Importing required packages

In [1]:

```
import os
import cv2
import gc
from skimage import color, data, restoration
import cv2
import numpy as np
from skimage.restoration import estimate_sigma
from skimage.filters import median
import config
import imutils
```

adding function that process the input images

In [2]:

```
def weiner_noise_reduction(img):
    # data.astronaut()
    img = color.rgb2gray(img)
    from scipy.signal import convolve2d
    psf = np.ones((5, 5)) / 25
    img = convolve2d(img, psf, 'same')
    img += 0.1 * img.std() * np.random.standard_normal(img.shape)
    deconvolved_img = restoration.wiener(img, psf, 1100)

    return deconvolved_img



def estimate_noise(img):
    # img = cv2.imread(image_path)
    return estimate_sigma(img, multichannel=True, average_sigmas=True)


def preprocess_image(image):
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

    enoise = estimate_noise(image)
    noise_free_image = weiner_noise_reduction(image)
    gray = cv2.cvtColor(image, cv2.COLOR_RGB2GRAY)
    fingerprint = gray - noise_free_image
    fingerprint = fingerprint / 255
    filtered_img = median(fingerprint, selem=None, out=None, mask=None, shift_x=Fal
                          shift_y=False, mode='nearest', cval=0.0, behavior='rank')
    colored = cv2.cvtColor(filtered_img, cv2.COLOR_GRAY2BGR)
    # print('-----------------')
    # cv2.imshow('filtered_image', filtered_img)
    # colored = cv2.cvtColor(filtered_img, cv2.COLOR_GRAY2BGR)
    # print(colored)
    # cv2.imshow('colored', colored)
    return colored
```

function to process casia one image dataset and store them into numpy matrix with file name **'dataset/np_casia_one_forged.npy'**

In [3]:

```python
def prepare_casia_one_dataset():
    casia_one_au_arr = []
    casia_one_forged_arr = []

    # np.save('data.npy', num_arr) # save
    for image in os.listdir(CASIA_ONE_AUTHENTIC_PATH):
        imagepath = os.path.join(CASIA_ONE_AUTHENTIC_PATH, image)
        cv_image = cv2.imread(imagepath)
        print(str(image) + 'processing...')
        h, w = cv_image.shape[:2]
        if h != 256 and w != 384:
            continue
            # cv_image = imutils.resize(cv_image, width=384, height=256)
        if h == 256 and w == 384:
            processed_image = preprocess_image(cv_image)
            casia_one_au_arr.append(np.array(processed_image))
        else:
            print('Dimention mismatch')

    np_casia_one_au = np.array(casia_one_au_arr)
    np.save('dataset/np_casia_one_au.npy', np_casia_one_au)  # save
    print('CASIA1 Authentic Data Processed..')
    gc.collect()

    for image in os.listdir(CASIA_ONE_FORGED_PATH):
        imagepath = os.path.join(CASIA_ONE_FORGED_PATH, image)
        cv_image = cv2.imread(imagepath)
        print(str(image) + 'processing...')
        h, w = cv_image.shape[:2]
        if h != 256 and w != 384:
            continue
            # cv_image = imutils.resize(cv_image, width=384, height=256)
        if h == 256 and w == 384:
            processed_image = preprocess_image(cv_image)
            casia_one_forged_arr.append(np.array(processed_image))
        else:
            print('Dimention mismatch')

    np_casia_one_forged = np.array(casia_one_forged_arr)
    np.save('dataset/np_casia_one_forged.npy', np_casia_one_forged)  # save
    print('CASIA1 Forged Data Processed..')
    gc.collect()
```

CASIA 1 database contains **800 authentic** and **921 forged** images.

The size s **384X256** pixels.

In [4]:

```
CASIA_ONE_AUTHENTIC_PATH = 'casia-dataset/CASIA1/Au/'
CASIA_ONE_FORGED_PATH = 'casia-dataset/CASIA1/Sp/'
```

Checking... is there already process numpy array exist or not. If not exists then creating new one.

