MINI PROJECT 2A REPORT
ON

"IMAGE FORGERY DETECTION"

Submitted In Partial Fulfilment of The Requirements
For the Degree of

BACHELOR OF COMPUTER SCIENCE AND ENGINEERING (IoT AND CYBER SECURITY INCLUDING BLOCKCHAIN TECHNOLOGY)

BY

Soman Sukale	**TE - IoT51**
Jisan Khan	**TE - IoT28**
Shardul Vanage	**TE - IoT63**
Abhishek Vishwakarma	**TE - IoT64**

UNDER THE GUIDANCE OF
DR. SHEEBA P. S.

LOKMANYA TILAK COLLEGE OF ENGINEERING

Department of Computer Science and Engineering
(IoT and Cyber Security Including Blockchain Technology)
AY 2022-2023

CERTIFICATE

This is to Certify that the Mini Project 2A Entitled " **IMAGE FORGERY DETECTION** " is a Bonafide work of Abhishek Vishwakarma(TE - 64) Shardul Vanage (TE - 63), Jisan Khan (TE- 28), Soman Sukale (TE - 51). It is Submitted to the University of Mumbai in Partial Fulfilment of the Requirement for the award.

External Examiner **Internal Examiner**

Dr.Sheeba P.S.

(H.O.D – CSE (IoT and CSBT)

ACKNOWLEDGEMENT

Here we Gladly Present This Mini Project Report on " **IMAGE FORGERY DETECTION** " as a part of the TE, 6th Semester in Computer Science and Engineering (IoT and Cyber Security Including Blockchain Technology). At this Time of Submitting this Report We use this opportunity to mention those People who were with us for this work. We extend our sincere and heartfelt thanks to our esteemed guide, **Dr. Sheeba P.S.** for providing us with the Right Guidance and Advice at the Crucial Junctures and for Showing us the right way.

Jisan Khan **Abhishek Vishwakarma**

Shardul Vanage **Soman Sukale**

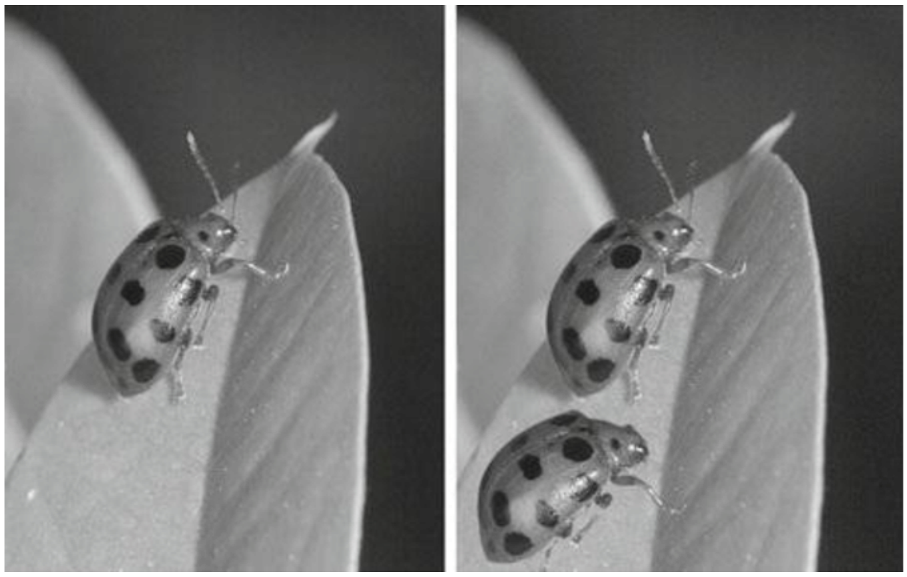
Contents

Chapter 1

Introduction

1.1 Need of the Project/ Problem Definition

Number of images over the internet is increasing at an exponential rate with the availability of the internet and camera to the large proportion of the population. It is effortless to make forgery images using advanced software like Adobe photoshop in which we can copy and paste one image onto an original image. Doctored images are a primary concern for social media companies. These doctored images are the primary source of fake news and are often used for mob incitement. With the advancement of photo and video editing software, it is becoming challenging for a normal human being to differentiate between a real image and a forged image we need forensic experts to do it. So we will try to build a network that will tell us which image is real and which is fake. If an image is fake I will also try to predict the region in the fake image which is tempered.



1.2 Research Objective

This project's goal is to investigate and analyse various techniques and algorithms for identifying fake images. With the development of digital photography, image modification has grown easier and more widely available, which has led to a number of societal problems, such as the media's use of doctored images and the altering of photos to improve body image.

Even if there are several ways to spot fake images, it can be difficult to decide which ones are the most useful and effective. According to their techniques of detection, the project attempts to classify algorithms into five different categories JPEG Compression Quantization, Edge Detection, Clone Detection, Resampling Detection, and Light Colour Anomaly Detection. These groups will be studied in order to assess the effectiveness of the algorithms and establish their dependability.

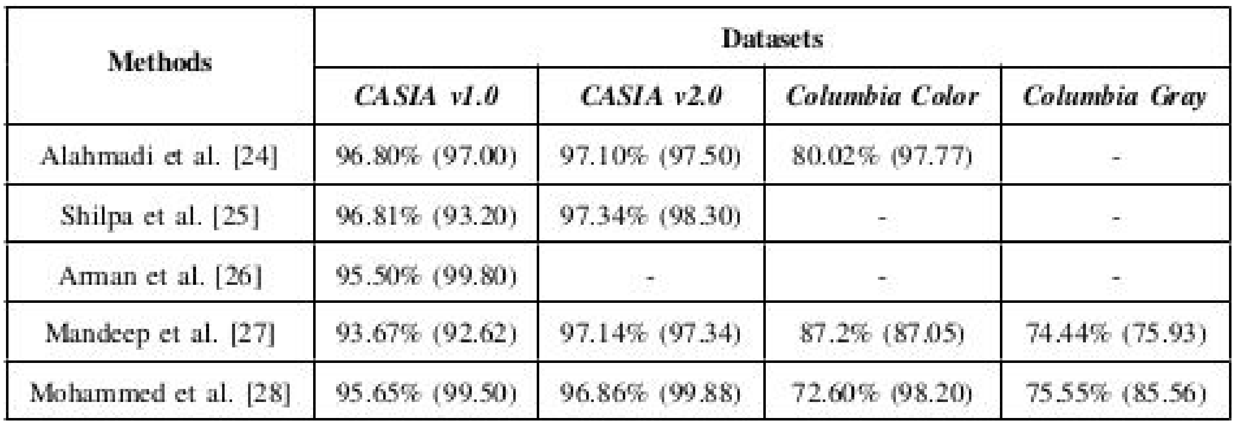
The project will implement and test the most reliable algorithm from within the identified groups, with a focus on detecting different types of image forgery. Extensive testing using a library of images will be performed to assess the success rate of the implemented algorithms, their false positive rates, and runtime. Variants of the same algorithm will also be tested to determine how changes in their parameters affect their performance on different types of images.

The results of this research will be valuable for improving the credibility of images used in the media and in daily life. The project aims to provide insight into detecting image forgery and to identify the most effective algorithms for different types of images. The research will not involve creating or testing new algorithms, but rather improving and comparing pre-existing algorithms. The project's outcome will depend on the type of image being analyzed, and while conclusions will be drawn, the effectiveness of an algorithm will vary depending on individual user needs.

The project's overall objective is to give a thorough review of various picture forgery detection techniques and their efficacy in various contexts, with the ultimate purpose of enhancing the dependability and trustworthiness of images portrayed in the media and in everyday life.

