LIVERPOOL JOHN MOORES UNIVERSITY

FACULTY OF ENGINEERING AND TECHNOLOGY

FAKE IMAGE FORGERY DETECTION USING DEEP LEARNING

By

SHUBHANGI TRIVEDI

Supervisor: Mr Bikash Santra

Final Thesis Report

Submitted on

DECEMBER 2021

A thesis submitted in accordance with the statutes and regulations of the University in part fulfilment of the requirements for the Degree of Master of Science

DEDICATION

This work is dedicated to my son, who has made me feel better and more fulfilled. I am now stronger and more determined than I could ever imagine. I saw you grow as this project progressed, and that satisfaction is indescribable.

ACKNOWLEDGEMENT

It is my great joy to express my gratitude to the Almighty God for providing me with a sound mind in which to complete my MS Research Report.

Mr. Bikash Santra, my thesis supervisor, deserves special recognition for his unwavering support and patience during this research. He was always accessible to answer any questions I had, and having his help ensured that my research was completed on time.

Finally, I'd want to thank my wonderful and supporting husband, Ankit. Your words of support during difficult times were much appreciated and recorded. Thank you from the bottom of my heart.

ABSTRACT

Now we depend a lot on multimedia content and with the availability of increased need of content, there is a lot of intentional manipulation is happening. With advancement, we have multiple options available in the market for image editing. Image tempering has become very easy and detecting these forgeries through naked eyes has become a difficult task. Thus, with such advancements in technology, it gets very difficult to judge the credibility of any image available, and such images are accepted by the public without questioning their integrity. So, in this dissertation, we will try to detect real and forged images using deep learning models. For training purposes, we have used the publicly available dataset CASIA V2.0. We used the passive image forgery detectors that would use ELA information of the image to classify an image into fake or real. When an image is forged it is generally the metadata information of the image that gets manipulated and on a portion of the image, an error is introduced due to recompression that happens on each image save.

By using error level analysis, we will calculate the difference between compression levels between different images. We will be using different state of Art models for feature extraction we used the ELA technique. We have also used various pre-processing techniques like denoising, image augmentation that helped in achieving maximum accuracy.

We have used a custom CNN model also along with state of art models like VGG16and RESNET50, but we achieved the maximum validation accuracy of 94% with the custom CNN model and, that model we have considered the final model for image forgery detection.

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AMFE - APPROXIMATED MACHADO FRACTIONAL ENTROPY
CASIA - CHINESE ACADEMY OF SCIENCES' INSTITUTE OF AUTOMATION.
CCD - CHARGE-COUPLED DEVISE
CFA – COLOR FILTER ARRAY
CFM – COLOUR FILTER MOSAIC
CMFD - COPY MOVE FORGERY DETECTION
CMOS - COMPLEMENTARY METAL-OXIDE SEMICONDUCTOR
CNN- CONVOLUTIONAL NEURAL NETWORK
DCT - DIGITAL COSINE TRANSFORM
DNN - DEEP NEURAL NETWORKS
DL -DEEP LEARNING
ELA – ERROR LEVEL ANALYSIS
EDA- EXPLORATORY DATA ANALYSIS
FPR - FALSE POSITIVE RATE
FCN – FULLY CONVOLUTIONAL NETWORK
GAN - GENERATIVE ADVERSARIAL NETWORK
MFCN - MULTI-TASK FULLY CONVOLUTIONAL NETWORK
PCA - PRINCIPAL COMPONENT ANALYSIS
PCT - PRINCIPAL COMPONENT TRANSFORMATION
PDF – PROBABILITY DENSITY FUNCTION
RNN- RECURRENT NEURAL NETWORKS
SIFT - SCALE-INVARIANT FEATURE TRANSFORM
SVD: SINGULAR VALUE DECOMPOSITION
SVM - SUPPORT VECTOR MACHINE
SURF- SPEEDED UP ROBUST FEATURES
TIFF - TAGGED IMAGE FILE FORMAT
VGG - VISUAL GEOMETRY GROUP FROM OXFORD

CHAPTER 1: INTRODUCTION

1.1 Background

The use of digital media has become a part of our daily life with the advent of technology-handheld devices and fast flow of information. Internet and connectivity have exposed us to Exabyte of data especially graphical data such as images, videos, etc. Mobile sharing of images has become so common throughout the world that people have by default exposed themselves to cyber predators who keep a close watch on a typical user's activity. A typical mobile/computer user is not aware of what these predators can do with the images they are sharing. These predators use different types of techniques of changing the data within the media and use the exploited media for malicious purposes.

This situation portrays a definitive need for a comprehensive solution to protect the end user from such digital predators. For providing a flawless solution to the problem, developers/security experts need to have some sort of basis to find out whether the given sample media (photo or a video) is a tampered one or the original one.

The developer may simply ask two questions to determine the authenticity of a given sample image. Firstly, a developer may ask whether the image still depicts the captured original scene. Secondly, he/she may ask whether the image was captured from the device from which it is claimed to be generated. The second question is of prime importance as it allows determining definitive information (user/device which generated the media) about the source of the image. Since it is never easy to determine the actual source of the image, it becomes a great challenge to know anything about the original image.

The developer, therefore, needs to blindly follow what is available in his/her hands i.e., Tampered/original Image. Security experts have classified two methods to overcome challenges related to source determination, first is active in which information acquired from the source (camera, computer, etc.) is exploited to determine the authenticity of the available information. The second is passive in which no information of source is available for exploitation only the media available at disposal has to be checked directly. Types of image forgery detection techniques are shown in Figure 1.1.

An active approach is dependent upon a trustworthy camera i.e. A device that somehow grants a digital signature/watermark to the image automatically. If any cyber predator tries to tamper with the image, the act can easily be traced as the watermark/signature gets tampered with. But the active method bears a major drawback as this method requires a trustworthy camera with a standard protocol to provide a watermark to every image which is captured by it; however, such a requirement is next to impossible as providing a fixed watermark as a standard practice by

every manufacturer is not feasible. Hence, this method is limited to a very scarce number of scenarios.

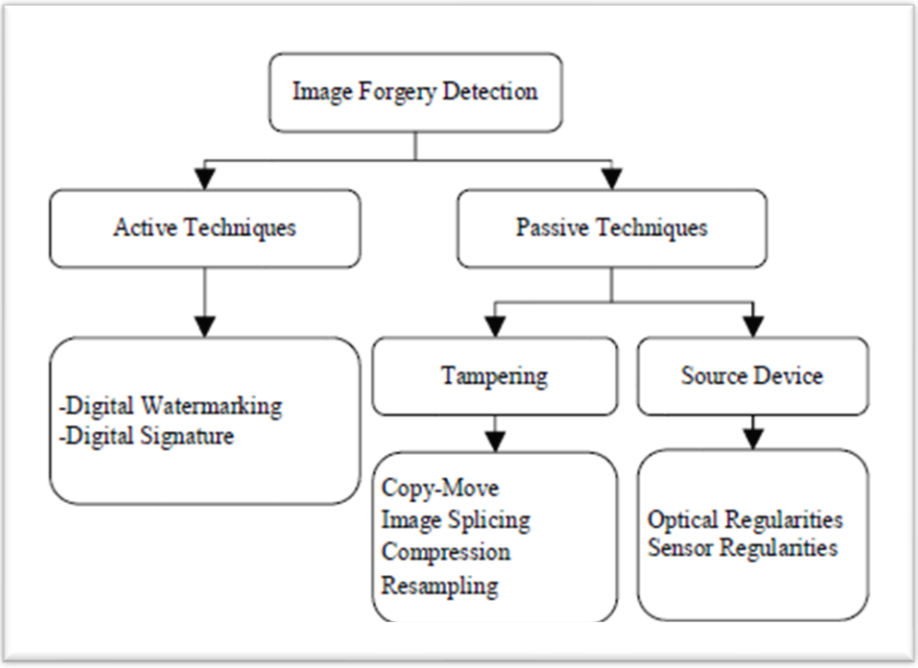


Figure 1.1 Image forgery techniques

Such problems can easily be overcome by the use of techniques that do not require any prior information about the image and are hence called passive techniques. These techniques work on the principle of determining traces of data left whenever an image is acquired, compressed, stored, or modified. When such features of data are analyzed, it becomes easy to check for any kind of incongruence in the available data.

A typical image can be edited in a variety of ways and therefore, poses a challenge for developers/security experts. An image can be edited for various reasons such as for the improvement of image quality etc. such editing is known as innocent editing, in which the available information becomes more streamlined. On the contrary, when the information is changed by hiding or adding something then such editing is known as malicious editing which is of prime interest for security experts.

Image editing is broadly classified into three major categories, these categories comprise both operators i.e., innocent and malicious. The categories are Enhancement, Geometric and Content Modification. Here enhancement operator is primarily used for innocent editing, while other 02 categories (Geometric and Content modification) are intended for malicious attacks.

Enhancement techniques involve basic features such as color modification, contrast adjustments, etc. Geometric modifications involve zoom, rotation, cropping, etc. finally content modification involves copy-move, cut and paste, etc. From the above explanation, it can easily be sought those the latter two techniques will one way or the other involve malicious editing.

Figure 1.2 Forged Images Examples, the first example shows a change of person (image splicing), the second image shows the addition of fountain (Copy-move) and the third image shows the removal of a person(removal) (Diallo et al., 2020)

Most common malicious attacks involve cropping, cut-paste, etc. as these techniques allow the forger to remove the content of an image and paste it somewhere else, be it any media. Figure 1.2 shows an example of the most common type of forgery. The most common categories of image forgeries are:

1) **Copy-Move Forgery:** This kind of forgery involves transforming an original image by tampering with its contents by the use of editing features such as moving and copying. This kind of forgery becomes handy for forgers who want to make use of images in carrying out their political agendas, spreading misinformation, or creating illusions among the audience. This kind of forgery involves no change in the color and background of the original image. Many tools have been developed to detect such a forgery.

2) **Image Splicing:** This kind of forgery mainly involves merging the contents of the same image or different images to compose a forged image. This technique enables the forger to change the backgrounds of the image or use a background of the different images in

the subject image. This technique also allows enhancing the field of vision of any particular image. This technique provides greater flexibility to the forger and allows the forger to make multiple changes in a single image. The detection of this technique is a cumbersome task but can be identified if the forger has left any shadows, reflections in a given image. Figure 1.3 shows an example of image splicing.

(a) *Image Original "donor"* (b) *Image Original "recipient"* (c) *Image Result*

Figure 1.3 Splicing

3) **Image Retouching:** This method allows altering common features of any image such as changing the color of the image, changing the scale of a given image, stretching, or squeezing a particular part of an image. Detection of this technique is very difficult as it involves multiple changes in multiple areas of a given image. Image retouching can be detected by detecting different lighting conditions in a spliced image. Figure 1.4 shows an example of image retouching.

Figure 1.4 Example of Image retouching (Armas Vega et al., 2020)

1.2 Problem statement

When talking about image forgery using copy-move techniques, then splicing is considered difficult to be detected. Spicing is an operation in which a portion of one image is copied by an attacker and pasted on another image thus altering the image. What further makes its recognition difficult is that this copy-move is also followed by some level of compression, with

some blur effect and then smoothening of boundaries. Thus, detecting forgery in the case of splicing is comparatively tricky than forgery techniques.

ELA is the best technique that uses the difference between compression levels among different regions of the image. Similarly, deep learning models are considered best for image recognition. Hence, in this study we will try to use CNN for the identification of forgery.

1.3 Aims and Objectives

The main aim of the research is to identify tampered images correctly using the CASIA V 2.0 dataset which involves the use of a Convolutional Neural Network for training purposes.

