Detection of Image Splicing using Image Forgery Techniques

Submitted By :-
Kartik Nema (IIT2018156)
Bhupendra (IIT2018163)
Prakhar Srivastava (IIT2018172)
Ashish Patel (IIT2018175)
Shubham Soni (IIT2018177)

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY, ALLAHABAD

(A UNIVERSITY ESTABLISHED UNDER SEC.3 OF UGC ACT, 1956 VIDE NOTIFICATION NO. F.9-4/99-U.3 DATED 04.08.2000 OF THE GOVT. OF INDIA)
A CENTRE OF EXCELLENCE IN INFORMATION TECHNOLOGY ESTABLISHED BY GOVT.OF INDIA

November, 2020

Index

List of Figures

Abstract :- *In this paper we discuss an approach to detect splicing forgery in an image. For this purpose we consider all windows of 64x64 in an image and try to classify those windows as fake or authentic.*

Keywords :- *Image forgery, splicing, edgemasks, Convolutional Neural Networks, pickling*

1. Introduction & Motivation

With advancement in digital image processing techniques and with a large dataset of images available, websites like flickr, facebook host millions of websites. Image forgery methods are also utilized in forensics to detect a fake image. However, with the advancement in image manipulation tools assessment of open media is becoming more difficult. Since blurs are mostly viewed as low pass filters,detection of blurs have been a challenging part. We lack the automation of the tools to assess an image authenticity. Image forgery could be done using many techniques such as image retouching forgery, spliced image forgery, copy move forgery.

We discuss image forgery using splicing and methods to detect it. These techniques help to determine the credibility of the image i.e. if it has been manipulated.

1.1 *What is Image forgery?*

Image forgery refers to the manipulation of the digital image so as to hide or conceal some information in the image. Image forgery thus refers to the alteration of some content in an image. Two common methods are used for image forgery, they are copy-move and splicing.

(i) Copy-move :- A part of picture is picture is copied and pasted over region of the same image in order to hide details of that region.

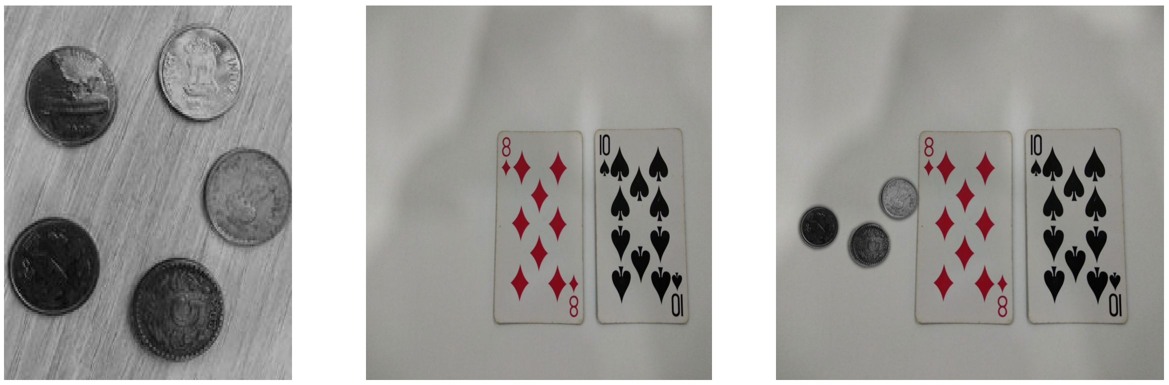
(i) Splicing :- It involves composition of 2 or more images as one.

In the modern world the manipulation of digital images has become increasingly easier due to advanced softwares and computers etc.

1.2 *Image Splicing*

In splicing some part of an image is copied and pasted into some region of another image, it results in loss of image information in the second image (over which the copied section is pasted). It is different from copy-move forgery because the latter involves only one image. The following images show image splicing, where the individual images of coins and playing cards are combined to obtain the third (composite) image.

Image splicing is harder to detect than other image forgery techniques like copy-move forgery.



(Splicing example, credits :- GeeksForGeeks, images taken from here)

1.3 *Problem Definition & Objectives*

In this project we aim to design a system, which takes in an image as input and outputs whether the image is spliced or not. This is a binary classifier i.e. it tells whether the image is spliced (yes or no).

1.4 *Scope*

Detection of image manipulation is extremely important, this is because fraud images can be used as legal evidence, shared on social media etc. Hence with the advent of technology modern image forgery detection techniques have been invented. Determining the authenticity of images i.e. detecting tempering without the presence of any extra information or any embedded watermarks is the challenge of modern times in the field of digital image processing. This project is a step in that direction which attempts to detect image splicing.