Chapter 2

Comparison with the existing Implementations



Methods	Datasets			
	CASIA v1.0	*CASIA v2.0*	*Columbia Color*	*Columbia Gray*
Alahmadi et al. [24]	96.80% (97.00)	97.10% (97.50)	80.02% (97.77)	-
Shilpa et al. [25]	96.81% (93.20)	97.34% (98.30)	-	-
Arman et al. [26]	95.50% (99.80)	-	-	-
Mandeep et al. [27]	93.67% (92.62)	97.14% (97.34)	87.2% (87.05)	74.44% (75.93)
Mohammed et al. [28]	95.65% (99.50)	96.86% (99.88)	72.60% (98.20)	75.55% (85.56)

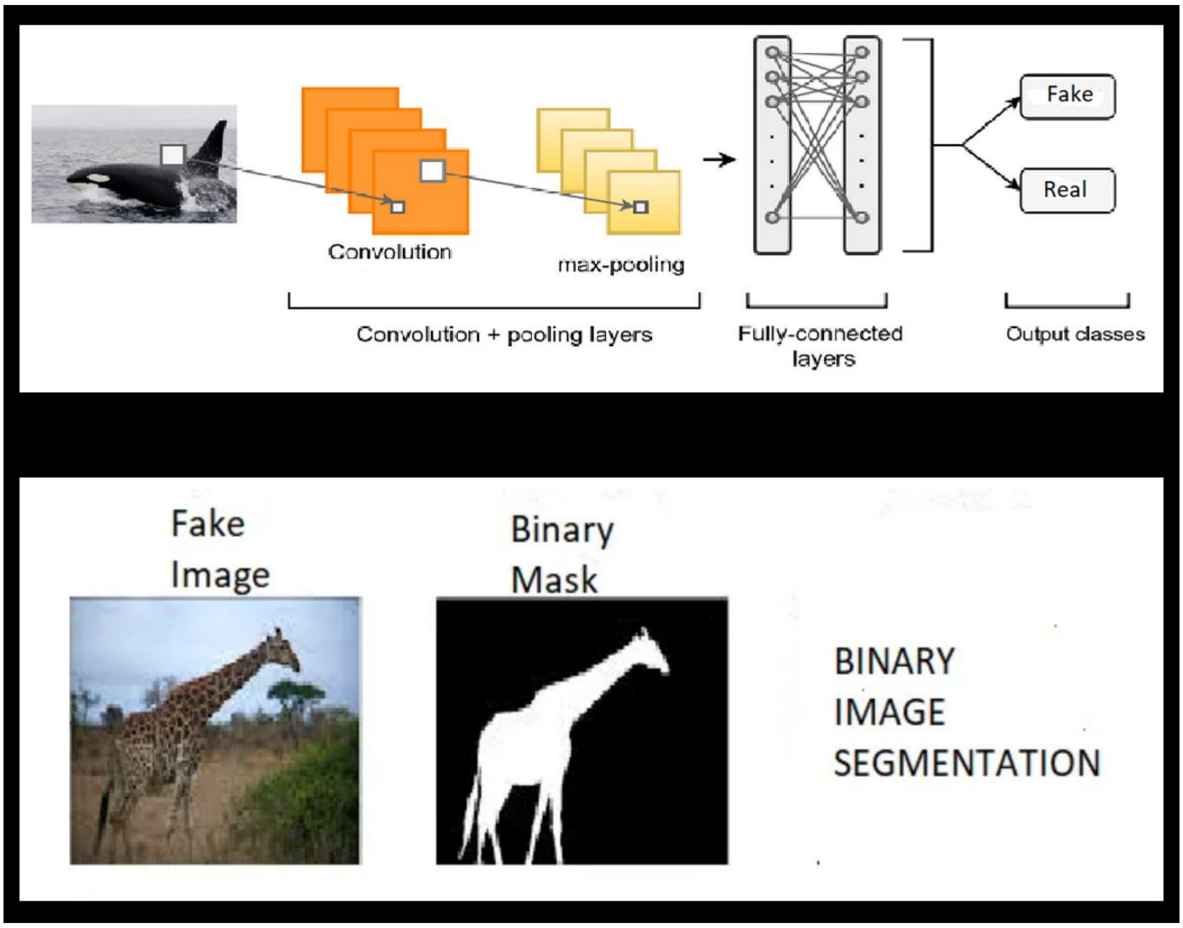
TABLE II: Reproduced Results

Chapter 3

Implementation

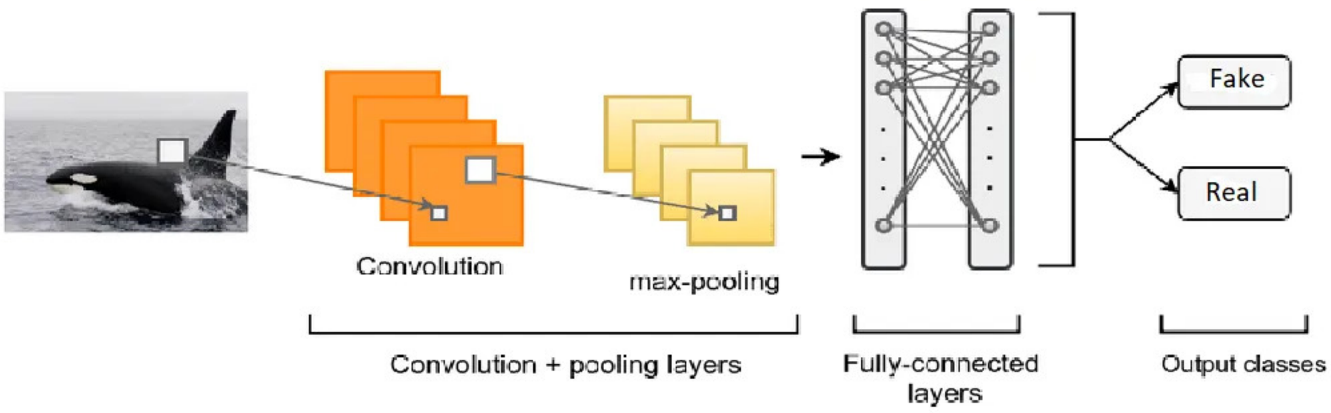
3.1 System Overview

WE are using CASIA 2.0 dataset which is readily available on Kaggle. There are two phases of this problem. In the first phase, we will try to detect whether an image is real or fake which is a classification problem. In the second phase, we will try to predict the region of the image which is tempered (Image2Image) which can be thought of Binary Image Segmentation problem. There is no latency requirement as such because we don't need instant results from our model but it should not take a lot of time to detect the tempered region and whether the image is real or fake.



3.2 phase 1

We will start with phase 1 of this problem in which we will try to predict whether an image is real or fake (Classification problem).



3.2.1 Basic Data Exploration

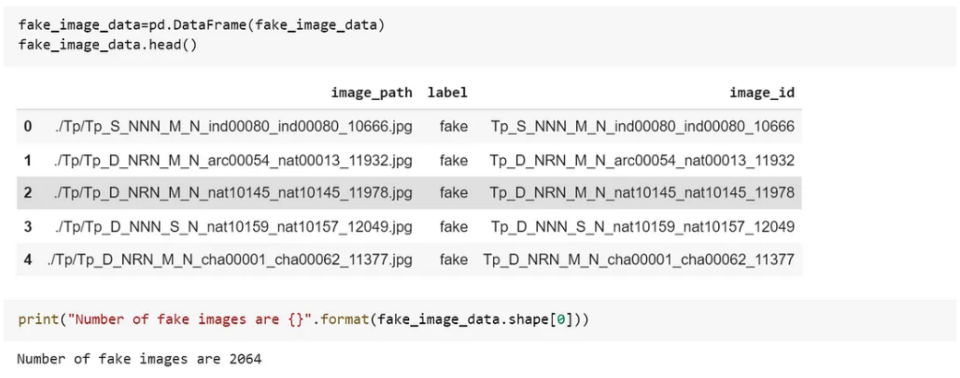
Let's explore !!!!!!!!!!!

1) Importing basic Python libraries for data exploration, manipulation, and model building.