The objectives of the study based on the aims are following:

- To identify the best preprocessing techniques that help in identifying image forgeries better and can generalize images effectively.
- To identify if ELA can help in effective feature extraction
- To identify the best state of the art model that can help in detecting image forgeries with better accuracy.
- To give an overview of different types of techniques that are currently used for image forgery.

1.4 Significance of study

The result of this study will help in identifying the image forgery or tampering with high accuracy and precision. With the increased usage of social media, incidences of image manipulation are increasing and with advancement, in editing technology, these manipulations have become very hard to detect via the traditional approach and have started to erode the trust in digital content. This phenomenon demands the need for a way to help us verify the truthfulness and credibility of the image. Though color modification, contrast adjustment, and filtering are part of innocent image editing, geometric modification and content modification lead to malicious image editing.

If a forged image is spliced or copy-move, then interpolation is a necessary step. When a forged image contains areas from different sources, or another part of the same image, rescaling and /or rotation are often involved. In general, these changes are mostly done on a part of the image, not on the entire image. Thus, dividing the image into multiple blocks, and then detecting the changes compared to neighboring blocks will help us to detect the manipulated area. Apart from this whenever an image has tampered the tampering process introduces a mild level of compression and hence compression can also help in identifying tampering. There is a

significant level of change that may occur to image metadata also where the source of images is different. Hence by using this understanding of image forgery, we can develop a model that can help us identify image forgery.

In many cases, images are proof of medical reports and crime scene evidence and such forged image can result in loss of life or escape of convict. Thus, by this paper, we will try to create a model that could detect image forgery effectively and efficiently and thus we can easily verify the authenticity of an image.

1.5 Scope of Study

The scope of this work focuses on proposing the use of a pre-trained model and its performance will be decided then based on the accuracy of detection of fake images.

Preprocessed images with labels real and fake will be fed to the pre-trained models for feature extraction using ELA and then PCA would be used for feature reduction. Later the extracted features will be used for the classification of images using different classifiers. The proposed work will be using python and the CASIA V2.0 dataset. Figure 1.5 shows the high-level approach that we will use in this study.

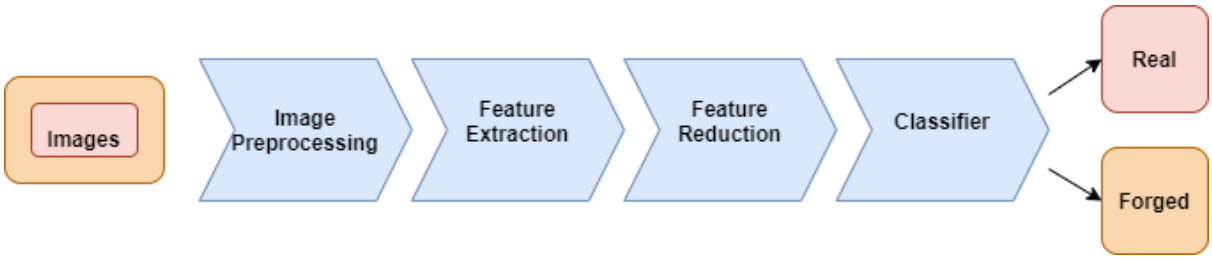


Figure 1.5 Flow Chart for the proposed approach

1.6 Structure of Study

This work is majorly classified into six chapters.

The first chapter is an introductory chapter that covers what is image forgery, how image forgery can be categorized and achieved, what issues does image forgery causes, and how to detect image forgery, and the scope of this report. The second chapter covers all the related work done in the field of image forensics and have also covered details about the origin of digital photography, formats of images and all the techniques developed so far in detecting image forgeries. The third chapter discusses dataset description and the proposed research

methodology which covers preprocessing methods and existing models. The fourth chapter is an extension of chapter three and covers in-depth the EDA performed and preprocessing techniques implementation and model building. The fifth chapter discusses the results and evaluation metrics. The sixth chapter discusses the conclusion, learnings, and future recommendations. The last section covers the references.

CHAPTER 2: LITERATURE REVIEW

2.1 Introduction

This chapter covers the basics of digital photography, including the production process, image formats, and file compression. It also covers some of the techniques and procedures used to forge digital images, as well as some of the techniques and methods used to detect image forgery.

Digital Photography

The method of capturing light and recording the captured light to a digital medium from the electromagnetic spectrum is known as digital photography. A sensor in a digital camera stores and saves photographic images in digital form.

The camera captures millions of "dots" called pixels in digital images, which a computer uses to show the image by splitting the image into a corresponding grid of pixels, each comprising RGB (Red, Blue, Green) areas called sub-pixels. The brightness of each of these sub-pixels is then specified using the values saved in the digital snapshot, and the pixel's combined brightness is viewed as a single colour (Dennis P Curtin, 2011).

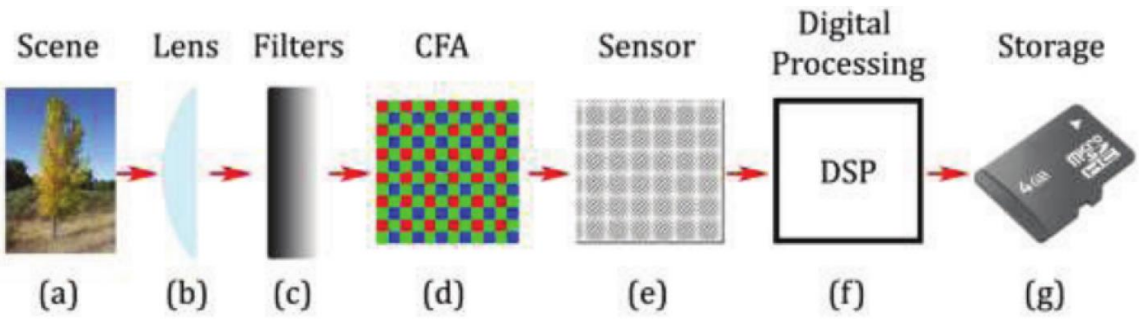


Figure 2.1 Digital Photography

The digital photography model is depicted in Figure 8. An optical lens [Figure 2.1(b)] focuses upon the light from the scene [Figure 2.1(a)] into the camera. The quantity of light that enters the camera and the quantity of the scene captured, as well as focusing on the image are all controlled by the lens. The lenses are classified into three groups: Wide-angle lenses capture a large arc of the scene's content and do have a shorter focal length than a standard lens. Macro lenses photograph objects that are extremely close to the lens. Telephoto lenses (Wikipedia, 2021) capture images from a great distance.

The light passes through a succession of filters after passing through the lens, therefore preparing it for conversion into the digital realm [Figure 2.1(c)]. The anti-aliasing filter, sitting on top of a camera's sensor and preventing high spatial frequencies from going through, which corresponds to very tiny details in the scene, is one of these filters. The anti-aliasing filter, on the other hand, restricts the frequencies that pass through the sensor and is commonly referred to as a "low-pass" filter since it only allows lower frequencies to pass through(Anderson, 2011).

The image sensor is made up of pixels, which are picture elements that register the amount of light that falls on them. After that, the elements convert the amount of light they received into the corresponding number of electrons, which are subsequently translated into voltage and finally into digital values. CCD (Charge-Coupled Devise) and CMOS (Complementary Metal-oxide Semiconductor) are the two types of sensors typically used in cameras. Although their basic functions are comparable, most digital cameras employ a CMOS sensor since it provides faster speed and reduced power usage (Moynihan, 2011).

A Colour Filter Array (CFA) is employed to help discriminate between different frequency ranges of light because pixels are not colour sensitive elements [Figure 2.1(d)]. The colour filter array (CFA) is a mosaic of microscopic colour filters placed on top of monochrome image sensors, commonly on top of CCD and CMOS sensors, to split the light into distinct pixels by colour frequency. The Bayer RGB colour filter array, which consists of a mosaic of red, green, and blue, is one of the most prevalent forms of filter arrays. The filter pattern is 50 percent green, 25 percent blue, and 25 percent red since human eyes are more sensitive to green colours (Lukac and Plataniotis, 2005)

Following the CFA, the digital sensor collects the information by quantizing light intensity levels, which converts a continuous signal into a finite set of intensity values. Following the sensor's conversion of light intensity for each colour, a "mosaicing" method is required to compute colour values for each of the three basic colours represented at each pixel location using interpolation [Figure 2.1(f)], (Kimmel, 1998)

The onboard camera processor then performs several actions, including white balance and gamma correction [Figure 2.1(f)]. [Figure 2.1(g)] The camera processes the sensor data and creates a digital image file for storing on a digital memory device.

2.2 Image Compression

Consider a general CT picture, with 512 x 512 pixels size and has a pixel resolution of 12 bits per pixel. The CT picture would require a minimum of 393,216 bytes or 3,145,728 bits of

storage space, ignoring any further metadata. When you consider the amount of storage required for a 100-slice study and ten such studies per week, it's simple to see why digital storage capacity is such a major worry when working with digital imaging. Rather than mindlessly purchasing gear to enhance accessible storage capacity, it makes sense to find a way to make better use of what we already have.

The goal behind picture compression is to take advantage of redundancies in image data and re-encode it in such a way that it becomes more compact. Lossless and lossy compression are the two basic methods of data compression.

2.2.1 Lossless

From (Karam, n.d.) Lossless compression reduces mathematical redundancy by compressing data without sacrificing quality. This implies the algorithm scans the photos for recurrent patterns or sequences and simplifies them to a simple formula. In a typical CT image, for example, the surrounding boundaries are usually completely black (air), which is represented by a continuous string of zeros (0 0 0 0 0 0 0 0 0 0 ...). The lossless method would recognise the pattern and replace it with an encoded formula of the type "repeat zero 10 times" as shown in Figure 2.2. When a compressed image is decompressed for viewing, the resulting image is identical to the original source image, implying that no data was lost.

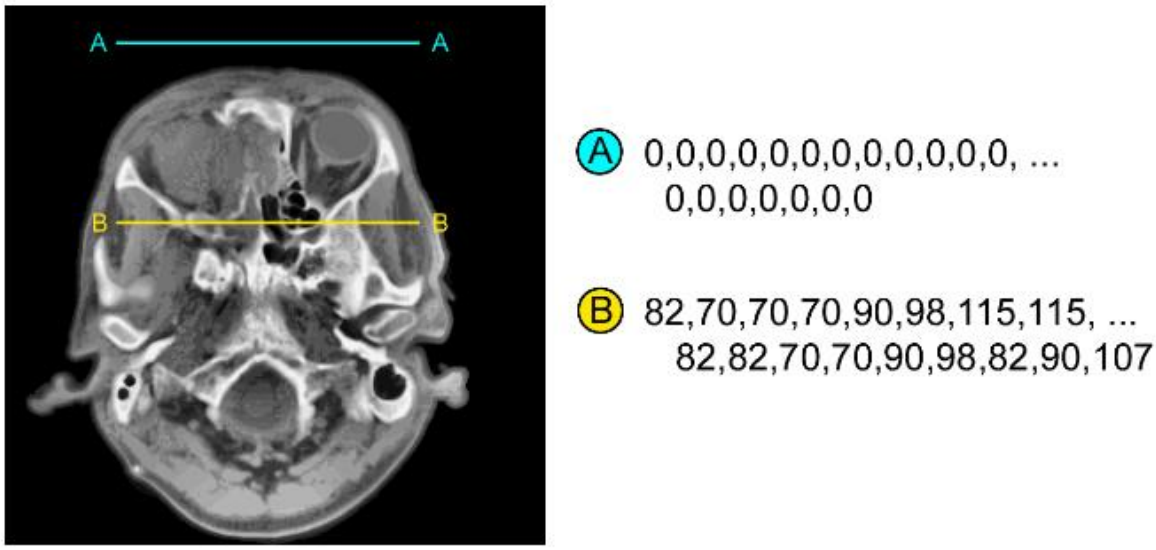


Figure 2.2 Line profile A–extended A's stream of zeros (black) will easily give significant compression savings, however, line profile B–seemingly B's random profile will see a modest benefit. Data patterns and repetitions are exploited in lossless compression. Tan e

When the compressed image is decompressed for viewing, the resulting image is identical to the original source image, indicating that no data was lost during compression. Lossless formats save about 1:2 in space and work best with very simple images like schematics and line art, as well as images with clean lines and flat colour.