In [5]:

```
filename = os.path.join('dataset', 'np_casia_one_forged.npy')
if not os.path.exists(filename):
    print('Processing Casia I dataset...')
    prepare_casia_one_dataset()
else:
    print(filename + ' already processed...')
```

```
dataset/np_casia_one_forged.npy already processed...
```

The **CASIA 2** database contains more than **7400 authentic** and **5000 forged images**. The images are in either JPEG, TIFF, or BMP format.

In [6]:

```
CASIA_TWO_AUTHENTIC_PATH = 'casia-dataset/CASIA2/Au/'
CASIA_TWO_FORGED_PATH = 'casia-dataset/CASIA2/Tp/'
```

function to process casio two dataset

In [7]:

```python
def prepare_casia_two_dataset():
    casia_two_au_arr = []
    casia_two_forged_arr = []

    # np.save('data.npy', num_arr) # save
    for image in os.listdir(CASIA_TWO_AUTHENTIC_PATH):
        imagepath = os.path.join(CASIA_TWO_AUTHENTIC_PATH, image)
        cv_image = cv2.imread(imagepath)
        try:
            print(str(image) + 'processing...')
            h, w = cv_image.shape[:2]
            if h != 256 and w != 384:
                continue
                # cv_image = imutils.resize(cv_image, width=384, height=256)
            if h == 256 and w == 384:
                processed_image = preprocess_image(cv_image)
                casia_two_au_arr.append(np.array(processed_image))
            else:
                print('Dimention mismatch')
        except Exception as err:
            print(err)

    np_casia_two_au = np.array(casia_two_au_arr)
    np.save('dataset/np_casia_two_au.npy', np_casia_two_au)  # save
    print('CASIA2 Authentic Data Processed..')
    gc.collect()

    for image in os.listdir(CASIA_TWO_FORGED_PATH):
        imagepath = os.path.join(CASIA_TWO_FORGED_PATH, image)
        cv_image = cv2.imread(imagepath)
        try:
            print(str(image) + 'processing...')
            h, w = cv_image.shape[:2]
            if h != 256 and w != 384:
                continue
                # cv_image = imutils.resize(cv_image, width=384, height=256)
            if h == 256 and w == 384:
                processed_image = preprocess_image(cv_image)
                casia_two_forged_arr.append(np.array(processed_image))
            else:
                print('Dimention mismatch')
        except Exception as err:
            print(err)


    np_casia_two_forged = np.array(casia_two_forged_arr)
    np.save('dataset/np_casia_two_forged.npy', np_casia_two_forged)  # save
    print('CASIA2 Forged Data Processed..')
    gc.collect()
```

Checking... is there already process numpy array exist or not. If not exists then creating new one.

In [8]:

```
ilename = os.path.join('dataset', 'np_casia_two_forged.npy')
if not os.path.exists(filename):
    print('Processing Casia II dataset...')
    prepare_casia_two_dataset()
else:
    print(filename + ' already processed...')
```

```
dataset/np_casia_one_forged.npy already processed...
```