2) In our dataset there are three folders

- Tp:- This folder contains fake images
- CASIA 2 Groundtruth:- This folder contains a fake image mask
- Au:- This folder contains real images

3) Let's analyze fake images first. Tp contains all the fake images plus it also contains some other files so will only read jpg files or png files.



```
fake_image_data=pd.DataFrame(fake_image_data)
fake_image_data.head()
```

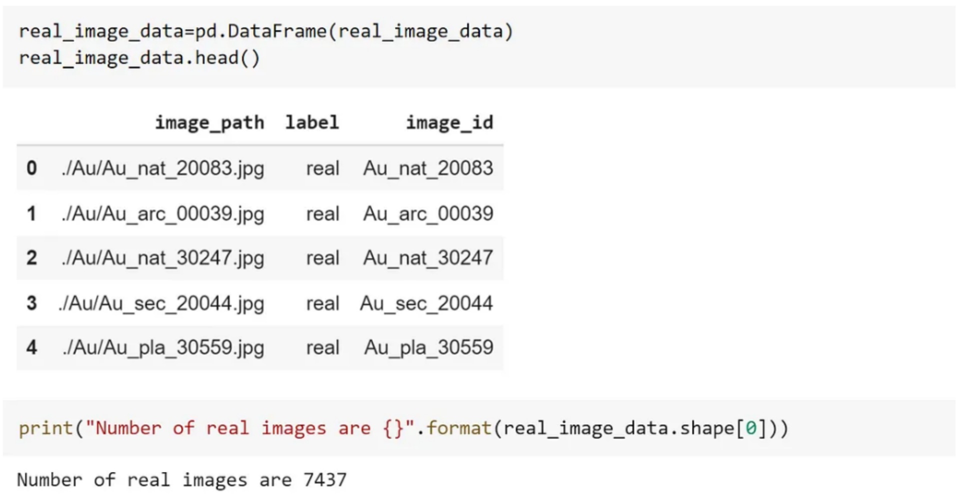
	image_path	label	image_id
0	./Tp/Tp_S_NNN_M_N_ind00080_ind00080_10666.jpg	fake	Tp_S_NNN_M_N_ind00080_ind00080_10666
1	./Tp/Tp_D_NRN_M_N_arc00054_nat00013_11932.jpg	fake	Tp_D_NRN_M_N_arc00054_nat00013_11932
2	./Tp/Tp_D_NRN_M_N_nat10145_nat10145_11978.jpg	fake	Tp_D_NRN_M_N_nat10145_nat10145_11978
3	./Tp/Tp_D_NNN_S_N_nat10159_nat10157_12049.jpg	fake	Tp_D_NNN_S_N_nat10159_nat10157_12049
4	./Tp/Tp_D_NRN_M_N_cha00001_cha00062_11377.jpg	fake	Tp_D_NRN_M_N_cha00001_cha00062_11377

```
print("Number of fake images are {}".format(fake_image_data.shape[0]))
```

```
Number of fake images are 2064
```

The total number of fake images comes out to be 2064. Each fake image has a mask image(stored in CASIA 2 Groundtruth folder)that shows us the region which is tempered but we don't need it in phase 1.

4) Let's analyze pristine images



```
real_image_data=pd.DataFrame(real_image_data)
real_image_data.head()
```

	image_path	label	image_id
0	./Au/Au_nat_20083.jpg	real	Au_nat_20083
1	./Au/Au_arc_00039.jpg	real	Au_arc_00039
2	./Au/Au_nat_30247.jpg	real	Au_nat_30247
3	./Au/Au_sec_20044.jpg	real	Au_sec_20044
4	./Au/Au_pla_30559.jpg	real	Au_pla_30559

```
print("Number of real images are {}".format(real_image_data.shape[0]))
```

```
Number of real images are 7437
```

The number of real images is 7437 which is significantly higher than the number of fake images which can lead to the problem of an imbalanced dataset. Thus we will reduce the number of pristine images this technique to handle imbalanced datasets is called undersampling. Now we have perfectly half of the total images belonging to the fake class and half belonging to the real class.

3.2.2 Data Preprocessing

So the idea is to do error level analysis on the input image and pass the ELA of the input image to a simple convolutional neural network which is trained to differentiate between a fake image or a pristine image.

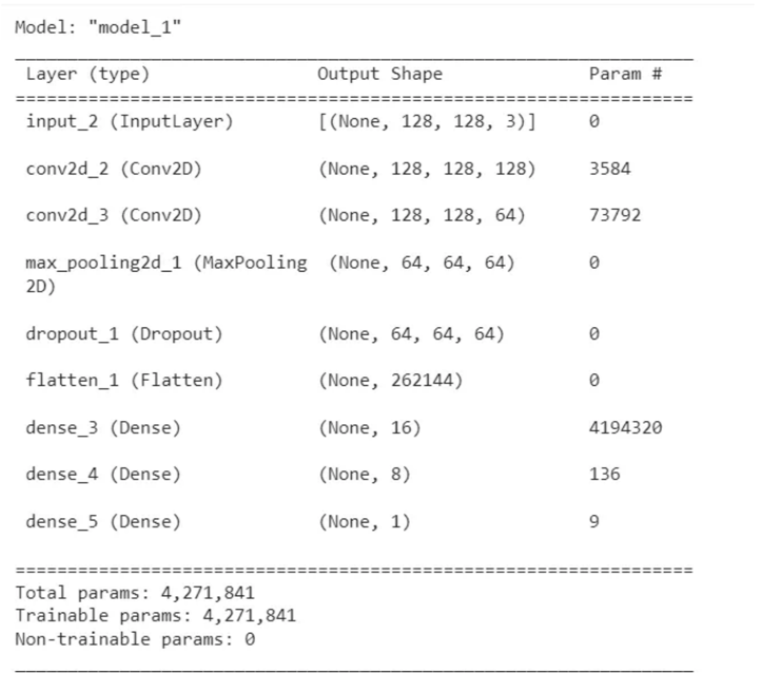
Error level analysis is one technique for knowing images that have been manipulated by storing images at a certain quality level and then calculating the difference from the compression level. When JPEG was first saved, then it will compress the image the first time, most editing software like adobe photoshop, gimp, and adobe lightroom support JPEG compressing operations. If the image is rescheduled using image editing software, it the compressed again.

Neal Krawetz invented the concept of Error Level Analysis for images when he noticed how errors spread when a JPEG image is saved. When cutting out a section of an image and pasting it into another image, the ELA for the pasted section often detects a more significant error, which means it is brighter than the rest of the image higher ELA level.