2.2.2 Lossy

Perceptual redundancy is reduced by lossy compression. This means that the algorithm considers the limits of the human eye and discards data that is regarded non-essential to the overall image's perceived quality. Colour details, as well as very bright and very dark tones, for example, are less sensitive to the human eye. The lossy method may then diminish the spatial resolution of the colour channels and smooth the image's highly brilliant and very dark areas.

It's worth noting that, unlike lossless compression, the final decoded lossy image isn't exactly the same as the original source. The more forceful the compression, the more information is lost, and the difference between the compressed and original image becomes more visible. Lossy compression, on the other hand, usually produces better results, with a space savings of around 1:10 or so. Lossy compression works best with photographic images or images with few sharp edges, such as gradients and tones (Figure 2.3).

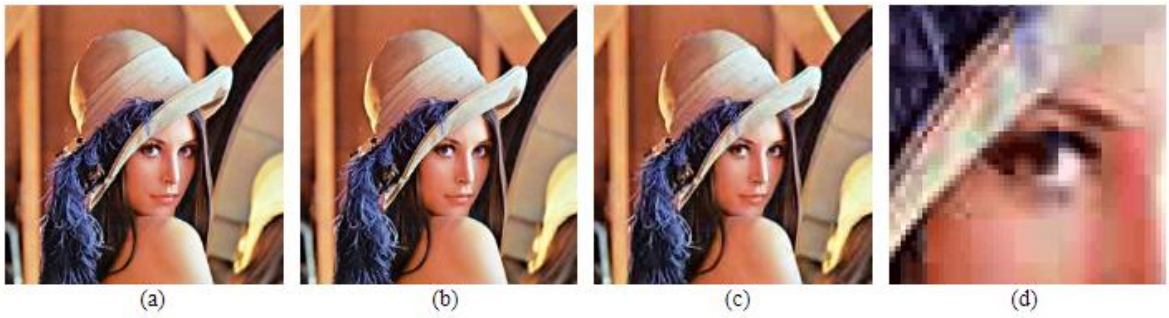


Figure 2.3 Lossy compression artifacts (a, b, and c): the example image is saved at successively greater levels of compression, initially at 1:1, then the second image with 1:10, and the final image with 1:30 of compression level. The image is quite legible even at 1:30, though there are apparent visual flaws; (d) the image with 1:30 compression level when zoomed 5 times. image is showing visible smudging, which is a really important feature of the lossy compression of the JPEG. (Tan, 2006)

2.3 Image Formats

When capturing photographs, one of the most crucial workflow decisions you make is which image file format to utilize. A standard specification for encoding information about an image

into bits of data for storage is known as an image file format. (Tan, 2006) describes an image saved and encoded in a recognized image format identifies itself as an image and offers essential information like matrix size and bit depth. There are so many different pictures file formats that it might be difficult to figure out which one is ideal for your purposes.

Some, like TIFF, are suitable for printing, while others, like JPEG and PNG, are best for web graphics. We'll go over some of the most prevalent image file formats.

2.3.1 JPEG

Joint Photographic Experts Group

(Wiggins et al., 2001) JPEG was developed in the early 1990s to be the forerunner of the next generation of image compression techniques. JPEG is the name of the compression method developed by the Independent JPEG Group, not a file format. JPEG, like GIF, employs pixel files to hold bit-mapped information; however, it does not use indexed colour. The JPEG format's strength is its ability to compress large image files greatly, allowing for faster electronic movement. The fundamental flaw

JPEG is that it is a lossy compression technology, which means that data is lost with each compression, potentially resulting in image degradation.

2.3.2 TIFF

Tagged Image File Format

(Wiggins et al., 2001) TIFF is a trademark that was first registered by Aldus, which amalgamated with Adobe Systems later on (San Jose, Calif). TIFF was developed primarily by manufacturers of input and output devices such as printers, monitors, and scanners; as a result, it is designed to work with a variety of image processing devices. TIFF stands for "Tagged Image File Format," which refers to the format's sophisticated file structure. The file's initial header is followed by "chunks" of data known as "tags," which send picture information to the computer that displays the file.

TIFF's major advantage is that it can handle a wide range of picture sizes, resolutions, and colour resolution without losing the detail thanks to lossless compression.

2.3.3 BMP

Windows Bitmap Format

Microsoft® introduced BMP as the native bitmap picture format for their Microsoft Windows® operating system. BMP is a relatively simple format that lacks many of the capabilities of other, more powerful formats. However, it is supported by most applications and is sometimes

referred to as the lowest common denominator format for transferring images across programs. (Tan, 2006)

2.4 Methods of detecting image tampering:

The set of image forensic tools used earlier as stated by (Farid, 2009) can be grouped into 5 categories namely

• Pixel-based techniques that detect statistical abnormalities introduced at the pixel level

• Physics-based techniques that explicitly model and detect anomalies in the three-dimensional interaction between physical objects, light, and the camera

• Format-based techniques that leverage the statistical correlations introduced by a specific lossy compression scheme

• Camera-based techniques that exploit artifacts introduced by the camera lens, sensor, or on-chip postprocessing

• Geometry-based techniques that explicitly model and detect anomalies in the three-dimensional interaction between physical objects, light, and the camera

2.4.1 Physics-based techniques

To detect photos that have been manipulated and to discover changing regions in an image, the simplest method is to employ human observation or physical laws. To do so, we can look for elements in the image that are:

2.4.1.1 Lighting and shadows:

Because shadows and sharp highlights reveal the source and direction of light when multiple photos are combined into a single image, the shadows may be uneven or the lighting may differ. As seen in Figure 2.4 a recognised counterfeit of John Kerry and Jane Fonda sharing the stage at an anti-war rally is shown above. Kerry's estimated light direction is 123, while Fonda's estimated light direction is 86. An actual photograph of Richard Nixon and Elvis Presley is shown here. Nixon and Presley's estimated directions are 98 and 93, respectively.

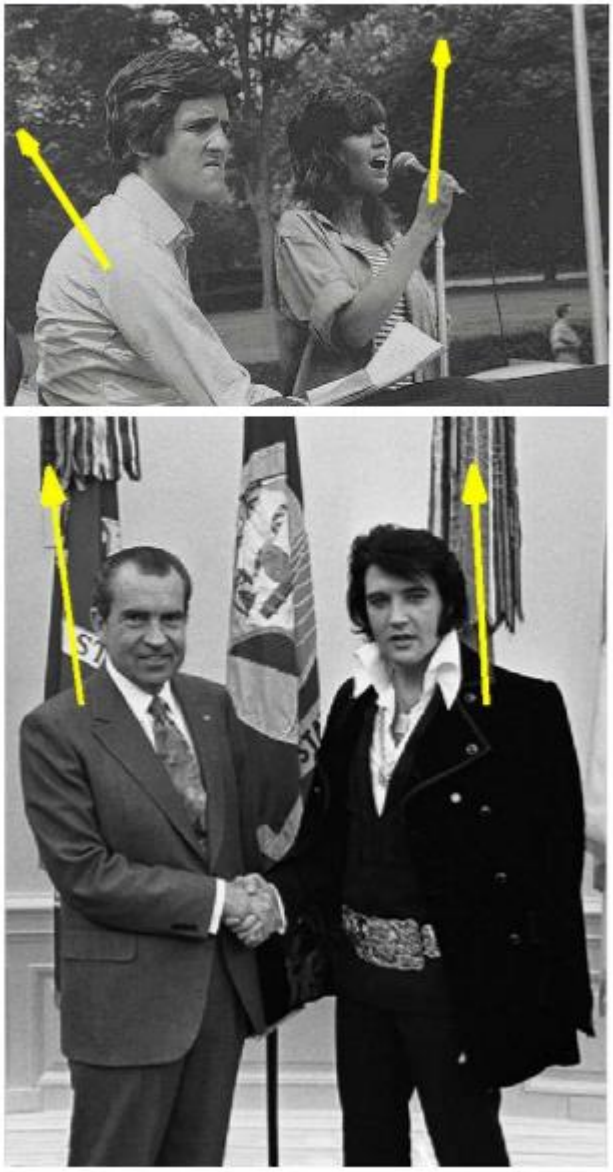


Figure 2.4 A recognised counterfeit of Jane Fonda & John Kerry sharing the stage at an anti-war rally is shown above. Kerry's estimated light direction is 123, while Fonda's light direction is 86 estimated. An actual photograph of Richard Nixon and Elvis Presley (Johnson, 2005)

2.4.1.2 Floating and Scale:

When you combine too many photographs, the sizes may become erratic. Splicing a person into an image can also make it appear as if that person is floating or not rooted to the earth.

2.4.2 Format Based

2.4.2.1 Metadata Analysis:

As previously stated, images can be saved in a variety of image file formats, each of which contains some information. Changes to the image can sometimes alter the file format in image forgeries. (N.Krawetz, 2007) stated in his paper for the JPEG file format that JPEG comprises feature sets, and therefore changing the image will change the feature set. As a result, if the feature sets are manipulated, those manipulated feature sets can help in identifying tampering.
The majority of image files contain more than just a picture. They also carry information about the image (metadata). Metadata describes a photograph's provenance, such as the type of camera used, colour space information, and application notes. Different sorts of metadata are included in different photo formats. Beyond the image size and colour space, several formats, such as BMP, PPM, and PBM, contain very little information. A JPEG from a camera, on the other hand, typically carries a lot of information, such as the camera's manufacturer and model, focal and aperture information, and timestamps.
Unless the image was converted from a JPEG or altered, PNG files often contain relatively little information. Some information that we must look for as stated in (Afsal Villan et al., 2017) are:

1) The manufacturer, model, and software: These are used to identify the device or application that took the photo. The EXIF metadata block on most digital cameras includes a Make and Model. The camera's firmware version or application information may be described in the software.
2) Image Dimensions The dimensions of the image are frequently recorded in the metadata. Is the rendered picture size (at the bottom of the metadata) consistent with the other sizes? Many programs crop or resize images without altering the metadata.
3) Date and time stamps: Look for fields that contain timestamp information. These are used to determine when a photograph was taken or edited. Do the timestamps correspond to the expected timeline?
4) Different types of metadata: There are numerous sorts of metadata. Some are created solely by cameras, while others are created solely by software.
5) Descriptive text: Many photos have embedded annotations that describe the image, identify the people in it, and so on.
6) Missing Metadata: Are there any metadata fields that are missing? If the photo was taken with a digital camera, it should include camera-specific data. Metadata is stripped from several programs and internet services. A lack of relevant metadata usually suggests that the image has been resaved rather than being original.
7) Altered Metadata: The chain of custody for evidence handling is equivalent to metadata. It can tell you how a photo was created, processed, and saved. Some people, on the other hand, modify metadata on purpose. In an attempt to deceive, they may alter timestamps or photo information.