Training the keras classifier

In [9]:

```
import keras
from keras import Model, Sequential, optimizers, applications
from keras.applications import ResNet50
from keras.layers import GlobalAveragePooling2D, Dropout, Dense, Flatten
from keras_applications import resnet50
from keras import backend as K
import matplotlib.pyplot as plt
import numpy as np
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
import config
```

```
Using TensorFlow backend.
/usr/local/lib/python3.5/dist-packages/tensorflow/python/framework/dty
pes.py:526: FutureWarning: Passing (type, 1) or '1type' as a synonym o
f type is deprecated; in a future version of numpy, it will be underst
ood as (type, (1,)) / '(1,)type'.
  _np_qint8 = np.dtype([("qint8", np.int8, 1)])
/usr/local/lib/python3.5/dist-packages/tensorflow/python/framework/dty
pes.py:527: FutureWarning: Passing (type, 1) or '1type' as a synonym o
f type is deprecated; in a future version of numpy, it will be underst
ood as (type, (1,)) / '(1,)type'.
  _np_quint8 = np.dtype([("quint8", np.uint8, 1)])
/usr/local/lib/python3.5/dist-packages/tensorflow/python/framework/dty
pes.py:528: FutureWarning: Passing (type, 1) or '1type' as a synonym o
f type is deprecated; in a future version of numpy, it will be underst
ood as (type, (1,)) / '(1,)type'.
  _np_qint16 = np.dtype([("qint16", np.int16, 1)])
/usr/local/lib/python3.5/dist-packages/tensorflow/python/framework/dty
pes.py:529: FutureWarning: Passing (type, 1) or '1type' as a synonym o
f type is deprecated; in a future version of numpy, it will be underst
ood as (type, (1,)) / '(1,)type'.
  _np_quint16 = np.dtype([("quint16", np.uint16, 1)])
/usr/local/lib/python3.5/dist-packages/tensorflow/python/framework/dty
pes.py:530: FutureWarning: Passing (type, 1) or '1type' as a synonym o
f type is deprecated; in a future version of numpy, it will be underst
ood as (type, (1,)) / '(1,)type'.
  _np_qint32 = np.dtype([("qint32", np.int32, 1)])
/usr/local/lib/python3.5/dist-packages/tensorflow/python/framework/dty
pes.py:535: FutureWarning: Passing (type, 1) or '1type' as a synonym o
f type is deprecated; in a future version of numpy, it will be underst
ood as (type, (1,)) / '(1,)type'.
  np_resource = np.dtype([("resource", np.ubyte, 1)])
```

build the VGG16 network

In [10]:

```
img_height = 256
img_width = 384

# build the VGG16 network
model = applications.VGG16(weights='imagenet', include_top=False, input_shape=(img_
```

```
WARNING:tensorflow:From /usr/local/lib/python3.5/dist-packages/tensorf
low/python/framework/op_def_library.py:263: colocate_with (from tensor
flow.python.framework.ops) is deprecated and will be removed in a futu
re version.
Instructions for updating:
Colocations handled automatically by placer.
```

build a classifier model to put on top of the convolutional model

In [11]:

```
top_model = Sequential()
top_model.add(Flatten(input_shape=model.output_shape[1:]))
top_model.add(Dense(256, activation='relu'))
top_model.add(Dropout(0.5))
top_model.add(Dense(1, activation='sigmoid'))
```

```
WARNING:tensorflow:From /usr/local/lib/python3.5/dist-packages/keras/b
ackend/tensorflow_backend.py:3445: calling dropout (from tensorflow.py
thon.ops.nn_ops) with keep_prob is deprecated and will be removed in a
future version.
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate
= 1 - keep_prob`.
```

add the model on top of the convolutional base

In [12]:

```
# model.add(top_model) this throws error alternative is below

new_model = Sequential() #new model
for layer in model.layers:
    new_model.add(layer)

new_model.add(top_model) # now this works
```

set the first 25 layers (up to the last conv block) to non-trainable (weights will not be updated)

LOCK THE TOP CONV LAYERS

In [13]:

```
for layer in new_model.layers[:15]:
    layer.trainable = False

print('Model loaded.')

print(new_model.summary())
```

```
Model loaded.
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
block1_conv1 (Conv2D)        (None, 256, 384, 64)      1792      
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 256, 384, 64)      36928     
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 128, 192, 64)      0         
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 128, 192, 128)     73856     
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 128, 192, 128)     147584    
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 64, 96, 128)       0         
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 64, 96, 256)       295168    
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 64, 96, 256)       590080    
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 64, 96, 256)       590080    
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 32, 48, 256)       0         
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 32, 48, 512)       1180160   
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 32, 48, 512)       2359808   
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 32, 48, 512)       2359808   
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 16, 24, 512)       0         
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 16, 24, 512)       2359808   
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 16, 24, 512)       2359808   
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 16, 24, 512)       2359808   
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 8, 12, 512)        0         
_________________________________________________________________
sequential_1 (Sequential)    (None, 1)                 12583425  
=================================================================
Total params: 27,298,113
Trainable params: 17,303,041
Non-trainable params: 9,995,072
_________________________________________________________________
None
```