58/100
 - 12s - loss: 0.3021 - accuracy: 1.0000 - val_loss: 0.3654 - val_accuracy:
0.5455
Epoch 59/100
 - 11s - loss: 0.2973 - accuracy: 1.0000 - val_loss: 0.3222 - val_accuracy:
0.5455
Epoch 60/100
 - 11s - loss: 0.2932 - accuracy: 1.0000 - val_loss: 0.3149 - val_accuracy:
0.5455
Epoch 61/100
 - 11s - loss: 0.2893 - accuracy: 1.0000 - val_loss: 0.3307 - val_accuracy:
0.5455
Epoch 62/100
 - 11s - loss: 0.2846 - accuracy: 1.0000 - val_loss: 0.3259 - val_accuracy:
0.5455
Epoch 63/100
 - 11s - loss: 0.2802 - accuracy: 1.0000 - val_loss: 0.6215 - val_accuracy:
0.5455
Epoch 64/100
 - 11s - loss: 0.2759 - accuracy: 1.0000 - val_loss: 1.3193 - val_accuracy:
0.5455
Epoch 65/100
 - 11s - loss: 0.2719 - accuracy: 1.0000 - val_loss: 1.0281 - val_accuracy:
0.5455
Epoch 66/100
```

```
 - 11s - loss: 0.2679 - accuracy: 1.0000 - val_loss: 1.0359 - val_accuracy:
0.5455
Epoch 67/100
 - 11s - loss: 0.2642 - accuracy: 1.0000 - val_loss: 1.3675 - val_accuracy:
0.5455
Epoch 68/100
 - 11s - loss: 0.2598 - accuracy: 1.0000 - val_loss: 1.0507 - val_accuracy:
0.5455
Epoch 69/100
 - 11s - loss: 0.2558 - accuracy: 1.0000 - val_loss: 1.3870 - val_accuracy:
0.5455
Epoch 70/100
 - 11s - loss: 0.2523 - accuracy: 1.0000 - val_loss: 1.4004 - val_accuracy:
0.5455
Epoch 71/100
 - 11s - loss: 0.2483 - accuracy: 1.0000 - val_loss: 0.6025 - val_accuracy:
0.5455
Epoch 72/100
 - 11s - loss: 0.2447 - accuracy: 1.0000 - val_loss: 1.4284 - val_accuracy:
0.5455
Epoch 73/100
 - 11s - loss: 0.2408 - accuracy: 1.0000 - val_loss: 1.4421 - val_accuracy:
0.5455
Epoch 74/100
 - 11s - loss: 0.2373 - accuracy: 1.0000 - val_loss: 0.2926 - val_accuracy:
0.5455
Epoch 75/100
 - 11s - loss: 0.2336 - accuracy: 1.0000 - val_loss: 1.4699 - val_accuracy:
0.5455
Epoch 76/100
 - 11s - loss: 0.2303 - accuracy: 1.0000 - val_loss: 1.4846 - val_accuracy:
0.5455
Epoch 77/100
 - 11s - loss: 0.2265 - accuracy: 1.0000 - val_loss: 0.9415 - val_accuracy:
0.5455
Epoch 78/100
 - 11s - loss: 0.2231 - accuracy: 1.0000 - val_loss: 1.5196 - val_accuracy:
0.5455
Epoch 79/100
 - 11s - loss: 0.2200 - accuracy: 1.0000 - val_loss: 0.2532 - val_accuracy:
0.5455
Epoch 80/100
 - 11s - loss: 0.2167 - accuracy: 1.0000 - val_loss: 0.2372 - val_accuracy:
0.5455
Epoch 81/100
 - 11s - loss: 0.2132 - accuracy: 1.0000 - val_loss: 1.7218 - val_accuracy:
0.5455
Epoch 82/100
```

```
 - 11s - loss: 0.2101 - accuracy: 1.0000 - val_loss: 0.2297 - val_accuracy:
0.5455
Epoch 83/100
 - 11s - loss: 0.2068 - accuracy: 1.0000 - val_loss: 0.2384 - val_accuracy:
0.5455
Epoch 84/100
 - 11s - loss: 0.2036 - accuracy: 1.0000 - val_loss: 1.6039 - val_accuracy:
0.5455
Epoch 85/100
 - 12s - loss: 0.2008 - accuracy: 1.0000 - val_loss: 1.6118 - val_accuracy:
0.5455
Epoch 86/100
 - 11s - loss: 0.1975 - accuracy: 1.0000 - val_loss: 1.8041 - val_accuracy:
0.5455
Epoch 87/100
 - 11s - loss: 0.1944 - accuracy: 1.0000 - val_loss: 1.8206 - val_accuracy:
0.5455
Epoch 88/100
 - 11s - loss: 0.1913 - accuracy: 1.0000 - val_loss: 1.6622 - val_accuracy:
0.5455
Epoch 89/100
 - 11s - loss: 0.1886 - accuracy: 1.0000 - val_loss: 0.9975 - val_accuracy:
0.5455
Epoch 90/100
 - 11s - loss: 0.1858 - accuracy: 1.0000 - val_loss: 0.2335 - val_accuracy:
0.5455
Epoch 91/100
 - 11s - loss: 0.1830 - accuracy: 1.0000 - val_loss: 0.1901 - val_accuracy:
0.5455
Epoch 92/100
 - 11s - loss: 0.1802 - accuracy: 1.0000 - val_loss: 0.5549 - val_accuracy:
0.5455
Epoch 93/100
 - 11s - loss: 0.1772 - accuracy: 1.0000 - val_loss: 1.9209 - val_accuracy:
0.5455
Epoch 94/100
 - 11s - loss: 0.1746 - accuracy: 1.0000 - val_loss: 0.1810 - val_accuracy:
0.5455
Epoch 95/100
 - 11s - loss: 0.1720 - accuracy: 1.0000 - val_loss: 1.0254 - val_accuracy:
0.5455
Epoch 96/100
 - 12s - loss: 0.1693 - accuracy: 1.0000 - val_loss: 0.1751 - val_accuracy:
0.5455
Epoch 97/100
 - 11s - loss: 0.1665 - accuracy: 1.0000 - val_loss: 1.2755 - val_accuracy:
0.5455
Epoch 98/100
```

