Training Data

In [1]:

```
!wget --header="Host: uca90aae79a2a63c4a7676237a37.dl.dropboxusercontent.com" --header="User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_5) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/85.0.4183.121 Safari/537.36" --header="Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,ap
ation/signed-exchange;v=b3;q=0.9" --header="Accept-Language: en,en-US;q=0.9,fr;q=0.8" --header="Referer: https://www.dropbox.com/"
"https://uca90aae79a2a63c4a7676237a37.dl.dropboxusercontent.com/cd/0/get/BB21N2JqsWd89mk6Bp_2RDw3BN
XKxzFo-aVPylwg7BdJEoXyrSTtBR7980zJbqZjHQY6GwvuyIAO8ohdUHRT09bbmqQxfF1qgMDiCrFC7Q/file?
_download_id=755669963604364929128501311126350512870306693387303132817591595&_notify_domain=www.dro
.com&dl=1" -c -O 'phase-01-training.tar.gz'
```

```
--2020-10-23 16:49:34--
https://uca90aae79a2a63c4a7676237a37.dl.dropboxusercontent.com/cd/0/get/BB21N2JqsWd89mk6Bp_2RDw3BNa
KxzFo-aVPylwg7BdJEoXyrSTtBR7980zJbqZjHQY6GwvuyIAO8ohdUHRT09bbmqQxfF1qgMDiCrFC7Q/file?
_download_id=755669963604364929128501311126350512870306693387303132817591595&_notify_domain=www.dro
.com&dl=1
Resolving uca90aae79a2a63c4a7676237a37.dl.dropboxusercontent.com
(uca90aae79a2a63c4a7676237a37.dl.dropboxusercontent.com)... 162.125.65.15,
2620:100:6021:15::a27d:410f
Connecting to uca90aae79a2a63c4a7676237a37.dl.dropboxusercontent.com
(uca90aae79a2a63c4a7676237a37.dl.dropboxusercontent.com)|162.125.65.15|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2749754446 (2.6G) [application/binary]
Saving to: ‘phase-01-training.tar.gz’

phase-01-training.t 100%[===================>]   2.56G  29.2MB/s    in 95s

2020-10-23 16:51:11 (27.5 MB/s) - ‘phase-01-training.tar.gz’ saved [2749754446/2749754446]
```

In [ ]:

```
!wget --header="Host: storage.googleapis.com" --header="User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_5) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/85.0.4183.121 Safari/537.36" --header="Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,ap
ation/signed-exchange;v=b3;q=0.9" --header="Accept-Language: en,en-US;q=0.9,fr;q=0.8" --header="Referer: https://www.kaggle.com/" "https://storage.googleapis.com/kaggle-data-
sets/59500/115146/bundle/archive.zip?X-Goog-Algorithm=GOOG4-RSA-SHA256&X-Goog-Credential=gcp-kaggl
e-com%40kaggle-161607.iam.gserviceaccount.com%2F20201003%2Fauto%2Fstorage%2Fgoog4_request&X-Goog-D
ate=20201003T083507Z&X-Goog-Expires=259199&X-Goog-SignedHeaders=host&X-Goog-
Signature=372af04a74f9495d024520b3524949d5f25f8829e98b942ce871b70ac61536d39299fa4a786976e7ee7e0a2a6
50c337715a3740d896de06e358cc906f6e55f3ab5c1f6a4d58dcd6ead653d50c02b49c4235f0ffaf92a6329c8d1f7b4677a
ee3453d4416c649c23bd005dacf9a3104f388bf4fdcd13030045f0680b48b8ba95f82787c8fd31af7515b8a963933cbc9e1
3a7b5ebac625f826046060b824bad79a90593652fb7d074324945d76b70d4f35b4a981e9707d15383eed1cb6be1f2ed5884
4143f915470af1c6fd6e14738fd6dffc62f5a23d3625086e21ce95b3d1e8bfbdc13198a3d03449948e00520cf626e789295
ef1144d" -c -O 'archive.zip'
```

```
--2020-10-06 15:15:00--  https://storage.googleapis.com/kaggle-data-
sets/59500/115146/bundle/archive.zip?X-Goog-Algorithm=GOOG4-RSA-SHA256&X-Goog-Credential=gcp-kaggl
e-com%40kaggle-161607.iam.gserviceaccount.com%2F20201003%2Fauto%2Fstorage%2Fgoog4_request&X-Goog-D
ate=20201003T083507Z&X-Goog-Expires=259199&X-Goog-SignedHeaders=host&X-Goog-
Signature=372af04a74f9495d024520b3524949d5f25f8829e98b942ce871b70ac61536d39299fa4a786976e7ee7e0a2a6
50c337715a3740d896de06e358cc906f6e55f3ab5c1f6a4d58dcd6ead653d50c02b49c4235f0ffaf92a6329c8d1f7b4677a
ee3453d4416c649c23bd005dacf9a3104f388bf4fdcd13030045f0680b48b8ba95f82787c8fd31af7515b8a963933cbc9e1
3a7b5ebac625f826046060b824bad79a90593652fb7d074324945d76b70d4f35b4a981e9707d15383eed1cb6be1f2ed5884
4143f915470af1c6fd6e14738fd6dffc62f5a23d3625086e21ce95b3d1e8bfbdc13198a3d03449948e00520cf626e789295
ef1144d
Resolving storage.googleapis.com (storage.googleapis.com)... 74.125.20.128, 74.125.142.128,
74.125.195.128, ...
Connecting to storage.googleapis.com (storage.googleapis.com)|74.125.20.128|:443... connected.
HTTP request sent, awaiting response... 400 Bad Request
2020-10-06 15:15:00 ERROR 400: Bad Request.
```

In [ ]:

```
import os
os.environ['KAGGLE_CONFIG_DIR'] = "/content"
%cd /content
```

```
/content
```

In [ ]:

```
!kaggle datasets download -d sophatvathana/casia-dataset
```

```
Warning: Your Kaggle API key is readable by other users on this system! To fix this, you can run '
chmod 600 /content/kaggle.json'
Downloading casia-dataset.zip to /content
  3% 152M/5.22G [00:03<01:49, 49.5MB/s]
User cancelled operation
Error in atexit._run_exitfuncs:^C
```

Unzipping Data

In [2]:

```
!tar -xf  '/content/phase-01-training.tar.gz' -C '/content/'
```

In [ ]:

```
!unzip -qq '/content/archive.zip'
```

In [ ]:

```
!unzip -qq '/content/casia-dataset.zip'
```

In [ ]:

```
!rm -rf '/content/CASIA1' '/content/CASIA2' '/content/casia' '/content/dataset-dist'
```

Importing Libraries

In [2]:

```
!pip install segmentation-models
```

```
Collecting segmentation-models
  Downloading
https://files.pythonhosted.org/packages/da/b9/4a183518c21689a56b834eaaa45cad242d9ec09a4360b5b10139f
3f4/segmentation_models-1.0.1-py3-none-any.whl
Collecting efficientnet==1.0.0
  Downloading
https://files.pythonhosted.org/packages/97/82/f3ae07316f0461417dc54affab6e86ab188a5a22f33176d352716
6e0/efficientnet-1.0.0-py3-none-any.whl
Collecting keras-applications<=1.0.8,>=1.0.7
  Downloading
https://files.pythonhosted.org/packages/71/e3/19762fdfc62877ae9102edf6342d71b28fbfd9dea3d2f96a882ce
03f/Keras_Applications-1.0.8-py3-none-any.whl (50kB)
     |████████████████████████████████| 51kB 2.7MB/s
Collecting image-classifiers==1.0.0
  Downloading
https://files.pythonhosted.org/packages/81/98/6f84720e299a4942ab80df5f76ab97b7828b24d1de5e9b2cbbe60
8b7/image_classifiers-1.0.0-py3-none-any.whl
Requirement already satisfied: scikit-image in /usr/local/lib/python3.6/dist-packages (from
efficientnet==1.0.0->segmentation-models) (0.16.2)
Requirement already satisfied: h5py in /usr/local/lib/python3.6/dist-packages (from keras-
applications<=1.0.8,>=1.0.7->segmentation-models) (2.10.0)
Requirement already satisfied: numpy>=1.9.1 in /usr/local/lib/python3.6/dist-packages (from keras-
applications<=1.0.8,>=1.0.7->segmentation-models) (1.18.5)
Requirement already satisfied: imageio>=2.3.0 in /usr/local/lib/python3.6/dist-packages (from
scikit-image->efficientnet==1.0.0->segmentation-models) (2.4.1)
Requirement already satisfied: scipy>=0.19.0 in /usr/local/lib/python3.6/dist-packages (from
scikit-image->efficientnet==1.0.0->segmentation-models) (1.4.1)
Requirement already satisfied: networkx>=2.0 in /usr/local/lib/python3.6/dist-packages (from
scikit-image->efficientnet==1.0.0->segmentation-models) (2.5)
Requirement already satisfied: matplotlib!=3.0.0,>=2.0.0 in /usr/local/lib/python3.6/dist-packages
(from scikit-image->efficientnet==1.0.0->segmentation-models) (3.2.2)
Requirement already satisfied: pillow>=4.3.0 in /usr/local/lib/python3.6/dist-packages (from
scikit-image->efficientnet==1.0.0->segmentation-models) (7.0.0)
Requirement already satisfied: PyWavelets>=0.4.0 in /usr/local/lib/python3.6/dist-packages (from
scikit-image->efficientnet==1.0.0->segmentation-models) (1.1.1)
Requirement already satisfied: six in /usr/local/lib/python3.6/dist-packages (from h5py->keras-
applications<=1.0.8,>=1.0.7->segmentation-models) (1.15.0)
Requirement already satisfied: decorator>=4.3.0 in /usr/local/lib/python3.6/dist-packages (from
networkx>=2.0->scikit-image->efficientnet==1.0.0->segmentation-models) (4.4.2)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in
/usr/local/lib/python3.6/dist-packages (from matplotlib!=3.0.0,>=2.0.0->scikit-image-
>efficientnet==1.0.0->segmentation-models) (2.4.7)
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.6/dist-packages
(from matplotlib!=3.0.0,>=2.0.0->scikit-image->efficientnet==1.0.0->segmentation-models) (2.8.1)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.6/dist-packages (from
matplotlib!=3.0.0,>=2.0.0->scikit-image->efficientnet==1.0.0->segmentation-models) (1.2.0)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.6/dist-packages (from
matplotlib!=3.0.0,>=2.0.0->scikit-image->efficientnet==1.0.0->segmentation-models) (0.10.0)
Installing collected packages: keras-applications, efficientnet, image-classifiers, segmentation-m
odels
Successfully installed efficientnet-1.0.0 image-classifiers-1.0.0 keras-applications-1.0.8
segmentation-models-1.0.1
```

In [3]:

```
import numpy as np
from prettytable import PrettyTable
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from keras.utils.np_utils import to_categorical
from keras.models import Sequential
from keras.layers import Dense, Flatten, Conv2D, MaxPool2D, Dropout
from keras.optimizers import Adam
from keras.preprocessing.image import ImageDataGenerator
from keras.callbacks import EarlyStopping,ReduceLROnPlateau
from PIL import Image, ImageChops, ImageEnhance
import os
import itertools
import seaborn as sns
import shutil
from imageio import imread
import imageio
import pandas as pd
import tqdm
import matplotlib.pyplot as plt
import seaborn as sns
from keras.applications.resnet import ResNet50, ResNet101, ResNet152
import datetime
from keras.callbacks import TensorBoard
import cv2
from skimage.transform import resize
import PIL
from numpy import save,load
from keras.optimizers import Adam,SGD
%env SM_FRAMEWORK=tf.keras
# from tensorflow import keras
import tensorflow as tf
import segmentation_models as sm
from keras import backend as K
import tensorflow_addons as tfa
from segmentation_models import Unet
from tensorflow.keras.layers import Input, Add, Dropout,Dense, Activation, ZeroPadding2D, BatchNorm
alization, Flatten, Conv2D, AveragePooling2D, MaxPooling2D, GlobalMaxPooling2D, concatenate,Conv2DT
ranspose,GlobalMaxPool2D,GlobalAveragePooling2D
from tensorflow.keras.models import Model, load_model
from tensorflow.keras.initializers import glorot_uniform
from segmentation_models.metrics import iou_score
import gc
```

```
env: SM_FRAMEWORK=tf.keras
```

```
env: SM_FRAMEWORK=tf.keras
Segmentation Models: using `tf.keras` framework.
```

In [8]:

```
 # Dice_Coeff or F1 score
def metric(y_true, y_pred):
    y_true_f = K.flatten(y_true)
    y_pred_f = K.flatten(y_pred)
    intersection = K.sum(y_true_f * y_pred_f)
    return (2. * intersection + 1) / (K.sum(y_true_f) + K.sum(y_pred_f) + 1)
```

In [ ]:

```
os.makedirs('/content/dataset-dist/phase-01/training/filtered_images/')
```

In [ ]:

```
!pip install basic-image-eda
```

```
Collecting basic-image-eda
  Downloading
https://files.pythonhosted.org/packages/a9/35/183057a3723d4c0b9c0f1431af2ee4765b0b98d57ad5dd18205d3
bf1/basic_image_eda-0.0.3-py3-none-any.whl
Installing collected packages: basic-image-eda
Successfully installed basic-image-eda-0.0.3
```

Exploratory Data Analysis

Dataset 1 (CASIA2)

In [6]:

```
def generate_ela(path,quality):
  temp_file = 'temp_file.jpg'
  # ela_filename = 'temp_ela.png'

  image = Image.open(path).convert('RGB')
  image.save(temp_file, 'JPEG', quality = quality)
  temp_image = Image.open(temp_file)

  ela_img = ImageChops.difference(image, temp_image)

  extrema = ela_img.getextrema()
  max_diff = max([ex[1] for ex in extrema])
  if max_diff == 0:
      max_diff = 1
  scale = 255.0 / max_diff

  ela_img = ImageEnhance.Brightness(ela_img).enhance(scale)

  return ela_img
```

In [ ]:

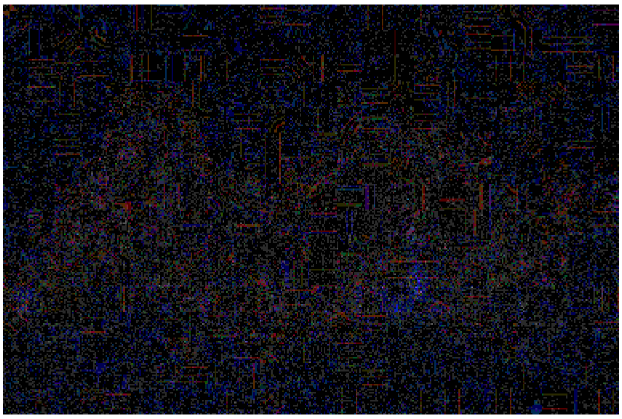
```
real_img = '/content/casia/CASIA2/Au/Au_ani_00082.jpg'
Image.open(real_img)
```

Out[ ]:

In [ ]:

```
generate_ela(real_img, 90)
```

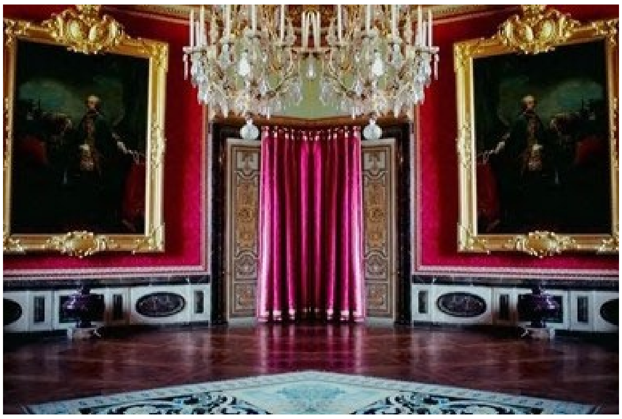
Out[ ]:



In [ ]:

```
fake_img = '/content/casia/CASIA2/Tp/Tp_D_CNN_M_N_ind00091_ind00091_10648.jpg'
Image.open(fake_img)
```

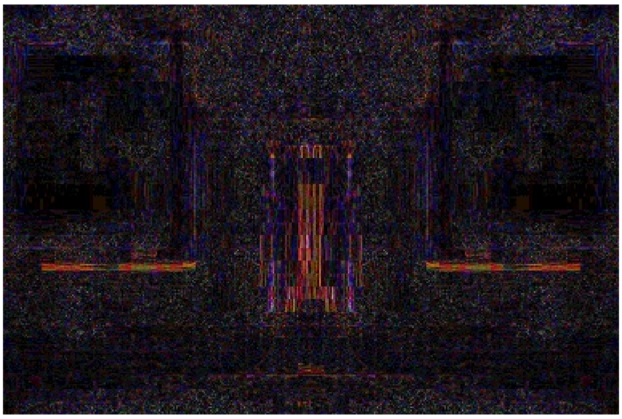
Out[ ]:



In [ ]:

```
generate_ela(fake_img, 90)
```

Out[ ]:



- What is ELA (Error level analysis)? -> Error Level Analysis (ELA) permits identifying areas within an image that are at different compression levels. With JPEG images, the entire picture should be at roughly the same level. If a section of the image is at a significantly different error level, then it likely indicates a digital modification. -> ELA highlights differences in the JPEG compression rate. Regions with uniform coloring, like a solid blue sky or a white wall, will likely have a lower ELA result (darker color) than high-contrast edges. -> Look around the picture and identify the different high-contrast edges, low-contrast edges, surfaces, and textures. Compare those areas with the ELA results. If there are significant differences, then it identifies suspicious areas that may have been digitally altered. ->Resaving a JPEG removes high-frequencies and results in less differences between high-contrast edges, textures, and surfaces. A very low quality JPEG will appear very dark. ->Scaling a picture smaller can boost high-contrast edges, making them brighter under ELA. Similarly, saving a JPEG with an Adobe product will automatically sharpen high-contrast edges and textures, making them appear much brighter than low-texture surfaces.
- Above we can observe the real image and manipulated image with corresponding ELA.

In [ ]:

```
path, dirs, files = next(os.walk("/content/casia/CASIA2/Au"))
file_count = len(files)
print('Number of Authentic images = ',file_count)

path2, dirs2, files2 = next(os.walk("/content/casia/CASIA2/Tp"))
file_count2 = len(files2)
print('Number of Fake images = ',file_count2)
```

```
Number of Authentic images =  7492
Number of Fake images =  5124
```

In [ ]:

```
from basic_image_eda import BasicImageEDA
BasicImageEDA.explore('/content/casia/CASIA2/Au', dimension_plot = True, channel_hist = True)
```

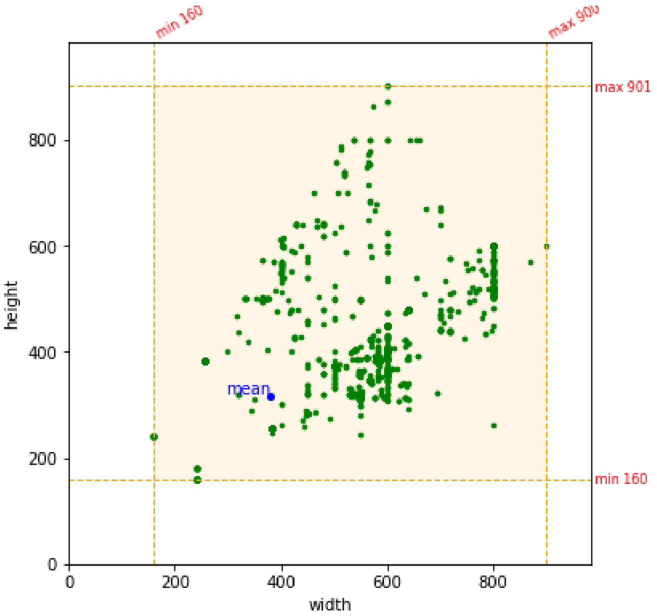
```
found 7408 images.
```

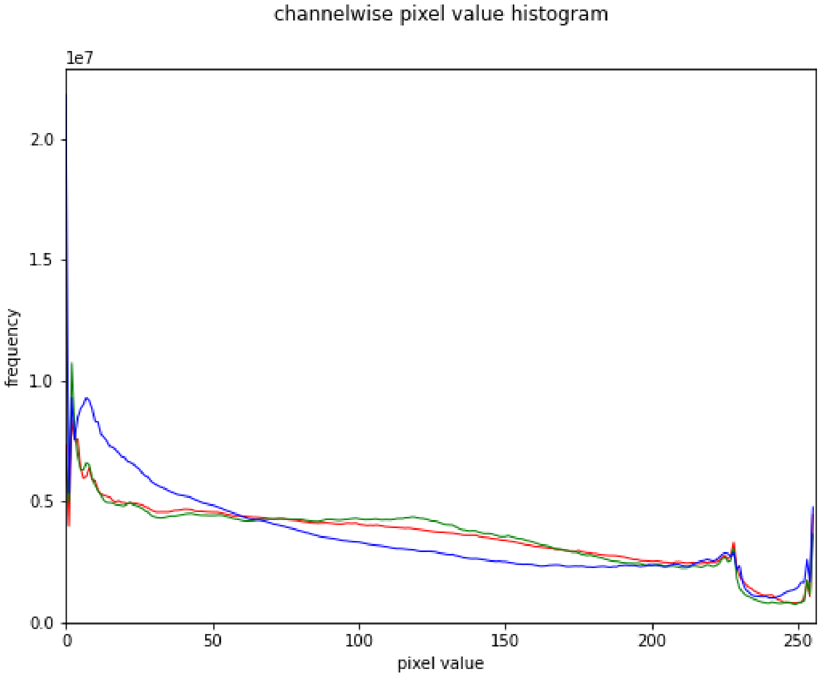
```
  0%|          | 14/7408 [00:00<00:53, 136.96it/s]
```

```
Using 4 threads. (max:4)
```

```
100%|██████████| 7408/7408 [00:55<00:00, 134.67it/s]
```

```
*--------------------------------------------------------------------------------------*
number of images                         |  7408

dtype                                    |  uint8
channels                                 |  [3]
extensions                               |  ['jpg', 'bmp']

min height                               |  160
max height                               |  901
mean height                              |  316.3999730021598
median height                            |  256

min width                                |  160
max width                                |  900
mean width                               |  381.2722732181426
median width                             |  384

mean height/width ratio                  |  0.8298530872218277
median height/width ratio                |  0.6666666666666666
recommended input size(by mean)          |  [320 384] (h x w, multiples of 8)
recommended input size(by mean)          |  [320 384] (h x w, multiples of 16)
recommended input size(by mean)          |  [320 384] (h x w, multiples of 32)

channel mean(0~1)                        |  [0.3924019  0.3833333  0.35072565]
channel std(0~1)                         |  [0.26803446 0.26113018 0.27882704]
*--------------------------------------------------------------------------------------*
eda ended in 00 hours 00 minutes 55 seconds
```





☐ remove_zero

☐ remove_maxval

Out[ ]:

```
{'channels': [3],
 'dtype': 'uint8',
 'extensions': ['jpg', 'bmp'],
 'max_h': 901,
 'max_w': 900,
 'mean': array([0.3924019 , 0.3833333 , 0.35072565], dtype=float32),
 'mean_h': 316.3999730021598,
 'mean_hw_ratio': 0.8298530872218277,
 'mean_w': 381.2722732181426,
 'median_h': 256,
 'median_hw_ratio': 0.6666666666666666,
 'median_w': 384,
 'min_h': 160,
 'min_w': 160,
 'rec_hw_size_16': array([320, 384]),
 'rec_hw_size_32': array([320, 384]),
 'rec_hw_size_8': array([320, 384]),
 'std': array([0.26803446, 0.26113018, 0.27882704], dtype=float32)}
```

In [ ]:

```
BasicImageEDA.explore('/content/casia/CASIA2/Tp', dimension_plot = True, channel_hist = True)
```