2. Literature Review

2.1 *Research papers considered*

1. **Forgery Detection of Spliced Images Using Machine Learning Classifiers and color Illumination** [1]:- In this method the illuminant color inconsistency and machine learning classifiers like Support Vector Machines and Least square Support Vector Machine and Perceptrons are used for forgery detection.
2. **A Review on Splicing Image Forgery Detection Techniques** [2]:- Different methods of forgery detection which are based on statistics, physics or or only algorithmic methods.
3. **Image Forgery Identification using Convolution Neural Network** [3]:- For image forgery detection, we project a technique that merges ZM-polar and block discrete cosine transform as motivated by the Works. Over copy-move forgery and splicing.
4. **Image Splicing Detection via Camera Response Function Analysis** [4] (base paper) :- Here classifiers like CNN which are used for image classification and recognition and SVM classifiers are used to detect image forgery.
5. **Investigation of image forgery based on multiscale retinex under illumination variations** [5]:- The method utilizes statistical region merging (SRM) segmentation method to segment an image. Then, a multiscale retinex (MSR) with color distribution is employed to improve the color of regions
6. **Digital image forgery detection using a deep learning approach** [6]:- Patches of images are given as input to classify them as real or fake. The developed approach involves the analysis of the image in a sliding window and the classification of each image fragment corresponding to the position of the window.
7. **Image Forgery Detection Based on statistical features of block DCT coefficients** [7]:- Here the image is converted from RGB to YCbCr color space and a feature matrix is formed by creating standard deviation and ones for each component and combined.This is passed to the classifiers to identify the image.

Forgery Detection of Spliced Images Using Machine Learning Classifiers and color Illumination

The concept of the illuminant color inconsistency and machine learning classifiers are used to predict the results. First of all the illuminant color of the images is estimated. For further processing all the faces in the image are extracted. Features like HOG and SASI are extracted from the image. The features mentioned above are used as inputs to the classifier (SVM, LS-SVM). Support vector machines (SVM) are supervised learning algorithms, which is useful for non-linear classifications.

Results :- The results obtained by using LS-SVM were much more accurate than those obtained by SVM

A Review on Splicing Image Forgery Detection Techniques

This paper deals with many methodologies :-

(i) First methods is detecting the sharp edges or the pixel which do not have any relation to the neighborhood pixels.Generally there is some patterns in the pixels of image and we try to recognize them and if we recognized that pattern then we can apply those pattern and find there is some unrelated stuff in the image.this does not work very good in all cases although it can work well in images which do not have very wide range of features.

(ii) Second technique is based on the Local Binary Pattern (LBP).In this method we divide the chrominance component of input image in overlapping blocks and then for each block we calculate the LBP and transform it into frequency domain .finally we calculate the standard deviations of each block and use them to as features .Now we feed these features to a support vector machines which gives us the final result.

(iii) Third technique is very specific and it only focuses on detecting the forgery in images which contains faces .The estimate of illuminant color is extracted independently from the different mini regions and for classification it uses the SVM.

Results:-Using second method gives us the 97% accuracy which is best and third technique does not perform well even in the images having faces.

Image Forgery Identification using Convolution Neural Network

Splicing forgery recognition:-

To distinguish the splicing image and a real image is a tough task as compared to copy-move With images feature, the basic idea of different splicing detection methods is to search the inconsistency area in the image.For controlling a produced image, the area is constantly compacted more than once and the area is blurred in sampling. At pixel level over various scales, noise level is tested. Any region which is not in noise level is a suspicious region in the image and noise level conflicting is a way of splicing an image.

Copy-move forgery detection:-

It uses a method based on Keypoint basebandBlock. This method is basically an uninformed search over the image to find key points and interest points in the image for image categorization.

Results :- Spliced image detection using the SVM has attained an accuracy of 85-86%. The second method showed lack of rotation and scaling in the image. Now the third one which uses CNN gives the accuracy result as with the formulae

$$\text{Accuracy} = (TP + TN)/(TP + TN + FN + FP) * 100$$

CNN accurately classifies with an accuracy of 99.03 whereas a maximum accuracy of 99.11 is attained for the spliced images.

Investigation of image forgery based on multiscale retinex under illumination variations

The method utilizes statistical region merging (SRM) segmentation method to segment an image. Then, a multiscale retinex (MSR) with color distribution is employed to improve the color of regions. A multiscale integral center symmetric census transform (MI-CSCT) is then applied. MI-CSCT is a normalization technique aimed at rendering regions in such a way that a post-processed image is free of illumination variations. The main contribution of this work is that the SRM algorithm is adapted to partition the image into small regions, where the least frequent regions can be identified.

Then, the proposed method perceives objects in an invariant manner despite variations in illumination. Finally, matching feature constraints are utilized to improve the detection rate of the proposed method via adaptive fuzzy C means (FCM) clustering technique. The main aim of the proposed method is to expose region duplication forgery against illumination changes.