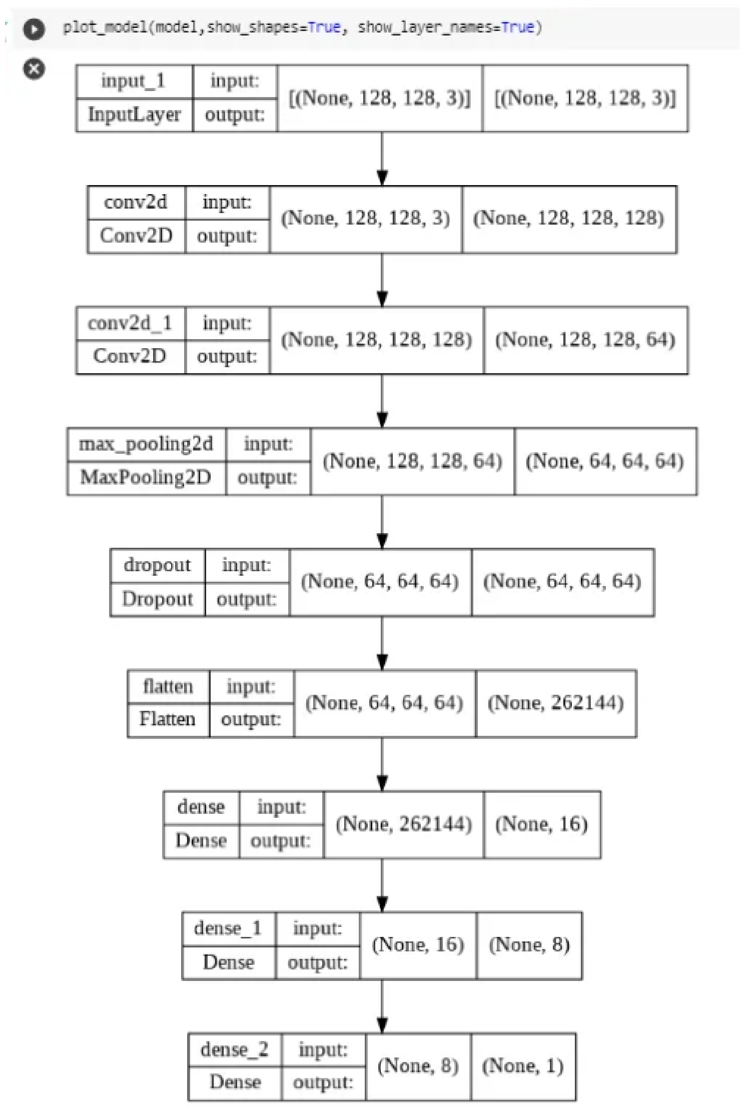
- Function for generating ELA image of an input image.
- We write a function that will generate an ELA image for an image plus it will normalize the pixel value of the ELA image(from 0–255 to 0–1). It will also resize the image to 128*128.
- We iterate over my dataset to generate X (input) and Y(output). X contains a 128*128 long vector and Y is the corresponding label for that X (0 for fake and 1 for real)
- Next we will separate our data (X and Y) into training and validation sets. We will not build a testing set as there are not enough data points available.
- We build a custom data generator so that we don't need to preprocess all the data points at a time and store them in our RAM. We can preprocess a batch of data points and pass it through our Convolution Neural Network.

3.2.3 Model Building And Training

1) We build a simple neural network with 2 convolution layers with a max pooling layer and a dropout layer followed by two dense layers and one output layer.



```
Model: "model_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 128, 128, 3)]     0

 conv2d_2 (Conv2D)           (None, 128, 128, 128)     3584

 conv2d_3 (Conv2D)           (None, 128, 128, 64)      73792

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

 dropout_1 (Dropout)         (None, 64, 64, 64)        0

 flatten_1 (Flatten)         (None, 262144)            0

 dense_3 (Dense)             (None, 16)                4194320

 dense_4 (Dense)             (None, 8)                 136

 dense_5 (Dense)             (None, 1)                 9

=================================================================
Total params: 4,271,841
Trainable params: 4,271,841
Non-trainable params: 0
_________________________________________________________________
```



```
plot_model(model,show_shapes=True, show_layer_names=True)
```

input_1 | InputLayer | input: [(None, 128, 128, 3)] | output: [(None, 128, 128, 3)]
conv2d | Conv2D | input: (None, 128, 128, 3) | output: (None, 128, 128, 128)
conv2d_1 | Conv2D | input: (None, 128, 128, 128) | output: (None, 128, 128, 64)
max_pooling2d | MaxPooling2D | input: (None, 128, 128, 64) | output: (None, 64, 64, 64)
dropout | Dropout | input: (None, 64, 64, 64) | output: (None, 64, 64, 64)
flatten | Flatten | input: (None, 64, 64, 64) | output: (None, 262144)
dense | Dense | input: (None, 262144) | output: (None, 16)
dense_1 | Dense | input: (None, 16) | output: (None, 8)
dense_2 | Dense | input: (None, 8) | output: (None, 1)

So there is a total 4,271,841 number of parameters.

2) We compile the model with Adam (learning rate=0.0001) as optimizer and binary cross entropy as loss function because there are only two classes.

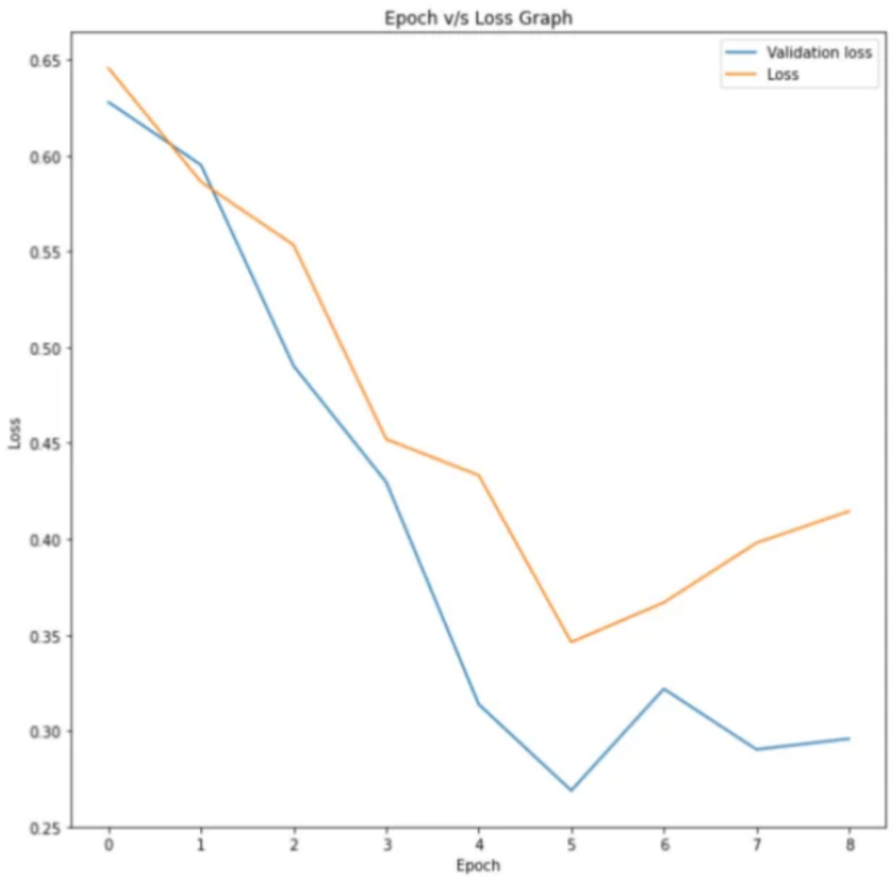
3) We define two callbacks (A callback is a set of functions to be applied at given stages of the training procedure. You define and use a callback when you want to automate some tasks after every training/epoch that helps you have control over the training process)

- LearningRateScheduler It will decrease the learning rate by 10 percent after every 5 epochs
- Early stopping:- It will monitor validation loss if it didn't decrease after 3 epochs it will stop the training model.

4) We put the model to train for 30 epochs but it stopped after 9 epochs because

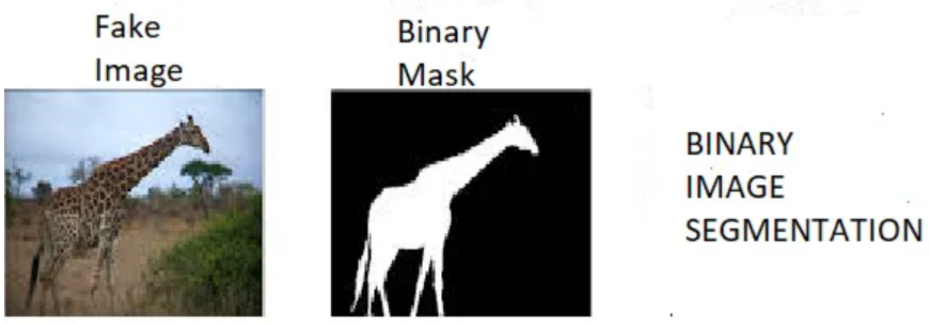
of our early stopping callback as our validation loss didn't decrease for 3 epochs.