```
 - 11s - loss: 0.1642 - accuracy: 1.0000 - val_loss: 1.8010 - val_accuracy:
0.5455
Epoch 99/100
 - 11s - loss: 0.1614 - accuracy: 1.0000 - val_loss: 0.1660 - val_accuracy:
0.5455
Epoch 100/100
 - 11s - loss: 0.1591 - accuracy: 1.0000 - val_loss: 1.8368 - val_accuracy:
0.5455
```

[25]:
```
<keras.callbacks.callbacks.History at 0x7f448a6e2ef0>
```

[0]:
```python
def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):

    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')

    print(cm)

    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    fmt = '.2f' if normalize else 'd'
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], fmt),
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
```

[27]:
```python
newModel.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 28, 28, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0
_________________________________________________________________
flatten (Flatten)            (None, 25088)             0
_________________________________________________________________
fc1 (Dense)                  (None, 4096)              102764544
_________________________________________________________________
fc2 (Dense)                  (None, 4096)              16781312
_________________________________________________________________
predictions (Dense)          (None, 1000)              4097000
_________________________________________________________________
dense_1 (Dense)              (None, 2)                 2002
=================================================================
Total params: 138,359,546
Trainable params: 2,002
```

```
Non-trainable params: 138,357,544
_________________________________________________________________
```

[28]:
```python
test_imgs, test_labels = next(test_batches)


test_labels = test_labels[:,0]

test_labels
```

[28]:
```
array([1., 0., 1., 1.], dtype=float32)
```

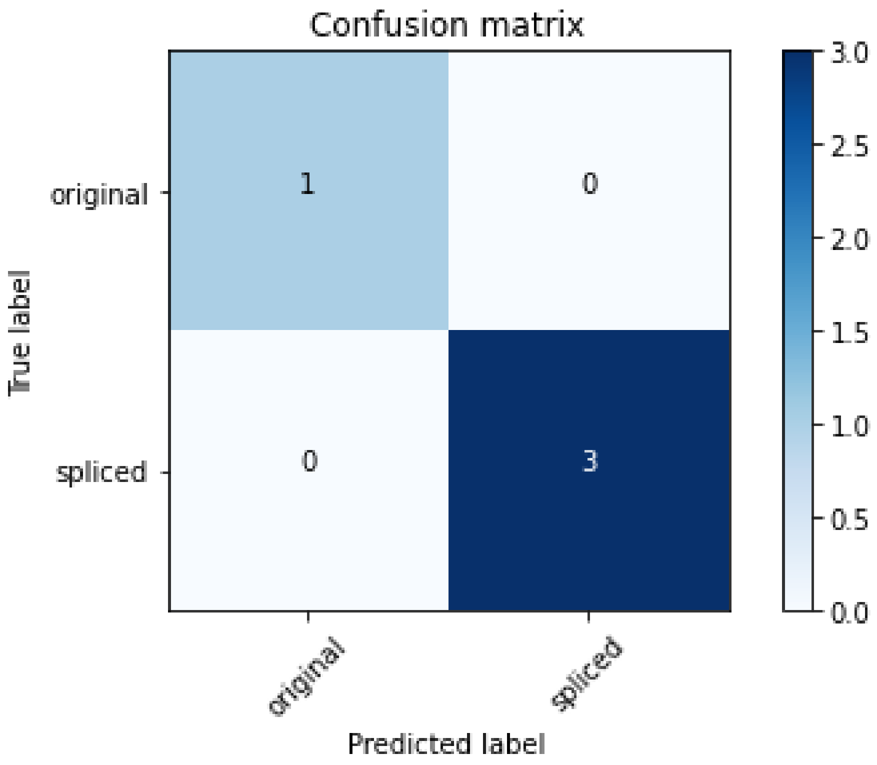
[0]:
```python
predections = newModel.predict_generator(test_batches, steps=1, verbose=0)
```

[30]:
```python
print (test_labels)
print (np.round(predections[:,0]))

cn =  confusion_matrix(test_labels, np.round(predections[:,0]))
```

```
[1. 0. 1. 1.]
[1. 0. 1. 1.]
```

[31]:
```python
cn_plot_labels = ['original', 'spliced']


plot_confusion_matrix(cn , cn_plot_labels, title='Confusion matrix')
```

```
Confusion matrix, without normalization
[[1 0]
 [0 3]]
```


Confusion matrix
True label
original
spliced
Predicted label
original
spliced
1
0
0
3
3.0
2.5
2.0
1.5
1.0
0.5
0.0