Results :- The proposed method yields promising results with block sizes (of segmented region) of 4 x 4, 8 x 8. Results show that the proposed method can detect two types of forged regions: (i) uniform and (ii) non-uniform regions.

Digital image forgery detection using a deep learning approach

The proposed model takes picture patches as info and gets characterization results for a fix: unique or fraud. On the preparation stage we select patches from original picture areas and on the outskirts of implanted joining(splicing). The acquired outcomes show high grouping exactness (97.8% precision for adjusted model and 96.4% exactness for the zero-phase prepared) for a lot of pictures containing artificial distortions in examination with existing arrangements.To train the modal which have complex architectures we need lots of data which is the problem because generally any dataset does not have this amount of data.To solve this problem in this paper researchers used the 40X40 patches(fragments) to train the dataset.

Results:- Results were quite satisfactory ,researchers achieved 97.8% accuracy using the final model.

Image Forgery Detection Based on statistical features of block DCT coefficients

This paper proposes a model to distinguish a forge image and an authentic image.The image is passed through several stages to perform the operation.The steps involved are:

(i) Pre-Processing stage:

The test image is first converted to YCbCr (Y-luminance component, Cb and Cr- Chroma channels).Since Y contributes more in recognition of the luminance than Cb and Cr to the human eye, therefore mostly tampering traces are hidden in the chroma channels.

(ii) Modelling the tampering traces in DCT domain:

The image after being passed through the preprocessing stage each of the components of the image(Y,Cb,Cr) are stored in a matrix Q. The sub-image of size J*K is divided into non-overlapping 8*8 pixels size and a 2D-DCT operation is performed on each block.Further from each block the DC coefficients are removed and rest of the 63 AC coefficients are put into a column vector Ai . Each of these Ai are then combined to form a matrix M.Then row wise standard deviation and a vector S is formed, row wise number of one's are computed and vector O is formed.Next, the image is cropped by removing the first 4 rows and columns. Create standard deviation Sc and ones vector Oc for the cropped sub-image.These vectors are concatenated to form a feature vector.Feature vectors for all the components Y,Cb,Cr are formed and concatenated.This final vector is passed to the next stage Image classifier and classified as authentic or forged image.

(iii) Image classification: Forgery image detection is a two-class(authentic vs. forged) problem. As an image classifier we have used a support vector machine(SVM) , since it has shown promising performance results in many applications.

Results:- After obtaining the results from the experiments it is seen that the approach performed well on the CASIA dataset for both kinds of forgery.

2.2 *Base paper description*

To detect image forgery the image is first spliced where the contents of the image is extracted and copy it into a new image. This is done so that there is a harsh boundary between the image and the background.This will detect image forgery which involve artificially blurred images.Further classifiers like SVM and CNN are used to recognize the image and detect image forgery.

(i) CNN: A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other.

To detect image splicing IGH or Intensity Gradient Histogram and then it is passed to CNN classifier. Intensity gradient histogram is a plot of the intensity vs number of pixels that have that intensity value. Image is either blurred or sharpened for detection of image forgery. Possible outcomes are:

(ii) Authentic Blurred edges: By blurring an authentic image and plotting the Intensity Gradient Histogram we find that the plot is asymmetric in nature.

(iii) Authentic Sharp edges: Here if we sharpen the image and plot the IGH we observe that the plot comes symmetric.

(iv) Forged Blurred edges: The image is blurred and IGH is plotted.We note that the plot here also comes out to be symmetric

(v) Forged Sharp edges: There are only two intensities a and b, both having the same(large) gradient. The IGH of a forged sharp edge will only have all pixels fall in only two bins.

Now to automate this process classifiers like CNN or SVMs are used to identify whether the plot is asymmetric or symmetric and to draw the edge of the forged image.A simple is a SVM is a natural classifier and if the CNN classifier is applied there is lack of training data limits its performance.

The results are formulated by using the SVM and CNN classifier. The accuracy for the IGH with CNN classifier comes out to be highest.