5) Let's visualize how your loss decreases with an increase in epochs.



3.3 phase 2

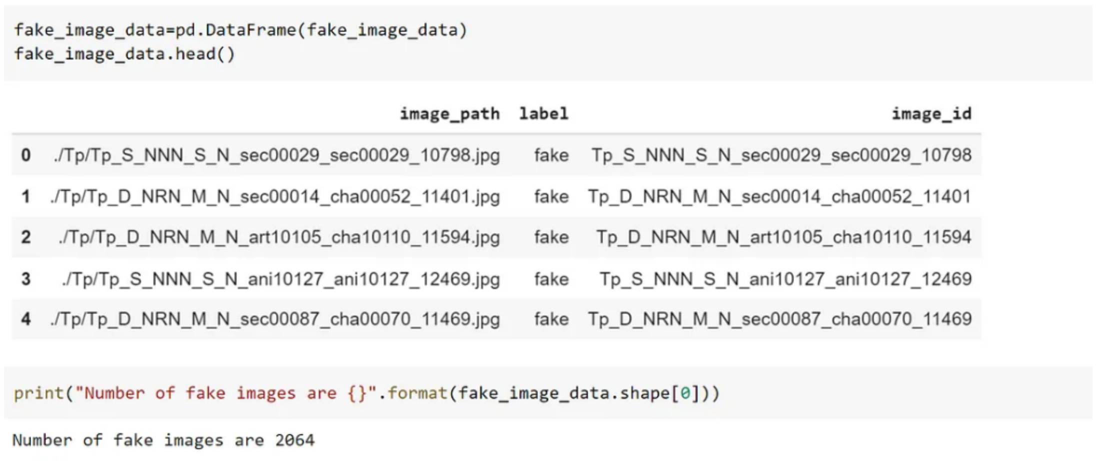
In phase 2 We will try to predict the region of the fake image where it is tempered. Essentially it is a binary image segmentation problem in which we will give a fake image as input and our output will be a binary image.



3.3.1 Data Exploration

1) Importing necessary libraries for data manipulation and building a deep learning model.

2) Here we will be using only fake images for which we have masks available that show us the region of the image which is tempered.



```
fake_image_data=pd.DataFrame(fake_image_data)
fake_image_data.head()
```

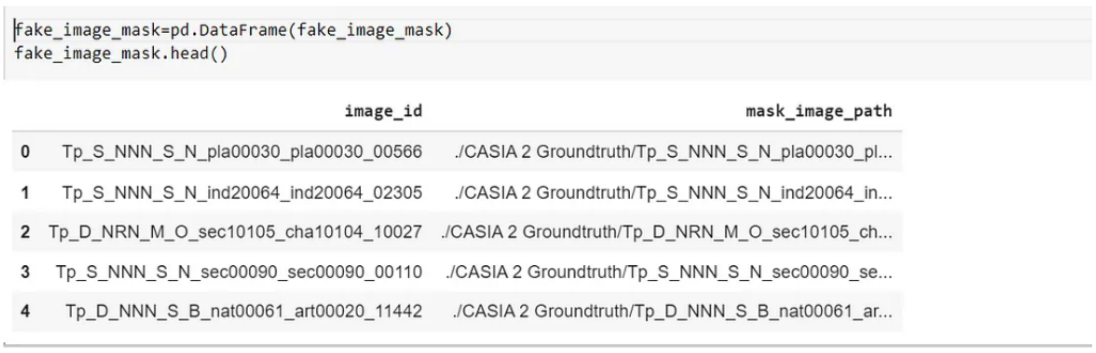
	image_path	label	image_id
0	./Tp/Tp_S_NNN_S_N_sec00029_sec00029_10798.jpg	fake	Tp_S_NNN_S_N_sec00029_sec00029_10798
1	./Tp/Tp_D_NRN_M_N_sec00014_cha00052_11401.jpg	fake	Tp_D_NRN_M_N_sec00014_cha00052_11401
2	./Tp/Tp_D_NRN_M_N_art10105_cha10110_11594.jpg	fake	Tp_D_NRN_M_N_art10105_cha10110_11594
3	./Tp/Tp_S_NNN_S_N_ani10127_ani10127_12469.jpg	fake	Tp_S_NNN_S_N_ani10127_ani10127_12469
4	./Tp/Tp_D_NRN_M_N_sec00087_cha00070_11469.jpg	fake	Tp_D_NRN_M_N_sec00087_cha00070_11469

```
print("Number of fake images are {}".format(fake_image_data.shape[0]))
```

```
Number of fake images are 2064
```

so essentially here we are reading each file at a time extracting the image path, creating a label(fake in this case), and image id.

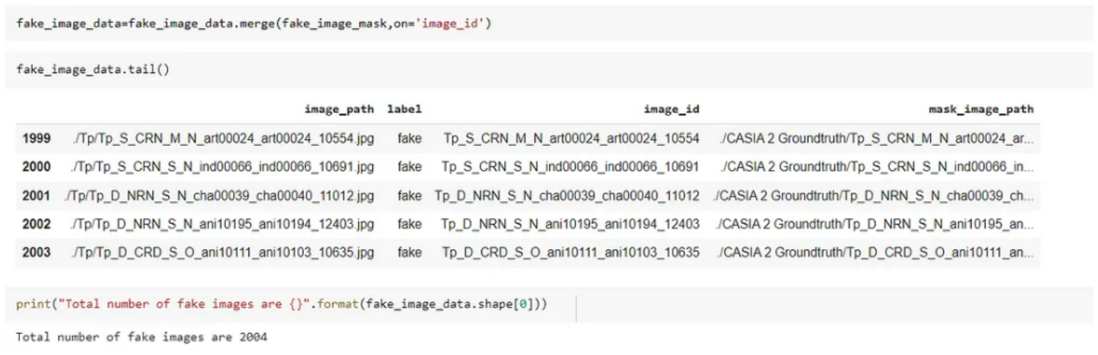
3) Every fake image has a mask image corresponding to it. We will iterate through CASIA 2 Groundtruth folder which contains all the mask images. We will save the image id and the path of the mask image.



```
fake_image_mask=pd.DataFrame(fake_image_mask)
fake_image_mask.head()
```

	image_id	mask_image_path
0	Tp_S_NNN_S_N_pla00030_pla00030_00566	./CASIA 2 Groundtruth/Tp_S_NNN_S_N_pla00030_pl...
1	Tp_S_NNN_S_N_ind20064_ind20064_02305	./CASIA 2 Groundtruth/Tp_S_NNN_S_N_ind20064_in...
2	Tp_D_NRN_M_O_sec10105_cha10104_10027	./CASIA 2 Groundtruth/Tp_D_NRN_M_O_sec10105_ch...
3	Tp_S_NNN_S_N_sec00090_sec00090_00110	./CASIA 2 Groundtruth/Tp_S_NNN_S_N_sec00090_se...
4	Tp_D_NNN_S_B_nat00061_art00020_11442	./CASIA 2 Groundtruth/Tp_D_NNN_S_B_nat00061_ar...

4) Now we have two datasets one contains a fake image path and image id second contains a mask image path and image id. So the idea is to merge both of them(on image id) into one which contains the image path and mask image path.



```
fake_image_data=fake_image_data.merge(fake_image_mask,on='image_id')
```

```
fake_image_data.tail()
```

	image_path	label	image_id	mask_image_path
1999	./Tp/Tp_S_CRN_M_N_art00024_art00024_10554.jpg	fake	Tp_S_CRN_M_N_art00024_art00024_10554	./CASIA 2 Groundtruth/Tp_S_CRN_M_N_art00024_ar...
2000	./Tp/Tp_S_CRN_S_N_ind00066_ind00066_10691.jpg	fake	Tp_S_CRN_S_N_ind00066_ind00066_10691	./CASIA 2 Groundtruth/Tp_S_CRN_S_N_ind00066_in...
2001	./Tp/Tp_D_NRN_S_N_cha00039_cha00040_11012.jpg	fake	Tp_D_NRN_S_N_cha00039_cha00040_11012	./CASIA 2 Groundtruth/Tp_D_NRN_S_N_cha00039_ch...
2002	./Tp/Tp_D_NRN_S_N_ani10195_ani10194_12403.jpg	fake	Tp_D_NRN_S_N_ani10195_ani10194_12403	./CASIA 2 Groundtruth/Tp_D_NRN_S_N_ani10195_an...
2003	./Tp/Tp_D_CRD_S_O_ani10111_ani10103_10635.jpg	fake	Tp_D_CRD_S_O_ani10111_ani10103_10635	./CASIA 2 Groundtruth/Tp_D_CRD_S_O_ani10111_an...