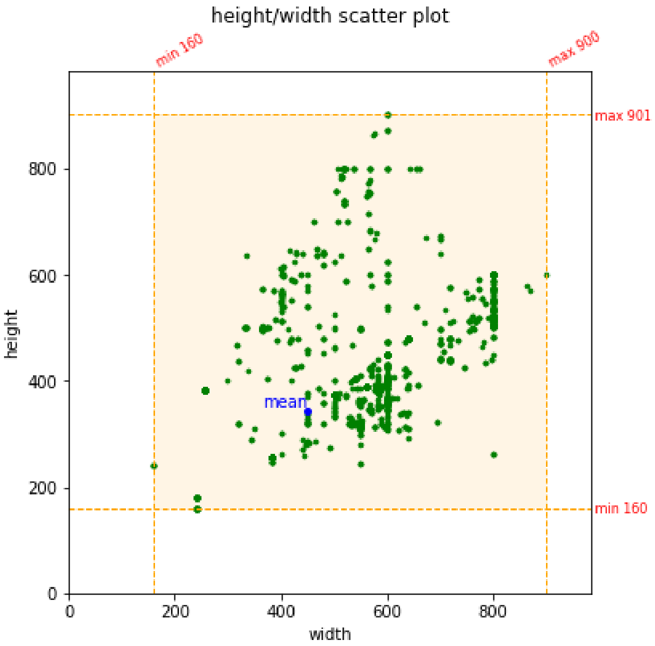
```
found 5123 images.
```

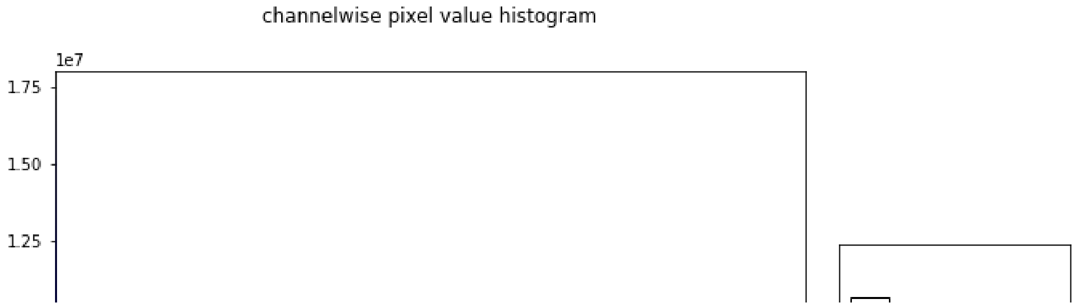
```
  0%|          | 10/5123 [00:00<00:52, 97.35it/s]
```

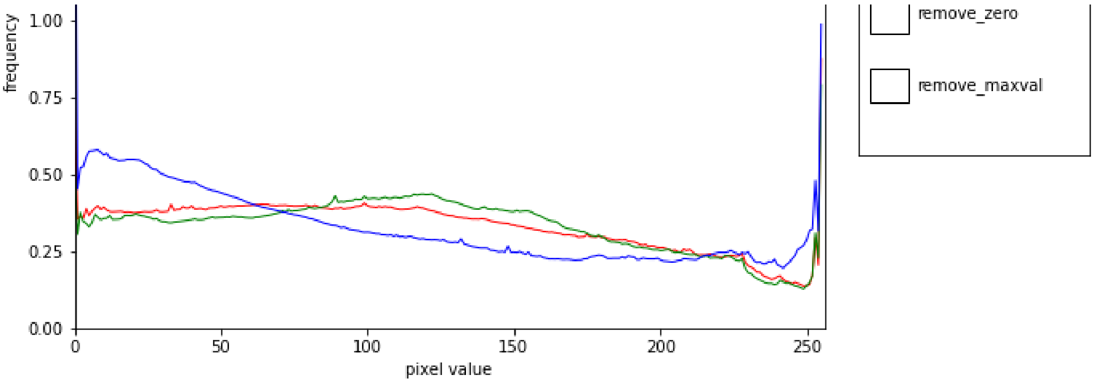
```
Using 4 threads. (max:4)
```

```
100%|██████████| 5123/5123 [00:46<00:00, 110.19it/s]
```

```
*-------------------------------------------------------------------------------------*
number of images                         |  5123

dtype                                    |  uint8
channels                                 |  [3, 4]
extensions                               |  ['jpg', 'tif']

min height                               |  160
max height                               |  901
mean height                              |  343.9918016787039
median height                            |  256

min width                                |  160
max width                                |  900
mean width                               |  450.2896740191294
median width                             |  384

mean height/width ratio                  |  0.7639344660257306
median height/width ratio                |  0.6666666666666666
recommended input size(by mean)          |  [344 448]  (h x w, multiples of 8)
recommended input size(by mean)          |  [336 448]  (h x w, multiples of 16)
recommended input size(by mean)          |  [352 448]  (h x w, multiples of 32)

channel mean(0~1)                        |  [0.44204736 0.43905905 0.39562663]
channel std(0~1)                         |  [0.2699287  0.2630554  0.29094294]
*-------------------------------------------------------------------------------------*
eda ended in 00 hours 00 minutes 46 seconds
```







Out[ ]:

```
{'channels': [3, 4],
 'dtype': 'uint8',
 'extensions': ['jpg', 'tif'],
 'max_h': 901,
 'max_w': 900,
 'mean': array([0.44204736, 0.43905905, 0.39562663], dtype=float32),
 'mean_h': 343.9918016787039,
 'mean_hw_ratio': 0.7639344660257306,
 'mean_w': 450.2896740191294,
 'median_h': 256,
 'median_hw_ratio': 0.6666666666666666,
 'median_w': 384,
 'min_h': 160,
 'min_w': 160,
 'rec_hw_size_16': array([336, 448]),
 'rec_hw_size_32': array([352, 448]),
 'rec_hw_size_8': array([344, 448]),
 'std': array([0.2699287 , 0.2630554 , 0.29094294], dtype=float32)}
```

- Above we can observe various properties of our authentic and fake image data.
- We can easily observe that image sizes are not equal.

Dataset 2 (IEEE IFS-TC Image Forensics Challenge)

In [ ]:

```
pristine_path = '/content/dataset-dist/phase-01/training/pristine'
tampered_path = '/content/dataset-dist/phase-01/training/fake'
full_path = '/content/dataset-dist/phase-01/training'
```

In [ ]:

```
path, dirs, files = next(os.walk(pristine_path))
file_count = len(files)
print('Number of Pristine images = ',file_count)

path2, dirs2, files2 = next(os.walk(tampered_path))
file_count2 = len(files2)
print('Number of tampered images and their masks = ',file_count2)
```

```
Number of Pristine images =  1050
Number of tampered images and their masks =  901
```

In [ ]:

```
pris_img = '/content/dataset-dist/phase-01/training/pristine/0001d52e2fd94f30c2bca0449763a752.png'
Image.open(pris_img)
```

Out[ ]:

In [ ]:

```
tamp_img = '/content/dataset-dist/phase-01/training/fake/010543abfbd0db1e9aa1b24604336e0c.png'
Image.open(tamp_img)
```

Out[ ]:

In [ ]:

```
tamp_mask_img = '/content/dataset-dist/phase-01/training/fake/010543abfbd0db1e9aa1b24604336e0c.mask.png'
Image.open(tamp_mask_img)
```

Out[ ]:

- The dataset is divided into 2 parts pristie i.e. original images and second part is fake images which are tampered.
- Tampered images are also provided with corresponding masks.
- The dataset containes 1050 prisitne images and 450 tampered images with their 450 masks.

In [ ]:

```
BasicImageEDA.explore('/content/dataset-dist/phase-01/training/pristine', dimension_plot = True,
channel_hist = True)
```

```
found 1050 images.
```

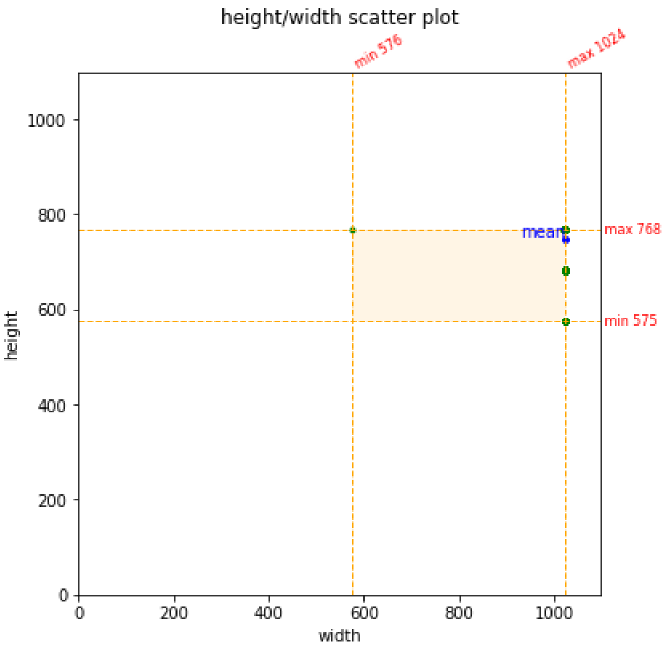
```
  0%|          | 0/1050 [00:00<?, ?it/s]
```

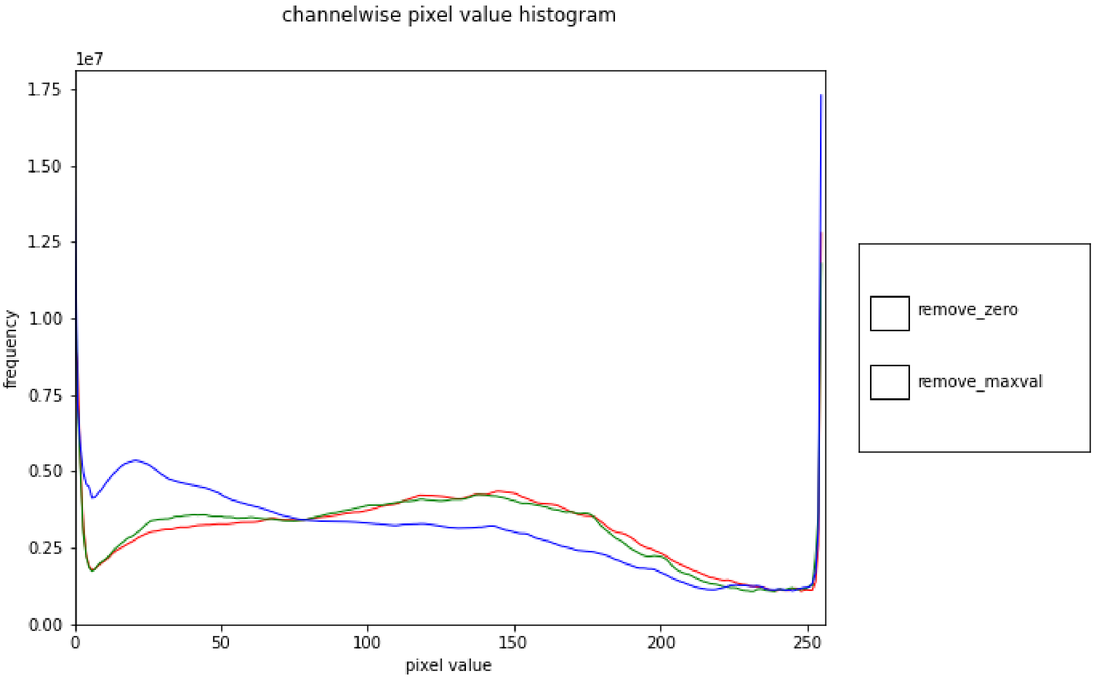
```
Using 4 threads. (max:4)
```

```
100%|██████████| 1050/1050 [00:46<00:00, 22.59it/s]
```

```
*--------------------------------------------------------------------------------------*
number of images                          |  1050

dtype                                     |  uint8
channels                                  |  [3]
extensions                                |  ['png']
```

```
min height                              |  575
max height                              |  768
mean height                             |  745.5419047619048
median height                           |  768

min width                               |  576
max width                               |  1024
mean width                              |  1022.6095238095238
median width                            |  1024

mean height/width ratio                 |  0.7290582450127592
median height/width ratio               |  0.75
recommended input size(by mean)         |  [ 744 1024] (h x w, multiples of 8)
recommended input size(by mean)         |  [ 752 1024] (h x w, multiples of 16)
recommended input size(by mean)         |  [ 736 1024] (h x w, multiples of 32)

channel mean(0~1)                       |  [0.45755294 0.4466957  0.39164594]
channel std(0~1)                        |  [0.26054084 0.2610299  0.27889916]
*-------------------------------------------------------------------------------------*
eda ended in 00 hours 00 minutes 46 seconds
```





Out[ ]:

```
{'channels': [3],
 'dtype': 'uint8',
 'extensions': ['png'],
 'max_h': 768,
 'max_w': 1024,
 'mean': array([0.45755294, 0.4466957 , 0.39164594], dtype=float32),
 'mean_h': 745.5419047619048,
 'mean_hw_ratio': 0.7290582450127592,
 'mean_w': 1022.6095238095238,
 'median_h': 768,
 'median_hw_ratio': 0.75,
 'median_w': 1024,
 'min_h': 575,
 'min_w': 576,
 'rec_hw_size_16': array([ 752, 1024]),
 'rec_hw_size_32': array([ 736, 1024]),
 'rec_hw_size_8': array([ 744, 1024]),
 'std': array([0.26054084, 0.2610299 , 0.27889916], dtype=float32)}
```

In [ ]:

```
#Separating tampered images from mask
tamp_imgs = os.listdir(tampered_path)
if not os.path.isdir(tampered_path+'masks/'):
  os.mkdir(tampered_path+'masks/')
for tamp in tamp_imgs:
  if len(tamp.split('.'))==3:
      shutil.move(tampered_path+'/'+tamp, tampered_path+'masks/')
```

In [ ]:

```
BasicImageEDA.explore('/content/dataset-dist/phase-01/training/fake', dimension_plot = True,
channel_hist = True)
```

```
found 450 images.
```

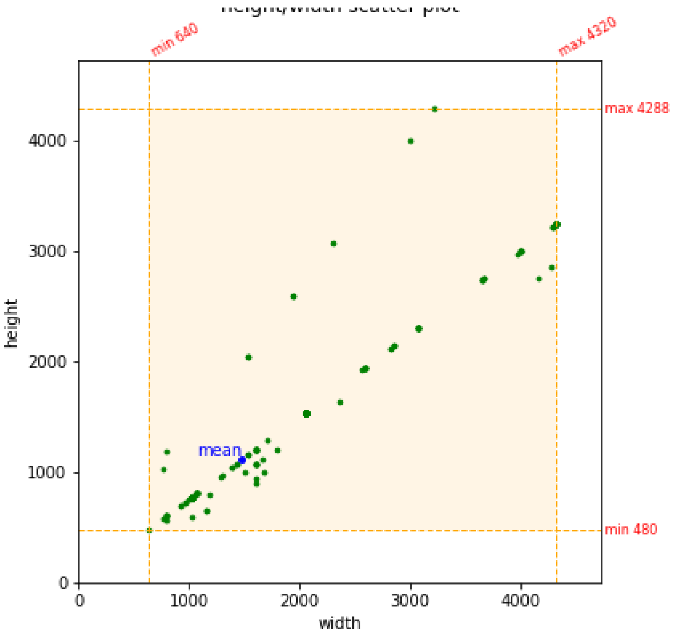
```
  0%|          | 0/450 [00:00<?, ?it/s]
```

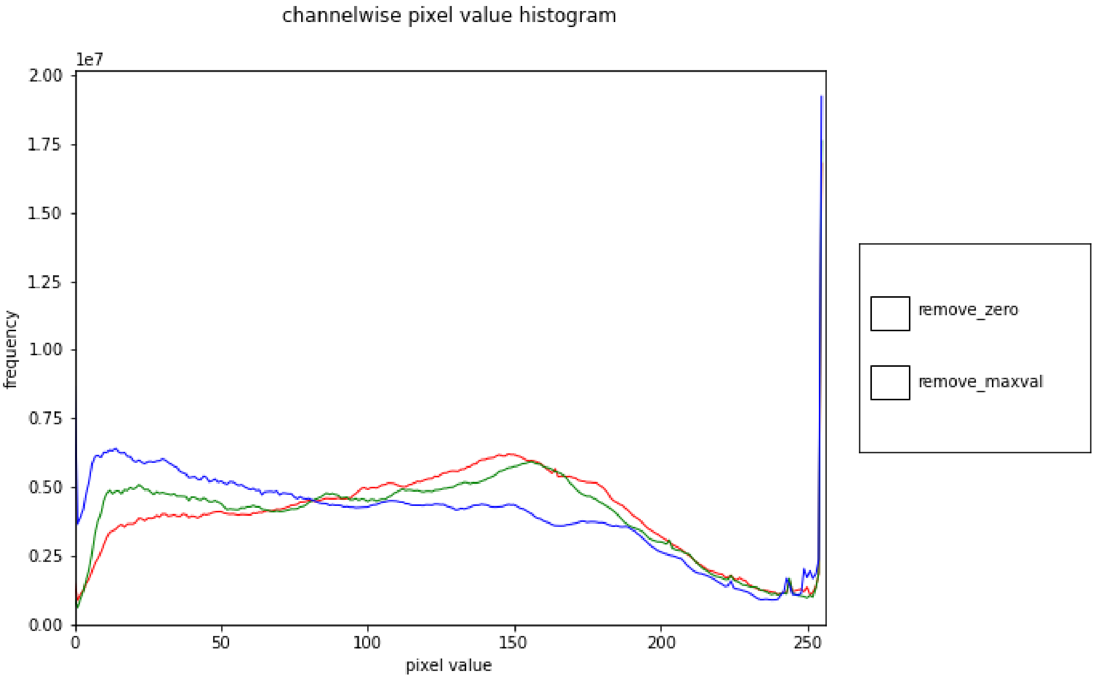
```
Using 4 threads. (max:4)
```

```
100%|██████████| 450/450 [00:57<00:00,  7.78it/s]
```

```
*--------------------------------------------------------------------------------------*
number of images                          |  450

dtype                                     |  uint8
channels                                  |  [3, 4]
extensions                                |  ['png']

min height                                |  480
max height                                |  4288
mean height                               |  1120.1733333333334
median height                             |  768

min width                                 |  640
max width                                 |  4320
mean width                                |  1473.5288888888888
median width                              |  1024

mean height/width ratio                   |  0.7601977414762445
median height/width ratio                 |  0.75
recommended input size(by mean)           |  [1120 1472] (h x w, multiples of 8)
recommended input size(by mean)           |  [1120 1472] (h x w, multiples of 16)
recommended input size(by mean)           |  [1120 1472] (h x w, multiples of 32)

channel mean(0~1)                         |  [0.48769164 0.47090828 0.41975728]
channel std(0~1)                          |  [0.25392324 0.25708836 0.2749937 ]
*--------------------------------------------------------------------------------------*
eda ended in 00 hours 00 minutes 58 seconds
```

height/width scatter plot




channelwise pixel value histogram

Out[ ]:

```
{'channels': [3, 4],
 'dtype': 'uint8',
 'extensions': ['png'],
 'max_h': 4288,
 'max_w': 4320,
 'mean': array([0.48769164, 0.47090828, 0.41975728], dtype=float32),
 'mean_h': 1120.1733333333334,
 'mean_hw_ratio': 0.7601977414762445,
 'mean_w': 1473.5288888888888,
 'median_h': 768,
 'median_hw_ratio': 0.75,
 'median_w': 1024,
 'min_h': 480,
 'min_w': 640,
 'rec_hw_size_16': array([1120, 1472]),
 'rec_hw_size_32': array([1120, 1472]),
 'rec_hw_size_8': array([1120, 1472]),
 'std': array([0.25392324, 0.25708836, 0.2749937 ], dtype=float32)}
```

In [ ]:

```
BasicImageEDA.explore('/content/dataset-dist/phase-01/training/fakemasks', dimension_plot = True,
channel_hist = True)
channel_hist = True)
```

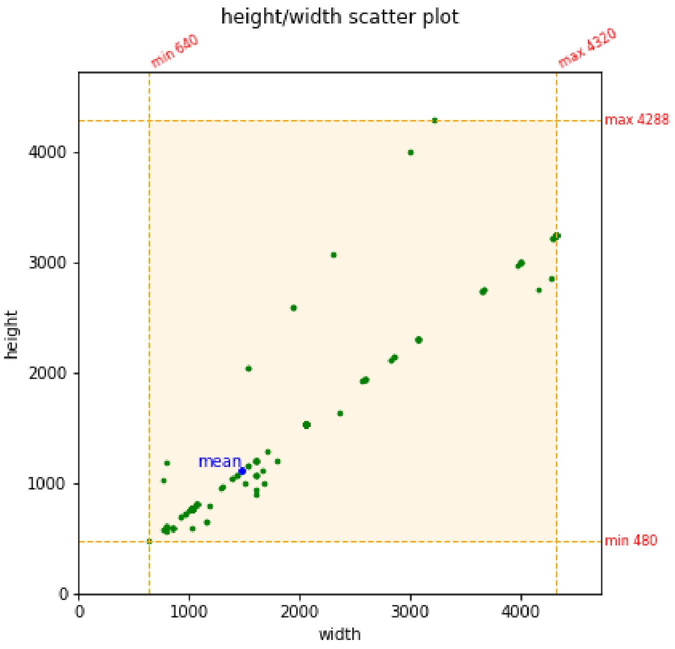
```
found 450 images.
```

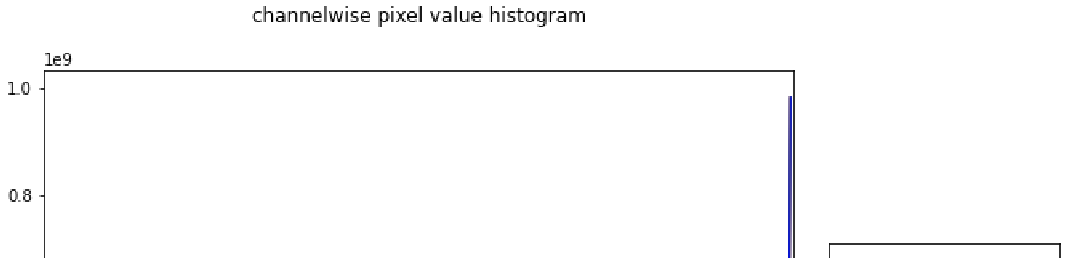
```
  1%|          | 3/450 [00:00<00:18, 24.39it/s]
```

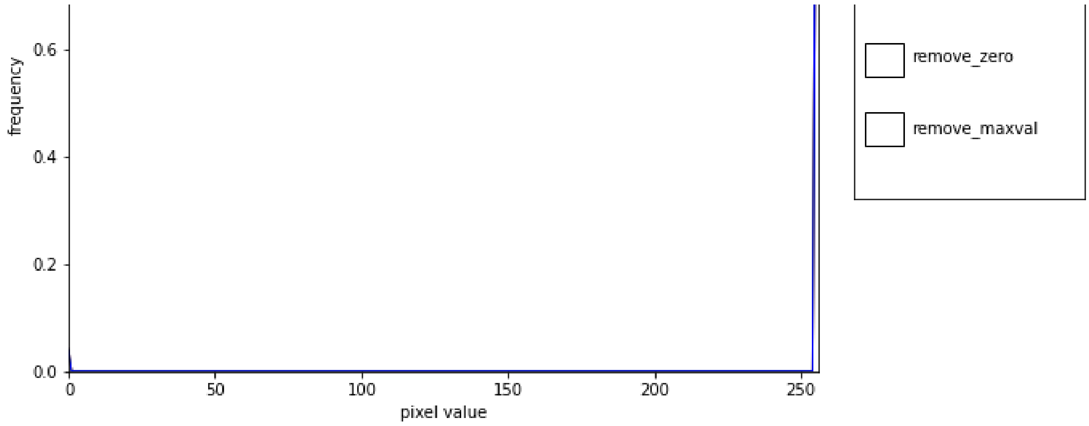
```
Using 4 threads. (max:4)
```

```
100%|██████████| 450/450 [00:43<00:00, 10.35it/s]
```

```
*------------------------------------------------------------------------------------------*
number of images                          |  450

dtype                                     |  uint8
channels                                  |  [1, 3, 4]
extensions                                |  ['png']

min height                                |  480
max height                                |  4288
mean height                               |  1119.2355555555555
median height                             |  768

min width                                 |  640
max width                                 |  4320
mean width                                |  1473.0666666666666
median width                              |  1024

mean height/width ratio                   |  0.7597996620806179
median height/width ratio                 |  0.75
recommended input size(by mean)           |  [1120 1472] (h x w, multiples of 8)
recommended input size(by mean)           |  [1120 1472] (h x w, multiples of 16)
recommended input size(by mean)           |  [1120 1472] (h x w, multiples of 32)

channel mean(0~1)                         |  [0.93324006 0.9332418  0.93324596]
channel std(0~1)                          |  [0.24928743 0.24928407 0.24928   ]
*------------------------------------------------------------------------------------------*
eda ended in 00 hours 00 minutes 43 seconds
```







Out[ ]:

```
{'channels': [1, 3, 4],
 'dtype': 'uint8',
 'extensions': ['png'],
 'max_h': 4288,
 'max_w': 4320,
 'mean': array([0.93324006, 0.9332418 , 0.93324596], dtype=float32),
 'mean_h': 1119.2355555555555,
 'mean_hw_ratio': 0.7597996620806179,
 'mean_w': 1473.0666666666666,
 'median_h': 768,
 'median_hw_ratio': 0.75,
 'median_w': 1024,
 'min_h': 480,
 'min_w': 640,
 'rec_hw_size_16': array([1120, 1472]),
 'rec_hw_size_32': array([1120, 1472]),
 'rec_hw_size_8': array([1120, 1472]),
 'std': array([0.24928743, 0.24928407, 0.24928   ], dtype=float32)}
```