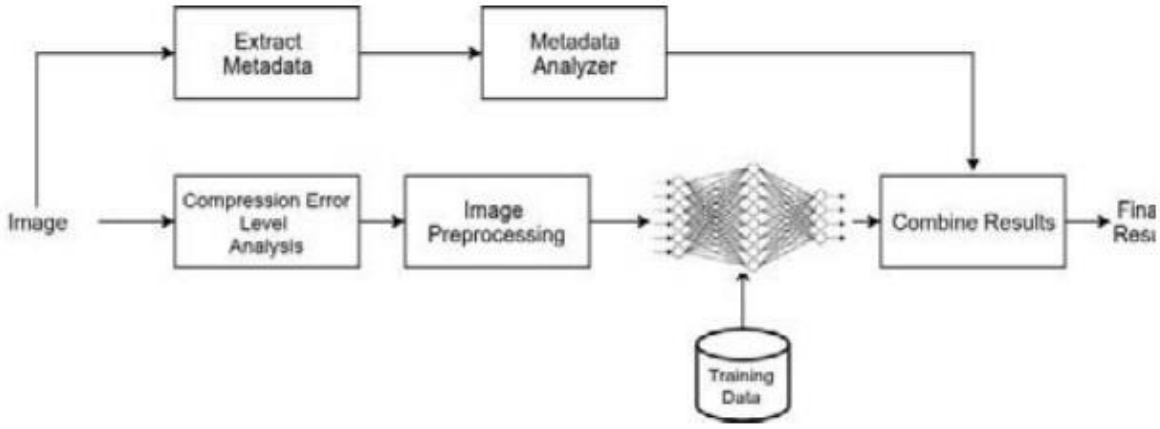


Figure 2.5 Block diagram of the ELA method used by (Afsal Villan et al., 2017)

Figure 2.5 shows the flow chart of the system that is used by (Afsal Villan et al., 2017). This system uses a metadata analyser as feature extractor and the features extracted by metadata analyser is used in combination with feature extracted through compression error level analysis. Results from both the components are used for the final output.

Although metadata can provide useful information about a photograph, it does have several limitations:

- Because some metadata fields are plain text, photo editors may attempt to modify the metadata, which may go undetected.
- Misleading metadata information. Concatenating two separate photos in Photoshop, for example, can result in the resultant image containing metadata from one of the concatenated images. Because the metadata information didn't match the new stored image, this results in hazy details.

2.4.2.2 JPEG Quantization using DCT:

Most of the images used these days are in JPEG format. Author (Lin et al., 2009b) suggested the use of the DQ effect. DQ effect is the presence of periodic peaks and valleys in the histograms of DCT coefficients. The method uses a DCT block of 8X8. For each frequency, the DCT(Karam, n.d.) coefficients of all the blocks can be gathered to build a histogram. Though this approach was limited to only JPEG format since this method worked without fully decompressing the image, this method was faster. Figure 2.5 shows how DCT can be used to highlight tampered regions.

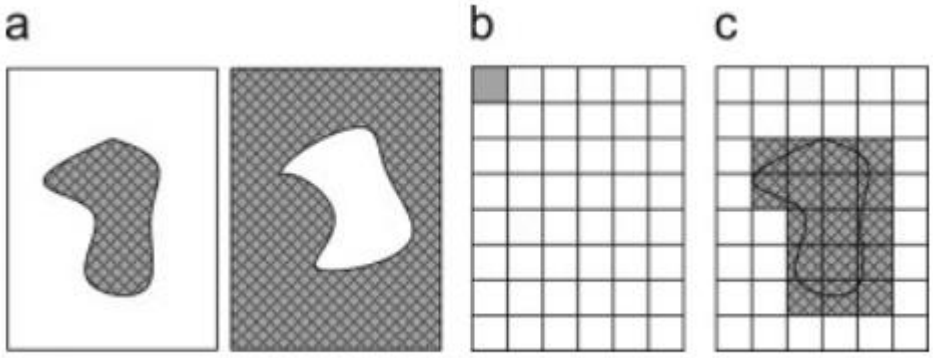
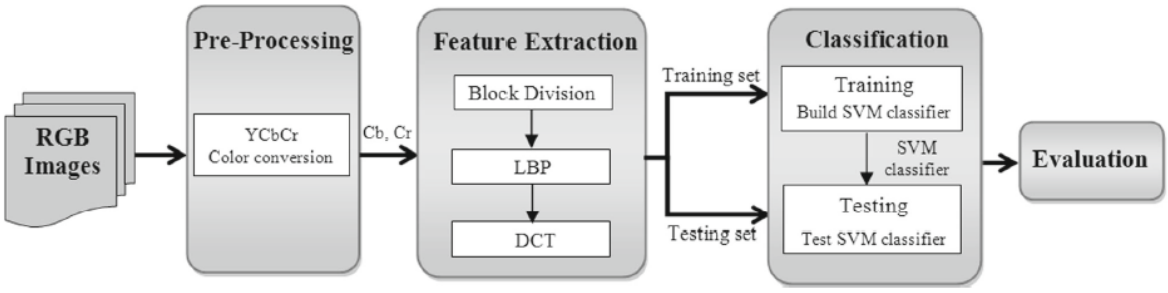


Figure 2.6 a. Shows the unchanged region of a tampered image as a blank region and the changed or forged region as a shaded region.

With further improvement, for localization of tampered region (Wang et al., 2014) proposed the use of Laplacian Mixture model for the distribution of AC DCT coefficient and later EM (Expectation-Maximization) algorithm was employed to find the probability of an image being tampered.

(Huang et al., 2011) DCT is applied to each block to reflect its features in an upgraded DCT technique. Truncating is a technique for reducing the size of features. The feature vectors are then lexicographically sorted, and duplicated image blocks in the sorted list will be new neighbours. In the matching process, duplicated image blocks will be compared. The proposed method is capable of detecting tampering via JPEG compression, blurring, or additive white Gaussian noise. (Alahmadi et al., 2017) used DCT in combination with LBP (Local binary pattern) for tampered part detection.



2.7 Block diagram of DCT method (Alahmadi et al., 2017)

2.4.3 Camera-based techniques:

2.4.3.1 Colour Filter Array Pattern

A colour filter array (CFA) or colour filter mosaic (CFM) is a mosaic of small colour filters that are put over the pixel sensors of an image sensor to collect colour information in photography.

In the CFA demosaicing artifacts, tampering leaves a trace in the image. The manipulated region can be determined by identifying these modifications. CFA number pattern estimate was proposed in (Dirik and Memon, 2009). It required the usage of several candidate patterns obtained through picture re-interpolation. For each candidate pattern, the Mean Square Error between the input and interpolated image is calculated. If all of the MSE values are the same, the image has been tampered. Figure 2.6 and Figure 2.7 illustrate how altered regions can be identified.

Figure 2.8 Original Image

Figure 2.9 Tampered region detected using CFA pattern estimation (Regina et al., 2010)

It is also suggested in (Armas Vega et al., 2020) that the sensor noise in interpolated pixels should be muted if a picture is interpolated. This is related to the low-pass interpolation's nature. Sensor noise variance in interpolated pixels is much lower than the sensor's noise power in non-interpolated pixels. As a result, the interpolation algorithm's artifacts may be quantified by comparing the ratio of interpolated and non-interpolated pixels' noise variances. It's safe to conclude that the input image was modified if this ratio is close to 1.

This method is used to detect tampering that occurs as a result of modifications including resizing, recompression, and filtering. There are, nevertheless, genuine photographs that do not include CFA artifacts. This method cannot be used to verify the authenticity of such photographs.

2.4.3.2 Camera response

Because most digital camera sensors are approximately linear, the amount of light measured by each sensor element and the associated final pixel value should have a linear relationship. Most cameras, on the other hand, use pointwise nonlinearity to improve the final image. In (Hsu and Chang, 2010) authors explain how to estimate this mapping, known as a response function, from a single image in their paper. Tampering is detected by looking for differences in the response function across the image.

2.4.4 Geometric based techniques:

These techniques are based on the main point, which is the projection of the camera centre onto the image plane, which allows for the measurement of objects in the real world and their position relative to the camera. (Kashyap et al., n.d.)

Grooves in handgun barrels give the shot a twist, which improves accuracy and range. These grooves acquaint the bullet shot with certain markings to some extent, and can thus be used with a specific firearm. Several picture forensic approaches have been developed by the same soul.

Principle point and metric measurement detection methods are two types of geometry-based picture forgery detection systems. (Farid, 2009)

2.5 Advance Image tampering detection

2.5.1 ELA (Error Level Analysis):

As mentioned in (Sudiatmika et al., 2019) by storing photos at a given quality level and then evaluating the difference from the compression level, error level analysis is one way for identifying photographs that have been modified. When a JPEG file is saved for the first time, it compresses the image. Most editing software, such as Adobe Photoshop, Gimp, and Adobe Lightroom, enable JPEG compression. If we use image-editing software to reschedule the image, we'll have to compress it again.

(Bakiah et al., 2015) The image is broken into 8X8 chunks and recompressed individually using ELA at a pre-set error rate of 95%. If the image has not been altered in any way, each square

should be given the same quality rating. The degree of error will also be increased during resave procedures. Following resave procedures, the error level potential was reduced, as evidenced by darker ELA results. The square grid may reach its lowest error level after a certain amount of resaving operations. As a result, each resaving procedure may miss out on frequency and specifics. Figure 2.8 depicts ELA in a coloured image where ELA is able to detect tampering quite easily, whereas Figure 2.9 depicts weakness of ELA on greyscale images because the precision of error levels is reduced as the degree of information decreases, error levels generated by high frequency components and error levels generated by various quality are indistinguishable.

Figure 2.10 a) the actual image, b) Forged image with tampering done around eyes and lips and a flower added on the hat, and c) ELA transformed image for the forged image. (Jeronymo et al., 2017)

Figure 2.11 a) shows the original image, b) shows the forgery, with a zeppelin in the background and c) shows ELA for the tampered image. (Jeronymo et al., 2017)

Later (Jeronymo et al., 2017) suggested the use of wavelet thresholding as a denoising technique that helps in removing noise without affecting any other component. The approach successfully reduces noise and improves error levels, allowing for better identification of areas of the image where tampering has occurred. In ELA analysis, the Daubechies wavelet transform outperforms the log Gabor technique. The statistical strategy of calculating a threshold value is not ideal, although producing a better approximation. This is due to the fact that error level analysis builds a noise map from the image by definition. Regions with more noise are likely to reflect forgeries, but ELA also generates noise in regions with high frequency components, such as hair or borders. Handling high-frequency regions are proposed as future work.

(Sudiatmika et al., 2019) proposed the use of the system that integrates the ELA (Error Level Analysis) method and pre-trained model VGG-16, which produced 92.2% training accuracy after using 100 epochs. ELA (Error Level Analysis) uses the difference in compression level between different subsections of an image to identify forgery. (Armas Vega et al., 2020) proposed use of two algorithms for image forgery detection. Error Level Analysis (ELA) algorithm, was used to detect splicing in an image which highlighted those pixels that have a different level of compression whereas another algorithm was based on the quadratic mean error of the Colour Filter Array (CFA) interpolation pattern which determined the level of interpolation in the manipulated image. They used the CASIA V1.0 dataset and got an accuracy of 73.3% with a high-resolution image. The second algorithm didn't prove useful in the case of low-resolution images.

Similarly (Qurat-Ul-Ain et al., 2021) proposed the use of ELA along with multiple state of art models and found out that VGG 16 yields the best accuracy among all state of art models with a training accuracy of 91.97%.