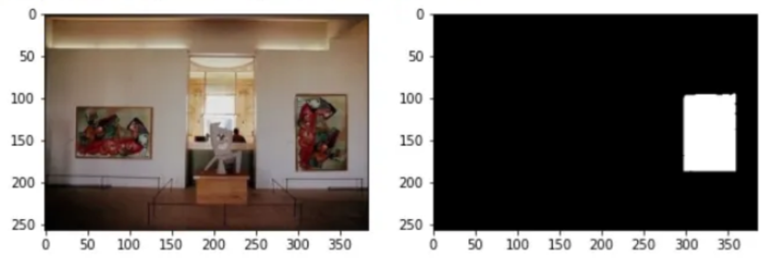
```
print("Total number of fake images are {}".format(fake_image_data.shape[0]))
```

```
Total number of fake images are 2004
```

5) Now we have a dataset that contains fake image paths and mask image paths corresponding to each fake image. The total number of fake images is 2004. Let's see one of the random images.

```
#Let's look at one example
ran_num=np.random.randint(0,fake_image_data.shape[0])
fig=plt.figure(figsize=(15,15))
ax1=fig.add_subplot(131)
ax2=fig.add_subplot(132)
temp_arr=Image.open(fake_image_data.iloc[ran_num]['image_path'])
temp_mask_arr=cv2.imread(fake_image_data.iloc[ran_num]['mask_image_path'],0)
ax1.imshow(temp_arr)
ax2.imshow(temp_mask_arr,cmap='gray')
```

```
<matplotlib.image.AxesImage at 0x7f0adafed690>
```

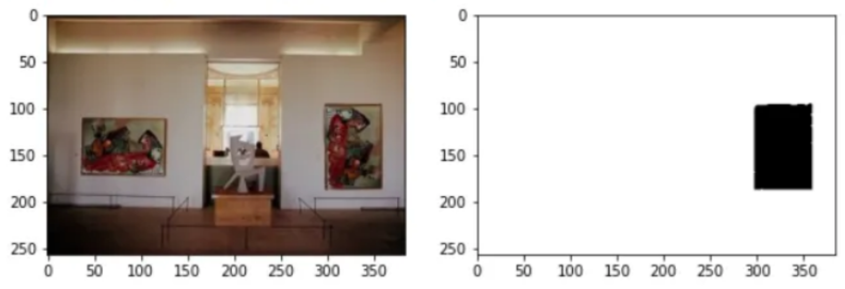


3.3.2 Data Preprocessing

1) I will make a small change to the mask image. I will show the tempered region as black and the non-tempered region as white. Initially, our mask image contains pixel values from 0 to 255 I will also scale it down to 0 or 1 so that our sigmoid function can work properly which we will use as an activation function in the output layer of our model.
Essentially this function converts all the pixel value that was 255 initially(white) into 0.0(black) and all the pixel values that are not equal to 255 into 1.0(white).
Now let's see the same random image and its mask.

```
fig=plt.figure(figsize=(15,15))
ax1=fig.add_subplot(131)
ax2=fig.add_subplot(132)
temp_mask_arr_mod=change_pixel_value(temp_mask_arr)
ax1.imshow(temp_arr)
ax2.imshow(temp_mask_arr_mod,cmap='gray')
```

```
<matplotlib.image.AxesImage at 0x7f0ad81b7690>
```



2) Splitting our dataset(fake image data) into training and validation datasets.

3) For this task we are going to use dual-stream UNET. The first stream input will be the RGB image second stream input will be a noise feature map which we will obtain by passing the image through SRM (Steganalysis Rich Model)filters.

3.3.3 Customer Data Loader

I will create custom data loaders so that we don't have to do all preprocessing and save the data in our RAM . we can do preprocessing on a batch of input and feed it to our neural network.In this class, we are preprocessing one image at a time and created batches out of it. Preprocessing contains the following things.Resize RGB image to 512*512*3 .Generate a Noise image by passing RGB image through SRM filters. Change pixel value of mask image(Black region as tempered region) and resize it into 512*512

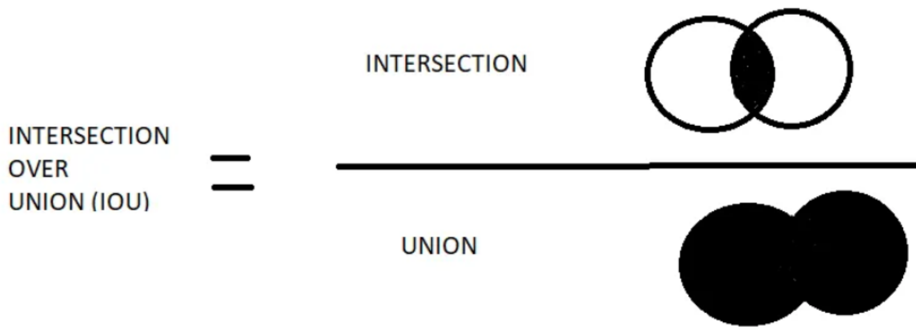
3.3.4 Model Building and Training

I will build a dual stream UNET. The first stream takes a normal RGB image as input and the second stream takes a processed image by passing the image through SRM filters. Output from both streams is then concatenated and passed through a CNN layer with a sigmoid activation function. The output will be a 512*512 binary image.

3.3.5 Model Compiling And Training

1) The loss function that I used for this model is binary cross entropy as the output is a binary image having pixel values of either 0 or 1.

2) The Metric that I used for this model is IOU(Intersection over Union) which is a very common metric used in Image Segmentation. Let's see what is IOU.



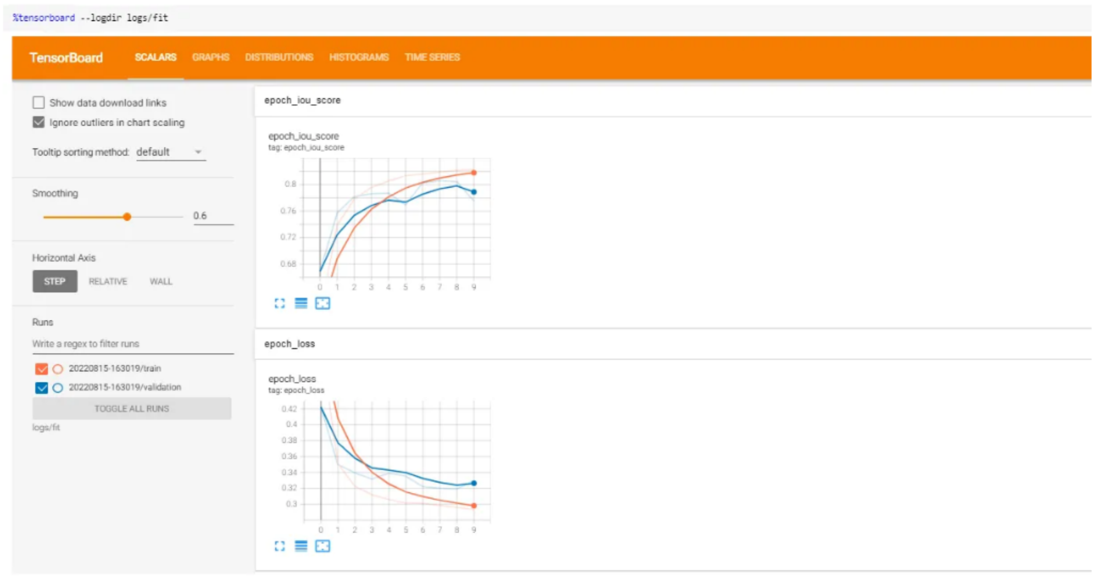
so we have a true mask image and a predicted mask image(threshold=0.5). we iterate through each pixel (Both true mask image and predicted mask image have the same number of pixels)and find how many pixels in both images have the same pixel value (Intersection). Union is how many pixel values are in total (count number of pixels in any of the images both have the same number of pixels). IOU range from [0, 1] .1 if both the predicted mask image and true mask image are exactly the same 0 when the predicted mask image and true mask image are totally different.

