main.py

import numpy as np import os

import matplotlib

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

from scipy import ndimage

from skimage.measure import regionprops from skimage import io

from skimage.filters import threshold\_otsu import tensorflow as tf

import pandas as pd import numpy as np from time import time import keras

from tensorflow.python.framework import ops import tensorflow.compat.v1 as tf tf.disable\_v2\_behavior()

# paths to images genuine\_image\_paths = "real" forged\_image\_paths ="forged"

def rgbgrey(img):

greyimg = np.zeros((img.shape[0], img.shape[1]))

for row in range(len(img)):

for col in range(len(img[row])): greyimg[row][col]=np.average(img[row][col])

return greyimg

def greybin(img):

# Converts grayscale to binary blur\_radius = 0.8

img = ndimage.gaussian\_filter(img, blur\_radius

img = ndimage.binary\_erosion(img).astype(img.dtype) thres = threshold\_otsu(img)

binimg = img > thres

binimg = np.logical\_not(binimg) return binimg

def preproc(path, img=None, display=True): if img is None:

img = mpimg.imread(path) if display:

plt.imshow(img) plt.show()

grey = rgbgrey(img) #rgb to grey if display:

plt.imshow(grey, cmap = matplotlib.cm.Greys\_r) plt.show()

binimg = greybin(grey) #grey to binary if display:

plt.imshow(binimg, cmap = matplotlib.cm.Greys\_r) plt.show()

r, c = np.where(binimg==1)

signimg = binimg[r.min(): r.max(), c.min(): c.max()] if display:

plt.imshow(signimg, cmap = matplotlib.cm.Greys\_r)

plt.show() return signimg

if img[row][col]==True:

b = np.array([row,col])

a= np.add(a,b) numOfWhites += 1

rowcols = np.array([img.shape[0], img.shape[1]]) centroid = a/numOfWhites

centroid = centroid/rowcols return centroid[0], centroid[1] def EccentricitySolidity(img:

r=regionprops(img.astype("int8")) return r[0].eccentricity, r[0].solidity