In [ ]:

```
#Getting
pris_img = os.listdir('/content/dataset-dist/phase-01/training/pristine')
pris_shapes=[]
for pris in pris_img:
  try:
    pris_shapes.append(imageio.imread(pristine_path+'/'+pris).shape)
  except:
    continue
```

In [ ]:

```
not_3_channel_pris = []
for idx,temp in enumerate(pris_shapes):
  try:
    if(pris_shapes[idx][2]!=3):
      print(str(idx) +'\t'+ str(pris_shapes[idx]) +'\t'+pris_img[idx])
      not_3_channel_pris.append(pris_img[idx])
  except:
    print(str(idx) +'\t'+ str(pris_shapes[idx]) +'\t'+pris_img[idx])
    not_3_channel_pris.append(pris_img[idx])
    continue
```

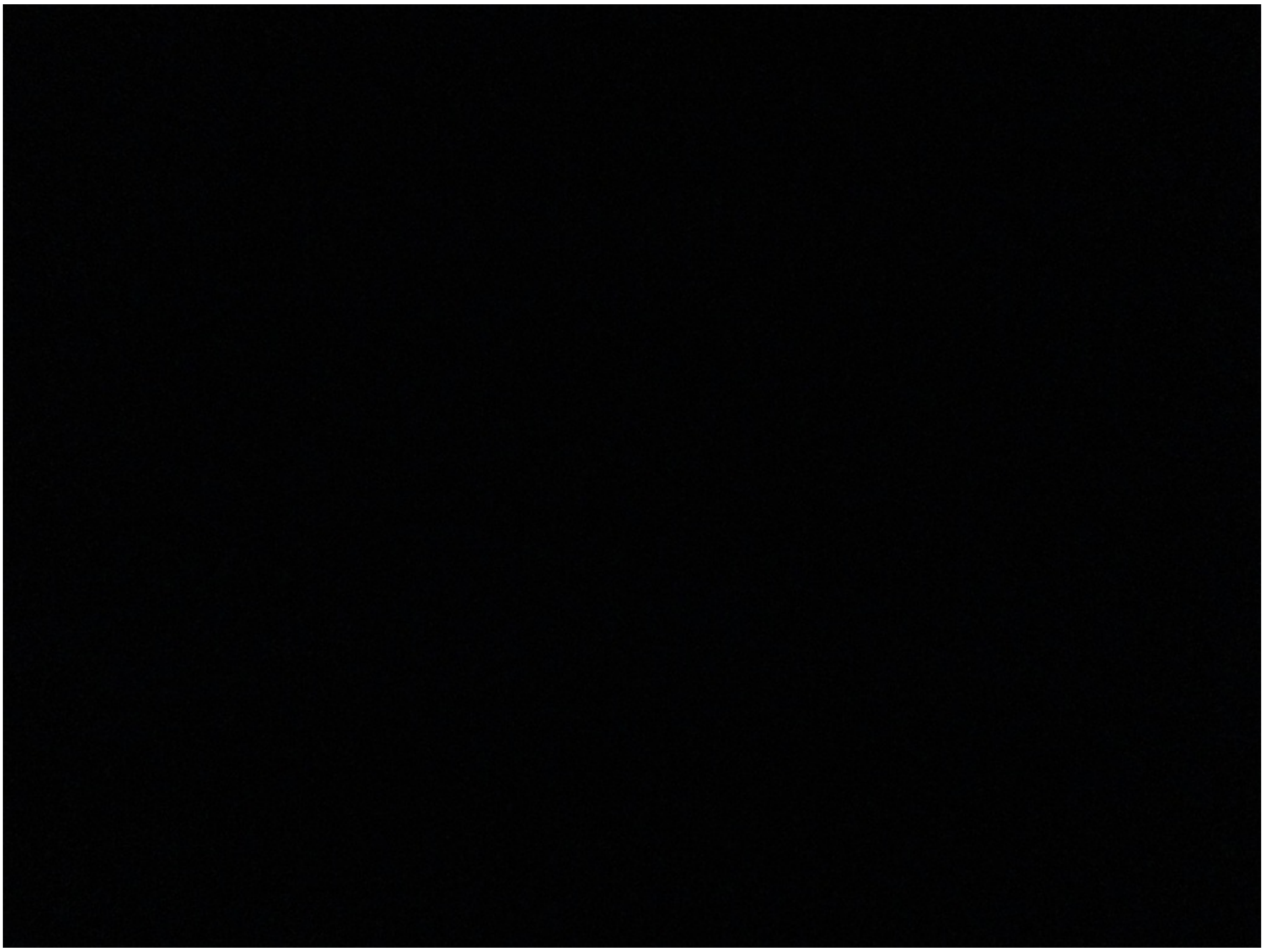
```
98 (768, 1024, 4) 3de6cffce5e0922289bc60c4928acc83.png
109 (768, 1024) 4a1bc3dfa2890535b3c20c233486dbb0.png
221 (768, 1024, 4) 5fc0e8c3c7c1f08a17ed513e274cb493.png
231 (768, 1024, 4) 4b0328c1c0c0a20cabd216e72f0e1002.png
288 (768, 1024, 4) 6d50883e5382834a11493df107cff293.png
315 (768, 576, 4) 6f3b895a3e6b79a827b3ab2a3ec16f97.png
317 (768, 576, 4) 1f3cb8e3a0e12cd2108c52ebb00efcc4.png
335 (768, 1024, 4) 0fcab1fdfc928d55e7eb2914d2ec1afe.png
425 (681, 1024, 4) 4f8d241407dc6607b903a871d1935a3c.png
470 (768, 1024, 4) 2d4c57a4942c0eada19b5231e48aa5c7.png
485 (768, 1024, 4) 1a1d66303c66705aed0c480852a4b7ca.png
628 (768, 576, 4) 6c66a2aa9c0f8f6c9895be73090f1d72.png
633 (768, 1024, 4) 2f4fc27e6917dade2020ceb35a0a1c3d.png
```

```
680 (768, 1024, 4) 03ecdd7f56567b5e64a9f9c768af4295.png
688 (681, 1024, 4) 2cade8c2315a9f03c0193521c1899000.png
703 (768, 1024, 4) 1f563ab49891d72b7d2cd06dd069f689.png
715 (768, 1024, 4) 6f5c74d3e536815751c5706f564b9800.png
735 (768, 1024) 5e5de13ef0adcf7ffe963b039b2467da.png
854 (768, 1024, 4) 2f9a8e339a7334014fb90a84e66489e0.png
877 (768, 1024, 4) 5a25dce69a98d43ba6e9006a27876124.png
896 (768, 1024) 6bfd67cde29d7bf424c265dad69c990a.png
961 (768, 1024, 4) 3e61c9e80fb244378943e49012aaeea0.png
976 (768, 1024, 4) 0e53cd9ad8eeb7e42730e80624a3a0dc.png
977 (768, 1024, 4) 6e304babb685244e0a3f3c60e75e1cd6.png
983 (768, 1024, 4) 0e991f4a4c3d73a2c7c8d5f8d48bcca9.png
```

In [ ]:

```
#4 channel pristine image
Image.open('/content/dataset-dist/phase-01/training/pristine/1a1d66303c66705aed0c480852a4b7ca.png'
)
```

Out[ ]:



In [ ]:

```
tamp_imgs2 = os.listdir('/content/dataset-dist/phase-01/training/fake')
tamp_shapes=[]
for tamp in tamp_imgs2:
  try:
    tamp_shapes.append(imageio.imread(tampered_path+'/'+tamp).shape)
  except:
    continue
```

In [ ]:

```
not_3_channel_fake = []
for idx,temp in enumerate(tamp_shapes):
  if(tamp_shapes[idx][2]!=3):
    print(str(idx) +'\t'+ str(tamp_shapes[idx]) +'\t'+tamp_imgs2[idx])
    not_3_channel_fake.append(tamp_imgs2[idx])
```

```
5 (1536, 2048, 4) a572d8a52f46accacc8eef19acb8759c.png
16 (768, 1024, 4) bedffe5f780e3c2be2cc032450ce0800.png
17 (933, 1600, 4) 901b75381945c55845a0371a576e94d3.png
25 (768, 1024, 4) da51285c1f90d4b22b4be4e9d0ac63aa.png
32 (768, 1024, 4) acb0d5f4885a2dc4cd37f8e7671ec852.png
37 (600, 800, 4) 0294345b3b2324b195cb3b30e91d7678.png
42 (768, 1024, 4) caa67ad39a42b14c19c572ccc5fd2243.png
51 (768, 1024, 4) c7aaa7d7e4c6c0693b4cdd222cb10353.png
53 (768, 1024, 4) b9988711e0a2880bb4fa8cf549dc55bc.png
54 (768, 1024, 4) be51269d525b84339af824a319814cda.png
57 (570, 760, 4) 994526d452af15e3802e1a752a2af020.png
61 (1536, 2048, 4) 31682dc53f670f43d4308f7e99a1d3f8.png
78 (1536, 2048, 4) 207763f02c2c5881c898386782da1728.png
80 (1536, 2048, 4) 30096e4b0cdadb88b548f4fa2aee4b95.png
81 (1536, 2048, 4) b42396107373446a2ab013cd61460b04.png
83 (1536, 2048, 4) 65356dc39286084901856e5cae427764.png
84 (994, 1680, 4) 871d73986285b6092cc16cfe7ab4b7ed.png
95 (768, 1024, 4) 729184490532feff1bea99534924c91d.png
98 (1536, 2048, 4) a0942fb0a31b0f782d5d67a92e6f782c.png
99 (1536, 2048, 4) bacb43ad2dffd3f9cc5fae9c28747020.png
102 (2758, 4150, 4) 6870d1aa15aa6be76dac6931583bf998.png
108 (1024, 768, 4) ca76eefff717b507a0eba5353324d9e5.png
123 (1536, 2048, 4) 2170480823e668068b78bce75afe07a1.png
132 (1536, 2048, 4) c239a9a08c352ec88ad81d4099e0ebcb.png
133 (1536, 2048, 4) c9df6c726d44834138a89684b2454dbb.png
135 (768, 1024, 4) 422896874343197d07d448cfff92ddea.png
146 (1536, 2048, 4) b35d925d02a6792e66e475860372aaae.png
157 (1200, 1800, 4) a470313562508266ecf17a4a5410fc6f.png
173 (480, 640, 4) a24271ffaff5625d6d77bd1b8db7f06f.png
190 (768, 1024, 4) ad0018fb4cec9dc007f89d94902d1bf1.png
193 (768, 1024, 4) b957bf062b8006317f24bdba5dd0abfe.png
194 (1536, 2048, 4) c0a07e462960bd1eb37b5b0c1753c0fa.png
197 (595, 1025, 4) bc6699010f8031e38623087db86466b1.png
223 (1536, 2048, 4) ae62c81c35af25f591fee642ade58245.png
241 (768, 1024, 4) a2e715a637a43b33199c0991785adab7.png
256 (768, 1024, 4) c22d2d0168ef819997238082b12d4149.png
261 (768, 1024, 4) a67721b7b84cb6e9649c67168d02274b.png
279 (563, 798, 4) b12fb5198adce69091a77caac294016c.png
284 (1536, 2048, 4) c31172b1b6021532294f78da3b65fdc3.png
287 (600, 800, 4) aa61a96b0a18b8dbc65fd20af3644958.png
288 (1536, 2048, 4) 44061b655cd1191b0f0e8a19a0e7e69d.png
293 (768, 1024, 4) cdce0701fa96db632af5facf074fe654.png
294 (1637, 2352, 4) c6d6089b4ea367333ca025d8cd1c8d33.png
299 (768, 1024, 4) c78759a7396dd709da30a25683b6219c.png
318 (600, 800, 4) c6ae44ca12707711f81c0380408db48d.png
320 (1536, 2048, 4) d0122ee951d52ee1198b9c402c0a75bf.png
324 (600, 800, 4) 2472627d9b38bce396254ac17b9b3655.png
330 (1536, 2048, 4) be07c00a9019c724132920b410951478.png
332 (1536, 2048, 4) 87327cd1756383fefd53a135523d1e5c.png
341 (768, 1024, 4) 660234fb74717f1874bf604b7d3e3818.png
347 (1536, 2048, 4) d4c52c98840e8128d50e0cd73068ccd3.png
350 (1536, 2048, 4) b9b5386221a626f791fdd6cf400b73df.png
387 (1536, 2048, 4) cd22077ab23004ae566c5e09cecee05f.png
393 (1536, 2048, 4) c26d812059636151e2d7264fa0a9fdbc.png
394 (768, 1024, 4) a9411d7195ec9ac242d5fcb2f6ebf396.png
421 (1536, 2048, 4) 8330772517186ab2c21c9e80ddd3daf1.png
422 (768, 1024, 4) bb187a066f29895b69dc38298ffd72f5.png
423 (768, 1024, 4) c9699a289977ce2d80c6e073eadf8b2a.png
430 (1536, 2048, 4) b09992fe2065d07847925b93505ed296.png
432 (768, 1024, 4) e1795634a3bf20d9c2e313a92c048bad.png
439 (768, 1024, 4) ca8f5bc4dfc11f10d823230e1c800caf.png
441 (1536, 2048, 4) f0d05a2fce59b068846bd1c8453d7d89.png
444 (1536, 2048, 4) ab1e91e9a308f42d3cd0baa59f7c7c71.png
447 (1536, 2048, 4) b185aab42bdb43b9f6baec60f9a5ba27.png
```

In [ ]:

```
#4 channel tampered image
Image.open('/content/dataset-dist/phase-01/training/fake/c86cb7e7cf51b7b182a6ffa8b253ed2b.png')
```

Out[ ]:

In [ ]:

```
tamp_masks2 = os.listdir('/content/dataset-dist/phase-01/training/fakemasks')
tamp_mask_shapes=[]
for tamp in tamp_masks2:
  try:
    tamp_mask_shapes.append(imageio.imread('/content/dataset-dist/phase-01/training/fakemasks/' + t
amp).shape)
  except:
    continue
```

```
/usr/local/lib/python3.6/dist-packages/PIL/Image.py:932: UserWarning: Palette images with
Transparency expressed in bytes should be converted to RGBA images
  "Palette images with Transparency expressed in bytes should be "
```

In [ ]:

```
not_3_channel_mask = []
for idx,temp in enumerate(tamp_mask_shapes):
  try:
    if(tamp_mask_shapes[idx][2]!=1):
      print(str(idx) +'\t'+ str(tamp_mask_shapes[idx]) +'\t'+tamp_masks2[idx])
      not_3_channel_mask.append(tamp_masks2[idx])
  except:
    # if(tamp_mask_shapes[idx][2]):
    #   print(str(idx) +'\t'+ str(tamp_mask_shapes[idx]) +'\t'+tamp_masks2[idx])
    continue
```

```
0 (2592, 1936, 3) 55774d613ff0e35e640172a35fdd6c96.mask.png
7 (1536, 2048, 4) b91fb5eb5b2ef55ad665fb6cd7f7b657.mask.png
8 (1536, 2048, 4) bacb43ad2dffd3f9cc5fae9c28747020.mask.png
15 (4000, 3000, 3) 588da262d375acd1ee48212fb2760ae2.mask.png
30 (3240, 4320, 3) b3410ded36f35bda24e18c5362f69d63.mask.png
37 (3240, 4320, 3) fce1efb32a85a7f67f959a7c37f1f52b.mask.png
38 (2304, 3072, 3) b8cecf240477353bdf87aac6521e066f.mask.png
41 (1200, 1600, 3) f95d94a3a6384059a64725f89677e885.mask.png
44 (1536, 2048, 4) 31682dc53f670f43d4308f7e99a1d3f8.mask.png
```

```
44 (1536, 2048, 4) 51082dc551070145d450817e99a1d518.mask.png
51 (1536, 2048, 4) b0175f62b84472eab8d177aa3d0364f9.mask.png
54 (1536, 2048, 3) 84fd5a243a63e25013ef3d6fe8eeaf12.mask.png
57 (3240, 4320, 3) 7125cec169f3635cd07db90b16e848d5.mask.png
60 (3216, 4288, 3) 2744a9427d865323a62eff7fdfbb9ed5.mask.png
63 (3240, 4320, 3) d8fd021d1ca1e21880a0b84effa7157b.mask.png
65 (1536, 2048, 4) 2564876058a684e6d899f3575081b16f.mask.png
68 (2304, 3072, 3) 364f0d4ece36860de6c2ad8cf943b800.mask.png
69 (598, 848, 3) b0060704d02f1229b75cbd550c7267b4.mask.png
86 (600, 800, 4) b47c970b9a25c103951ab48c55727ecc.mask.png
88 (2748, 3664, 3) dcc2c41c810408c4cb08ce092a712d02.mask.png
89 (600, 800, 4) 0294345b3b2324b195cb3b30e91d7678.mask.png
90 (1536, 2048, 4) 010543abfbd0db1e9aa1b24604336e0c.mask.png
92 (2304, 3072, 3) 95545cc4da4cae23ec42672bbbd3bfa7.mask.png
96 (3000, 4000, 3) e6764ef0fb7a2a0b624b554c89c62137.mask.png
102 (600, 800, 4) b5413c246d39c53af7e53b1d56f64946.mask.png
106 (3240, 4320, 3) d87fb93838434f8a1d668c0ed1bd8824.mask.png
107 (1536, 2048, 4) c9df6c726d44834138a89684b2454dbb.mask.png
110 (1536, 2048, 4) 17030938cc2cb314cb87552dcc9516a4.mask.png
111 (1536, 2048, 4) bd102f5038b9c07483b7e76eb614de0a.mask.png
115 (1536, 2048, 4) ac821291d07a9c317f2dbf5e424cb8cb.mask.png
116 (1200, 1600, 3) d7632c418ab10443dbeddc264ec032fb.mask.png
122 (2748, 3664, 3) a9667850a5652972443b765ae3ccf3ff.mask.png
125 (994, 1680, 4) 871d73986285b6092cc16cfe7ab4b7ed.mask.png
128 (4288, 3216, 3) 229f447c1a26a74005b3f058201bde3f.mask.png
130 (1536, 2048, 4) a572d8a52f46accacc8eef19acb8759c.mask.png
132 (768, 1024, 4) 1848de26a06a7831457609429c92e2e7.mask.png
133 (2848, 4272, 3) c66c01633c25f6b9861578432638508d.mask.png
134 (2748, 3664, 3) d96f2bcd580fa5de490377169d6a45bd.mask.png
136 (2304, 3072, 4) bb7ed6b43f565a1fe2ebcbf99886d1d4.mask.png
144 (3240, 4320, 3) e2a45c5c31400c8a16df285a45be6900.mask.png
146 (1536, 2048, 4) c6fd81dc1179711b70fa379eeb714028.mask.png
147 (563, 798, 4) 91c79965316355431c3c8fed22a115ae.mask.png
152 (1536, 2048, 4) 30096e4b0cdadb88b548f4fa2aee4b95.mask.png
156 (1637, 2352, 4) 244a7433a307b9a2c839cefe14c0ba1d.mask.png
157 (1536, 2048, 4) 729184490532feff1bea99534924c91d.mask.png
158 (1536, 2048, 4) 207763f02c2c5881c898386782da1728.mask.png
165 (598, 848, 3) d4aff0ad5f4f99fc6cad4243b926eda7.mask.png
170 (2304, 3072, 3) 93644a457afc64b27a692c1ecd9df773.mask.png
171 (1536, 2048, 4) 152681a0017a5fded699c43cd6df97d1.mask.png
179 (1536, 2048, 4) 65356dc39286084901856e5cae427764.mask.png
180 (1536, 2048, 4) c7aaa5080c49117748fce73bff068573.mask.png
181 (598, 848, 3) d9b9f5db7d29a3855cceef574145b595.mask.png
182 (1536, 2048, 4) aa61a96b0a18b8dbc65fd20af3644958.mask.png
185 (3240, 4320, 3) cab8ac89fc001f1adb2ff4d8b3f9f9a9.mask.png
190 (2144, 2848, 3) fedd664fb16748292deb66f75e1da4bc.mask.png
191 (1200, 1600, 3) 2508f9cfb3c5f96d4539dcb1fa049d6a.mask.png
192 (480, 640, 4) 31311633f92518299051f6c846919af1.mask.png
197 (2048, 1536, 4) ab59e5fdefc229a4a07592c4376e2ffc.mask.png
201 (2592, 1936, 3) 237db9303fe590d8104510e36dfaa4a7.mask.png
202 (2112, 2816, 3) 2472627d9b38bce396254ac17b9b3655.mask.png
206 (1536, 2048, 4) 839128f5837a4d3614e9f1f6b4cf087e.mask.png
207 (3000, 4000, 3) 371e6f86e51ab2258b69547dd7657b30.mask.png
215 (1936, 2592, 3) bea810fbe5f0ee59c79b4ebd4732f1a0.mask.png
222 (3240, 4320, 3) b9149cc8a64a111bbc1b30cb1bdc37d4.mask.png
227 (1920, 2560, 3) ce571df8dd7ac27523b2cca4cbc32194.mask.png
229 (1944, 2592, 3) fdf110fa789efa05114811412f54b2a9.mask.png
237 (2304, 3072, 3) 6870d1aa15aa6be76dac6931583bf998.mask.png
249 (1536, 2048, 4) 687764119688d5ee49717027e6145bee.mask.png
251 (1536, 2048, 4) b42396107373446a2ab013cd61460b04.mask.png
254 (3240, 4320, 3) 756a2f39b0b82013ee00b825d66ad0bc.mask.png
259 (3240, 4320, 3) df2cf775afbb1d2880aa22f5f3c43995.mask.png
265 (3240, 4320, 3) 87327cd1756383fefd53a135523d1e5c.mask.png
276 (1536, 2048, 4) ac478f5c3c6c77c12764362388773da7.mask.png
277 (1944, 2592, 3) eded2f92d413246c29001c448dbddd1b.mask.png
280 (3000, 4000, 3) da87f75ad935467d3c8d0ab08a559e76.mask.png
285 (1200, 1800, 4) 1990a2ed067b8c537d8fe36d1ab4a7f2.mask.png
287 (2758, 4150, 4) 72366b10b23899d659b3b0fa92d3a73c.mask.png
289 (1536, 2048, 4) 8403960a267cea6cec8473736454c4bd.mask.png
292 (1536, 2048, 4) bb187a066f29895b69dc38298ffd72f5.mask.png
299 (3240, 4320, 3) d6388ee9f63e1111d41ce66ddf06ff41.mask.png
300 (1200, 1600, 3) 35458ca671876a5bad7f87419fe53b4c.mask.png
301 (3240, 4320, 3) edb9414156ff96adf906cbba292e6cd4.mask.png
302 (3216, 4288, 3) ce6a3e19dfcd8e8b162faf8511b920ae.mask.png
304 (3000, 4000, 3) 881be478b340b647b959481b9148534e.mask.png
308 (1536, 2048, 4) afe1529ef8aaec50a917345e7280f9c2.mask.png
309 (3216, 4288, 3) 707642c1c1a36f8fb28274f2484b11fe.mask.png
314 (3240, 4320, 3) d1e6b20063dfb294189b8338541ba1cf.mask.png
```

```
314 (3240, 4320, 3) d1e6b20065d1b294189b8358541ba1c1.mask.png
327 (1536, 2048, 4) a768fceead79102801131160d76dc08b.mask.png
329 (598, 848, 3) aa3b4f7caf9de8c1d6551c33045fb4c1.mask.png
344 (598, 848, 3) bc04da26ab41ce92565dd3c686dae6c8.mask.png
349 (1536, 2048, 4) c7cbba822ab211cd0292b3688aa4b903.mask.png
356 (1536, 2048, 4) 906607152d984039c6baebdf6fa15c40.mask.png
364 (1536, 2048, 4) bc42f3fa484ad950692de70f31a1314c.mask.png
365 (1536, 2048, 4) 87238ad60578291fff62151eb6618adb.mask.png
376 (598, 848, 3) cc263a4c9ff9943acbb9049f637a0bed.mask.png
377 (1944, 2592, 3) cb5dece724ab947f6a615c7fe1f8c380.mask.png
379 (1536, 2048, 4) a607a69c1d589cf0e9d24ff6162abf01.mask.png
388 (2736, 3648, 3) 0908dafde12041540b70d688315df6e9.mask.png
390 (3240, 4320, 3) 6743af3f663bc4244b0f80b93541f542.mask.png
391 (2112, 2816, 3) eee3361e4c383c67789ef99c3a65fa8d.mask.png
394 (1536, 2048, 4) a874d91dfcbe35e7204dce0845bcc71a.mask.png
396 (2976, 3968, 3) 2680cb774242658dd33dd7dccccf6308.mask.png
399 (3240, 4320, 3) a749a9c6c906f0f57b9c91d13439bd6b.mask.png
401 (933, 1600, 4) 901b75381945c55845a0371a576e94d3.mask.png
402 (2048, 1536, 4) 49885ceb0d7868353754bcdc653fd85a.mask.png
403 (1536, 2048, 4) af8332f8c5b8cedeeb33963bb532e2f8.mask.png
407 (598, 848, 3) ca472f184807aded538221ac0b5ac27b.mask.png
412 (2144, 2848, 3) e3f10dc9b7dfcf91efd0bce8bf2c82b0.mask.png
414 (1536, 2048, 4) 8569531f0cfe6fed6f0911100c8c8d56.mask.png
417 (3072, 2304, 3) ce6fc3053fb6221f93f5c376a005e658.mask.png
418 (1536, 2048, 4) ad71b41fd8257bfb9bf303008a92f68e.mask.png
419 (1536, 2048, 4) 6524631d3b2df3c246e553dd55e3361b.mask.png
420 (2736, 3648, 3) 674cf83cd200c6936f54a0ed6894bf9c.mask.png
425 (3000, 4000, 3) dedfc7b72b5c7ce42bd3e93df033a531.mask.png
427 (1944, 2592, 3) 452f20323286faadad71a1c9ffae59f4.mask.png
428 (3240, 4320, 3) c26d812059636151e2d7264fa0a9fdbc.mask.png
431 (570, 760, 4) 994526d452af15e3802e1a752a2af020.mask.png
434 (2736, 3648, 3) 092b43f88eab0ae3ecc0eb0ccbe37c82.mask.png
439 (1536, 2048, 4) a9411d7195ec9ac242d5fcb2f6ebf396.mask.png
446 (1536, 2048, 4) c2e9e25b3f224a2bc80ca1f6dd86f465.mask.png
447 (600, 800, 4) 44061b655cd1191b0f0e8a19a0e7e69d.mask.png
```

- From above we can see the dataset consists of 1,3 and 4 channel images as well as masks.