3) I will use two callbacks to train my model

LrScheduler:- It will decrease the learning rate by 10 percentage after every 5 epochs

Tensorboard:- TensorBoard is a tool for providing the measurements and visualizations needed during the machine learning workflow. It enables tracking experiment metrics like loss and accuracy and visualizing the model graph.

4) Lets' generate tensorboard.



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

☐ Show data download links
☑ Ignore outliers in chart scaling
Tooltip sorting method: default

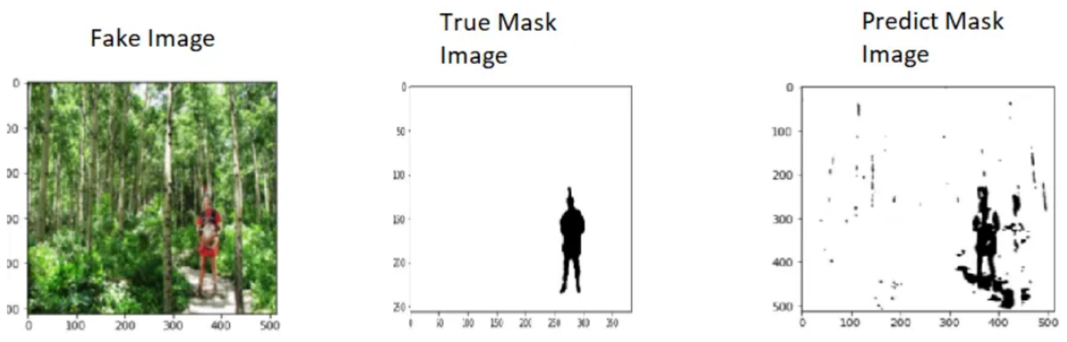
Smoothing 0.6

Horizontal Axis: [✓ STEP] RELATIVE WALL

Runs
Write a regex to filter runs
☑ 20220815-163019/train
☑ 20220815-163019/validation
logs/fit

5) Saving our model architecture and learned weights for the model to use in deployment.

6) I got a decent IOU score (around 0.80). Let's try to predict on a random image.



Chapter 4

Results

4.1 Interface

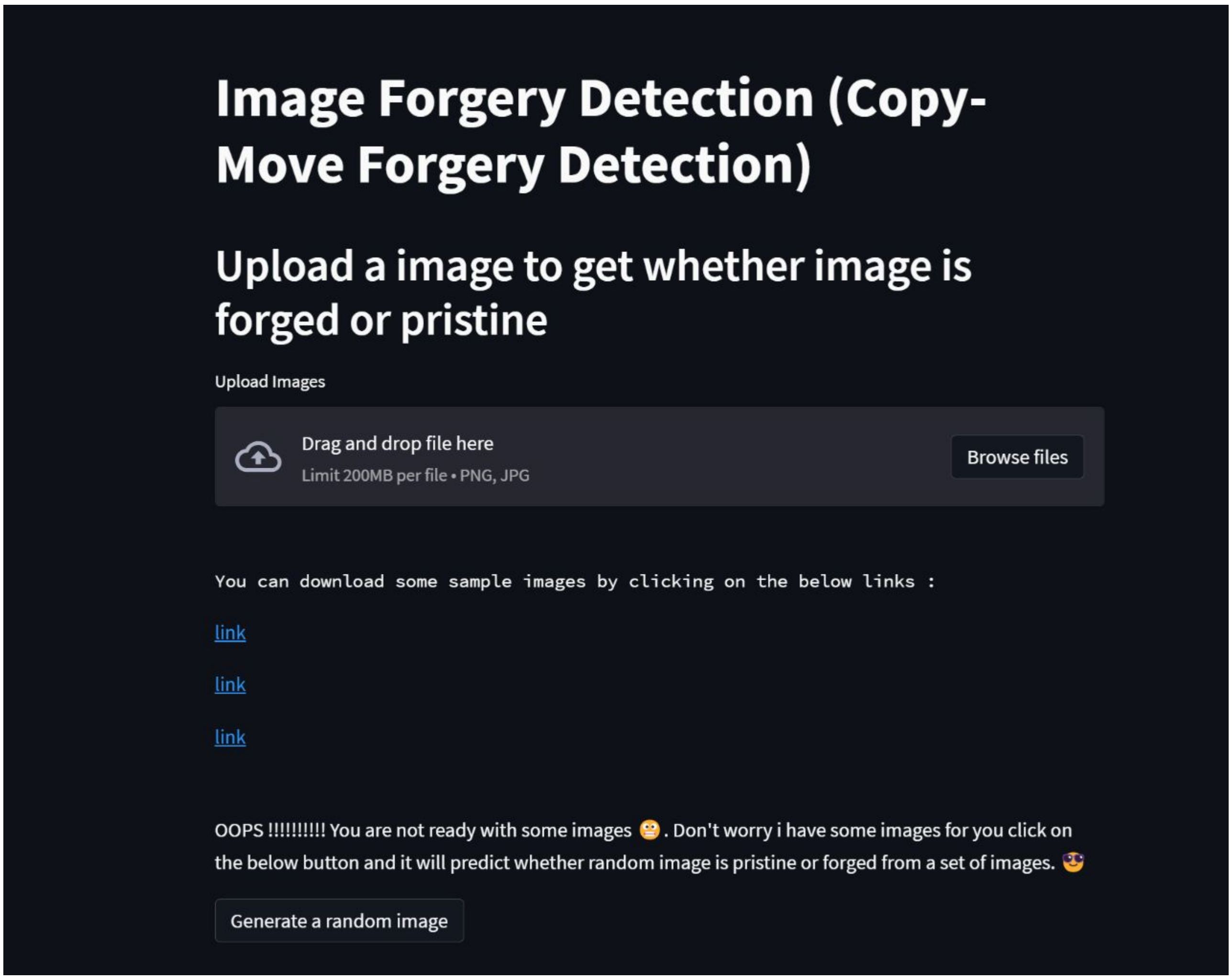


Image Forgery Detection (Copy-Move Forgery Detection)