3. Methodology

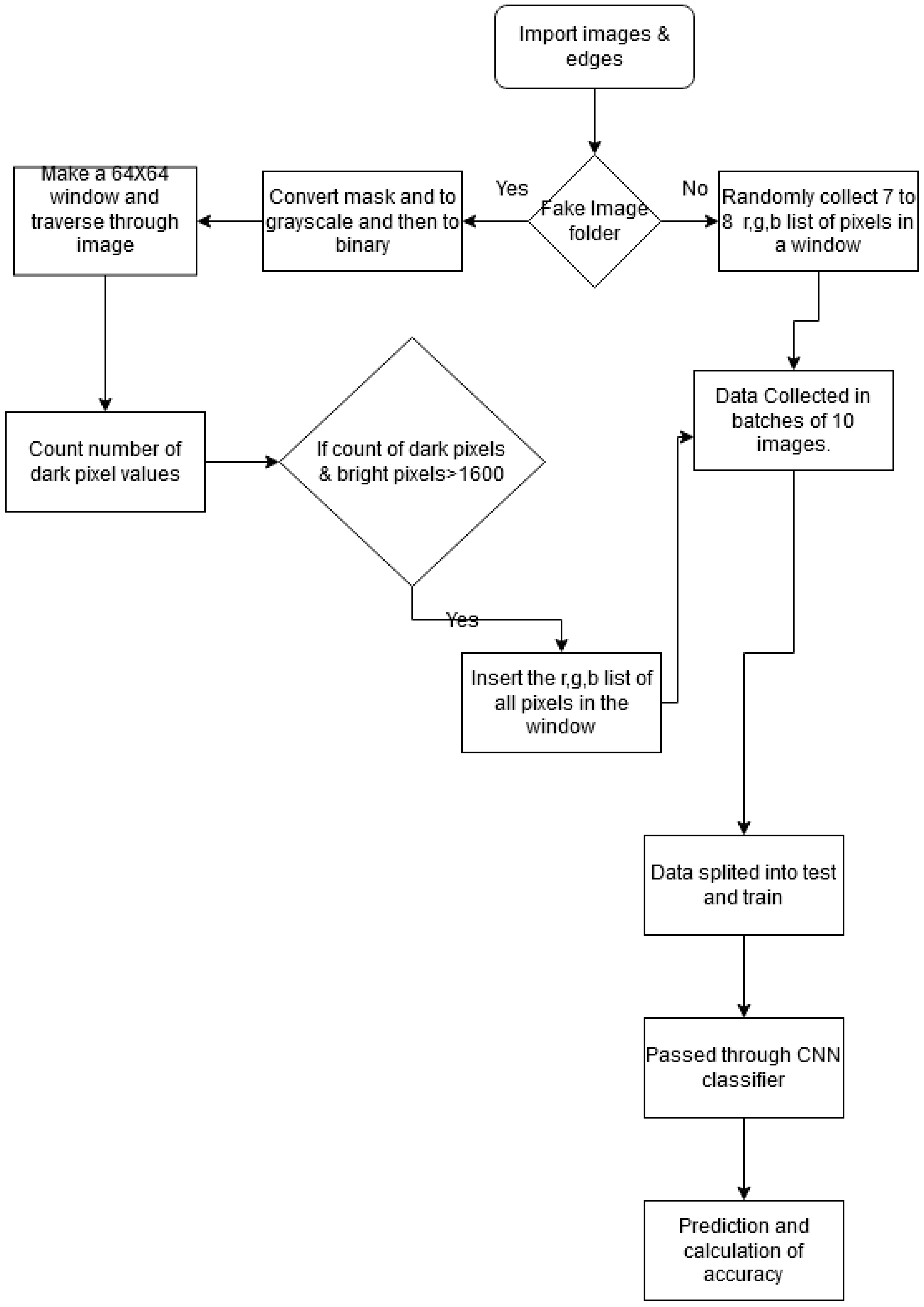
3.1 *Feature Extraction*:

The CUISDE Dataset used here is not fit for training a model. We are given that for every fake image there is a mask present.This mask is used to sample the fake image along the boundary of the spliced image to ensure that there is a contribution of around 25% from both the non-spliced and the spliced region.To make the coundaris more distinct the image is converted to grayscale and then it is further converted to binary image using Otsu's thresholding implemented in OpenCV module.Further sampling is done by creating a window of 64X64 and moving it through the fake image and counting the number of zero valued pixels(dark pixels).These boundaries are now to be learned by the classifier. Since the data was too large therefore different batches are created .Each batch consists of the data frame for 10 images. So, 16 batches are created from the CUISDE Dataset.

3.2 *Classification:*

The data collected in the form of batches from the Feature Extraction stage is now needed to be classified using a CNN classifier. First the data is splitted in Testing and Training data in the ratio of 1:4 respectively.This data is trained on the training dataset. Then to predict the labels in the test dataset it is passed through predict function.

3.3 *Flowchart:*


Import images & edges
Fake Image folder
Yes
No
Convert mask and to grayscale and then to binary
Make a 64X64 window and traverse through image
Randomly collect 7 to 8 r,g,b list of pixels in a window
Count number of dark pixel values
If count of dark pixels & bright pixels>1600
Yes
Insert the r,g,b list of all pixels in the window
Data Collected in batches of 10 images.
Data splited into test and train
Passed through CNN classifier
Prediction and calculation of accuracy

(proposed methodology, flowchart)

4. Implementation

4.1 *Code Description:*

4.1.1 *Preprocessing.py:*

Step 1: The paths for the authentic and forged images are created.