In [ ]:

```
print('Number of images with not 3 channels in prisitine = ',len(not_3_channel_pris))
print('Number of images with not 3 channels in fake = ',len(not_3_channel_fake))
print('Number of images with not 3 channels in masks = ',len(not_3_channel_mask))
```

```
Number of images with not 3 channels in prisitine =  25
Number of images with not 3 channels in fake =  64
Number of images with not 3 channels in masks =  120
```

In [ ]:

```
#4 channel mask
Image.open('/content/dataset-dist/phase-01/training/fake/839128f5837a4d3614e9f1f6b4cf087e.mask.png')
```

Out[ ]:

In [ ]:

```
pristine_data = {'name':[],'height':[], 'width':[], 'channels':[],'label':[]}
```

In [ ]:

```
for image in tqdm.tqdm(os.listdir('/content/dataset-dist/phase-01/training/pristine')):
  img = imread('/content/dataset-dist/phase-01/training/pristine'+'/'+image)
  if len(img.shape)==2:
      height, width = img.shape
      channels = 1
  else:
      height, width, channels = img.shape
  pristine_data['name'].append(image)
  pristine_data['height'].append(height)
  pristine_data['width'].append(width)
  pristine_data['channels'].append(channels)
  pristine_data['label'].append('pristine')
  pristine_df = pd.DataFrame.from_dict(pristine_data)
```

```
100%|██████████| 1050/1050 [00:34<00:00, 30.49it/s]
```

In [ ]:

```
pristine_df.head()
```

Out[ ]:

	name	height	width	channels	label
0	1c39af9d0be4ee3c5a069dfa866b8c5a.png	768	1024	3	pristine
1	2d6895ce19c339ec873636c5134bc754.png	768	1024	3	pristine
2	6e05833bce893af997d49b6fbc8b6215.png	768	1024	3	pristine
3	2cea5e8c6e865004b13bd3a7fa0cbad9.png	768	1024	3	pristine
4	6c494c868401b08e53c3d29e2b87b37b.png	768	1024	3	pristine

In [ ]:

```
mask_data = {'name':[],'height':[], 'width':[], 'channels':[],'label':[]}
```

In [ ]:

```
for image in tqdm.tqdm(os.listdir('/content/dataset-dist/phase-01/training/fakemasks')):
  img = imread('/content/dataset-dist/phase-01/training/fakemasks'+'/'+image)
  if len(img.shape)==2:
      height, width = img.shape
      channels = 1
  else:
      height, width, channels = img.shape
  mask_data['name'].append(image)
  mask_data['height'].append(height)
  mask_data['width'].append(width)
  mask_data['channels'].append(channels)
  mask_data['label'].append('fake')
  mask_df = pd.DataFrame.from_dict(mask_data)
```

```
 17%|██        | 76/450 [00:03<00:13, 28.36it/s]/usr/local/lib/python3.6/dist-
packages/PIL/Image.py:932: UserWarning: Palette images with Transparency expressed in bytes should
be converted to RGBA images
  "Palette images with Transparency expressed in bytes should be "
100%|██████████| 450/450 [00:14<00:00, 31.42it/s]
```

In [ ]:

```
mask_df.head()
```

Out[ ]:

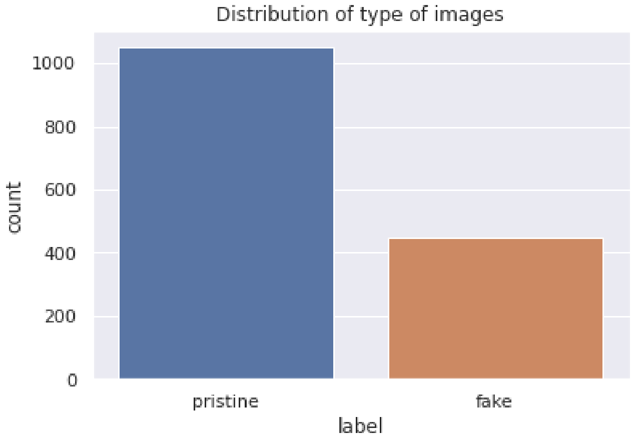
	name	height	width	channels	label
0	c6fd81dc1179711b70fa379eeb714028.mask.png	1536	2048	4	fake
1	901b75381945c55845a0371a576e94d3.mask.png	933	1600	4	fake
2	b0b2682b1b9c568c5050c4ad69243622.mask.png	771	1024	1	fake
3	8569531f0cfe6fed6f0911100c8c8d56.mask.png	1536	2048	4	fake
4	0363353570f16ff0a73aa0a03a7795b8.mask.png	765	1024	1	fake

In [ ]:

```
df_total = pristine_df.append(mask_df, ignore_index = True)
```

In [ ]:

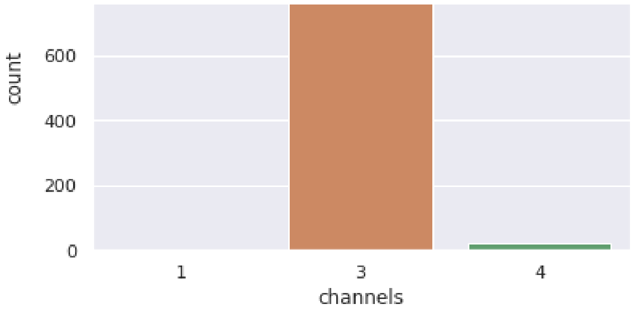
```
sns.set_theme(style="darkgrid")
plt.title('Distribution of type of images')
ax = sns.countplot(x="label", data=df_total)
```



In [ ]:

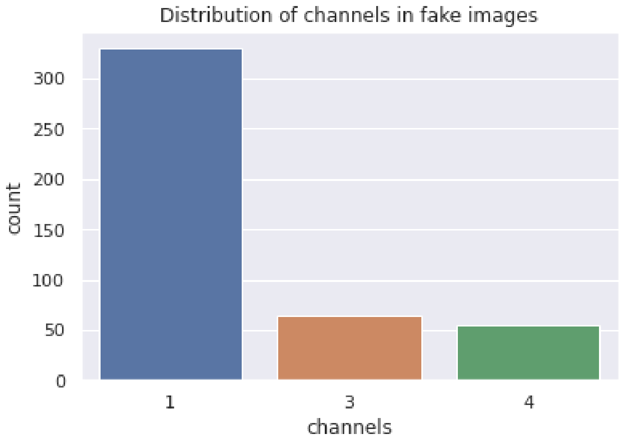
```
sns.set_theme(style="darkgrid")
plt.title('Distribution of channels in pristine images')
ax = sns.countplot(x="channels", data=pristine_df)
```

Distribution of channels in pristine images
1000
800



In [ ]:

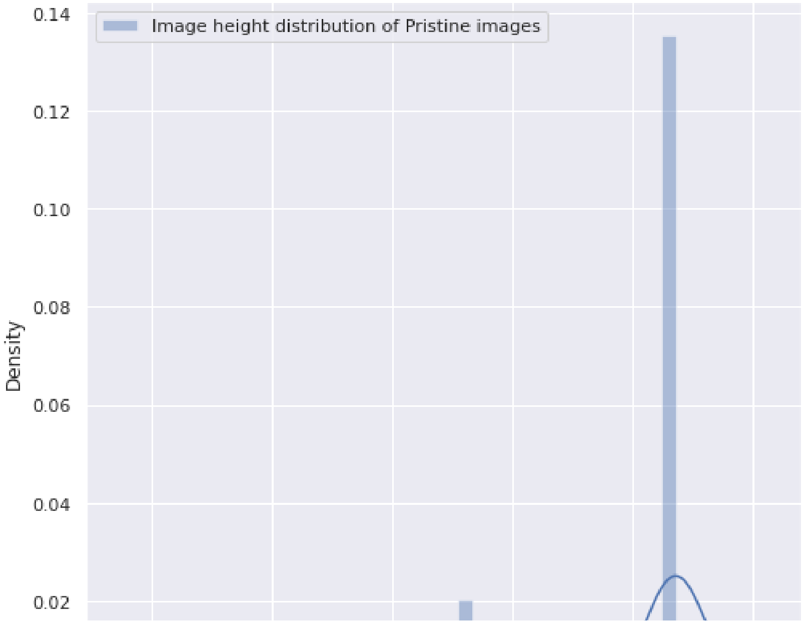
```
sns.set_theme(style="darkgrid")
plt.title('Distribution of channels in fake images')
ax = sns.countplot(x="channels", data=mask_df)
```

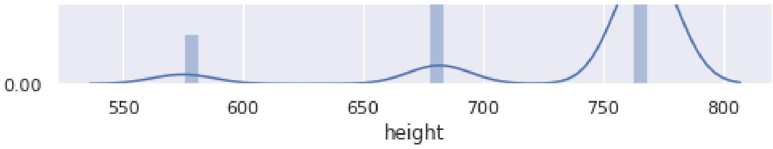


In [ ]:

```
plt.figure(figsize=(8,8))
sns.set(style="darkgrid")
ax = sns.distplot(pristine_df.height, label='Image height distribution of Pristine images')
plt.legend()
plt.show()
```

```
/usr/local/lib/python3.6/dist-packages/seaborn/distributions.py:2551: FutureWarning: `distplot` is
a deprecated function and will be removed in a future version. Please adapt your code to use eithe
r `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level
function for histograms).
  warnings.warn(msg, FutureWarning)
```

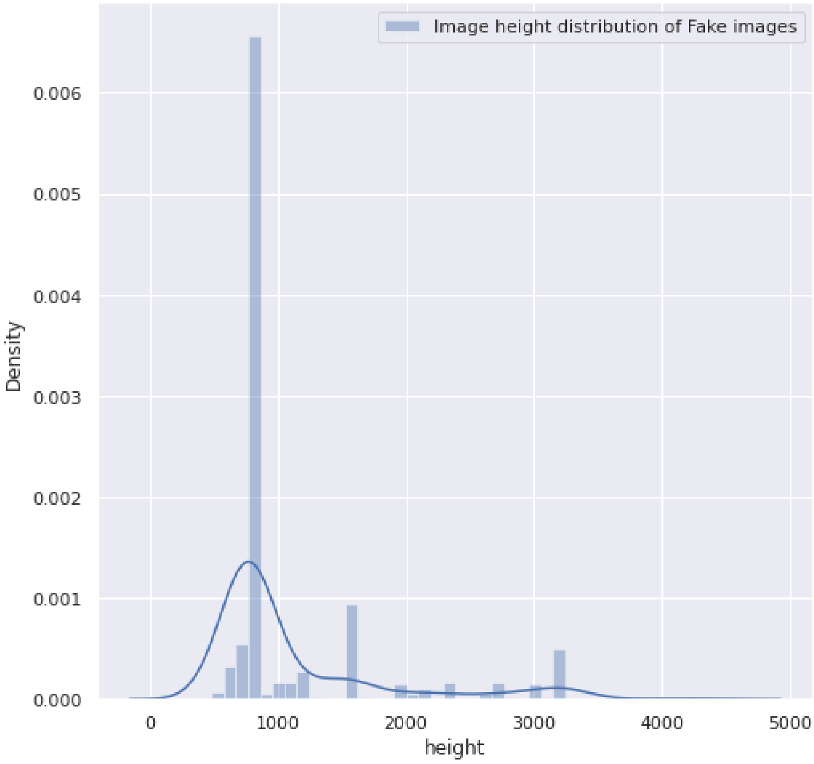




In [ ]:

```
plt.figure(figsize=(8,8))
sns.set(style="darkgrid")
ax = sns.distplot(mask_df.height, label='Image height distribution of Fake images')
plt.legend()
plt.show()
```

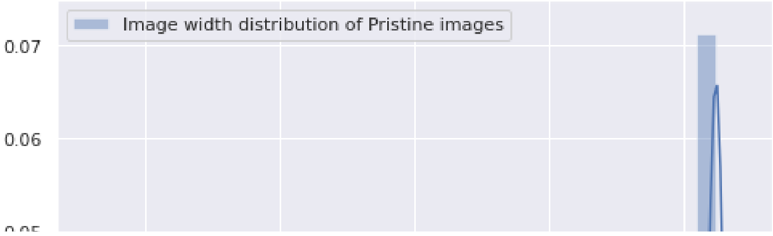
```
/usr/local/lib/python3.6/dist-packages/seaborn/distributions.py:2551: FutureWarning: `distplot` is
a deprecated function and will be removed in a future version. Please adapt your code to use eithe
r `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level
function for histograms).
  warnings.warn(msg, FutureWarning)
```

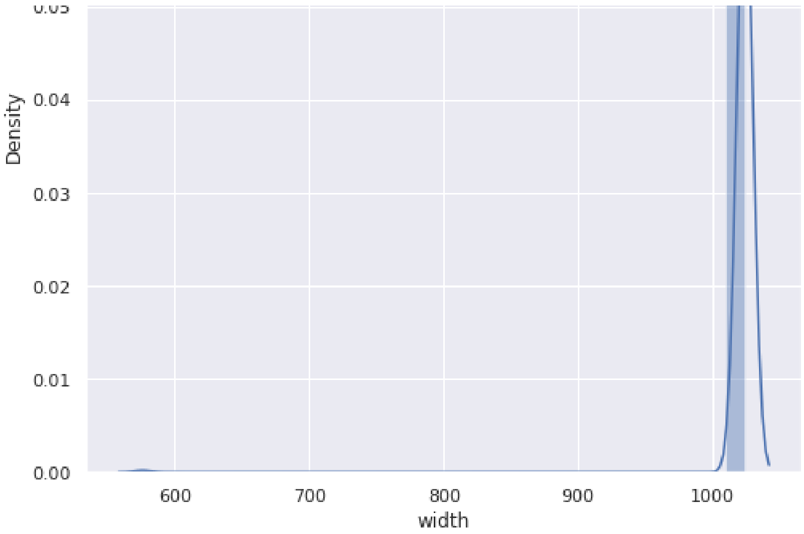


In [ ]:

```
plt.figure(figsize=(8,8))
sns.set(style="darkgrid")
ax = sns.distplot(pristine_df.width, label='Image width distribution of Pristine images')
plt.legend()
plt.show()
```

```
/usr/local/lib/python3.6/dist-packages/seaborn/distributions.py:2551: FutureWarning: `distplot` is
a deprecated function and will be removed in a future version. Please adapt your code to use eithe
r `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level
function for histograms).
  warnings.warn(msg, FutureWarning)
```

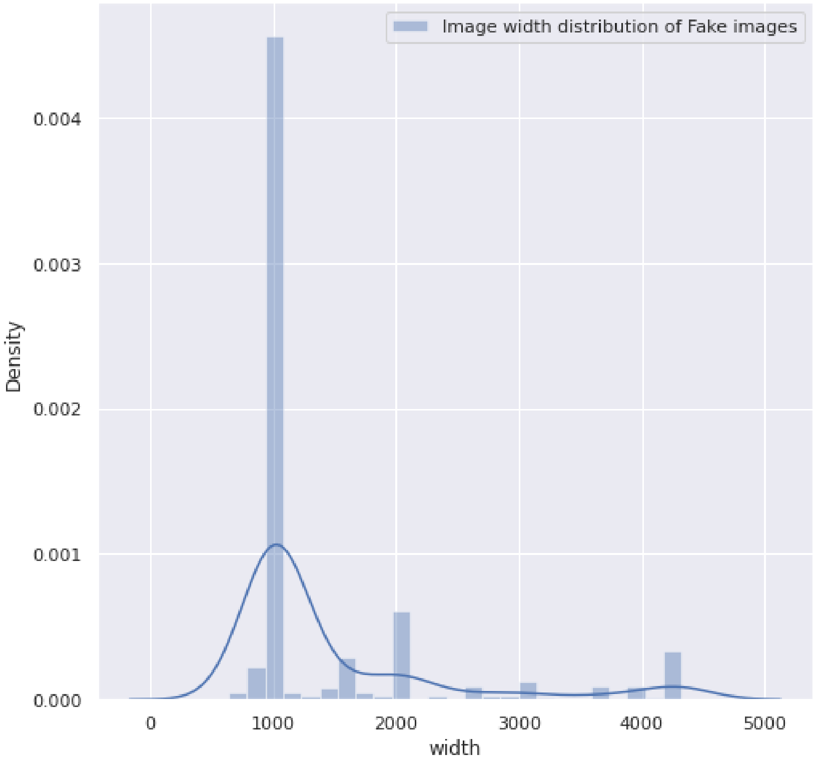




In [ ]:

```
plt.figure(figsize=(8,8))
sns.set(style="darkgrid")
ax = sns.distplot(mask_df.width, label='Image width distribution of Fake images')
plt.legend()
plt.show()
```

```
/usr/local/lib/python3.6/dist-packages/seaborn/distributions.py:2551: FutureWarning: `distplot` is
a deprecated function and will be removed in a future version. Please adapt your code to use eithe
r `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level
function for histograms).
  warnings.warn(msg, FutureWarning)
```



- Fake images have more images with greater height than pristine images.
- Almost all the pristine images have same width while fake images also have high density towards similar range.

Classification of images using CASIA dataset and ELA

In [7]:

```
#Generating ela from normal images and resizing them
```

```
def get_image(path):
    return np.array(generate_ela(path, 90).resize((128,128))).flatten()/255
```

In [ ]:

```
#Method to plot colored confusion matrix
def plot_confusion_matrix(cm, classes,normalize=False,title='Confusion matrix',cmap=plt.cm.Blues):
    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)


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

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

In [ ]:

```
X = []
Y = []
```

In [ ]:

```
#Adding image paths from pristine images with label 1 as they are ground truth
path = '/content/casia/CASIA2/Au'
for dir, paths, files in os.walk(path):
    for filename in files:
        if filename.endswith('jpg') or filename.endswith('png'):
            full_path = os.path.join(dir, filename)
            X.append(get_image(full_path))
            Y.append(1)

print(len(X), len(Y))
```

```
/usr/local/lib/python3.6/dist-packages/PIL/TiffImagePlugin.py:770: UserWarning: Possibly corrupt E
XIF data.  Expecting to read 8 bytes but only got 2. Skipping tag 41487
  " Skipping tag %s" % (size, len(data), tag)
/usr/local/lib/python3.6/dist-packages/PIL/TiffImagePlugin.py:770: UserWarning: Possibly corrupt E
XIF data.  Expecting to read 8 bytes but only got 0. Skipping tag 41988
  " Skipping tag %s" % (size, len(data), tag)
```

```
7354 7354
```

In [ ]:

```
#Adding image paths from tampered images with label 0
path = '/content/casia/CASIA2/Tp'
for dir, paths, files in os.walk(path):
    for filename in files:
        if filename.endswith('jpg') or filename.endswith('png'):
            full_path = os.path.join(dir, filename)
            X.append(get_image(full_path))
            Y.append(0)

print(len(X), len(Y))
```

```
9418 9418
```

In [ ]:

```
X = np.array(X)
print(X.shape)
print(X.shape)
```

```
(9418, 49152)
```

In [ ]:

```
Y = to_categorical(Y, 2)
X = X.reshape(-1, 128, 128, 3)
```

In [ ]:

```
print(X.shape)
```

```
(9418, 128, 128, 3)
```

In [ ]:

```
#Splitting the data
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2, random_state=20)
```

In [ ]:

```
print(X_train.shape,Y_train.shape)
print(X_test.shape,Y_test.shape)
```

```
(7534, 128, 128, 3) (7534, 2)
(1884, 128, 128, 3) (1884, 2)
```

Classification model 1

In [ ]:

```
model1 = Sequential()
model1.add(Conv2D(filters = 32, kernel_size = (5,5),padding = 'valid',activation =
'relu',input_shape = (128,128,3)))
model1.add(MaxPool2D(pool_size = (2,2)))
model1.add(Dropout(0.1))

model1.add(Conv2D(filters = 32, kernel_size = (5,5),padding = 'valid',activation = 'relu'))
model1.add(MaxPool2D(pool_size = (2,2)))
model1.add(Dropout(0.2))

model1.add(Flatten())
model1.add(Dense(256, activation = 'relu'))
model1.add(Dropout(0.3))
model1.add(Dense(2, activation = 'softmax'))
```

In [ ]:

```
model1.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d (Conv2D)              (None, 124, 124, 32)      2432      
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 62, 62, 32)        0         
_________________________________________________________________
dropout_3 (Dropout)          (None, 62, 62, 32)        0         
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 58, 58, 32)        25632     
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 29, 29, 32)        0         
_________________________________________________________________
dropout_4 (Dropout)          (None, 29, 29, 32)        0         
_________________________________________________________________
flatten (Flatten)            (None, 26912)             0         
```

```
_________________________________________________________________
dense_5 (Dense)              (None, 256)               6889728
_________________________________________________________________
dropout_5 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_6 (Dense)              (None, 2)                 514
=================================================================
Total params: 6,918,306
Trainable params: 6,918,306
Non-trainable params: 0
_________________________________________________________________
```

In [ ]:

```
!rm -rf logs/fit/
```

In [ ]:

```
epochs = 15
batch_size = 32
init_lr = 1e-4
optimizer = Adam(lr = init_lr, decay = init_lr/epochs)
model1.compile(optimizer = optimizer, loss = 'binary_crossentropy', metrics = ['accuracy'])
early_stopping = EarlyStopping(monitor = 'val_accuracy',min_delta = 0,patience = 2,mode = 'auto')
log_dir = "logs/fit/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard_callback = TensorBoard(log_dir=log_dir, histogram_freq=1)
```

In [ ]:

```
model1.fit(X_train,Y_train,batch_size = batch_size,epochs = epochs,validation_data = (X_test, Y_tes
t),callbacks = [early_stopping,tensorboard_callback])
```

```
Epoch 1/15
  2/236 [..............................] - ETA: 30s - loss: 0.6335 - accuracy:
0.6406WARNING:tensorflow:Callbacks method `on_train_batch_end` is slow compared to the batch time
(batch time: 0.0096s vs `on_train_batch_end` time: 0.2519s). Check your callbacks.
236/236 [==============================] - 4s 17ms/step - loss: 0.3730 - accuracy: 0.8053 - val_lo
ss: 0.2981 - val_accuracy: 0.8397
Epoch 2/15
236/236 [==============================] - 3s 15ms/step - loss: 0.2849 - accuracy: 0.8610 - val_lo
ss: 0.3216 - val_accuracy: 0.8546
Epoch 3/15
236/236 [==============================] - 3s 14ms/step - loss: 0.2539 - accuracy: 0.8792 - val_lo
ss: 0.5459 - val_accuracy: 0.6343
Epoch 4/15
236/236 [==============================] - 3s 14ms/step - loss: 0.2401 - accuracy: 0.8836 - val_lo
ss: 0.2417 - val_accuracy: 0.8875
Epoch 5/15
236/236 [==============================] - 3s 14ms/step - loss: 0.2188 - accuracy: 0.8922 - val_lo
ss: 0.1994 - val_accuracy: 0.8954
Epoch 6/15
236/236 [==============================] - 3s 15ms/step - loss: 0.1991 - accuracy: 0.9036 - val_lo
ss: 0.1846 - val_accuracy: 0.9076
Epoch 7/15
236/236 [==============================] - 3s 15ms/step - loss: 0.1760 - accuracy: 0.9251 - val_lo
ss: 0.1830 - val_accuracy: 0.9294
Epoch 8/15
236/236 [==============================] - 3s 15ms/step - loss: 0.1639 - accuracy: 0.9294 - val_lo
ss: 0.1509 - val_accuracy: 0.9416
Epoch 9/15
236/236 [==============================] - 3s 15ms/step - loss: 0.1531 - accuracy: 0.9326 - val_lo
ss: 0.1586 - val_accuracy: 0.9315
Epoch 10/15
236/236 [==============================] - 3s 15ms/step - loss: 0.1430 - accuracy: 0.9419 - val_lo
ss: 0.2107 - val_accuracy: 0.9214
```

Out[ ]:

```
<tensorflow.python.keras.callbacks.History at 0x7f15e129c668>
```

In [ ]:

```
%load_ext tensorboard
```

```
%tensorboard --logdir logs/fit
```

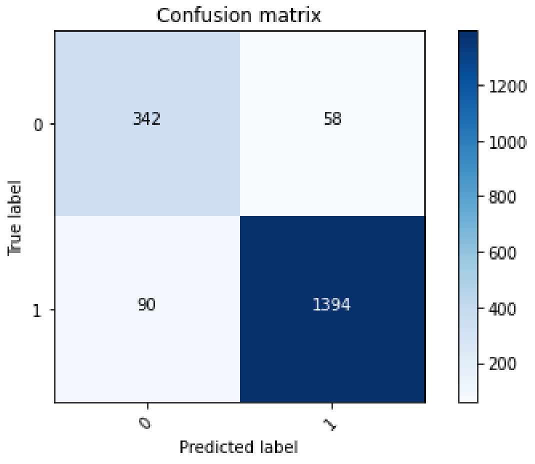
```
The tensorboard extension is already loaded. To reload it, use:
  %reload_ext tensorboard
```

In [ ]:

```
model1.save_weights("model1.h5")
```

In [ ]:

```
# Predict the values from the validation dataset and computing the confusion matrix
Y_pred = model1.predict(X_test)
Y_pred_classes = np.argmax(Y_pred,axis = 1)
Y_true = np.argmax(Y_test,axis = 1)

confusion_mtx = confusion_matrix(Y_true, Y_pred_classes)
plot_confusion_matrix(confusion_mtx, classes = range(2))
```



Classification model 2

In [ ]:

```
model2 = Sequential()
model2.add(ResNet50(include_top = False, pooling = 'avg', weights = 'imagenet'))
model2.add(Dense(256, activation = 'relu'))
model2.add(Dropout(0.1))
model2.add(Dense(128, activation='relu'))
model2.add(Dense(64, activation='relu'))
model2.add(Dropout(0.2))
model2.add(Dense(32, activation='relu'))
model2.add(Dropout(0.3))
model2.add(Dense(2, activation = 'softmax'))
```

In [ ]:

```
model2.summary()
```

```
Model: "sequential_29"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
resnet50 (Functional)        (None, 2048)              23587712  
_________________________________________________________________
dense_77 (Dense)             (None, 256)               524544    
_________________________________________________________________
dropout_69 (Dropout)         (None, 256)               0         
_________________________________________________________________
dense_78 (Dense)             (None, 128)               32896     
_________________________________________________________________
dense_79 (Dense)             (None, 64)                8256      
```

```
_________________________________________________________________
dropout_70 (Dropout)         (None, 64)                0
_________________________________________________________________
dense_80 (Dense)             (None, 32)                2080
_________________________________________________________________
dropout_71 (Dropout)         (None, 32)                0
_________________________________________________________________
dense_81 (Dense)             (None, 2)                 66
=================================================================
Total params: 24,155,554
Trainable params: 24,102,434
Non-trainable params: 53,120
_________________________________________________________________
```

In [ ]:

```
!rm -rf /content/logs2/fit
```

In [ ]:

```
epochs = 9
batch_size = 32
init_lr = 1e-4
optimizer = Adam(lr = init_lr, decay = init_lr/epochs)
model2.compile(optimizer = optimizer, loss = 'binary_crossentropy', metrics = ['accuracy'])
early_stopping = EarlyStopping(monitor = 'val_accuracy',min_delta = 0,patience = 4,mode = 'auto')
log_dir = "logs2/fit/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard_callback = TensorBoard(log_dir=log_dir, histogram_freq=1)
```

In [ ]:

```
model2.fit(X_train,Y_train,batch_size = batch_size,epochs = epochs,validation_data = (X_test, Y_tes
t),callbacks = [early_stopping,tensorboard_callback])
```

```
Epoch 1/9
  1/236 [..............................] - ETA: 0s - loss: 0.5391 - accuracy:
0.8125WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/tensorflow/python/ops/summary_ops_v2.py:1277: stop (from
tensorflow.python.eager.profiler) is deprecated and will be removed after 2020-07-01.
Instructions for updating:
use `tf.profiler.experimental.stop` instead.
  2/236 [..............................] - ETA: 30s - loss: 0.5436 - accuracy:
0.8281WARNING:tensorflow:Callbacks method `on_train_batch_end` is slow compared to the batch time
(batch time: 0.0634s vs `on_train_batch_end` time: 0.1940s). Check your callbacks.
236/236 [==============================] - 23s 99ms/step - loss: 0.2439 - accuracy: 0.8918 - val_l
oss: 1.8947 - val_accuracy: 0.7877
Epoch 2/9
236/236 [==============================] - 21s 90ms/step - loss: 0.1192 - accuracy: 0.9553 - val_l
oss: 3.2563 - val_accuracy: 0.7877
Epoch 3/9
236/236 [==============================] - 21s 90ms/step - loss: 0.0792 - accuracy: 0.9708 - val_l
oss: 3.2559 - val_accuracy: 0.7877
Epoch 4/9
236/236 [==============================] - 21s 90ms/step - loss: 0.0617 - accuracy: 0.9761 - val_l
oss: 3.2020 - val_accuracy: 0.7877
Epoch 5/9
236/236 [==============================] - 21s 91ms/step - loss: 0.0501 - accuracy: 0.9831 - val_l
oss: 1.0716 - val_accuracy: 0.7887
Epoch 6/9
236/236 [==============================] - 21s 90ms/step - loss: 0.0428 - accuracy: 0.9877 - val_l
oss: 0.2116 - val_accuracy: 0.9321
Epoch 7/9
236/236 [==============================] - 21s 90ms/step - loss: 0.0346 - accuracy: 0.9875 - val_l
oss: 0.1635 - val_accuracy: 0.9570
Epoch 8/9
236/236 [==============================] - 21s 90ms/step - loss: 0.0438 - accuracy: 0.9869 - val_l
oss: 0.1126 - val_accuracy: 0.9602
Epoch 9/9
236/236 [==============================] - 21s 90ms/step - loss: 0.0257 - accuracy: 0.9914 - val_l
oss: 0.1370 - val_accuracy: 0.9602
```

Out[ ]:

```
<tensorflow.python.keras.callbacks.History at 0x7f17f44fbe80>
```

In [ ]:

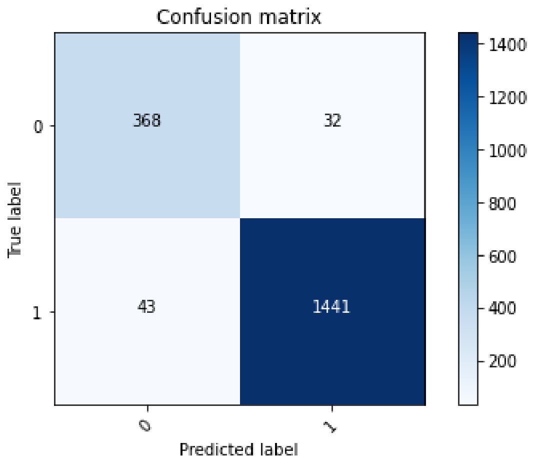
```
%load_ext tensorboard
%tensorboard --logdir logs2/fit
```

In [ ]:

```
model2.save_weights("model2.h5")
```

In [ ]:

```
# Predict the values from the validation dataset and computing the confusion matrix
Y_pred = model2.predict(X_test)
Y_pred_classes = np.argmax(Y_pred,axis = 1)
Y_true = np.argmax(Y_test,axis = 1)

confusion_mtx = confusion_matrix(Y_true, Y_pred_classes)
plot_confusion_matrix(confusion_mtx, classes = range(2))
```



Conclusion:

- We can see that model with Resnet performed well as compared to custom one for classification task.

Tampered region predicting with SRM

In [4]:

```
#defining SRM filter to be used
S = [[0, 0, 0, 0, 0],
    [0, -1, 2, -1, 0],
    [0, 2, -4, 2, 0],
    [0, -1, 2, -1, 0],
    [0, 0, 0, 0, 0]]
R = [[-1, 2, -2, 2, -1],
    [2, -6, 8, -6, 2],
    [-2, 8, -12, 8, -2],
    [2, -6, 8, -6, 2],
    [-1, 2, -2, 2, -1]]
M = [[0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0],
    [0, 1, -2, 1, 0],
    [0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0]]

S = np.asarray(S, dtype=float) / 4
R = np.asarray(R, dtype=float) / 12
M = np.asarray(M, dtype=float) / 2

srm_filter = S + R + M
print(srm_filter)
```

```
[[-0.08333333  0.16666667 -0.16666667  0.16666667 -0.08333333]
 [ 0.16666667 -0.75        1.16666667 -0.75        0.16666667]
 [-0.16666667  1.66666667 -3.          1.66666667 -0.16666667]
 [ 0.16666667 -0.75        1.16666667 -0.75        0.16666667]
 [-0.08333333  0.16666667 -0.16666667  0.16666667 -0.08333333]]
```

In [5]:

```
p = os.listdir('/content/dataset-dist/phase-01/training/fake')
p.remove('.DS_Store')
```

In [8]:

```
#Making directories
os.mkdir('/content/dataset-dist/phase-01/training/resized_images')
os.mkdir('/content/dataset-dist/phase-01/training/resized_images/fake_masks')
os.mkdir('/content/dataset-dist/phase-01/training/resized_images/fake_images')
os.mkdir('/content/dataset-dist/phase-01/training/resized_images/pristine_images')
os.mkdir('/content/dataset-dist/phase-01/training/SRM')
```

In [9]:

```
all_pristine = os.listdir('/content/dataset-dist/phase-01/training/pristine')
all_fake = os.listdir('/content/dataset-dist/phase-01/training/fake')
```

In [10]:

```
#resizing images
for fake_image in tqdm.tqdm(all_fake, position=0, leave=True):
    if(('.mask' in fake_image) and ('.DS_Store' not in fake_image)):
        img=Image.open('/content/dataset-dist/phase-01/training/fake/' + fake_image).convert("RGB")
        img = img.resize((512, 512), PIL.Image.ANTIALIAS)
        img.save('/content/dataset-dist/phase-01/training/resized_images/fake_masks/'+fake_image)
    else:
        if('.DS_Store' not in fake_image):
          img=Image.open('/content/dataset-dist/phase-01/training/fake/' + fake_image).convert("RGB
")
          img = img.resize((512, 512), PIL.Image.ANTIALIAS)
          img.save('/content/dataset-dist/phase-01/training/resized_images/fake_images/'+fake_image
)
```

```
 26%|██        | 231/901 [00:32<01:33,  7.19it/s]/usr/local/lib/python3.6/dist-
packages/PIL/Image.py:932: UserWarning: Palette images with Transparency expressed in bytes should
be converted to RGBA images
  "Palette images with Transparency expressed in bytes should be "
100%|██████████| 901/901 [02:08<00:00,  7.00it/s]
```

In [11]:

```
resized_fake = os.listdir('/content/dataset-dist/phase-01/training/resized_images/fake_images')
resized_mask = os.listdir('/content/dataset-dist/phase-01/training/resized_images/fake_masks')
resized_pristine = os.listdir('/content/dataset-dist/phase-
01/training/resized_images/pristine_images')
resized_fake_path = '/content/dataset-dist/phase-01/training/resized_images/fake_images/'
```

In [11]:

```
#making filtered images
for im in tqdm.tqdm(resized_fake, position=0, leave=True):
  try:
    img = imread('/content/dataset-dist/phase-01/training/fake/'+im)
    filtered_img = cv2.filter2D(img,-1,srm_filter)
    plt.imsave('/content/dataset-dist/phase-01/training/SRM/'+im,filtered_img)
  except:
    print('Bad file:', im)
```

```
100%|██████████| 450/450 [09:20<00:00,  1.25s/it]
```

In [15]:

```
fake_image_list_path = []
for i in os.listdir('/content/dataset-dist/phase-01/training/fake/'):
  if(('.mask.png' not in i) and ('.DS_Store' not in i)):
    fake_image_list_path.append('/content/dataset-dist/phase-01/training/fake/'+i)
```

In [16]:

```
fake_srm_list_path = ['/content/dataset-dist/phase-01/training/SRM/'+i for i in os.listdir('/conten
t/dataset-dist/phase-01/training/SRM/')]
fake_mask_list_path = ['/content/dataset-dist/phase-01/training/resized_images/fake_masks/'+i for i
in os.listdir('/content/dataset-dist/phase-01/training/resized_images/fake_masks')]
pristine_list_path = ['/content/dataset-dist/phase-01/training/resized_images/pristine_images/'+i
for i in os.listdir('/content/dataset-dist/phase-01/training/resized_images/pristine_images')]
# fake_image_list_path = ['/content/dataset-dist/phase-01/training/fake/'+i for i in
os.listdir('/content/dataset-dist/phase-01/training/fake')]
fake_srm_list_path.sort()
fake_mask_list_path.sort()
pristine_list_path.sort()
fake_image_list_path.sort()
```

In [21]:

```
os.mkdir('/content/dataset-dist/phase-01/training/srm_aug')
os.mkdir('/content/dataset-dist/phase-01/training/mask_aug')
os.mkdir('/content/dataset-dist/phase-01/training/fake_aug')
```

In [23]:

```
#Making augmentations
for i in tqdm.tqdm(range(len(fake_image_list_path)), position=0, leave=True):

    image=cv2.imread(fake_image_list_path[i])
    mask=cv2.imread(fake_mask_list_path[i])
    srm=cv2.imread(fake_srm_list_path[i])


    srm_name = fake_srm_list_path[i][44:]
    mask_name = fake_mask_list_path[i][66:]
    image_name = fake_image_list_path[i][45:]

    Image.fromarray(image).save('/content/dataset-dist/phase-01/training/fake_aug/'+ 'original_' +
image_name)
    Image.fromarray(image).save('/content/dataset-dist/phase-01/training/mask_aug/'+ 'original_' +
mask_name)
    Image.fromarray(image).save('/content/dataset-dist/phase-01/training/srm_aug/'+ 'original_' +
srm_name)


    hf=horizontalFlip(image,mask,srm)
    Image.fromarray(hf[0]).save(basic_path+'fake_aug/'+ 'hf_' + image_name)
    Image.fromarray(hf[1]).save(basic_path+'mask_aug/'+ 'hf_' + mask_name)
    Image.fromarray(hf[2]).save(basic_path+'srm_aug/'+ 'hf_' + srm_name)


    vf=verticalFlip(image,mask,srm)
    Image.fromarray(hf[0]).save(basic_path+'fake_aug/'+ 'vf_' + image_name)
    Image.fromarray(hf[1]).save(basic_path+'mask_aug/'+ 'vf_' + mask_name)
    Image.fromarray(hf[2]).save(basic_path+'srm_aug/'+ 'vf_' + srm_name)



    od=opticalDistortion(image,mask,srm)
    Image.fromarray(od[0]).save(basic_path+'fake_aug/'+ 'od_' + image_name)
    Image.fromarray(od[1]).save(basic_path+'mask_aug/'+ 'od_' + mask_name)
    Image.fromarray(od[2]).save(basic_path+'srm_aug/'+ 'od_' + srm_name)


    ed=elasticDistortion(image,mask,srm)
    Image.fromarray(ed[0]).save(basic_path+'fake_aug/'+ 'ed_' + image_name)
    Image.fromarray(ed[1]).save(basic_path+'mask_aug/'+ 'ed_' + mask_name)
    Image.fromarray(ed[2]).save(basic_path+'srm_aug/'+ 'ed_' + srm_name)
    ch=channelShuffle(image,mask,srm)
    Image.fromarray(ch[0]).save(basic_path+'fake_aug/'+ 'ch_' + image_name)
    Image.fromarray(ch[1]).save(basic_path+'mask_aug/'+ 'ch_' + mask_name)
    Image.fromarray(ch[2]).save(basic_path+'srm_aug/'+ 'ch_' + srm_name)




    rg=rGBShift(image,mask,srm)
    Image.fromarray(rg[0]).save(basic_path+'fake_aug/'+ 'rg_' + image_name)
    Image.fromarray(rg[1]).save(basic_path+'mask_aug/'+ 'rg_' + mask_name)
    Image.fromarray(rg[2]).save(basic_path+'srm_aug/'+ 'rg_' + srm_name)


    gd=gridDistortion(image,mask,srm)
    Image.fromarray(gd[0]).save(basic_path+'fake_aug/'+ 'gd_' + image_name)
    Image.fromarray(gd[1]).save(basic_path+'mask_aug/'+ 'gd_' + mask_name)
    Image.fromarray(gd[2]).save(basic_path+'srm_aug/'+ 'gd_' + srm_name)
```

```
100%|██████████| 450/450 [1:57:54<00:00, 15.72s/it]
```

In [3]:

```
fake_mask_list_path3 = os.listdir('/content/dataset-dist/phase-01/training/mask_aug')
fake_srm_list_path3 = os.listdir('/content/dataset-dist/phase-01/training/srm_aug')
fake_image_list_path3 = os.listdir('/content/dataset-dist/phase-01/training/fake_aug')

fake_mask_list_path3.sort()
fake_srm_list_path3.sort()
fake_image_list_path3.sort()
```

In [4]:

```
for idx,v in enumerate(fake_image_list_path3):
  temp=v
  fake_image_list_path3[idx]='/content/dataset-dist/phase-01/training/fake_aug/'+temp

for idx,v in enumerate(fake_mask_list_path3):
  temp=v
  fake_mask_list_path3[idx]='/content/dataset-dist/phase-01/training/mask_aug/'+temp

for idx,v in enumerate(fake_srm_list_path3):
  temp=v
  fake_srm_list_path3[idx]='/content/dataset-dist/phase-01/training/srm_aug/'+temp
```

In [5]:

```
#converting to numpy array and resizing images for final input
X_train1 = []
for filename in tqdm.tqdm(fake_image_list_path3[0:2600], position=0, leave=True):
  temp = resize(imread(filename),(256,256,3))
  X_train1.append(temp)
gc.collect()
X_train1 = np.array(X_train1)

X_train2 = []
for filename in tqdm.tqdm(fake_srm_list_path3[0:2600], position=0, leave=True):
  temp = resize(imread(filename),(256,256,3))
  X_train2.append(temp)
gc.collect()
X_train2 = np.array(X_train2)

Y_train = []
for filename in tqdm.tqdm(fake_mask_list_path3[0:2600], position=0, leave=True):
  temp = resize(imread(filename),(256,256,1))
  Y_train.append(temp)
gc.collect()
Y_train = np.array(Y_train)

X_val1 = []
for filename in tqdm.tqdm(fake_image_list_path3[2600:3600], position=0, leave=True):
  temp = resize(imread(filename),(256,256,3))
  X_val1.append(temp)
gc.collect()
X_val1 = np.array(X_val1)

X_val2 = []
for filename in tqdm.tqdm(fake_srm_list_path3[2600:3600], position=0, leave=True):
  temp = resize(imread(filename),(256,256,3))
  X_val2.append(temp)
gc.collect()
X_val2 = np.array(X_val2)

Y_val = []
for filename in tqdm.tqdm(fake_mask_list_path3[2600:3600], position=0, leave=True):
  temp = resize(imread(filename),(256,256,1))
  Y_val.append(temp)
gc.collect()
Y_val = np.array(Y_val)
```

```
100%|██████████| 2600/2600 [20:56<00:00,  2.07it/s]
100%|██████████| 2600/2600 [20:45<00:00,  2.09it/s]
100%|██████████| 2600/2600 [04:26<00:00,  9.77it/s]
100%|██████████| 1000/1000 [09:02<00:00,  1.84it/s]
100%|██████████| 1000/1000 [08:17<00:00,  2.01it/s]
100%|██████████| 1000/1000 [01:46<00:00,  9.40it/s]
```

In [ ]:

```
t1 = []
t2 = []
for im in tqdm.tqdm(all_fake):
  if('.mask.png' not in im):
    t1.append(im.split('.png')[0])
  else:
    t2.append(im.split('.mask.png')[0])
```

```
100%|██████████| 901/901 [00:00<00:00, 578015.89it/s]
```

In [ ]:

```
print(len(t1),len(t2))
```

```
451 450
```

In [ ]:

```
set(t1) ^ set(t2)
```

Out[ ]:

```
{'.DS_Store'}
```

In [ ]:

```
#preparing final datasets
input_fake_images = []
mask_images = []
filtered_images = []
for im in tqdm.tqdm(os.listdir('/content/dataset-dist/phase-01/training/fake')):
    if(('.mask.png' not in im) and ('.DS_Store' not in im)):
      input_fake_images.append('/content/dataset-dist/phase-01/training/fake/'+im)
      filtered_images.append('/content/dataset-dist/phase-01/training/filtered_images/'+im)
    if(('.mask.png' in im) and ('.DS_Store' not in im)):
      mask_images.append('/content/dataset-dist/phase-01/training/fake/'+im)
```

```
100%|██████████| 901/901 [00:00<00:00, 408106.69it/s]
```

In [ ]:

```
print(len(input_fake_images),len(mask_images),len(filtered_images))
```

```
450 450 450
```

In [ ]:

```
input_fake_images.sort()
mask_images.sort()
filtered_images.sort()
```

In [ ]:

```
#Splitting the data
X_train1 = np.array(input_fake_images[0:360])
X_train2 = np.array(filtered_images[0:360])
Y_train = np.array(mask_images[0:360])
X_val1 = np.array(input_fake_images[360:len(input_fake_images)])
X_val2 = np.array(filtered_images[360:len(filtered_images)])
Y_val = np.array(mask_images [360:len(mask_images)])
```

In [29]:

```
def reshape_data1(x):
  return np.array([resize(imread(file_name), (512, 512, 3)) for file_name in x])
```

In [30]:

```
def reshape_data2(y):
  return np.array([resize(imread(file_name), (512, 512, 1)) for file_name in y])
```

In [ ]:

```
X_train1 = reshape_data1(X_train1)
X_train2 = reshape_data1(X_train2)
X_val1 = reshape_data1(X_val1)
X_val2 = reshape_data1(X_val2)
Y_train = reshape_data2(Y_train)
Y_val = reshape_data2(Y_val)
```

In [6]:

```
save('/content/X_train1.npy',X_train1)
save('/content/X_train2.npy',X_train2)
save('/content/X_val1.npy',X_val1)
save('/content/X_val2.npy',X_val2)
save('/content/Y_train.npy',Y_train)
save('/content/Y_val.npy',Y_val)
```

In [3]:

```
X_train1 = load('/content/X_train1.npy')
X_train2 = load('/content/X_train2.npy')
X_val1 = load('/content/X_val1.npy')
X_val2 = load('/content/X_val2.npy')
Y_train = load('/content/Y_train.npy')
Y_val = load('/content/Y_val.npy')
```

Mask prediction with Resnet34 and Augmentation with SRM filter

In [11]:

```
#Training network with 2 inputs: one with fake image and second with filtered fake image
path_img = Unet(backbone_name='resnet34', encoder_weights='imagenet',
activation='sigmoid',input_shape=(512,512,3),decoder_use_batchnorm=True)
path_img._name = 'path_1'
out1 = Conv2D(3,(1,1), activation='sigmoid')(path_img.output)
drop1 = Dropout(0.1)(out1)

path_filter = Unet(backbone_name='resnet34', encoder_weights='imagenet',
activation='sigmoid',input_shape=(512,512,3),decoder_use_batchnorm=True)
path_filter._name = 'path_2'
out2 = Conv2D(3,(1,1), activation='sigmoid')(path_filter.output)
drop2 = Dropout(0.1)(out2)
```

In [12]:

```
for layer in path_img.layers:
    layer._name = layer.name + str("_img")
```

In [13]:

```
combined = concatenate([drop1, drop2])
drop3 = Dropout(0.2)(combined)
final = Conv2D(1,(1,1),activation='sigmoid')(drop3)
model = Model(inputs=[path_img.input,path_filter.input], outputs=[final])
```

In [23]:

```
!rm -rf '/content/logs3'
```

In [14]:

```
from segmentation_models.losses import bce_dice_loss,dice_loss
from keras.optimizers import Adam,SGD
optim = tf.keras.optimizers.Adam(0.0001)
focal_loss = sm.losses.cce_dice_loss
log_dir = "logs3/fit/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard_callback = TensorBoard(log_dir=log_dir, histogram_freq=1)
early_stopping = EarlyStopping(monitor = 'val_loss',min_delta = 0,patience = 3,mode = 'auto')
reduce_lr = ReduceLROnPlateau(monitor = 'val_loss', factor = 0.22, patience = 1, verbose = 1, min_d
elta = 0.0001)
model.compile(optimizer=optim, loss="binary_crossentropy", metrics=[metric])
```

In [15]:

```
model.fit([X_train1,X_train2], [Y_train],validation_data=([X_val1,X_val2], [Y_val]),epochs=18,
batch_size=1,callbacks=[tensorboard_callback,reduce_lr,early_stopping],verbose=1)
```

```
Epoch 1/18
  2/360 [..............................] - ETA: 2:56 - loss: 0.5434 - metric:
0.7392WARNING:tensorflow:Callbacks method `on_train_batch_end` is slow compared to the batch time
(batch time: 0.0621s vs `on_train_batch_end` time: 0.9228s). Check your callbacks.
360/360 [==============================] - 39s 107ms/step - loss: 0.5099 - metric: 0.7520 - val_lo
ss: 0.4465 - val_metric: 0.7787
Epoch 2/18
360/360 [==============================] - 35s 98ms/step - loss: 0.4580 - metric: 0.7810 - val_los
s: 0.3997 - val_metric: 0.8046
Epoch 3/18
360/360 [==============================] - 35s 98ms/step - loss: 0.4217 - metric: 0.8020 - val_los
s: 0.3635 - val_metric: 0.8252
Epoch 4/18
360/360 [==============================] - 35s 98ms/step - loss: 0.3927 - metric: 0.8193 - val_los
s: 0.3335 - val_metric: 0.8430
Epoch 5/18
360/360 [==============================] - 35s 98ms/step - loss: 0.3695 - metric: 0.8339 - val_los
s: 0.3107 - val_metric: 0.8567
Epoch 6/18
360/360 [==============================] - 35s 98ms/step - loss: 0.3503 - metric: 0.8468 - val_los
s: 0.2898 - val_metric: 0.8702
Epoch 7/18
360/360 [==============================] - 35s 98ms/step - loss: 0.3351 - metric: 0.8577 - val_los
s: 0.2742 - val_metric: 0.8805
Epoch 8/18
360/360 [==============================] - 35s 98ms/step - loss: 0.3225 - metric: 0.8668 - val_los
s: 0.2696 - val_metric: 0.8838
Epoch 9/18
360/360 [==============================] - 35s 98ms/step - loss: 0.3126 - metric: 0.8747 - val_los
s: 0.2516 - val_metric: 0.8966
Epoch 10/18
360/360 [==============================] - 35s 98ms/step - loss: 0.3041 - metric: 0.8811 - val_los
s: 0.2458 - val_metric: 0.9006
```

```
s: 0.2458 - val_metric: 0.9060
Epoch 11/18
360/360 [==============================] - 35s 98ms/step - loss: 0.2971 - metric: 0.8870 - val_los
s: 0.2357 - val_metric: 0.9077
Epoch 12/18
360/360 [==============================] - 35s 98ms/step - loss: 0.2907 - metric: 0.8916 - val_los
s: 0.2325 - val_metric: 0.9114
Epoch 13/18
360/360 [==============================] - 35s 98ms/step - loss: 0.2855 - metric: 0.8956 - val_los
s: 0.2255 - val_metric: 0.9163
Epoch 14/18
360/360 [==============================] - 35s 98ms/step - loss: 0.2801 - metric: 0.8994 - val_los
s: 0.2216 - val_metric: 0.9202
Epoch 15/18
360/360 [==============================] - ETA: 0s - loss: 0.2754 - metric: 0.9025
Epoch 00015: ReduceLROnPlateau reducing learning rate to 2.1999999444233253e-05.
360/360 [==============================] - 35s 98ms/step - loss: 0.2754 - metric: 0.9025 - val_los
s: 0.2221 - val_metric: 0.9203
Epoch 16/18
360/360 [==============================] - 35s 98ms/step - loss: 0.2720 - metric: 0.9044 - val_los
s: 0.2176 - val_metric: 0.9236
Epoch 17/18
360/360 [==============================] - 35s 98ms/step - loss: 0.2697 - metric: 0.9054 - val_los
s: 0.2170 - val_metric: 0.9248
Epoch 18/18
360/360 [==============================] - 35s 98ms/step - loss: 0.2682 - metric: 0.9060 - val_los
s: 0.2168 - val_metric: 0.9248
```

Out[15]:

```
<tensorflow.python.keras.callbacks.History at 0x7f9f97f98048>
```

In [16]:

```
model.save('model.h5')
```

```
---------------------------------------------------------------------------
RuntimeError                              Traceback (most recent call last)
<ipython-input-16-596723284980> in <module>()
----> 1 model.save('model.h5')

/usr/local/lib/python3.6/dist-packages/tensorflow/python/keras/engine/training.py in save(self, fi
lepath, overwrite, include_optimizer, save_format, signatures, options)
   1977     """
   1978     save.save_model(self, filepath, overwrite, include_optimizer, save_format,
-> 1979                     signatures, options)
   1980 
   1981   def save_weights(self,

/usr/local/lib/python3.6/dist-packages/tensorflow/python/keras/saving/save.py in save_model(model,
filepath, overwrite, include_optimizer, save_format, signatures, options)
    129           'or using `save_weights`.')
    130     hdf5_format.save_model_to_hdf5(
--> 131         model, filepath, overwrite, include_optimizer)
    132   else:
    133     saved_model_save.save(model, filepath, overwrite, include_optimizer,

/usr/local/lib/python3.6/dist-packages/tensorflow/python/keras/saving/hdf5_format.py in
save_model_to_hdf5(model, filepath, overwrite, include_optimizer)
    123     if (include_optimizer and model.optimizer and
    124         not isinstance(model.optimizer, optimizers.TFOptimizer)):
--> 125       save_optimizer_weights_to_hdf5_group(f, model.optimizer)
    126 
    127     f.flush()

/usr/local/lib/python3.6/dist-packages/tensorflow/python/keras/saving/hdf5_format.py in
save_optimizer_weights_to_hdf5_group(hdf5_group, optimizer)
    592     for name, val in zip(weight_names, weight_values):
    593       param_dset = weights_group.create_dataset(
--> 594           name, val.shape, dtype=val.dtype)
    595       if not val.shape:
    596         # scalar

/usr/local/lib/python3.6/dist-packages/h5py/_hl/group.py in create_dataset(self, name, shape,
dtype, data, **kwds)
    137             dset = dataset.Dataset(dsid)
    138             if name is not None:
--> 139                 self[name] = dset
    140             return dset
    141 

/usr/local/lib/python3.6/dist-packages/h5py/_hl/group.py in __setitem__(self, name, obj)
    371 
    372             if isinstance(obj, HLObject):
--> 373                 h5o.link(obj.id, self.id, name, lcpl=lcpl, lapl=self._lapl)
    374 
    375             elif isinstance(obj, SoftLink):

h5py/_objects.pyx in h5py._objects.with_phil.wrapper()

h5py/_objects.pyx in h5py._objects.with_phil.wrapper()

h5py/h5o.pyx in h5py.h5o.link()

RuntimeError: Unable to create link (name already exists)
```

In [17]:

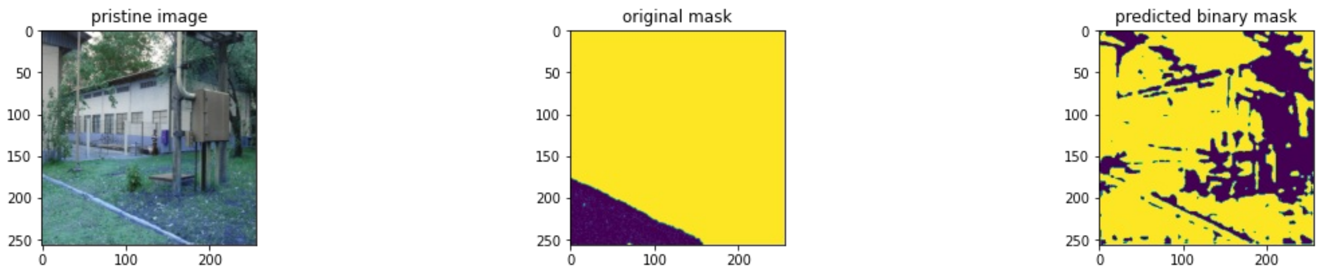
```
predict = model.predict([X_val1,X_val2])
```

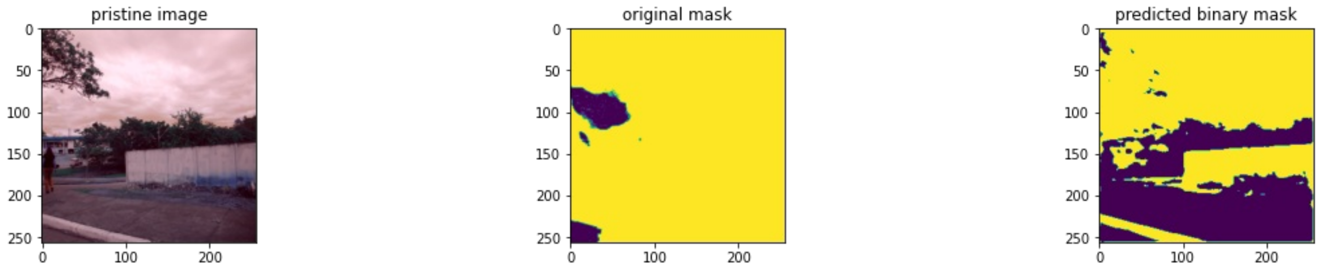
In [18]:

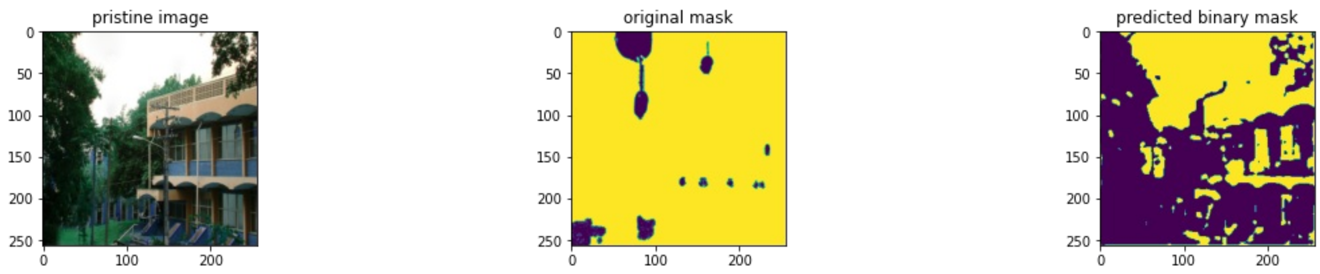
```
#Function to plot images
def plot_predicted_images(index):
    pred = np.squeeze(predict[index])
    plt.imsave('pred_mask.png',pred)
    im_gray = cv2.imread('pred_mask.png', cv2.IMREAD_GRAYSCALE)
    (thresh, im_bw) = cv2.threshold(im_gray, 220, 255, cv2.THRESH_BINARY | cv2.THRESH_OTSU)
    fig = plt.figure(figsize=(20,10))
    ax1 = fig.add_subplot(331)
    ax2 = fig.add_subplot(332)
    ax3 = fig.add_subplot(333)


    ax1.set_title("pristine image")
    ax2.set_title("original mask")
    ax3.set_title("predicted binary mask")
    ax1.imshow(X_val1[index])
    ax2.imshow(np.squeeze(Y_val[index]))
    ax3.imshow(im_bw)
```

In [19]:

```
plot_predicted_images(200)
plot_predicted_images(321)
plot_predicted_images(555)
```







Conclusion:

- We can observe a lot of noise in predicted mask which is not very good.

Tampered region predictor model with ELA filter

In [12]:

```
os.mkdir('/content/dataset-dist/phase-01/training/resized_images')
os.mkdir('/content/dataset-dist/phase-01/training/resized_images/fake_masks')
os.mkdir('/content/dataset-dist/phase-01/training/resized_images/fake_images')
os.mkdir('/content/dataset-dist/phase-01/training/resized_images/pristine_images')
os.mkdir('/content/dataset-dist/phase-01/training/ELA')
```

```
---------------------------------------------------------------------------
FileExistsError                           Traceback (most recent call last)
<ipython-input-12-b43a426c0630> in <module>()
----> 1 os.mkdir('/content/dataset-dist/phase-01/training/resized_images')
      2 os.mkdir('/content/dataset-dist/phase-01/training/resized_images/fake_masks')
      3 os.mkdir('/content/dataset-dist/phase-01/training/resized_images/fake_images')
      4 os.mkdir('/content/dataset-dist/phase-01/training/resized_images/pristine_images')
      5 os.mkdir('/content/dataset-dist/phase-01/training/ELA')

FileExistsError: [Errno 17] File exists: '/content/dataset-dist/phase-01/training/resized_images'
```

In [8]:

```
all_pristine = os.listdir('/content/dataset-dist/phase-01/training/pristine')
all_fake = os.listdir('/content/dataset-dist/phase-01/training/fake')
```

In [9]:

```
#resizing images
for fake_image in tqdm.tqdm(all_fake, position=0, leave=True):
    if(('.mask' in fake_image) and ('.DS_Store' not in fake_image)):
        img=Image.open('/content/dataset-dist/phase-01/training/fake/' + fake_image).convert("RGB")
        img = img.resize((512, 512), PIL.Image.ANTIALIAS)
        img.save('/content/dataset-dist/phase-01/training/resized_images/fake_masks/'+fake_image)
    else:
        if('.DS_Store' not in fake_image):
          img=Image.open('/content/dataset-dist/phase-01/training/fake/' + fake_image).convert("RGB
")
          img = img.resize((512, 512), PIL.Image.ANTIALIAS)
          img.save('/content/dataset-dist/phase-01/training/resized_images/fake_images/'+fake_image
)

for pristine_image in tqdm.tqdm(all_pristine, position=0, leave=True):
        img=Image.open('/content/dataset-dist/phase-01/training/pristine/' + pristine_image).conver
t("RGB")
        img = img.resize((512, 512), PIL.Image.ANTIALIAS)
        img.save('/content/dataset-dist/phase-01/training/resized_images/pristine_images/'+pristine
_image)
```

```
 39%|████      | 349/901 [00:50<01:35,  5.77it/s]/usr/local/lib/python3.6/dist-
packages/PIL/Image.py:932: UserWarning: Palette images with Transparency expressed in bytes should
be converted to RGBA images
  "Palette images with Transparency expressed in bytes should be "
100%|██████████| 901/901 [02:08<00:00,  7.04it/s]
100%|██████████| 1050/1050 [02:44<00:00,  6.40it/s]
```

In [14]:

```
fake_names_intersection = os.listdir('/content/dataset-dist/phase-
01/training/resized_images/fake_masks/')
```

In [17]:

```
#Adding gaussian blur to masks to reduce noise and converting to single channel
# os.mkdir('/content/dataset-dist/phase-01/training/resized_images/binary_masks')
bin_masks =[]
for mask in tqdm.tqdm(fake_names_intersection,position=0, leave=True):
    mask_img = cv2.imread('/content/dataset-dist/phase-01/training/resized_images/fake_masks/'+mask
)[:,:,:1]
    blur = cv2.GaussianBlur(mask_img,(5,5),0) #Adding gaussian blur
    ret,bin_mask = cv2.threshold(blur,0,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU) #modifing the values
to either 0 or 255
    cv2.imwrite('/content/dataset-dist/phase-
01/training/resized_images/binary_masks/'+mask+'.mask.png',bin_mask)
    bin_masks.append(bin_mask)
```

```
100%|██████████| 450/450 [00:02<00:00, 216.22it/s]
```

In [18]:

```
#method to convert to ela
def ELA(img_path):
    TEMP = 'ela_' + 'temp.jpg'
    SCALE = 10
    original = Image.open(img_path)
    try:
        original.save(TEMP, quality=90)
        temporary = Image.open(TEMP)
        diff = ImageChops.difference(original, temporary)

    except:

        original.convert('RGB').save(TEMP, quality=90)
        temporary = Image.open(TEMP)
        diff = ImageChops.difference(original.convert('RGB'), temporary)


    d = diff.load()

    WIDTH, HEIGHT = diff.size
    for x in range(WIDTH):
        for y in range(HEIGHT):
            d[x, y] = tuple(k * SCALE for k in d[x, y])
    return diff
```

In [19]:

```
resized_fake = os.listdir('/content/dataset-dist/phase-01/training/resized_images/fake_images')
# resized_mask = os.listdir('/content/dataset-dist/phase-01/training/resized_images/fake_masks')
resized_mask = os.listdir('/content/dataset-dist/phase-01/training/resized_images/binary_masks')
resized_pristine = os.listdir('/content/dataset-dist/phase-
01/training/resized_images/pristine_images')
resized_fake_path = '/content/dataset-dist/phase-01/training/resized_images/fake_images/'
```

In [21]:

```
#generating ela
os.mkdir('/content/dataset-dist/phase-01/training/ELA/')
for img in tqdm.tqdm(resized_fake, position=0, leave=True):
    ELA(resized_fake_path+img).save('/content/dataset-dist/phase-01/training/ELA/'+img)
```

```
100%|██████████| 450/450 [03:11<00:00,  2.35it/s]
```

In [22]:

```
fake_image_list_path = []
for i in os.listdir('/content/dataset-dist/phase-01/training/fake/'):
  if(('.mask.png' not in i) and ('.DS_Store' not in i)):
    fake_image_list_path.append('/content/dataset-dist/phase-01/training/fake/'+i)
```

In [23]:

```
len(fake_image_list_path)
```

Out[23]:

```
450
```

In [24]:

```
fake_ela_list_path = ['/content/dataset-dist/phase-01/training/ELA/'+i for i in os.listdir('/conten
t/dataset-dist/phase-01/training/ELA/')]
# fake_mask_list_path = ['/content/dataset-dist/phase-01/training/resized_images/fake_masks/'+i fo
r i in os.listdir('/content/dataset-dist/phase-01/training/resized_images/fake_masks')]
fake_mask_list_path = ['/content/dataset-dist/phase-01/training/resized_images/binary_masks/'+i
for i in os.listdir('/content/dataset-dist/phase-01/training/resized_images/binary_masks')]
pristine_list_path = ['/content/dataset-dist/phase-01/training/resized_images/pristine_images/'+i
for i in os.listdir('/content/dataset-dist/phase-01/training/resized_images/pristine_images')]
# fake_image_list_path = ['/content/dataset-dist/phase-01/training/fake/'+i for i in
os.listdir('/content/dataset-dist/phase-01/training/fake')]
```

In [25]:

```
fake_ela_list_path.sort()
fake_mask_list_path.sort()
pristine_list_path.sort()
fake_image_list_path.sort()
```

In [26]:

```
len('/content/dataset-dist/phase-01/training/ELA/')
```

Out[26]:

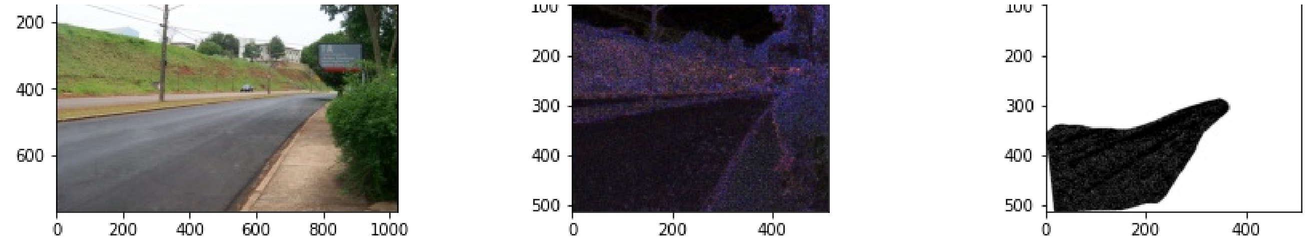
```
44
```

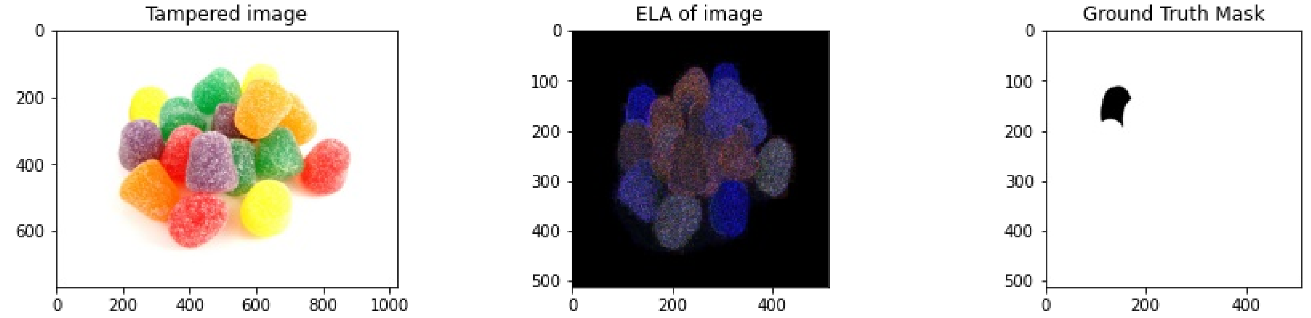
In [27]:

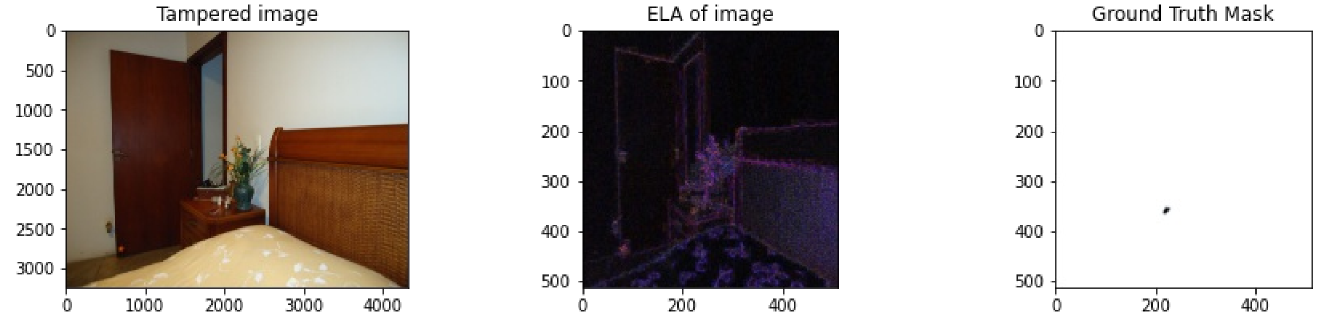
```
def display_img(index):

    fig = plt.figure(figsize=(15,10))
    ax1 = fig.add_subplot(331)
    ax2 = fig.add_subplot(332)
    ax3 = fig.add_subplot(333)
    ax1.set_title("Tampered image")
    ax2.set_title("ELA of image")
    ax3.set_title("Ground Truth Mask")

    ela_fake = Image.open(fake_ela_list_path[index])
    fake_mask = Image.open(fake_mask_list_path[index])
    tampered_image = Image.open('/content/dataset-dist/phase-01/training/fake/'+fake_ela_list_path[
index][44:])
    ax1.imshow(tampered_image)
    ax2.imshow(ela_fake)
    ax3.imshow(fake_mask)
```

In [ ]:

```
display_img(1)
display_img(12)
display_img(54)
```







In [31]:

```
X_train1 = np.array(fake_image_list_path[0:360])
X_train2 = np.array(fake_ela_list_path[0:360])
Y_train = np.array(fake_mask_list_path[0:360])
X_val1 = np.array(fake_image_list_path[360:len(fake_image_list_path)])
X_val2 = np.array(fake_ela_list_path[360:len(fake_ela_list_path)])
Y_val = np.array(fake_mask_list_path [360:len(fake_mask_list_path)])

X_train1 = reshape_data1(X_train1)
X_train2 = reshape_data1(X_train2)
X_val1 = reshape_data1(X_val1)
X_val2 = reshape_data1(X_val2)
Y_train = reshape_data2(Y_train)
Y_val = reshape_data2(Y_val)
```

In [32]:

```
print(X_train1.shape)
print(X_train2.shape)
print(Y_train.shape)
```

```
(360, 512, 512, 3)
(360, 512, 512, 3)
(360, 512, 512, 1)
```

In [34]:

```
print(X_val1.shape)
print(X_val2.shape)
print(Y_val.shape)
```

```
(90, 512, 512, 3)
(90, 512, 512, 3)
(90, 512, 512, 1)
```

In [33]:

```
save('/content/X_train1.npy',X_train1)
save('/content/X_train2.npy',X_train2)
save('/content/X_val1.npy',X_val1)
save('/content/X_val2.npy',X_val2)
save('/content/Y_train.npy',Y_train)
save('/content/Y_val.npy',Y_val)
```

In [4]:

```
X_train1 = load('/content/X_train1.npy')
X_train2 = load('/content/X_train2.npy')
X_val1 = load('/content/X_val1.npy')
X_val2 = load('/content/X_val2.npy')
Y_train = load('/content/Y_train.npy')
Y_val = load('/content/Y_val.npy')
```

Mask prediction model with VGG16 with Images and ELA filtered images without augmentation

In [ ]:

```
#Training network with 2 inputs: one with fake image and second with filtered fake image
path_img = Unet(backbone_name='vgg16', encoder_weights='imagenet',
activation='sigmoid',input_shape=(512,512,3))
path_img._name = 'path_1'
out1 = Conv2D(3,(1,1), activation='sigmoid')(path_img.output)

path_filter = Unet(backbone_name='vgg16', encoder_weights='imagenet',
activation='sigmoid',input_shape=(512,512,3))
path_filter._name = 'path_2'
out2 = Conv2D(3,(1,1), activation='sigmoid')(path_filter.output)
```

In [ ]:

```
for layer in path_img.layers:
    layer._name = layer.name + str("_img")
```

In [ ]:

```
combined = concatenate([out1, out2])
final = Conv2D(1,(1,1),activation='sigmoid')(combined)
model2 = Model(inputs=[path_img.input,path_filter.input], outputs=[final])
```

In [ ]:

```
!rm -rf '/content/logs4/fit'
```

In [ ]:

```
from segmentation_models.losses import bce_dice_loss,dice_loss
from keras.optimizers import Adam,SGD
optim = tf.keras.optimizers.Adam(0.0001)
focal_loss = sm.losses.cce_dice_loss
log_dir = "logs4/fit/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard_callback = TensorBoard(log_dir=log_dir, histogram_freq=1)
# early_stopping = EarlyStopping(monitor = 'val_accuracy',min_delta = 0,patience = 2,mode = 'auto'
)
# reduce_lr = ReduceLROnPlateau(monitor = 'val_loss', factor = 0.22, patience = 1, verbose = 1, mi
n_delta = 0.0001)
model2.compile(optimizer=optim, loss="binary_crossentropy", metrics=[metric])
```

In [ ]:

```
model2.fit([X_train1,X_train2], [Y_train],validation_data=([X_val1,X_val2], [Y_val]),epochs=30, bat
ch_size=1,callbacks=[tensorboard_callback],verbose=1)
```

```
Epoch 1/10
  2/360 [..............................] - ETA: 1:24 - loss: 0.1426 - metric:
0.9457WARNING:tensorflow:Callbacks method `on_train_batch_end` is slow compared to the batch time
(batch time: 0.0370s vs `on_train_batch_end` time: 0.2904s). Check your callbacks.
360/360 [==============================] - ETA: 0s - loss: 0.2541 - metric:
0.9209WARNING:tensorflow:Callbacks method `on_test_batch_end` is slow compared to the batch time (
batch time: 0.0087s vs `on_test_batch_end` time: 0.0376s). Check your callbacks.
360/360 [==============================] - 66s 185ms/step - loss: 0.2541 - metric: 0.9209 - val_lo
ss: 0.2109 - val_metric: 0.9311
Epoch 2/10
360/360 [==============================] - 66s 183ms/step - loss: 0.2536 - metric: 0.9218 - val_lo
ss: 0.2147 - val_metric: 0.9261
Epoch 3/10
360/360 [==============================] - 66s 183ms/step - loss: 0.2532 - metric: 0.9227 - val_lo
ss: 0.2154 - val_metric: 0.9255
Epoch 4/10
360/360 [==============================] - 66s 183ms/step - loss: 0.2530 - metric: 0.9232 - val_lo
ss: 0.2147 - val_metric: 0.9272
Epoch 5/10
360/360 [==============================] - 66s 183ms/step - loss: 0.2528 - metric: 0.9231 - val_lo
ss: 0.2080 - val_metric: 0.9342
Epoch 6/10
360/360 [==============================] - 66s 183ms/step - loss: 0.2525 - metric: 0.9239 - val_lo
ss: 0.2084 - val_metric: 0.9337
Epoch 7/10
360/360 [==============================] - 66s 183ms/step - loss: 0.2524 - metric: 0.9245 - val_lo
ss: 0.2090 - val_metric: 0.9324
Epoch 8/10
360/360 [==============================] - 66s 183ms/step - loss: 0.2522 - metric: 0.9245 - val_lo
ss: 0.2075 - val_metric: 0.9353
Epoch 9/10
360/360 [==============================] - 66s 183ms/step - loss: 0.2522 - metric: 0.9246 - val_lo
ss: 0.2125 - val_metric: 0.9304
Epoch 10/10
360/360 [==============================] - 66s 183ms/step - loss: 0.2520 - metric: 0.9253 - val_lo
ss: 0.2073 - val_metric: 0.9356
```

Out[ ]:

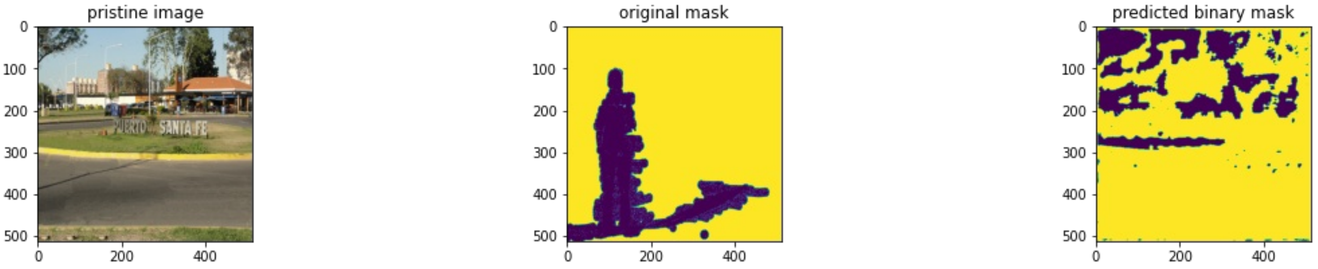
```
<tensorflow.python.keras.callbacks.History at 0x7fd0c1f57e10>
```

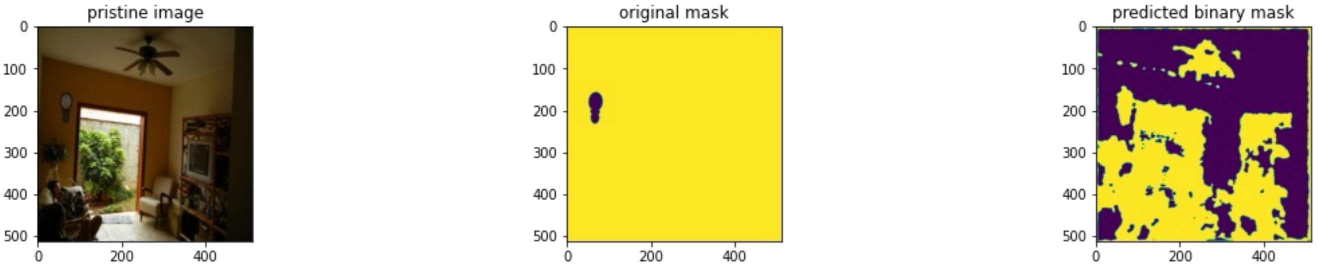
In [ ]:

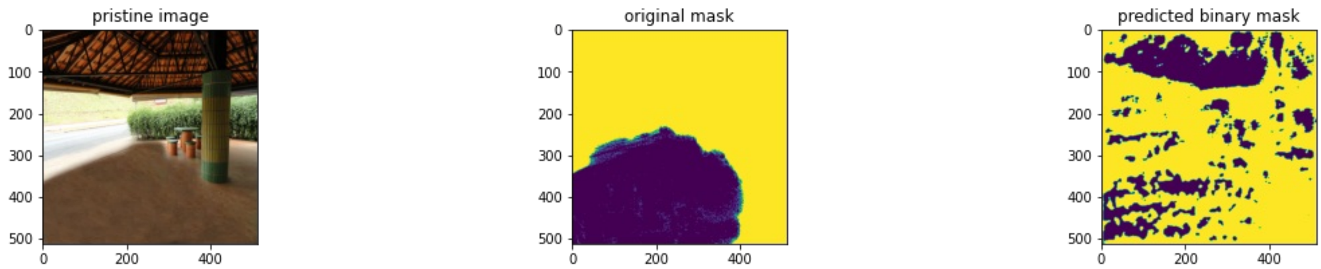
```
predict = model2.predict([X_val1,X_val2])
```

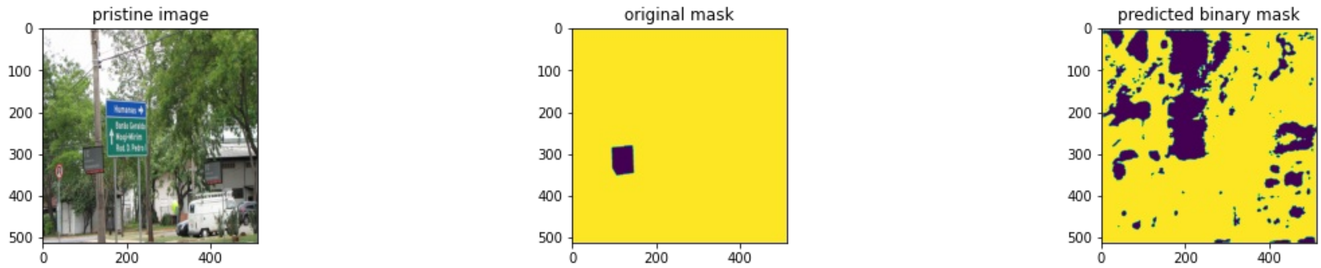
In [ ]:

```
plot_predicted_images(20)
plot_predicted_images(55)
plot_predicted_images(11)
plot_predicted_images(40)
```









Conclusion

- We can see still there is a lot of noise present in the predicted mask.

Adding augmentation

In [40]:

```
os.mkdir('/content/dataset-dist/phase-01/training/ela_aug')
os.mkdir('/content/dataset-dist/phase-01/training/mask_aug')
os.mkdir('/content/dataset-dist/phase-01/training/fake_aug')
```

```
---------------------------------------------------------------------------
FileExistsError                           Traceback (most recent call last)
<ipython-input-40-47d62efceec3> in <module>()
----> 1 os.mkdir('/content/dataset-dist/phase-01/training/ela_aug')
      2 os.mkdir('/content/dataset-dist/phase-01/training/mask_aug')
      3 os.mkdir('/content/dataset-dist/phase-01/training/fake_aug')

FileExistsError: [Errno 17] File exists: '/content/dataset-dist/phase-01/training/ela_aug'
```

In [36]:

```
#Defining augmentation functions
from albumentations import *
def horizontalFlip(image,mask,ela):
    aug = HorizontalFlip(p=1)
    augmented = aug(image=image, mask=mask,ela=ela)
    image_hflip = augmented['image']
    mask_hflip = augmented['mask']
    ela_hflip = augmented['ela']
    return image_hflip,mask_hflip,ela_hflip

def verticalFlip(image,mask,ela):
    aug = VerticalFlip(p=1)
    augmented = aug(image=image, mask=mask,ela=ela)
    image_vflip = augmented['image']
    mask_vflip = augmented['mask']
    ela_vflip = augmented['ela']
    return image_vflip,mask_vflip,ela_vflip

def randomRotate(image,mask,ela):
    aug = RandomRotate90(p=1)
    augmented = aug(image=image, mask=mask,ela=ela)
    image_rot90 = augmented['image']
    mask_rot90 = augmented['mask']
    ela_rot90 = augmented['ela']
    return image_rot90,mask_rot90,ela_rot90
```

```python
def transpose(image,mask,ela):
    aug = Transpose(p=1)
    augmented = aug(image=image, mask=mask,ela=ela)
    image_transpose = augmented['image']
    mask_transpose = augmented['mask']
    ela_transpose = augmented['ela']
    return image_transpose,mask_transpose,ela_transpose

def elasticDistortion(image,mask,ela):
    aug = ElasticTransform(p=1, alpha=120, sigma=120 * 0.05, alpha_affine=120 * 0.03)
    augmented = aug(image=image, mask=mask,ela=ela)
    image_ed = augmented['image']
    mask_ed = augmented['mask']
    ela_ed = augmented['ela']
    return image_ed,mask_ed,ela_ed

def opticalDistortion(image,mask,ela):
    aug = OpticalDistortion(p=1, distort_limit=2, shift_limit=0.5)
    augmented = aug(image=image, mask=mask,ela=ela)
    image_od = augmented['image']
    mask_od = augmented['mask']
    ela_od = augmented['ela']
    return image_od,mask_od,ela_od

def gridDistortion(image,mask,ela):
    aug = GridDistortion(p=1)
    augmented = aug(image=image, mask=mask,ela=ela)
    image_od = augmented['image']
    mask_od = augmented['mask']
    ela_od = augmented['ela']
    return image_od,mask_od,ela_od




def augment_flips_Color(image,mask,ela):
    aug = augment_flips_color(p=1)
    augmented = aug(image=image, mask=mask,ela=ela)
    image_od = augmented(image=image)['image']
    mask_od = augmented(image=mask)['image']
    ela_od = augmented(image=mask)['ela']
    return image_od,mask_od,ela_od



def channelShuffle(image,mask,ela):
    aug = ChannelShuffle(p=1)
    augmented = aug(image=image, mask=mask,ela=ela)
    image_od = augmented['image']
    mask_od = augmented['mask']
    ela_od = augmented['ela']
    return image_od,mask_od,ela_od



def rotate(image,mask,ela):

    aug = Rotate()
    augmented = aug(image=image, mask=mask,ela=ela)
    image_od = augmented['image']
    mask_od = augmented['mask']
    ela_od = augmented['ela']
    return image_od,mask_od,ela_od

def rGBShift(image,mask,ela):

    aug = RGBShift()
    augmented = aug(image=image, mask=mask,ela=ela)
    image_od = augmented['image']
    mask_od = augmented['mask']
    ela_od = augmented['ela']
    return image_od,mask_od,ela_od

def randomGamma(image,mask,ela):

    aug = RandomGamma()
    augmented = aug(image=image, mask=mask,ela=ela)
    image_od = augmented['image']
    mask_od = augmented['mask']
    ela_od = augmented['ela']
    return image_od,mask_od,ela_od

def flip(image,mask,ela):
    aug = Flip()
    augmented = aug(image=image, mask=mask,ela=ela)
    image_od = augmented['image']
    mask_od = augmented['mask']
    ela_od = augmented['ela']
    return image_od,mask_od,ela_od



def hueSaturationValue(image,mask,ela):
    aug = HueSaturationValue(p=1)
    augmented = aug(image=image, mask=mask,ela=ela)
    image_od = augmented['image']
    mask_od = augmented['mask']
    ela_od = augmented['ela']
    return image_od,mask_od,ela_od
```

In [18]:

```
pristine_list_path2 = os.listdir('/content/dataset-dist/phase-
01/training/resized_images/pristine_images')
fake_mask_list_path2 = os.listdir('/content/dataset-dist/phase-
01/training/resized_images/fake_masks')
fake_ela_list_path2 = os.listdir('/content/dataset-dist/phase-01/training/ELA')
fake_image_list_path2 = os.listdir('/content/dataset-dist/phase-
01/training/resized_images/fake_images')
```

In [19]:

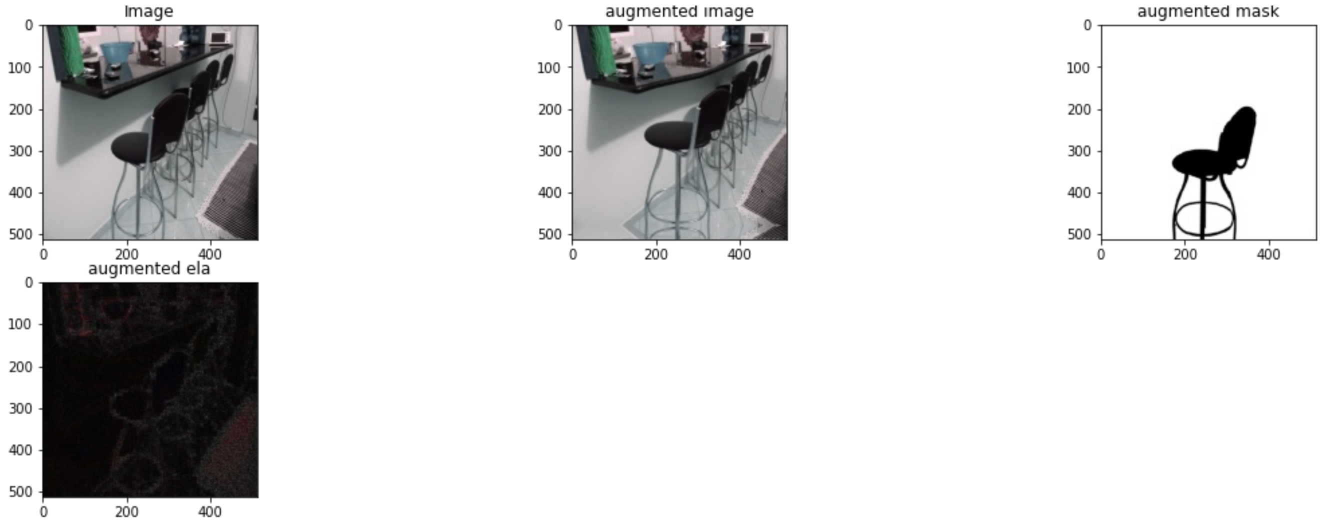
```
pristine_list_path2.sort()
fake_mask_list_path2.sort()
fake_ela_list_path2.sort()
fake_image_list_path2.sort()
```

In [ ]:

```
image = cv2.imread('/content/dataset-dist/phase-01/training/resized_images/fake_images/' +
fake_image_list_path2[14])
mask = cv2.imread('/content/dataset-dist/phase-01/training/resized_images/fake_masks/'+
fake_mask_list_path2[14])
ela = cv2.imread('/content/dataset-dist/phase-01/training/ELA/' + fake_ela_list_path2[14])

image_aug = gridDistortion(image,mask,ela)[0]
mask_aug = gridDistortion(image,mask,ela)[1]
ela_aug = gridDistortion(image,mask,ela)[2]

fig = plt.figure(figsize=(20,10))
ax1 = fig.add_subplot(331)
ax2 = fig.add_subplot(332)
ax3 = fig.add_subplot(333)
ax4 = fig.add_subplot(334)


ax1.set_title("Image")
ax2.set_title("augmented image")
ax3.set_title("augmented mask")
ax4.set_title("augmented ela")

ax1.imshow(image)
ax2.imshow(image_aug)
ax3.imshow(mask_aug)
ax4.imshow(ela_aug)
```

Out[ ]:

```
<matplotlib.image.AxesImage at 0x7fcf1b3a2d30>
```



In [37]:

```
basic_path = '/content/dataset-dist/phase-01/training/'
```

In [43]:

```
fake_mask_list_path[0][68:]
```

Out[43]:

```
'010543abfbd0db1e9aa1b24604336e0c.mask.png.mask.png'
```

In [44]:

```
#Generating augmented images
for i in tqdm.tqdm(range(len(fake_image_list_path)), position=0, leave=True):

    image=cv2.imread(fake_image_list_path[i])
    mask=cv2.imread(fake_mask_list_path[i])
    ela=cv2.imread(fake_ela_list_path[i])


    ela_name = fake_ela_list_path[i][44:]
    mask_name = fake_mask_list_path[i][68:]
    image_name = fake_image_list_path[i][45:]

    Image.fromarray(image).save('/content/dataset-dist/phase-01/training/fake_aug/'+ 'original_' +
image_name)
    Image.fromarray(image).save('/content/dataset-dist/phase-01/training/mask_aug/'+ 'original_' +
mask_name)
    Image.fromarray(image).save('/content/dataset-dist/phase-01/training/ela_aug/'+ 'original_' +
ela_name)


    hf=horizontalFlip(image,mask,ela)
    Image.fromarray(hf[0]).save(basic_path+'fake_aug/'+ 'hf_' + image_name)
    Image.fromarray(hf[1]).save(basic_path+'mask_aug/'+ 'hf_' + mask_name)
    Image.fromarray(hf[2]).save(basic_path+'ela_aug/'+ 'hf_' + ela_name)


    vf=verticalFlip(image,mask,ela)
    Image.fromarray(hf[0]).save(basic_path+'fake_aug/'+ 'vf_' + image_name)
    Image.fromarray(hf[1]).save(basic_path+'mask_aug/'+ 'vf_' + mask_name)
    Image.fromarray(hf[2]).save(basic_path+'ela_aug/'+ 'vf_' + ela_name)


    # tp=transpose(image,mask,ela)
    # Image.fromarray(vf[0]).save(basic_path+'fake_aug/'+ 'tp_' + image_name)
    # Image.fromarray(vf[1]).save(basic_path+'mask_aug/'+ 'tp_' + mask_name)
    # Image.fromarray(vf[2]).save(basic_path+'ela_aug/'+ 'tp_' + ela_name)


    # rr=randomRotate(image,mask,ela)
    # Image.fromarray(rr[0]).save(basic_path+'fake_aug/'+ 'rr_' + image_name)
    # Image.fromarray(rr[0]).save(basic_path+'fake_aug/'+ 'rr_' + image_name)
    # Image.fromarray(rr[1]).save(basic_path+'mask_aug/'+ 'rr_' + mask_name)
    # Image.fromarray(rr[2]).save(basic_path+'ela_aug/'+ 'rr_' + ela_name)


    od=opticalDistortion(image,mask,ela)
    Image.fromarray(od[0]).save(basic_path+'fake_aug/'+ 'od_' + image_name)
    Image.fromarray(od[1]).save(basic_path+'mask_aug/'+ 'od_' + mask_name)
    Image.fromarray(od[2]).save(basic_path+'ela_aug/'+ 'od_' + ela_name)


    ed=elasticDistortion(image,mask,ela)
    Image.fromarray(ed[0]).save(basic_path+'fake_aug/'+ 'ed_' + image_name)
    Image.fromarray(ed[1]).save(basic_path+'mask_aug/'+ 'ed_' + mask_name)
    Image.fromarray(ed[2]).save(basic_path+'ela_aug/'+ 'ed_' + ela_name)



    ch=channelShuffle(image,mask,ela)
    Image.fromarray(ch[0]).save(basic_path+'fake_aug/'+ 'ch_' + image_name)
    Image.fromarray(ch[1]).save(basic_path+'mask_aug/'+ 'ch_' + mask_name)
    Image.fromarray(ch[2]).save(basic_path+'ela_aug/'+ 'ch_' + ela_name)


    # hs=hueSaturationValue(image,mask,ela)
    # Image.fromarray(hs[0]).save(basic_path+'fake_aug/'+ 'hs_' + image_name)
    # Image.fromarray(hs[1]).save(basic_path+'mask_aug/'+ 'hs_' + mask_name)
    # Image.fromarray(hs[2]).save(basic_path+'ela_aug/'+ 'hs_' + ela_name)


    rg=rGBShift(image,mask,ela)
    Image.fromarray(rg[0]).save(basic_path+'fake_aug/'+ 'rg_' + image_name)
    Image.fromarray(rg[1]).save(basic_path+'mask_aug/'+ 'rg_' + mask_name)
    Image.fromarray(rg[2]).save(basic_path+'ela_aug/'+ 'rg_' + ela_name)


    gd=gridDistortion(image,mask,ela)
    Image.fromarray(gd[0]).save(basic_path+'fake_aug/'+ 'gd_' + image_name)
    Image.fromarray(gd[1]).save(basic_path+'mask_aug/'+ 'gd_' + mask_name)
    Image.fromarray(gd[2]).save(basic_path+'ela_aug/'+ 'gd_' + ela_name)
```

```
100%|██████████| 450/450 [1:23:53<00:00, 11.19s/it]
```

In [45]:

```
fake_mask_list_path3 = os.listdir('/content/dataset-dist/phase-01/training/mask_aug')
fake_ela_list_path3 = os.listdir('/content/dataset-dist/phase-01/training/ela_aug')
fake_image_list_path3 = os.listdir('/content/dataset-dist/phase-01/training/fake_aug')

fake_mask_list_path3.sort()
fake_ela_list_path3.sort()
fake_image_list_path3.sort()
```

In [46]:

```
print(len(fake_image_list_path3),len(fake_mask_list_path3),len(fake_ela_list_path3))
```

```
3600 3600 3600
```

In [47]:

```
for idx,v in enumerate(fake_image_list_path3):
  temp=v
  fake_image_list_path3[idx]='/content/dataset-dist/phase-01/training/fake_aug/'+temp

for idx,v in enumerate(fake_mask_list_path3):
  temp=v
  fake_mask_list_path3[idx]='/content/dataset-dist/phase-01/training/mask_aug/'+temp

for idx,v in enumerate(fake_ela_list_path3):
  temp=v
  fake_ela_list_path3[idx]='/content/dataset-dist/phase-01/training/ela_aug/'+temp
```

In [6]:

```
def reshape_data3(x):
  return np.array([resize(imread(file_name), (256, 256, 3)) for file_name in x])

def reshape_data4(x):
  return np.array([resize(imread(file_name), (256, 256, 1)) for file_name in x])
```

In [48]:

```
X_train1 = []
for filename in tqdm.tqdm(fake_image_list_path3[0:2600], position=0, leave=True):
  temp = resize(imread(filename),(256,256,3))
  X_train1.append(temp)
gc.collect()
X_train1 = np.array(X_train1)

X_train2 = []
for filename in tqdm.tqdm(fake_ela_list_path3[0:2600], position=0, leave=True):
  temp = resize(imread(filename),(256,256,3))
  X_train2.append(temp)
gc.collect()
X_train2 = np.array(X_train2)

Y_train = []
for filename in tqdm.tqdm(fake_mask_list_path3[0:2600], position=0, leave=True):
  temp = resize(imread(filename),(256,256,1))
  Y_train.append(temp)
gc.collect()
Y_train = np.array(Y_train)

X_val1 = []
for filename in tqdm.tqdm(fake_image_list_path3[2600:3600], position=0, leave=True):
  temp = resize(imread(filename),(256,256,3))
  X_val1.append(temp)
gc.collect()
X_val1 = np.array(X_val1)

X_val2 = []
for filename in tqdm.tqdm(fake_ela_list_path3[2600:3600], position=0, leave=True):
  temp = resize(imread(filename),(256,256,3))
  X_val2.append(temp)
gc.collect()
X_val2 = np.array(X_val2)

Y_val = []
for filename in tqdm.tqdm(fake_mask_list_path3[2600:3600], position=0, leave=True):
  temp = resize(imread(filename),(256,256,1))
  Y_val.append(temp)
gc.collect()
Y_val = np.array(Y_val)
```

```
100%|██████████| 2600/2600 [22:08<00:00,  1.96it/s]
100%|██████████| 2600/2600 [04:32<00:00,  9.55it/s]
100%|██████████| 2600/2600 [04:29<00:00,  9.66it/s]
100%|██████████| 1000/1000 [09:11<00:00,  1.81it/s]
100%|██████████| 1000/1000 [01:49<00:00,  9.09it/s]
100%|██████████| 1000/1000 [01:48<00:00,  9.25it/s]
```

In [25]:

```
save('/content/X_train1.npy',X_train1)
save('/content/X_train2.npy',X_train2)
save('/content/X_val1.npy',X_val1)
save('/content/X_val2.npy',X_val2)
save('/content/Y_train.npy',Y_train)
save('/content/Y_val.npy',Y_val)
```

In [26]:

```
from google.colab import drive
drive.mount('/content/gdrive')
```

```
Mounted at /content/gdrive
```

In [29]:

```
with open('/content/gdrive/My Drive/X_train1.npy', 'w') as f:
  f.write('/content/X_train1.npy')

with open('/content/gdrive/My Drive/X_train2.npy', 'w') as f:
  f.write('/content/X_train2.npy')

with open('/content/gdrive/My Drive/X_val1.npy', 'w') as f:
  f.write('/content/X_val1.npy')

with open('/content/gdrive/My Drive/X_val2.npy', 'w') as f:
  f.write('/content/X_val2.npy')

with open('/content/gdrive/My Drive/Y_train.npy', 'w') as f:
  f.write('/content/Y_train.npy')

with open('/content/gdrive/My Drive/Y_val.npy', 'w') as f:
  f.write('/content/Y_val.npy')
```

In [31]:

```
X_train1 = np.array(fake_image_list_path3[0:2000])
X_train2 = np.array(fake_ela_list_path3[0:2000])
Y_train = np.array(fake_mask_list_path3[0:2000])
X_val1 = np.array(fake_image_list_path3[2000:2400])
X_val2 = np.array(fake_ela_list_path3[2000:2400])
Y_val = np.array(fake_mask_list_path3 [2000:2400])
```

In [ ]:

```
print(X_train1.shape)
print(X_train2.shape)
print(Y_train.shape)

print(X_val1.shape)
print(X_val2.shape)
print(Y_val.shape)
```

```
(2500,)
(2500,)
(2500,)
(500,)
(500,)
(500,)
```

In [1]:

```
X_train1 = load('/content/X_train1.npy')
X_train2 = load('/content/X_train2.npy')
X_val1 = load('/content/X_val1.npy')
X_val2 = load('/content/X_val2.npy')
Y_train = load('/content/Y_train.npy')
Y_val = load('/content/Y_val.npy')
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-1-c04cbc90104b> in <module>()
----> 1 X_train1 = load('/content/X_train1.npy')
      2 X_train2 = load('/content/X_train2.npy')
      3 X_val1 = load('/content/X_val1.npy')
      4 X_val2 = load('/content/X_val2.npy')
      5 Y_train = load('/content/Y_train.npy')

NameError: name 'load' is not defined
```

Model Vgg16 with images and ELA filter with augmentation

In [49]:

```
#Training network with 2 inputs: one with fake image and second with filtered fake image
path_img = Unet(backbone_name='vgg16', encoder_weights='imagenet',classes=3, activation='sigmoid',i
nput_shape=(256,256,3))
path_img._name = 'path_1'
out1 = Conv2D(3,(1,1), activation='sigmoid')(path_img.output)

path_filter = Unet(backbone_name='vgg16', encoder_weights='imagenet',classes=3, activation='sigmoid
',input_shape=(256,256,3))
path_filter._name = 'path_2'
out2 = Conv2D(3,(1,1), activation='sigmoid')(path_filter.output)
```

```
Downloading data from
https://github.com/qubvel/classification_models/releases/download/0.0.1/resnet34_imagenet_1000_no_t
5
85524480/85521592 [==============================] - 3s 0us/step
```

In [50]:

```
for layer in path_img.layers:
    layer._name = layer.name + str("_img")
```

In [51]:

```
combined = concatenate([out1, out2])
final = Conv2D(1,(1,1),activation='sigmoid')(combined)
model2 = Model(inputs=[path_img.input,path_filter.input], outputs=[final])
```

In [56]:

```
optim = tf.keras.optimizers.Adam(0.001)
focal_loss = sm.losses.cce_dice_loss
log_dir = "logs6/fit/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard_callback = TensorBoard(log_dir=log_dir, histogram_freq=1)
filepath = 'model_checkpoints/model3_aug.hdf5'
checkpoint = tf.keras.callbacks.ModelCheckpoint(filepath,monitor='val_metric',save_best_only=True,
mode='max')
early_stop = tf.keras.callbacks.EarlyStopping(monitor='val_metric', patience=3)
reduce_lr = ReduceLROnPlateau(monitor = 'val_loss', factor = 0.2, patience = 1, verbose = 1, min_de
lta = 0.001)
model2.compile(optimizer=optim, loss="binary_crossentropy", metrics=[metric])
```

In [ ]:

```
model2.fit([X_train1,X_train2], [Y_train],validation_data=([X_val1,X_val2], [Y_val]),epochs=12, bat
ch_size=1,callbacks=[tensorboard_callback],verbose=1)
```

In [57]:

```
import gc
gc.collect()
```

Out[57]:

```
411
```

In [36]:

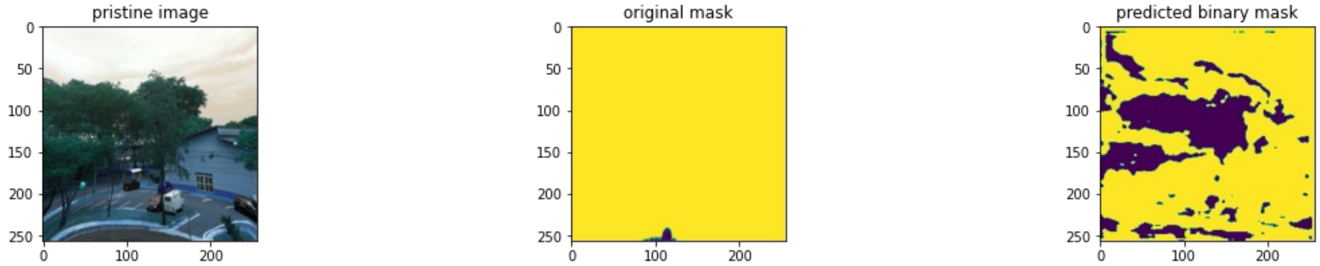
```
predicted = model2.predict([X_val1,X_val2])
```

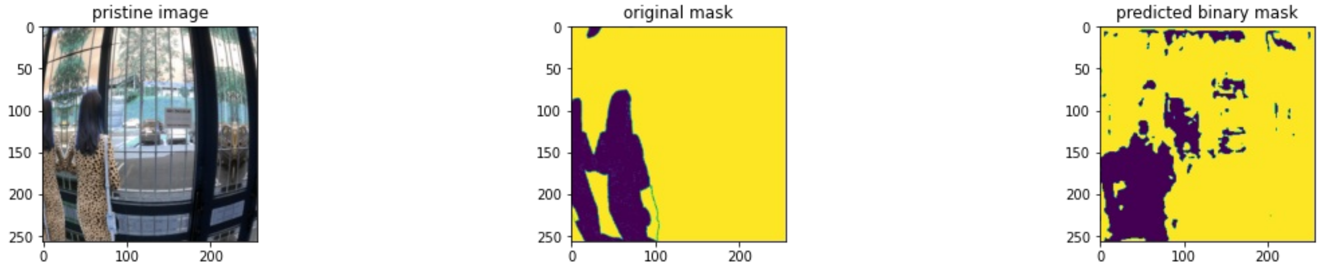
In [39]:

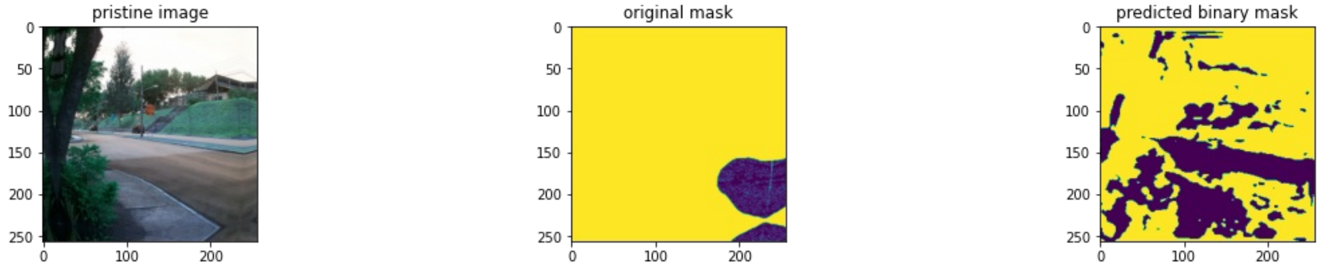
```
def plot_predicted_images(index):

    pred = np.squeeze(predicted[index])
    plt.imsave('pred_mask.png',pred)
    im_gray = cv2.imread('pred_mask.png', cv2.IMREAD_GRAYSCALE)
    (thresh, im_bw) = cv2.threshold(im_gray, 220, 255, cv2.THRESH_BINARY | cv2.THRESH_OTSU)
    fig = plt.figure(figsize=(20,10))
    ax1 = fig.add_subplot(331)
    ax2 = fig.add_subplot(332)
    ax3 = fig.add_subplot(333)

    ax1.set_title("pristine image")
    ax2.set_title("original mask")
    ax3.set_title("predicted binary mask")
    ax1.imshow(X_val1[index])
    ax2.imshow(np.squeeze(Y_val[index]))
    ax3.imshow(im_bw)
```

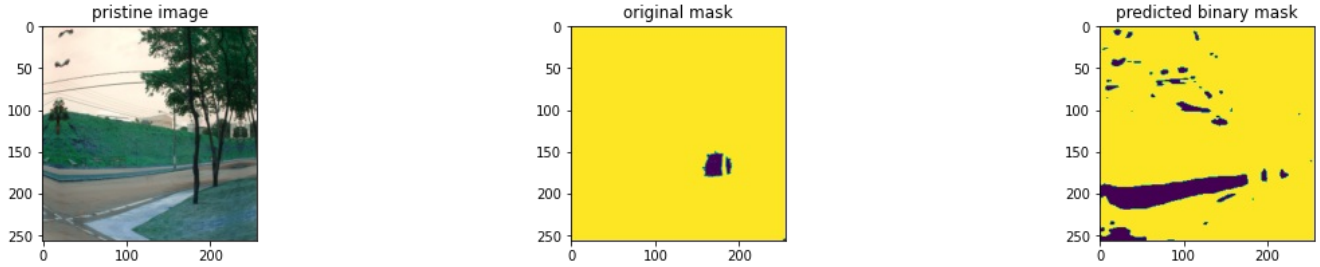
In [48]:

```
plot_predicted_images(31)
plot_predicted_images(178)
plot_predicted_images(89)
plot_predicted_images(66)
```









In [117]:

```
!rm -rf '/content/final_model_vgg16.hdf5'
```

Conclusion:

- Here we can see a little improvement in reduction of noise in predicted mask.

Model Resnet34 with images and ELA with augmentation

In [48]:

```
#Training network with 2 inputs: one with fake image and second with filtered fake image
path_img = Unet(backbone_name='resnet34', encoder_weights='imagenet',classes=3, activation='sigmoid
',input_shape=(256,256,3),decoder_use_batchnorm=True)
path_img._name = 'path_1'
out1 = Conv2D(3,(1,1), activation='sigmoid')(path_img.output)

path_filter = Unet(backbone_name='resnet34', encoder_weights='imagenet',classes=3, activation='sigm
oid',input_shape=(256,256,3),decoder_use_batchnorm=True)
path_filter._name = 'path_2'
out2 = Conv2D(3,(1,1), activation='sigmoid')(path_filter.output)
```

In [49]:

```
for layer in path_img.layers:
    layer._name = layer.name + str("_img")
```

In [50]:

```
combined = concatenate([out1, out2])
final = Conv2D(1,(1,1),activation='sigmoid')(combined)
model3 = Model(inputs=[path_img.input,path_filter.input], outputs=[final])
```

In [ ]:

```
optim = tf.keras.optimizers.Adam(0.01)
focal_loss = sm.losses.cce_dice_loss
log_dir = "logs6/fit/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard_callback = TensorBoard(log_dir=log_dir, histogram_freq=1)
filepath = 'model_checkpoints/model3_aug.hdf5'
checkpoint = tf.keras.callbacks.ModelCheckpoint(filepath,monitor='val_metric',save_best_only=True,
mode='max')
early_stop = tf.keras.callbacks.EarlyStopping(monitor='val_metric', patience=3)
reduce_lr = ReduceLROnPlateau(monitor = 'val_loss', factor = 0.1, patience = 2, verbose = 1, min_de
lta = 0.001)
model3.compile(optimizer=optim, loss="binary_crossentropy", metrics=[metric])
```

In [ ]:

```
model3.fit([X_train1,X_train2], [Y_train],validation_data=([X_val1,X_val2], [Y_val]),epochs=15, bat
ch_size=1,callbacks=[early_stop,reduce_lr,tensorboard_callback],verbose=1)
```

In [36]:

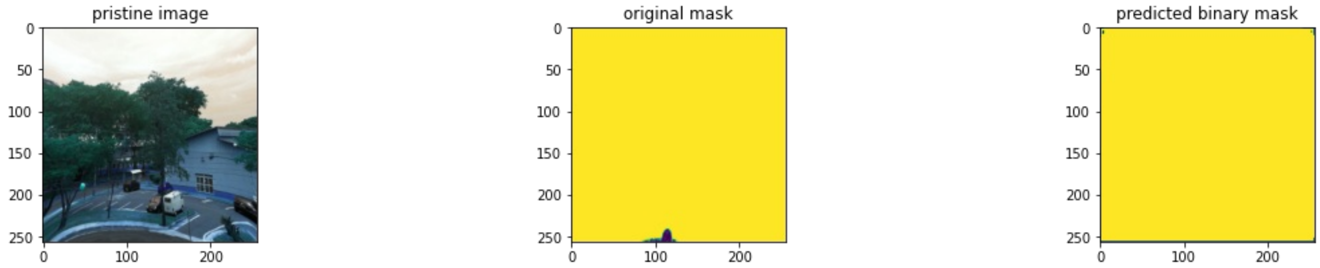
```
predicted2 = model3.predict([X_val1,X_val2])
```

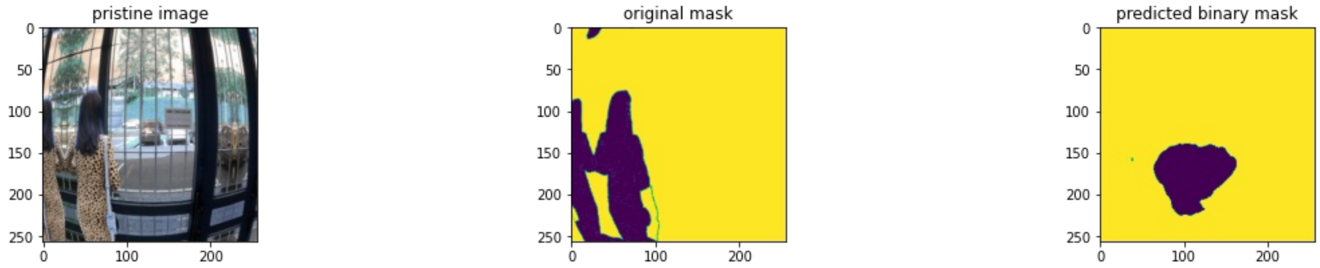
In [37]:

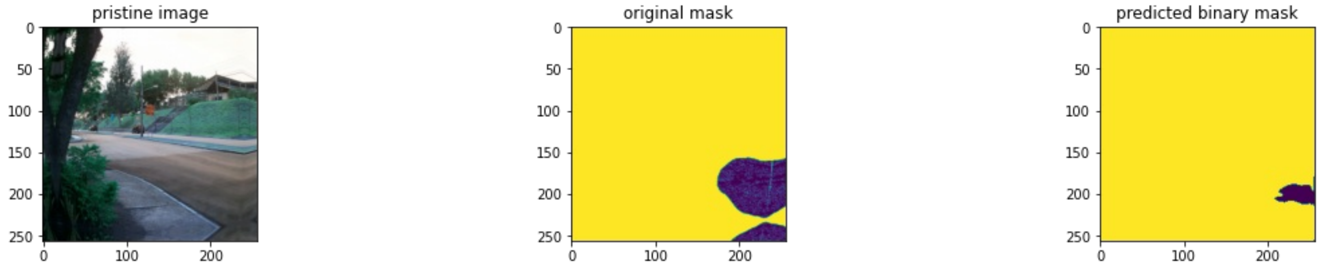
```
def plot_predicted_images(index):

    pred = np.squeeze(predicted2[index])
    plt.imsave('pred_mask.png',pred)
    im_gray = cv2.imread('pred_mask.png', cv2.IMREAD_GRAYSCALE)
    (thresh, im_bw) = cv2.threshold(im_gray, 220, 255, cv2.THRESH_BINARY | cv2.THRESH_OTSU)
    fig = plt.figure(figsize=(20,10))
    ax1 = fig.add_subplot(331)
    ax2 = fig.add_subplot(332)
    ax3 = fig.add_subplot(333)

    ax1.set_title("pristine image")
    ax2.set_title("original mask")
    ax3.set_title("predicted binary mask")
    ax1.imshow(X_val1[index])
    ax2.imshow(np.squeeze(Y_val[index]))
    ax3.imshow(im_bw)
```

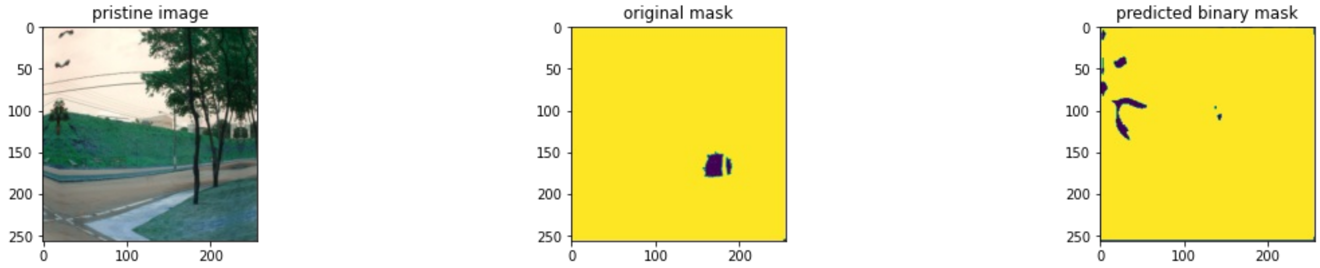
In [67]:

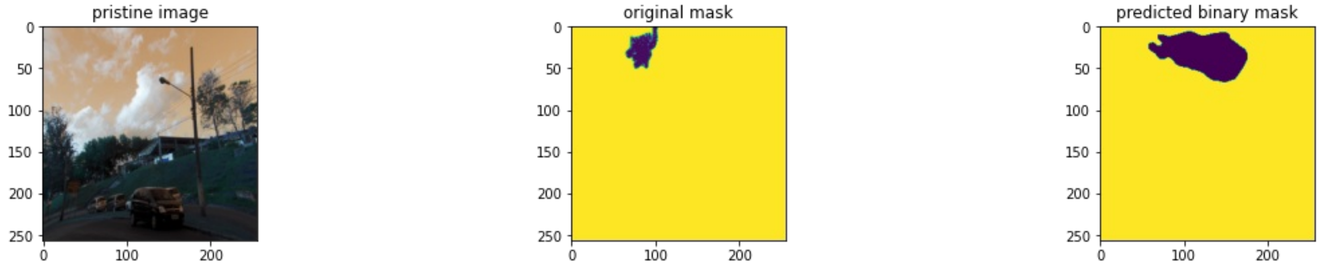
```
plot_predicted_images(31)
plot_predicted_images(178)
plot_predicted_images(89)
plot_predicted_images(66)
plot_predicted_images(198)
```











Conclusion:

- This model shows very good improvement over others as there is very less noise in predicted masks.

Summary of all the models in this notebook

Classification models

In [9]:

```
x = PrettyTable()
x.field_names = ["Model and architecture used", "Result"]
x.add_row(["Sequential model with custom architecture, trained on CASIA2", "Correct predictions = 
1736, Incorrect predictions = 148"])
x.add_row(["Sequential model with ResNet50 and imagenet weights and custom architecture, trained o
n CASIA2", "Correct predictions = 1827, Incorrect predictions = 75"])
print(x)
```

```
+------------------------------------------------------------------------------------------------+-
--------------------------------------------------------+
|                                  Model and architecture used                                   |
Result                              |
+------------------------------------------------------------------------------------------------+-
--------------------------------------------------------+
|                  Sequential model with custom architecture, trained on CASIA2                  |
orrect predictions = 1736, Incorrect predictions = 148 |
| Sequential model with ResNet50 and imagenet weights and custom architecture, trained on CASIA2 |
Correct predictions = 1827, Incorrect predictions = 75 |
+------------------------------------------------------------------------------------------------+-
--------------------------------------------------------+
```

Mask prediction models

In [8]:

```
x = PrettyTable()
x.field_names = ["Model and architecture used", "Result"]
x.add_row(["Resnet34 with imagenet weights + Concat output from two streams + SRM filter + With Au
gmentation", "A lot of noise in predicted mask is observed"])
x.add_row(["Vgg16 with imagenet weights + Concat output from two streams + ELA filter + Without Au
gmentation", "A lot of noise in predicted mask is observed"])
x.add_row(["Vgg16 with imagenet weights + Concat output from two streams + ELA filter + With Augme
ntation", "Reduction in noise but model still underfitting"])
x.add_row(["Resnet34 with imagenet weights + Concat output from two streams + ELA filter + With Au
gmentation", "Noise is totally reduced but region and mask detection can still be improved due to 
underfitting."])
print(x)
```

```
+--------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------+
|                                   Model and architecture used
Result                                                    |
+--------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------+
| Resnet34 with imagenet weights + Concat output from two streams + SRM filter + With Augmentation
|                            A lot of noise in predicted mask is observed
| Vgg16 with imagenet weights + Concat output from two streams + ELA filter + Without Augmentation
|                            A lot of noise in predicted mask is observed
|  Vgg16 with imagenet weights + Concat output from two streams + ELA filter + With Augmentation
|                          Reduction in noise but model still underfitting
| Resnet34 with imagenet weights + Concat output from two streams + ELA filter + With Augmentation
| Noise is totally reduced but region and mask detection can still be improved due to
underfitting. |
+--------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------+
```

References

- Research paper 1
- Reference 1
- Reference 2
- Reference 3
- Reference 4
- Reference 5
- Reference 6
- Reference 7
- Reference 8

- Reference 9
- Reference 10,of%20union%20in%20section%202)
- Reference 11
- Reference 12
- Reference 13
- Reference 14
- Reference 15
- Reference 16