Upload a image to get whether image is forged or pristine

Upload Images

Drag and drop file here
Limit 200MB per file • PNG, JPG

Browse files

```
You can download some sample images by clicking on the below links :
```

link

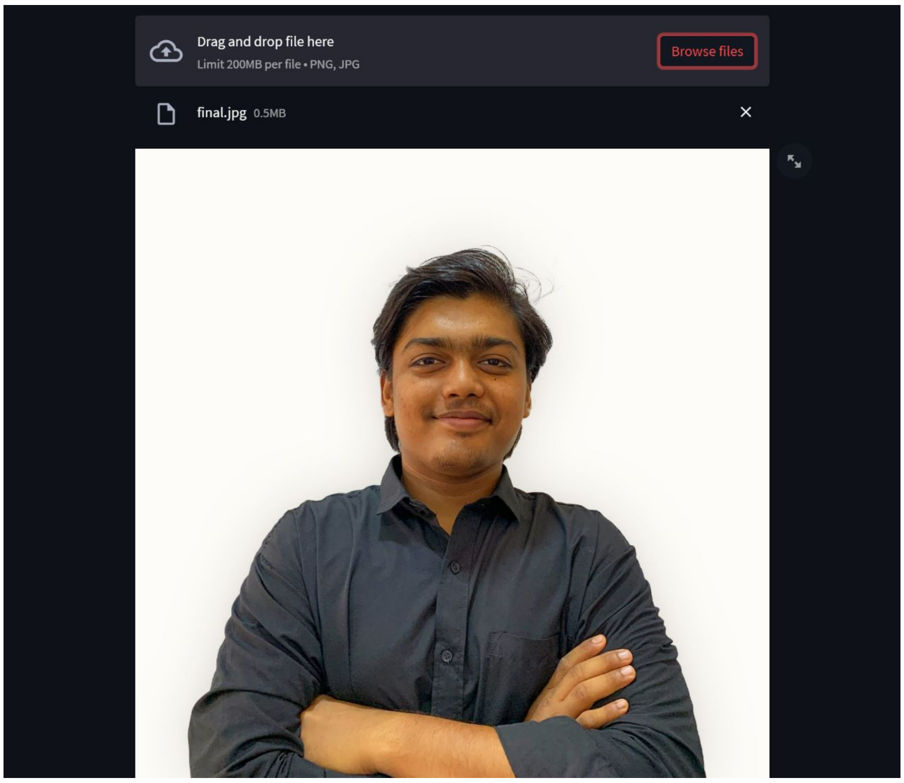
link

link

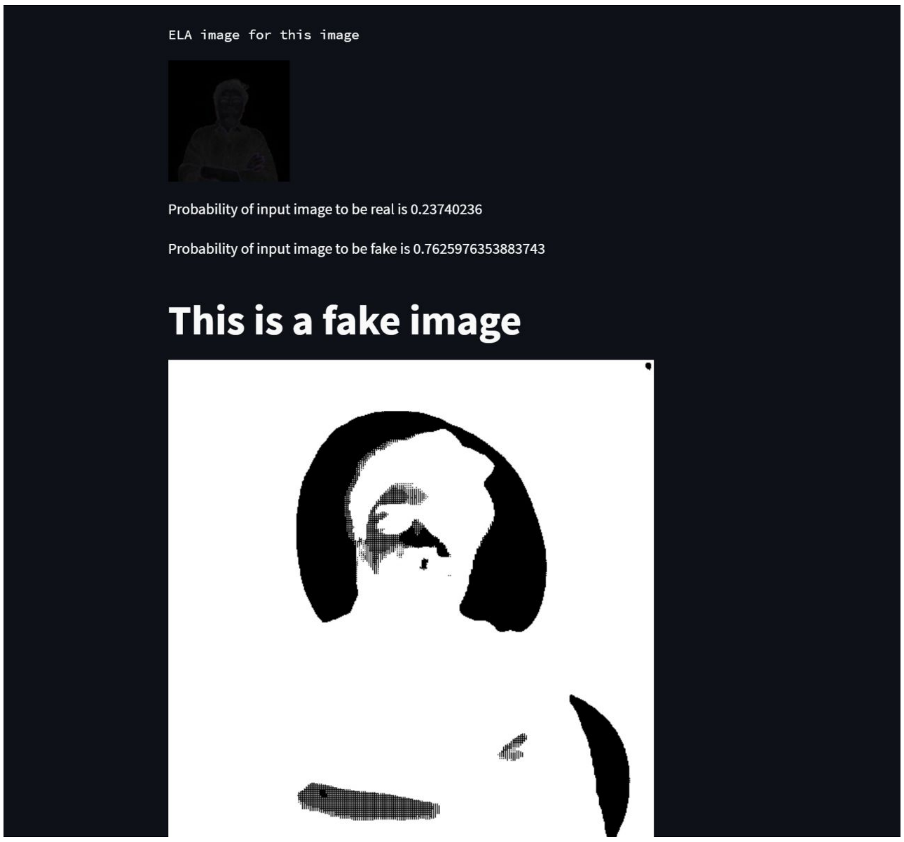
OOPS !!!!!!!!! You are not ready with some images 😬. Don't worry i have some images for you click on the below button and it will predict whether random image is pristine or forged from a set of images. 😎

Generate a random image

4.2 Input Image



4.3 Output Image



```
ELA image for this image
```

Probability of input image to be real is 0.23740236

Probability of input image to be fake is 0.7625976353883743

This is a fake image

Chapter 5

Conclusion and Future Scope

In conclusion, the detection of image manipulation is a crucial and challenging task in the field of digital image forensics. With the widespread availability of sophisticated image editing software, it has become increasingly easy to manipulate digital images in ways that are difficult to detect. As such, there is a pressing need for reliable and effective techniques for detecting image manipulation.

Research in this area has led to the development of a variety of techniques for detecting different types of image manipulation, including copy-move forgery, splicing, and retouching. These techniques often involve the extraction and analysis of features from the image, such as color, texture, and geometric properties. However, there is still much work to be done in this field, as existing techniques are not always effective in detecting all types of image manipulation.

Moving forward, it will be important for researchers to continue to develop and improve upon existing techniques for detecting image manipulation. This could involve exploring new methods for feature extraction and analysis, as well as incorporating advances in fields such as machine learning and computer vision. Ultimately, the goal should be to create a robust and reliable system for detecting image manipulation that can be used to ensure the authenticity and integrity of digital images.

There is a very limited dataset available for forgery detection that too has very few samples. However, we can tune this model more accurately using data augmentation and train for more epochs (obviously if you have no computational constraint)

References

[1] *Agus Gunawan[1], Holy Lovenia[2], Adrian Hartarto Pramudita[3] ”Detection og Image tampering With ELA and Deep learning” Informatics Engineering School of Electrical and Informatics Engineering, Bandung Institute of Technology.*

[2] *Nor Bakiah A. W.[1], Mohd. Yamani I. I. [2], Ainuddin Wahid A. W. [3], Rosli Salleh [4] ”An Evaluation of Error Level Analysis in Image Forensics” in IEEE 5th International Conference on System Engineering and Technology, Aug 2015. 10 - 11, UiTM, Shah Alam, Malaysia.*

[3] *J. Chen, X. Kang, Y. Liu and Z. J. Wang, ”Median Filtering Forensics Based on Convolutional Neural Networks,” in IEEE Signal Processing Letters, vol. 22, no. 11, pp. 1849-1853, Nov. 2015. doi: 10.1109/LSP.2015.2438008.*

[4] *Rao, Yuan and Jiangqun Ni. “A deep learning approach to detection of splicing and copy-move forgeries in images.” 2016 IEEE International Workshop on Information Forensics and Security (WIFS) (2016): 1-6.*

[5] *Fabio Carrara, Fabrizio Falchi, Roberto Caldelli, Giuseppe Amato, Roberta Fumarola, and Rudy Becarelli. 2017. Detecting adversarial example attacks to deep neural networks. In Proceedings of the 15th International Workshop on Content-Based Multimedia Indexing (CBMI ’17). ACM, New York, NY, USA, Article 38, 7 pages. DOI: 10.1145/3095713.3095753.*

[6] *Fabio Carrara, Fabrizio Falchi, Roberto Caldelli, Giuseppe Amato, Roberta Fumarola, and Rudy Becarelli. 2017. Detecting adversarial example attacks to deep neural networks. In Proceedings of the 15th International Workshop on Content-Based Multimedia Indexing (CBMI ’17). ACM, New York, NY, USA, Article 38, 7 pages. DOI: 10.1145/3095713.3095753.*

[7] *https://www.kaggle.com/datasets/sophatvathana/casia-dataset*

[8] *Capsule-Forensics: Using Capsule Networks to Detect Forged Images and Videos. Huy H. Nguyen, Junichi Yamagishi, Isao Echizen.*