The only issue with ELA is that works only on lossy compression.

2.5.2 CNN

DL combines the extraction and classification phases of features (low, mid, and high level). The method is data-driven and can learn abstract and sophisticated traits, which are required to detect tampered areas. Furthermore, it saves the effort and time required to locate forged characteristics in manipulated photos. Deep learning model training, on the other hand, is difficult and takes a lot of computer power and a lot of data. Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), and Deep Neural Networks (DNN) are examples of DL models. Among these DL models, Convolutional Neural Networks (CNN) are popular. The convolution layer on CNN functions as a feature extractor and a discriminator.

By suppressing image content, (Bayar and Stamm, 2016) introduced a novel convolutional layer that can learn the image’s tampered properties. Because manipulation changes some local relationships, this layer calculates the relationship between pixels in local structure rather than the content of a picture. This can detect several instances of image tampering. Any detection

approach has the drawback of being unable to provide appropriate results for multiple tampering attacks. Furthermore, the majority of the study is focused on JPEG images, with the tampered region being recognised utilising a clue given by the number of JPEG compression processes.

(Zhang et al., 2016) proposed a two-step detection mechanism for faked photos. They partitioned the image into patches initially and then used a Stacked Autoencoder model to learn the features for each patch. The contextual information is added to each patch in the second stage to ensure correct results.

(Salloum et al., 2018) proposed the use of Multi-task Fully Convolutional Network (MFCN) for image splicing localization, it uses 2 branches of multitask learning. One branch is used to learn the surface label, while the other branch is used to learn the edge or boundary of the spliced region. The base network architecture is the FCN VGG-16 architecture with skip connections. Although there was a considerable performance decrease in post-processing processes, this strategy still outperformed several existing solutions.

(Amerini et al., 2017) present a multi-domain convolutional neural network strategy to localise double JPEG compression. One CNN in the spatial domain, one in the frequency domain, and completely connected layers make up a multi-domain CNN. The input to a spatial domain-based CNN is n*n sized patches of RGB colour channels. It is made up of two completely linked layers and two convolutional blocks. Based on the frequency domain The DCT coefficients for each patch are sent into CNN. Based on the frequency domain Two convolutional layers, two pooling layers, and three complete connections make up CNN. Multi-domain CNN combines the outputs of these two networks' fully linked layers and classifies the patch into one of three categories: uncompressed, single, or double compressed.

When the image size is small and the image is compressed, detecting median filtering from it is a difficult operation. (Chen et al., 2015) presented a CNN-based strategy for extracting median filtering residuals from images to address this problem. The first CNN layer is a filter layer that decreases interference caused by the presence of edges and textures. The interference is removed, allowing the model to analyse the traces left by median filtering. The method was evaluated using a dataset of 15352 images that was created by combining five image datasets.

The traces of numerous devices may be spliced together in a spliced image. (Bondi et al., 2017) introduced a CNN-based technique that extracts features related to camera type from image patches to detect tampering based on the traces left by different camera models. The retrieved features are analysed using a clustering technique, and the image is classified as faked or authentic depending on the results. The method was put to the test using a dataset of 2000 photos from various camera models.

(Abdalla, 2019) discussed a copy-move forgery detection using CNN architecture that employs a pre-processing layer but the results were only satisfactory on the passive forged image.

(Jalab et al., 2019) focussed on increasing the accuracy of image splicing detection as well as also on the reduction of feature vector dimensionality by designing a new fractional texture descriptor using DWT. DWT breaks down an input image into multiple sub-images by applying a low and high pass filter. SVM was finally used as a classifier.

(Jaiswal and Srivastava, 2019) used a pre-trained state of art model RESNET-50 for training the model and then further used Multiclass Model using SVM Learner, K-NN, and Naïve Bayes as a classifier which produced accuracy as 70.26%, 59.91%, and 59.91% respectively.

(Kuznetsov, 2019) proposed the use of VGG 16 Convolution Neural Network for identifying image forgery-splicing. The author suggested feeding network architecture patches of an image from the original image and on the border of image splicing, obtaining results with training accuracy of 97.8 % and test accuracy of 96.8%.

(Hsu et al., 2020) proposed the use of CFFN using a network-based pairwise learning model to detect fake and real images. The pairwise learning approach helped in improving the generalization property of DeepFD. The dataset was generated using state of art GAN model. The model was able to detect fake images generated by new GAN models as well.

(Walia, 2021) proposed use of two streams one handcrafted and other deep features.

One stream uses discrete cosine transform of the image to compute Markov-based features. Another stream uses the luminance channel of YCbCr colorspace for feature extraction. CASIA v1 and CASIA v2 datasets were used. The accuracy achieved is 99.3% using the proposed fusion-based approach.

(Samir et al., 2020) proposed the use of the AlexNet model for the classification of image tampering. Instead of local response normalization, the model uses batch normalization and rectified linear unit (relu) was replaced by the maxout activation function and uses SoftMax as a classifier. Dataset used were NIST (Nimble 2017 Challenge Dataset), CASIA v2.0, DVMM, and CASIA v1.0 dataset. K folds classification for dividing the dataset into test and train.

2.6 Comparative study of existing forgery detections methods

Table 2.1 shows a comparative study of image forgeries techniques developed so far.

Table 2.1 Comparative study of image forgeries techniques

S. No.	Paper Title	Method Used	Pros/Cons	Year of Publication
1	Detection of copy-move forgery in digital image	DCT	Will not work in noisy image 2003	2003
2	Exposing digital forgeries by detecting duplicated image	PCA	Time Complexity	2004

	regions(Popescu and Farid, 2004)			
3	Exposing Digital Forgeries by Detecting Inconsistencies in Lighting (Johnson, 2005)	Physics Based Technique	Not helpful for detecting advanced forgery	2005
4	Identifying tampered regions using singular value decomposition in Digital image forensics (Xiaobing and Shengmin, 2008)	SVD	Will not work in highly noised and compressed image	2008
5	Fast, automatic and fine-grained tampered JPEG image detection via DCT coefficient analysis(Lin et al., 2009b)	DCT	Works well with JPEG format	2009
6	Image tamper detection based on demosaicing artifacts; Image tamper detection based on demosaicing artifacts(Dirik and Memon, 2009)	CFA	Use single feature and have low error rate	2009
7	Image forgery detection(Farid, 2009)	Geometric Based technique	Not efficient	2009
8	Fast copy-move forgery detection(Lin et al., 2009a)	Improved PCA	Works well with compressed and noisy images	2009
9	Improved DCT-based detection of copy-move forgery in images(Huang et al., 2011)	DCT	Works well with forgery caused by JPEG compression, blurring or additive white Gaussian noise	2011
10	An evaluation of popular copy-move Forgery detection approaches (Christlein et al., 2012)	DCT, DWT, KPCA, PCA	High performance and computationally effective	2012
11	An evaluation of Error Level Analysis in image forensics; An evaluation of Error Level Analysis in image forensics(Bakiah et al., 2015)	ELA	ELA showed reliability with JPEG compression, image splicing and image retouching forgery.	2015

12	Median Filtering Forensics Based on Convolutional Neural Networks(Chen et al., 2015)	CNN+MFR	Good accuracy for cut paste forgery	2015
13	A deep learning approach to universal image manipulation detection using a new convolutional layer(Bayar and Stamm, 2016)	CNN	Can work on images with multiple manipulations. High Accuracy	2016
14	Image region forgery detection: A deep learning approach(Zhang et al., 2016)	CNN	Uses two State of art tampering detection approaches	2016
15	Fake Image Detection Using Machine Learning(Afsal Villan et al., 2017)	Metadata Analysis	Reliable for detecting forgery	2017
16	Passive detection of image forgery using DCT and local binary pattern(Alahmadi et al., 2017)	DCT +LBP	Detect copy–move and splicing forgeries	2017
17	Image forgery detection by semi-automatic wavelet soft-Thresholding with error level analysis(Jeronymo et al., 2017)	ELA + Wavelength Thresholding	Works well with noisy images, not effective in handling high frequency region	2017
18	Localization of JPEG double compression through multi-domain convolutional neural networks(Amerini et al., 2017)	CNN	Works well for double compression JPEG	2017
19	Tampering Detection and Localization Through Clustering of Camera-Based CNN Features (Bondi et al., 2017)	CNN	Able to detect camera model traces from image patches	2017
20	Image Splicing Localization using a Multi-task Fully Convolutional Network (MFCN) (Salloum et al., 2018)	CNN+MFCN	MFCN helped in achieving better localization compared to other localization techniques	2018
21	Image forgery detection using error level analysis and deep learning (Qurat-Ul-Ain et al., 2021)	ELA + Metadata Analysis	High Accuracy	2019

22	Convolutional Neural Network for Copy-Move (Abdalla, 2019)	SVCNN	Improved result for active copy move forgery but only satisfactory results for passive forgery	2019
23	New texture descriptor based on modified fractional entropy for digital image splicing forgery detection(Jalab et al., 2019)	CNN+AMFE	Superior detection accuracy and positive and false positive rates were achieved	2019
24	Image Splicing Detection using Deep Residual Network (Jaiswal and Srivastava, 2019)	RESNET	Low Accuracy	2019
25	Digital image forgery detection using deep learning approach (Kuznetsov, 2019)	VGG16	High Accuracy	2019
26	Passive Image Forgery Detection Based on the Demosaicing Algorithm and JPEG Compression(Armas Vega et al., 2020)	CFA	Highly efficiency	2020
27	Deep fake image detection based on pairwise learning(Hsu et al., 2020)	CNN	High Accuracy on GAN generated dataset	2020
28	Optimization of a pre-trained AlexNet model for detecting and localizing image forgeries(Samir et al., 2020)	AlexNet +Batch Normalisation	Moderately effective	2020
29	Forged Face Detection using ELA and Deep Learning Techniques (Qurat-Ul-Ain et al., 2021)	ELA +CNN	Accurate and efficient	2021
30	Fusion of Handcrafted and Deep Features for Forgery Detection in Digital Images (Walia, 2021)	DCT + YCbCr colorspace	High Accuracy	2021

2.7 Summary

We covered some basics regarding digital photography, image formats, and file compression in this chapter. We also covered some of the strategies and procedures used to counterfeit digital photographs, as well as some of the techniques and methods used to identify image forgery. In Chapter 3, we'll go into research methodology, including several key tools for detecting picture forgeries and a convolutional neural network approach for classifying bogus and authentic photos.

CHAPTER 3: RESEARCH METHODOLOGY

3.1 Introduction

Deep learning is widely used in the field of image classification and recognition. For this work, images need to be classified as real and fake. CNN will be taking images as input and processed through various layers for feature extraction. Technically, images are passed through multiple CNN layers and filters. Thus, features extracted via the CNN network are used for training and testing purposes. Test data is used to predict the classes based on the model build using train data.

3.2 Methodology

The purpose of this research is to design and build a model that can effectively and efficiently be able to identify image forgery. We are using CASIA V2.0 dataset for model training and test purpose.

For the model building, we will be using CNN (State of art model using transfer learning) and ELA for feature extraction. Below is the block diagram to represent a flow of information and steps taken for the final outcome.