Step 2: All the fake images and their masks are passed to the extractWindowFake function which makes a window of 64X64 and counts the number of dark valued pixels in the mask image.If it is greater than a certain value then it is inserted into the 'data' section.

Step 3: All the authentic images are passed to the extractWindow function again in which a window is created and is traversed through the image. But here the R,G,B components of the image(in a window) are stored and returned.

Step 4: The data generated by these two functions are stored in a pickle file.Since the data was too large therefore the preprocessing stage code was implemented batch wise.

Step 5: After generating the pickle files for all the batches they were stored in Google Drive.

4.1.2 *Classify.ipynb:*

Step 1: The folder which contains all the pickle files is connected to the runtime of the Colab.

Step 2: The dataset is loaded into the dictionary dic from all the pickle files.The data and tag contains the dic['data'] and dic['tag'] respectively.

Step 3: The dataset is then splitted into training and testing data in the ratio of 4:1.

Step 4: The CNN classifier is further trained on the training data in 50 iterations and 32 batch size.

Step 5: For prediction x_test is passed to the predict function which will return an array.The values are rounded off.If it comes out to be 0 then the image is authentic else forged.

Step 6: To measure the accuracy predicted array is compared to the y_test and counting the number of correctly predicted values.

We observe that the accuracy by this method comes out to be 68.43%

4.1.3 *sample_output.py*

To enable easy access we created a graphical user interface.

Step 1: All the pickle files which were created are downloaded and stored in the folder.

Step 2: All the data in the pickle files are loaded into the dataset.Then our model is trained using the random forest classifier.

Step 3: The image to be classified is selected in the GUI.

Step 4: Using our model that image is classified as authentic or spliced.

Step 5: The message is displayed on the GUI

4.2 *Dataset description*

Our group found the **Columbia Uncompressed Image Splicing Detection Dataset (CUISDE Dataset) [9]** to be more insightful than the others, it contains two folders one having 183 authentic images and another containing 180 spliced images. The images cover both indoor and outdoor settings. They were taken using a variety of cameras like canon, nikon etc.

CUISDE contains 183 authentic and 180 spliced images and the masks.

Following are samples from the CUISDE dataset, the one on left is a spliced image present in the 4cam_splc folder, while the one on right is authentic present in the 4cam_auth folder.

Also each folder contains a sub-folder named edgemasks which stores edgemasks for all the images in the original directory, i.e. the dataset contains edgemasks for all the images be they spliced or authentic.

(CUISDE samples, Left: spliced image, Right: authentic image)

4.3 *Technologies used*

<u>Language </u>:- Python (Python 3)

<u>Tools and Libraries </u>:-

For development purposes we used coding environments like Google Colab, PyCharm, Jupyter Notebook.

A number of text editors like sublime text, Atom, Notepad++, VS code etc. were used.

Libraries include openCV, PIL, sklearn, numpy, pandas, matplotlib, pickle, tkinter, keras etc.

4.4 *Software & Hardware Requirements*

The code was developed (in addition to Google Colab) on local system having the following specifications :-

RAM :- 8 GB

Intel Core i5 processor

1 TB hard disk drive

Operating Systems :- Windows 10 (64 bit), Linux Ubuntu (LTS 18.04)

For interpreting and running the code on the local system a terminal or command prompt are required. In Windows 10 command prompt (cmd) and in Linux Ubuntu (bash terminal) were used. Anaconda, a cross platform for these operations, it also provides direct access to the other tools like Jupyter notebook.

Running the code (on local system) :-

To run the code python 3 was first installed in the system. The required libraries were then installed using pip (package installer). Softwares like pyCharm were installed as it makes maintaining the code much easier.

For collaboration tools like github and slack were used.

For file storage Google drive was extensively used.

4.5 ***Finished project structure and description***

The project contains 2 files (excluding the readme), the 2 files are :-

1. PreProcessing.py (python file)
2. Classify.ipynb (python notebook)
3. sample_output.py (python file)

The PreProcessing file contains code to make the **CUISDE [9]** suitable for our machine learning model. This code is used to pickle the dataset into various files, each file contains 10 (mostly) or 20 samples. Pickling is used in python to serialize and deserialize data in Python. The objective of this code was to serialize the entire dataset (363 entries at once), however due to limitations of the machine, we needed to repeat the process for many batches each of size 10. It results in 15 files containing serialized data.

Classify.ipynb takes the 15 files of serialized data and unserializes them. Then divide the data into training and testing data. Finally CNN is used for classification.

For development purposes the PreProcessing.py was run on a local machine multiple times and the resulting pickled files are stored on drive. To run this file basic libraries like cv2 (to read and manipulate images), sklearn (provides ready access to machine learning algorithms), pickle (as mentioned above) and other libraries like pandas and numpy need to be installed.