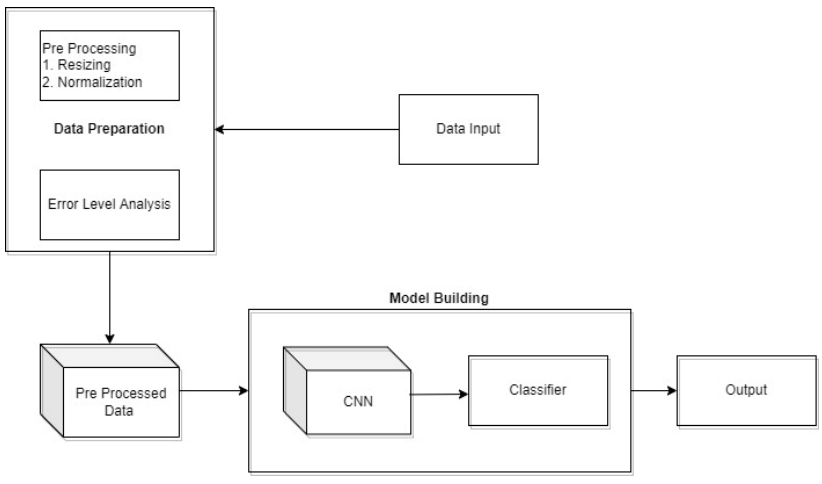


Figure 3.1 Proposed Approach

3.2.1 Data Set Description

CASIA V2 is a dataset for forgery classification. It contains a total of 12323 coloured images which are divided into 7491 original images and 5123 tampered images. It contains images from 320X240 to 800X600 pixel images. It contains images with JPEG, TIFF, and BMP formats. Tempered images are mostly splicing using Photoshop as well as blurred images, are also introduced. Original images contain an image of the following 9 categories: scene, article, animal, nature, character, architecture, plant, texture, and indoor.

Forged images were created using an original image or multiple original images. Region cropped from original is subjected to rotation, resize, or other distortions before pasting on another image. Figure 3.1 show some examples of images present in the CASIA V2.0 dataset.

Figure 3.2 Example of images captured from the CASIA v 2.0 dataset

Below is the statistical information about the forged/tampered image.

Table 3.1 Statistical information about the Tampered image (Jing Dong, 2013)

Category		**No of Images**
JPEG Format		2064
TIFF Format		3059
Source of tampered region	Same Image	3274
	Different Image	1849
	Rotation	568
	Resize	1648
	Distortion	196
	Rotation and resize	532
	Resize and Distortion	211
	Rotation and Distortion	42
	Rotation, Distortion, and Resize	83
Manipulation without pre-processing		1843

Manipulation with post-processing	Blurring along spliced edges	848
	Blurring on other regions	131
Manipulation without post-processing (Blurring)		4144
Size of Tampered Region	Small	3358
	Medium	819
	Large	946

3.2.2 Pre-Processing

Preprocessing of image: This is a very important step when dealing with image classification problems. So firstly, we will be resizing the image and checking if any rotation or orientation change is required for any image. Secondly, we will be applying normalization on the image because CNN learns by back-propagating through various weight matrices throughout the network, thus when the learning rate is applied on all the images then each correction will different each image proportionately.

Another important thing that needs to be kept in mind is the removal of noise. Denoising is a very important step before any further action is taken.

3.2.3 Proposed Method

We will discuss the deep learning that can be used for identifying the fake and real images

3.2.3.1 Convolutional Neural Network (CNN)

A convolutional neural network is a form of ANN that analyses image inputs and has learnable weights and biases for different regions of the image that may be separated from each other. Convolutional Neural Networks have the advantage of using Spatial and Temporal dependencies of the input images, allowing them to have reduced numbers of weights because of the reusability of some parameters. (Tammina, 2019)

In terms of memory and complexity, this procedure is efficient. Figure 3.3 shows the basic architecture of a convolutional neural network with each component highlighted.

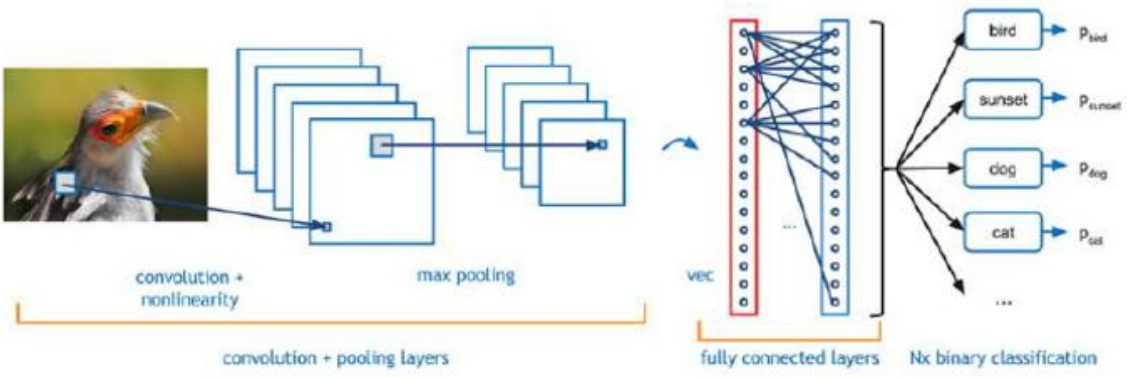


Figure 3.3 Architecture of Convolutional Neural Network

The following are the basic components of a convolutional neural network:

Convolution Layer: A kernel matrix is passed through the input matrix to build a feature map that is used by the following layer in a convolutional layer. By sliding the Kernel matrix over the input matrix, we perform a mathematical action known as convolution. Every location performs element-by-element matrix multiplication and the results are added onto the feature map. (Nielsen, 2015)

The convolution operation is frequently utilized in a range of fields such as image processing, statistics, and physics. Convolution is a technique that can be used on more than one axis. The convoluted image is calculated as follows if we have a 2-Dimensional image input, I, and a 2-Dimensional kernel filter, K:

$$S(i,j) = \sum\sum I(m, n)k(i - m,j - n)$$

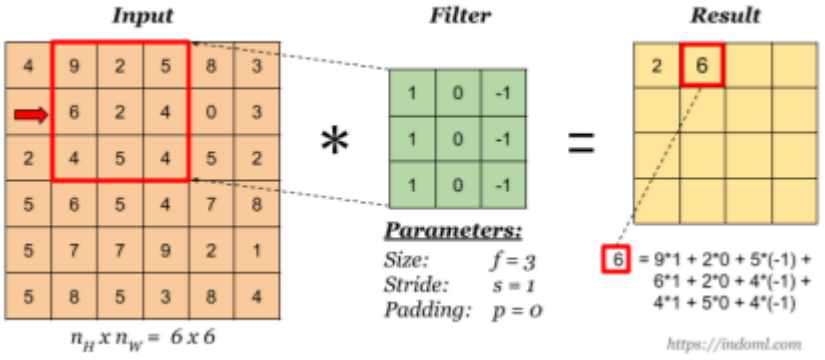


Figure 3.4 Element-wise multiplication and summation in CNN

Pooling layer: Like the Convolutional Layer, the Pooling layer, is responsible for reducing the spatial size of Convolved features. The computational power reduces with the reduction in dimensionality. It's also beneficial for extracting rotational and positional dominant features, which helps keep the model's training process smoothly. (Gitbook, n.d.)

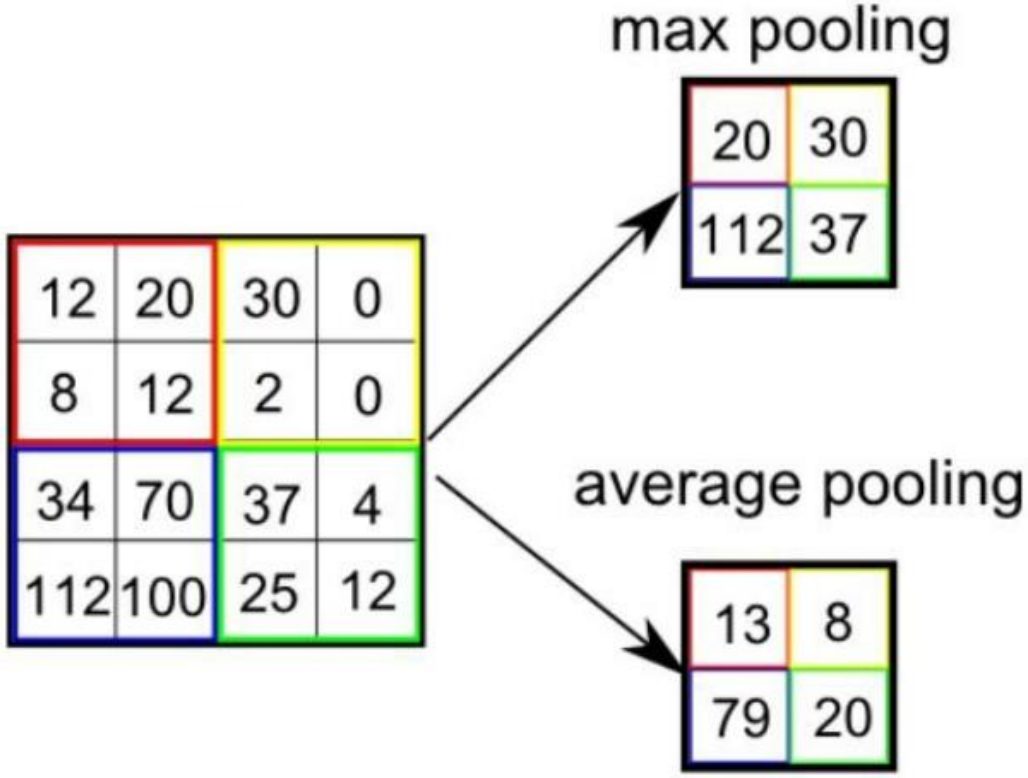


Figure 3.5 Pooling Types

As shown in Figure 3.5 pooling is categorized into two types: maximum pooling and average pooling. The maximum value from the portion of the image covered by the Kernel is returned by Max Pooling. Average Pooling, on the other hand, returns the average of all the values from the Kernel's section of the image.

Max Pooling also works as a Noise Suppressant. It performs de-noising and dimensionality reduction at the same time. Whereas Average Pooling reduces dimensionality as part of a noise-suppressing strategy. Thus, we can say that Max pooling is better than Average pooling. (Gholamalinezhad and Khosravi, 2020)

Fully Connected Layer: The final layer in a convolutional architecture is a fully linked layer. Following the convolution and pooling layers, it is the final layer. As shown in Figure 3.6, in typical neural networks, each neuron in the hidden layer is connected to all neurons in the previous layer, therefore a completely connected layer has neurons that connect to the whole input volume.