Finally colab is used for classification purposes. To read the files, google drive is mounted on google colab so that colab can directly read any content from the drive, else the entire serialized dataset will have to be updated on Google colab each and every time the session refreshes.

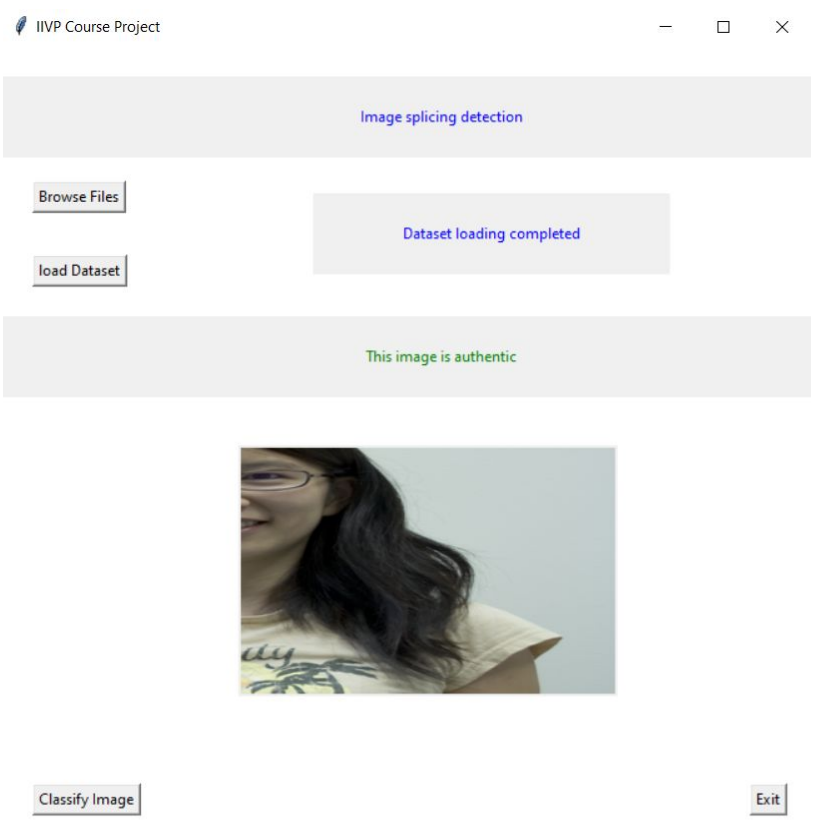
Since pre processing of data takes a long time, the link to drive containing all the processed data is provided in the README file.

Sample_output.py

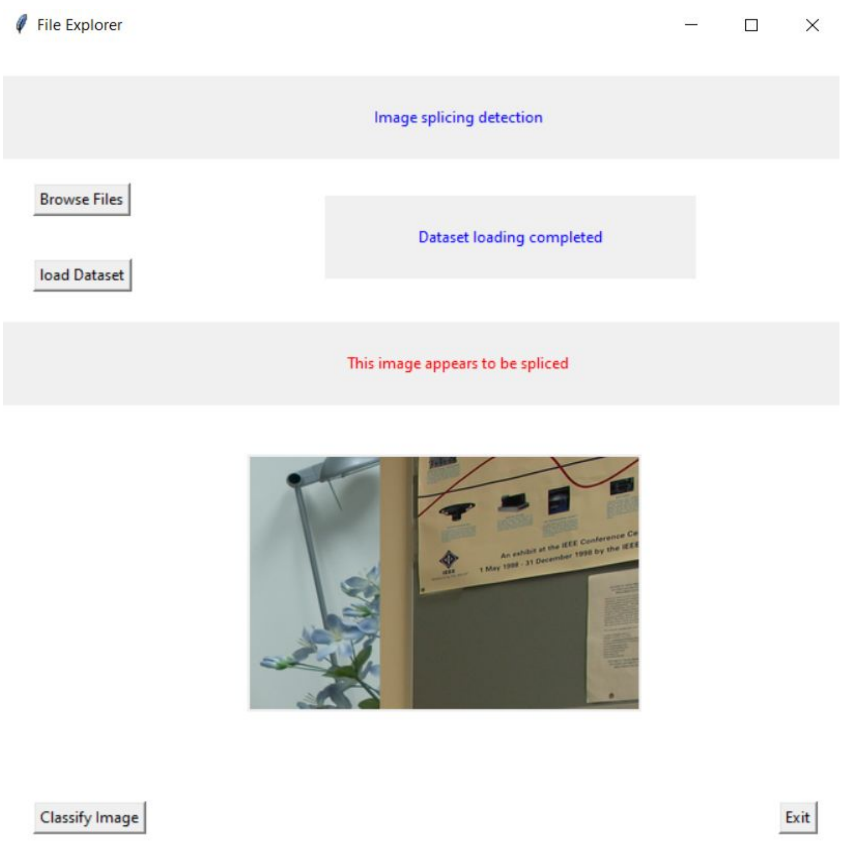
1. As we can see the image to be classified is selected from the GUI from Browse File option.Then the dataset is loaded in which all the data from all the pickle files are loaded.

2. After the dataset is loaded our random forest model is first trained on the dataset and then it is run on the input image.

3. The model then predicts our input image as authentic or spliced.


IIVP Course Project
Image splicing detection
Browse Files
Dataset loading completed
load Dataset
This image is authentic
Classify Image
Exit

Examples

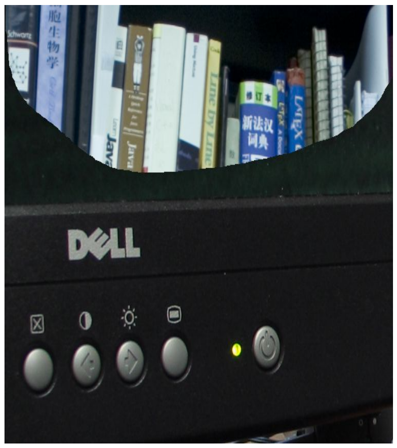
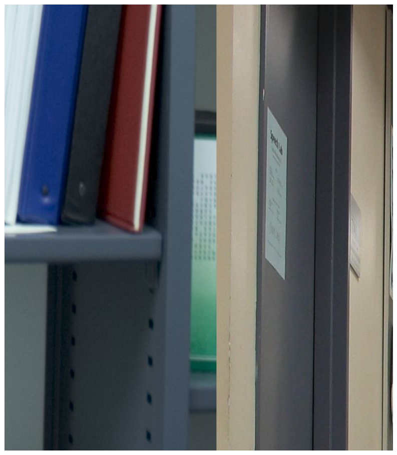

File Explorer
Image splicing detection
Browse Files
Dataset loading completed
load Dataset
This image appears to be spliced
Classify Image
Exit

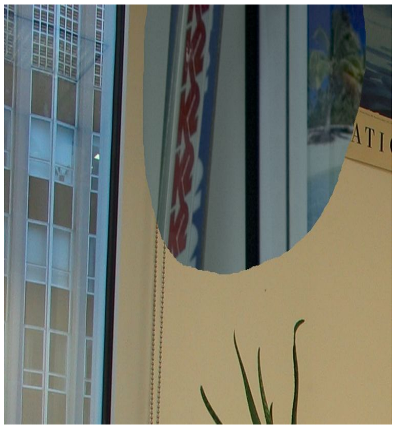
5. Analysis

5.1 Accuracy

To test our model we divided the dataset into two parts one containing 80% of samples used for training and one having 20% for testing. For this purpose of splitting the dataset into 2 parts we use sklearn's train_test_split. On testing our model on the above mentioned test set (20% of entire dataset), we get an accuracy of **68.43%**, i.e. our model was able to correctly sample 644 out of the total 941 samples.

5.2 Inputs and outputs

Image	Verdict	Expected
	Spliced	Spliced
	Spliced	Spliced

	Authentic	Authentic
	Spliced	Spliced
	Authentic	Authentic

(Images taken from CUISDE dataset, canong3_canonxt_sub_09, canong3_canonxt_sub_15, canonxt_23_sub_01, canong3_canonxt_sub_01, canonxt_32_sub_06 respectively)

5.3 *Comparison with other models*

Paper Title	Author(s)	Accuracy
A Review on Splicing Image Forgery Detection Techniques [2]	ChitwanBhalla, Surbhi Gupta	97%
Image Forgery Identification using Convolution Neural Network [3]	N. Hema Rajini	Spliced image detection using the SVM has attained an accuracy of 85-86%. CNN accurately classifies with an accuracy of 99.03%.
Image Splicing Detection via Camera Response Function Analysis [4]	Can Chen, Scott McCloskey, Jingyi Yu	According to the results, CNN classifiers on IGH give an accuracy of 97% on CUISIDE dataset,99% in SpLogo dataset and 97.8% on the combination of the two datasets.
Digital image forgery detection using deep learning approach [6]	A Kuznetsov	97.8%
Image Forgery Detection Based on statistical features of block DCT coefficients [7]	Shilpa Duaa, Jyotsna Singha, Harish Parthasarathya	98%

5.4 *Performance measures*

Posteriori and Priori analysis

As in the code for loading the dataset into the program, we are looping from 0 to 16 with the increment of +1 and in every iteration we are loading the dataset file which is names as tastyPickle" + str(i) (where i is the counter of loop).