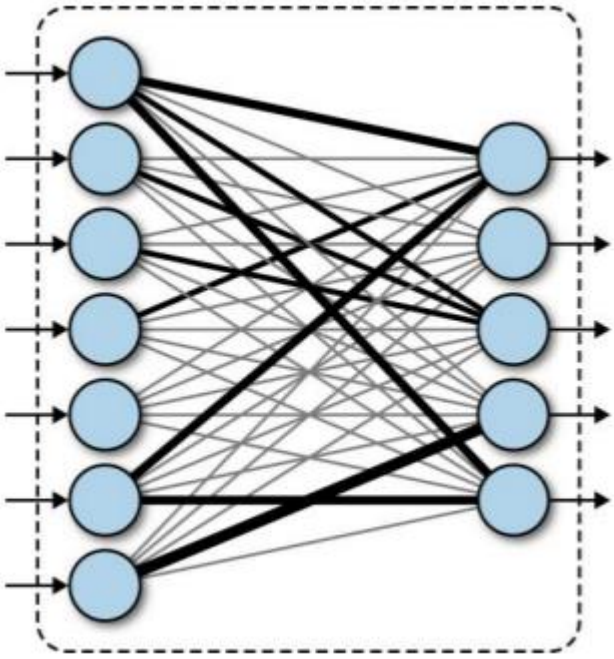


Figure 3.6 Fully Connected Layer (Tammina, 2019)

3.2.3.2 Transfer Learning

In this section, we will discuss various methods that have been used and have been proved helpful. We will discuss two methods, one with the use of a custom model building and another one with the use of transfer learning. (Qurat-Ul-Ain et al., 2021)

VGG -16:

VGG-16 is a well-known image classification model that was trained for the ImageNet Challenge (Russakovsky et al., 2015) with a subset of 1000 classes. The Visual Geometry Group at the University of Oxford proposed the VGG architecture, which is defined by a stack of convolutional layers preceding a Max Pooling layer. It has 13 convolutional layers and 3 FC layers, as well as 3X 3 and 1X1 filters with a stride of 1 pixel. (Simonyan and Zisserman, 2015)

The reasons why VGG 16 is selected as a method that can be used:

1. It has a sequential architecture
2. Recently, in many papers, it has been shown that VGG 16 has been proved useful in identifying fake images using different tampering styles e.g., copy-move, and splicing.
3. Although there is a large number of parameters and long inference time than other architectures like Alex Net, Inception, or VGGNet, VGG-16 can be pruned without any significant change in performance.

The architecture of VGG 16

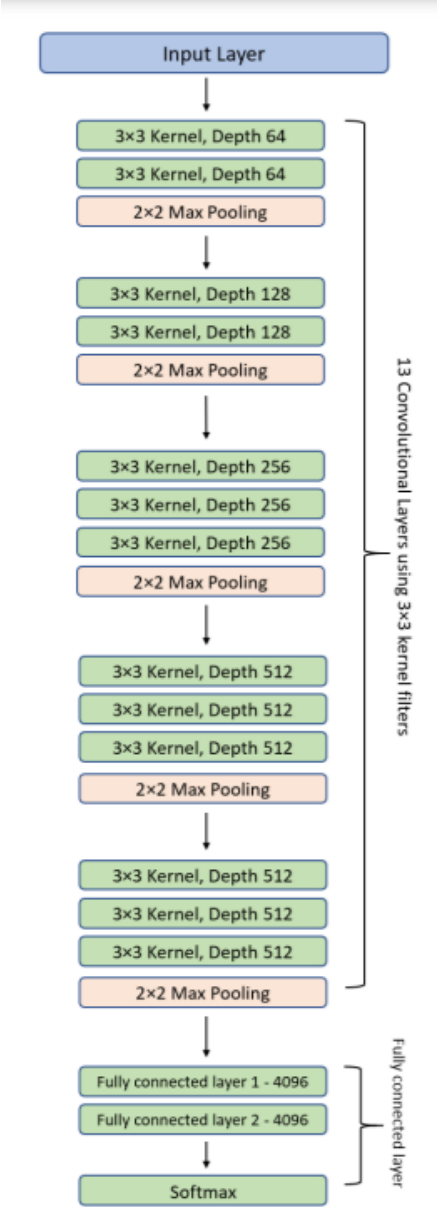


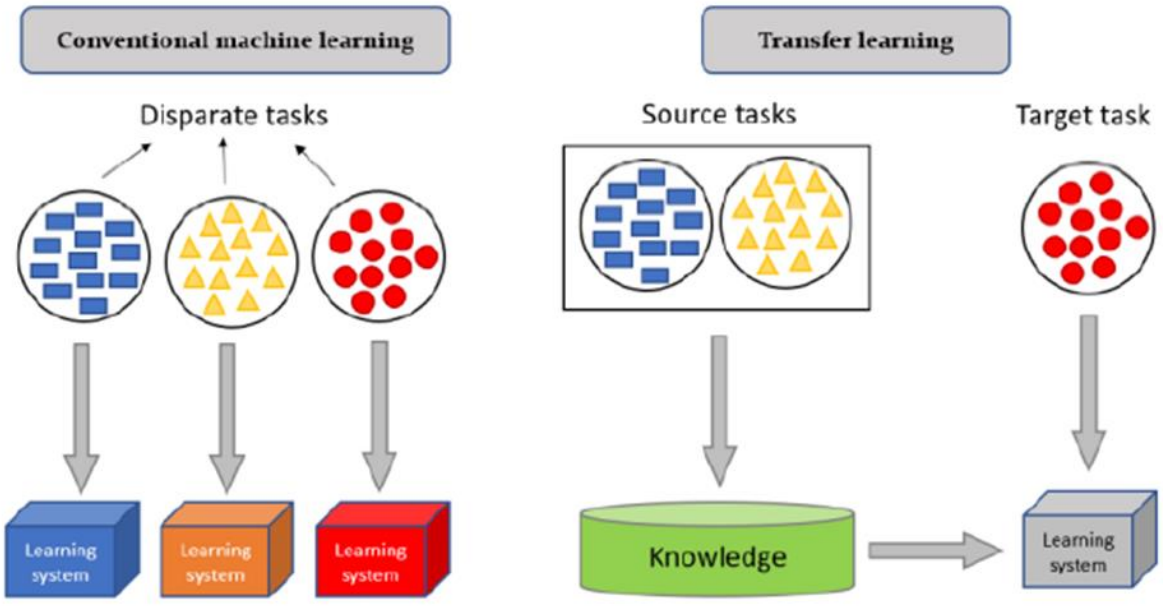
Figure 3.7 VGG Architecture (Tammina, 2019)

- The first and second convolutional layers are made up of 64 feature kernel filters, each of which is 33 pixels in size. The dimensions of the input picture (RGB image with depth 3) change to 224x224x64 as it passes through the first and second convolutional layers. The output is then sent to the max-pooling layer with a stride of two.

- The 124 feature kernel filters in the third and fourth convolutional layers have a filter size of 33%. After these two layers, a max-pooling layer with stride 2 is applied, and the output is shrunk to 56x56x128.
- Convolutional layers with a kernel size of 33 are used in the fifth, sixth, and seventh levels. 256 feature maps are used in all three. These layers are followed by a stride 2 max pooling layer.
- There are two sets of convolutional layers with kernel sizes of 33rd and thirteenth. There are 512 kernel filters in each of these convolutional layer sets. Following these layers is a max pooling layer with a stride of 1.
- The fourteen and fifteen layers are 4096-unit fully connected hidden layers, followed by a 1000-unit SoftMax output layer (sixteenth layer).

Leveraging Transfer Learning with Pre-Trained Model

There are numerous options for employing pre-trained models in transfer learning; one of them replaces part of the original architecture's layers while retaining the others. For example, the VGG-16 architecture can be used until the block4 pool layer, and that can then be connected to fully-connected layers with an outputs layer with two outputs. The proposed architecture will include pre-trained (frozen) parameters for the layers 1 to layer 4 pools, that will later connect to a new FC layer.



3.8 Comparative diagram of Conventional Machine Learning and Transfer Learning

3.2.3.3. Error Level Analysis

Error Level Analysis (ELA) is a technique developed by (N.Krawetz, 2007) which makes use of lossy compression present in forged images. Original image has its own property and any tampering done on these original images will certainly leave some traces of forgery and traces are used by ELA.

In a nutshell, ELA works by taking a lossy image and recompressing it with a known error rate, after which it computes the absolute difference between the analyzed and recompressed image. The following is a formal definition of ELA:

Error levels, $\mathrm{ELA}(\mathrm{n}_1, \mathrm{n}_2)$ where n1 and n2 are row and column indices, can be represented by

$$ELA(\mathrm{n}_1, \mathrm{n}_2) = |X\ (\mathrm{n}_1, \mathrm{n}_2) - \mathrm{X}rc(\mathrm{n}_1, \mathrm{n}_2)|$$

for each color channel, where X is the image suspected of forgery and X rc is the recompressed image.

Total error levels are error levels averaged across all color channels, as in

$$ELA(\mathrm{n}_1, \mathrm{n}_2) = \frac{1}{3}\sum_{\mathrm{i}=1}^{3} |X\ (\mathrm{n}_1, \mathrm{n}_2, \mathrm{i}) - \mathrm{X}rc(\mathrm{n}_1, \mathrm{n}_2, \mathrm{i})|$$

Where, i = 1, 2, 3, for a RGB image.

The error levels linked with the actual pixels show the difference between images; these error levels, expressed as a percentage change, are directly related to compression loss. The pixel has attained its local minima for error at the stated error rate if the amount of change is modest. If there is a significant amount of change, the pixels are likely not at their local minima and are foreign.

3.2.4 Classification

We will try using different classifiers such as SVM, XGBoost, Random Forest, and Naïve Bayes. The classifier that yields max accuracy will be considered for the final model.

3.2.5 Evaluation Metrics

For image forgery detection, the significant way of evaluating the performance of a model is by measuring how effectively the model was able to detect local tampered regions in an image. The whole model works as a classification problem that segregates fake and real images. Evaluation metrics selection should always be selected based on certain justification; it should not be randomly selected. Since ground truth image is not always available, evaluating on pixel level is difficult, thus we will evaluate on an image level. (Al-qershi and Khoo, 2018).

- TP (True Positive): Tampered images detected correctly as tampered images.
- FP (False Positive): Real images detected wrongly as tampered images.
- FN (False Negative): Tampered images falsely missed as real images.
- TN (True Negative): Real images detected correctly as real images

So, for image-level detection, we will make use of the metrics such as the Sum of Absolute Differences (SAD) or Sum of Squared Differences (SSD). Whereas, for pixel-based evaluation, we will use Precision, recall and F1-Score for evaluation purposes.

Precision can be defined as the probability of tampered pixel identified correctly to total tampered pixels in the ground truth image.

$$Precision = \frac{TP}{TP + FP}$$

Recall can be defined as the probability of the tampered pixels identified correctly among total pixels present in ground truth image.

$$Recall = \frac{TP}{TP + FN}$$

F_ Measure is the harmonic mean of precision and recall.

$$F - Measure = \frac{2X\ Precision\ X\ Recall}{Precision + Recall}$$

Accuracy can be defined as the probability of total pixels identified correctly.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

3.3 Summary

This chapter discusses the approach that we will be following for the implementation. The pre-trained model will be used for classifying images between real and fake images. In the next chapter, we will discuss the implementation and will identify the performance of the model using accuracy as a measure of performance.

CHAPTER 4: ANALYSIS AND IMPLEMENTATION

4.1 Introduction

The main intention of this chapter is to discuss the implementation of the model that can help in detecting image forgeries. We are classifying images into 2 categories, real and fake. In the previous chapter, we discussed in detail the methods and several tools that we are going to use for the forgery detection model. We also discussed existing models available.

For this report, we are proposing the approach as described in the Figure 4.1 flow chart. For the implementation of forgery detection, we are using the CNN (State of art) model and we are using the CASIA V2.0 dataset for the training purposes.