So time complexity for loading the image data:-

O(file_row_size0) + O(file_row_size2)..........O(file_row_size15)

If we generalize the size say n then:-

O(n*16) ⟹ **O(n).**

Now for training the model ,we are using the sequential class keras.model in python:-

As it is a one hidden layer classification:-

Time complexity for training the dataset:-

In Keras, the input dimension needs to be given excluding the batch-size (number of samples). In this neural network, the input shape is given as

=> np.array(x_train).shape[1:]) say n (it is the number of features).

Since each feature is associated with each node in the hidden layer the number of weights to calculate is => m(number of nodes in hidden layer)*n(number of features).

So if there are m number of nodes in the hidden layer , we need to calculate the bias parameter for each node ,So the total weights to calculate are:-

m*n + m.

So for epochs time complexity will become => **epochs(m*n + m).**

Now for predicting the output of the testing data we are using for loop for len(test_data) times. If the size of the testing data is n :-

Then its complexity is **O(n) .**

Actual time taken by code to execute is:-

In sec (for python program) => **79.9553234577179 sec.**

6. Conclusion

In this paper we discussed a solution to the problem of "Detection of image splicing using image forgery techniques", for this purpose we decided to use the CNN classification model on the CUISDE dataset. The method involves considering all 64*64 windows in the image. In case of authentic images all windows are considered while in case of spliced images only the ones with a certain threshold of 1s are considered. Finally we split the data for training and testing purposes and classify it using the CNN algorithm achieving an accuracy of 68.43%. We aim to utilize the time before final submission to work on some remaining (some optional) factors like :-

(i) Training and testing model on a larger dataset like the CASIA [8] dataset.

(ii) Improving performance of the model by experimenting with more algorithms and parameters configurement.

(ii) Optionally trying to provide a gui for the project.

7. References

[1] Tamana Sharma, Er.Mandeep Kaur "Forgery Detection of Spliced Images Using Machine Learning Classifiers and color Illumination", International Journal of Innovative Research in Science, Engineering and Technology http://www.ijirset.com/upload/2016/june/186_54_Forgery.pdf

[2] Chitwan Bhalla Surbhi Gupta, "A Review on Splicing Image Forgery Detection Techniques", International Journal of Computer Science and Information Technology & Security (IJCSITS), ISSN: 2249-9555 Vol.6, No.2, Mar-April 2016 https://pdfs.semanticscholar.org/f6a9/877a29d2b90601d320cabe491c16ddc0270a.pdf

[3] N. Hema Rajini, "Image Forgery Identification using Convolution Neural Network" ,International Journal of Recent Technology and Engineering (IJRTE) ISSN: 2277-3878, Volume-8, Issue-1S4, June 2019 https://www.ijrte.org/wp-content/uploads/papers/v8i1s4/A10550681S419.pdf

[4] **Can Chen, Scott McCloskey, Jingyi Yu, "Image Splicing Detection via Camera Response Function Analysis", Computer Vision foundation https://openaccess.thecvf.com/content_cvpr_2017/papers/Chen_Image_Splicing_Detection_CVPR_2017_paper.pdf**

[5] Diaa M. Uliyan, Mohammad T.Alshammari, "Investigation of image forgery based on multiscale retinex under illumination variations", Department of Information and Computer Science, College of Computer Science and Engineering, University of Ha'il, Ha'il, 81481, Saudi Arabia https://www.researchgate.net/publication/341728677_Investigation_of_image_forgery_based_on_multiscale_retinex_under_illumination_variations

[6] A Kuznetsov, "Digital image forgery detection using deep learning approach", Journal of Physics: Conference Series https://www.researchgate.net/publication/337550730_Digital_image_forgery_detection_using_deep_learning_approach

[7] Shilpa Duaa, Jyotsna Singha, Harish Parthasarathya, “Image forgery detection based on statistical features of block DCT coefficients”, Third International Conference on Computing and Network Communications (CoCoNet’19)
https://www.sciencedirect.com/science/article/pii/S1877050920310048

[8] Jing Dong, Wei Wang and Tieniu Tan, “CASIA IMAGE TAMPERING DETECTION EVALUATION DATABASE”, Institute of Automation, Chinese Academy of Sciences P.O.Box 2728, Beijing, 10190
http://ir.ia.ac.cn/bitstream/173211/12318/1/06625374.pdf

[9] Y.-F. Hsu and S.-F. Chang. Detecting image splicing using geometry invariants and camera characteristics consistency. In 2006 IEEE International Conference on Multimedia and Expo, pages 549–552. IEEE, 2006”. Columbia Uncompressed Image Splicing Detection Dataset is available here.

[10]
https://towardsdatascience.com/image-forgery-detection-2ee6f1a65442

[11]

https://github.com/vishu160196/image-forgery-detection/blob/master/top_model.ipynb

[12] https://www.geeksforgeeks.org/python-gui-tkinter/