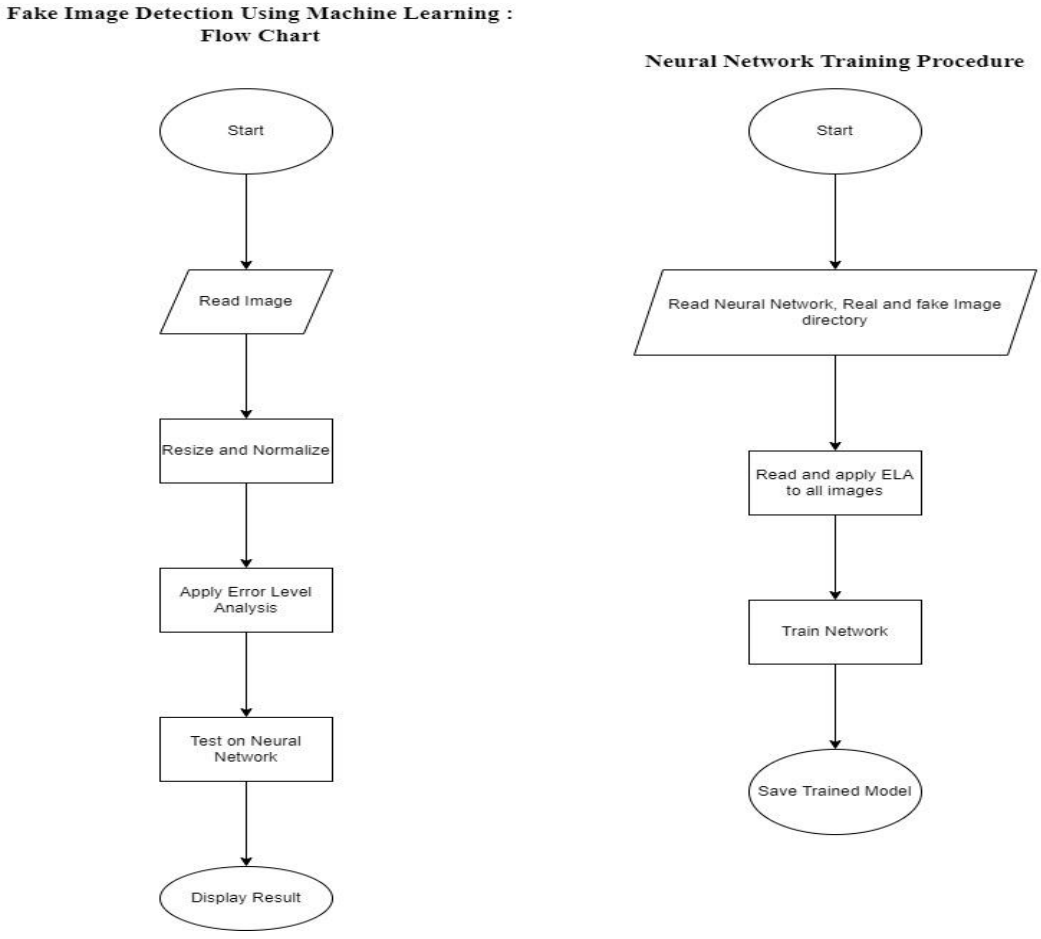


Figure 4.1 Flow chart for final proposed approach of training the model

Section 4.2 covers the dataset description. Section 4.3 covers exploratory data analysis performed on the CASIA dataset. Section 4.4 covers image preprocessing that includes image

augmentation, denoising, K fold CV, and ELA. Section 4.5 covers details regarding final model building using CNN that classify images into fake and real.

4.2 Data set description

CASIA V2 is a dataset for forgery classification. It contains a total of 12323 colored images which are divided into 7491 original images and 5123 tampered images. It contains images from 320X240 to 800X600 pixel images. It contains images with JPEG, TIFF, and BMP formats. Tempered images are mostly splicing using Photoshop as well as blurred images, are also introduced. Original images contain an image of the following 9 categories: scene, article, animal, nature, character, architecture, plant, texture, and indoor.

CASIA V2.0 dataset contain two folders: Au and Tp, Au contains all the Authentic images and Tp folder contains all the tampered images.

The naming convention used in the CASIA V2.0 dataset is as below:

1. Authentic images:

e.g., **Au_ani_00001.jpg** Au: Authentic ani: animal category other categories: nat (nature), arc (architecture), pla (plants), art, cha (characters), ind (indoor), txt (texture)

2. Tampered images:

a. Spliced image

Tp_D_CRN_S_N_cha00045_art00013_11817.jpg

Tp: Tampering

D: Different (means the tampered region was copied from the different image)

cha00045: the source image

art00013: the target image

11817: tampered image ID

b. Copy-move images

Tp_S_NRN_M_N_pla00020_pla00020_10988.jpg

Tp: Tampering

S: Same (means the tampered region was copied from the same image)

And the rest is similar to case a.

4.3 EDA

Exploratory Data Analysis provides a brief description of the dataset and helps us in getting some insight about data and help us in answering some preliminary questions. EDA also helps in the model building by providing some guidance e.g., in case of classification problem, the number of classes will help in identifying activation functions to be used. We have performed some EDA on the CASIA V2.0 dataset.

4.3.1 Raw Image comparison

Firstly, we started with plotting a few randomly sampled images. Figure 4.2 shows a screenshot of random images captured.

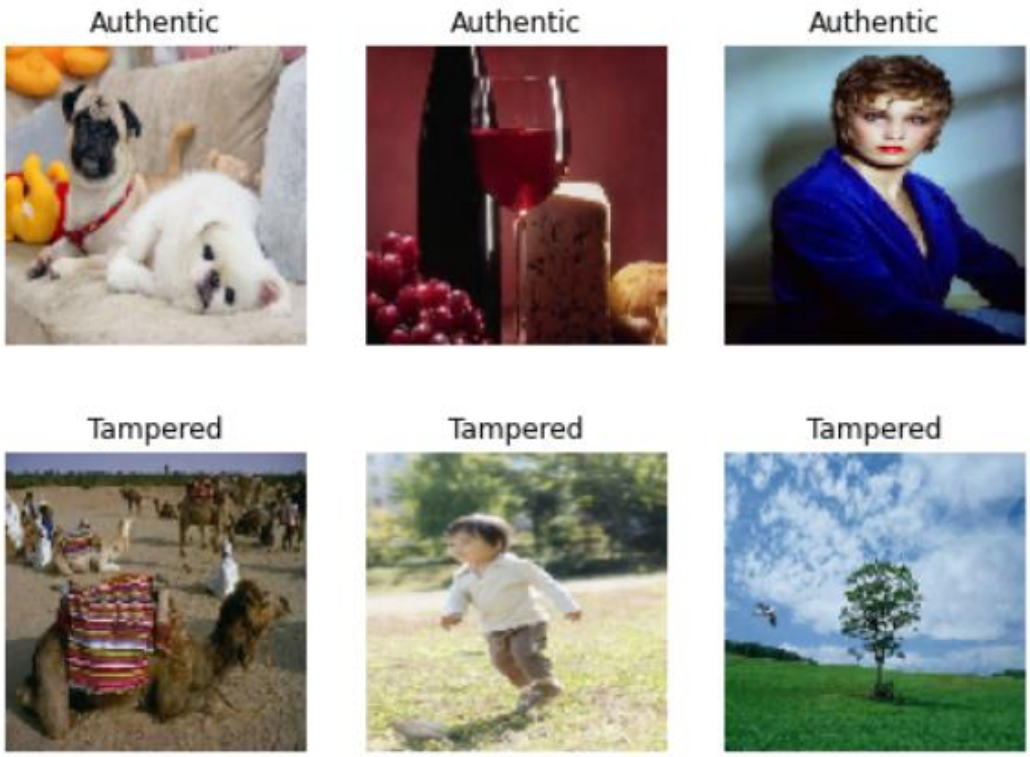


Figure 4.2 Samples of Authentic and Tampered images

4.3.2 Average Image and Contrast:

We tried to see what an average image looks like for both real and fake image. To find the average we have taken an average value of all pixel across the image. Using these average

values, we have tried to see a contrast between average real and fake images. Average images do not show much difference over grayscale. Figure 4.3 shows the result of averaging real and fake images and Figure 4.4 shows a contrast between average images.

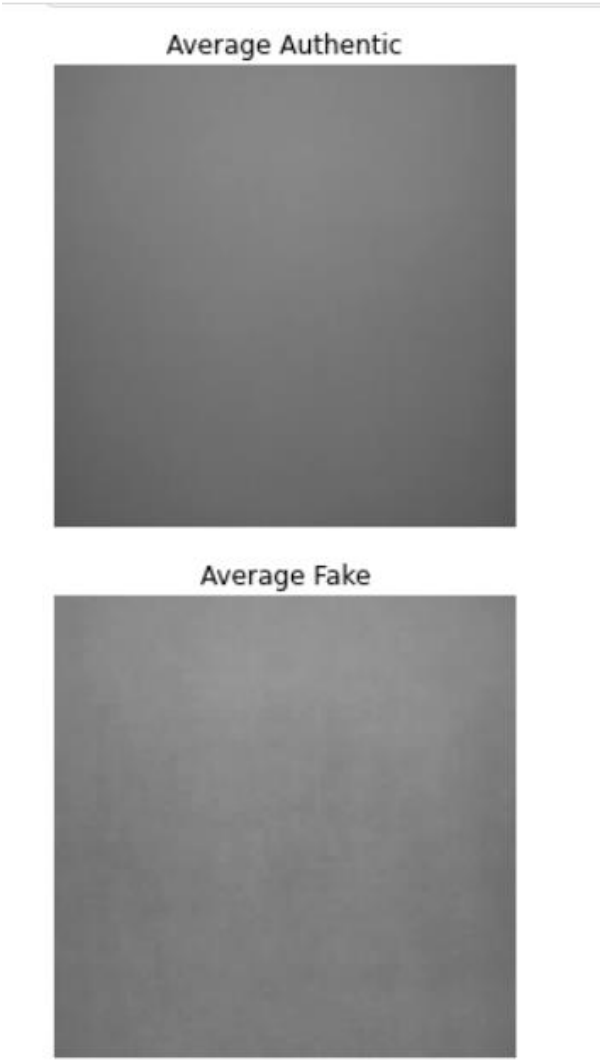


Figure 4.3 Average image of fake and real images

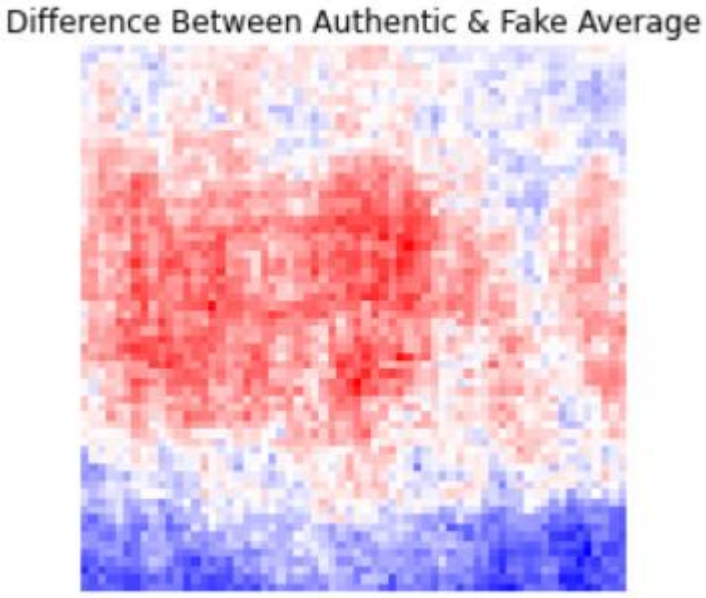


Figure 4.4 Contrast between real and fake images average values

4.3.3 Eigenimages:

Finally, we can visualize the components that best define each class using a dimension reduction technique such as principal component analysis (PCA). The eigenimages of our image matrix, which are effectively the eigenvectors (components) of PCA, can be reshaped into a matrix and plotted. Because this approach was first employed for facial recognition research, it's also known as eigenfaces. For each class, we will see the primary components that account for 70% of the variability. Figure 4.5 and 4.6 shows eigen images of real and fake images.

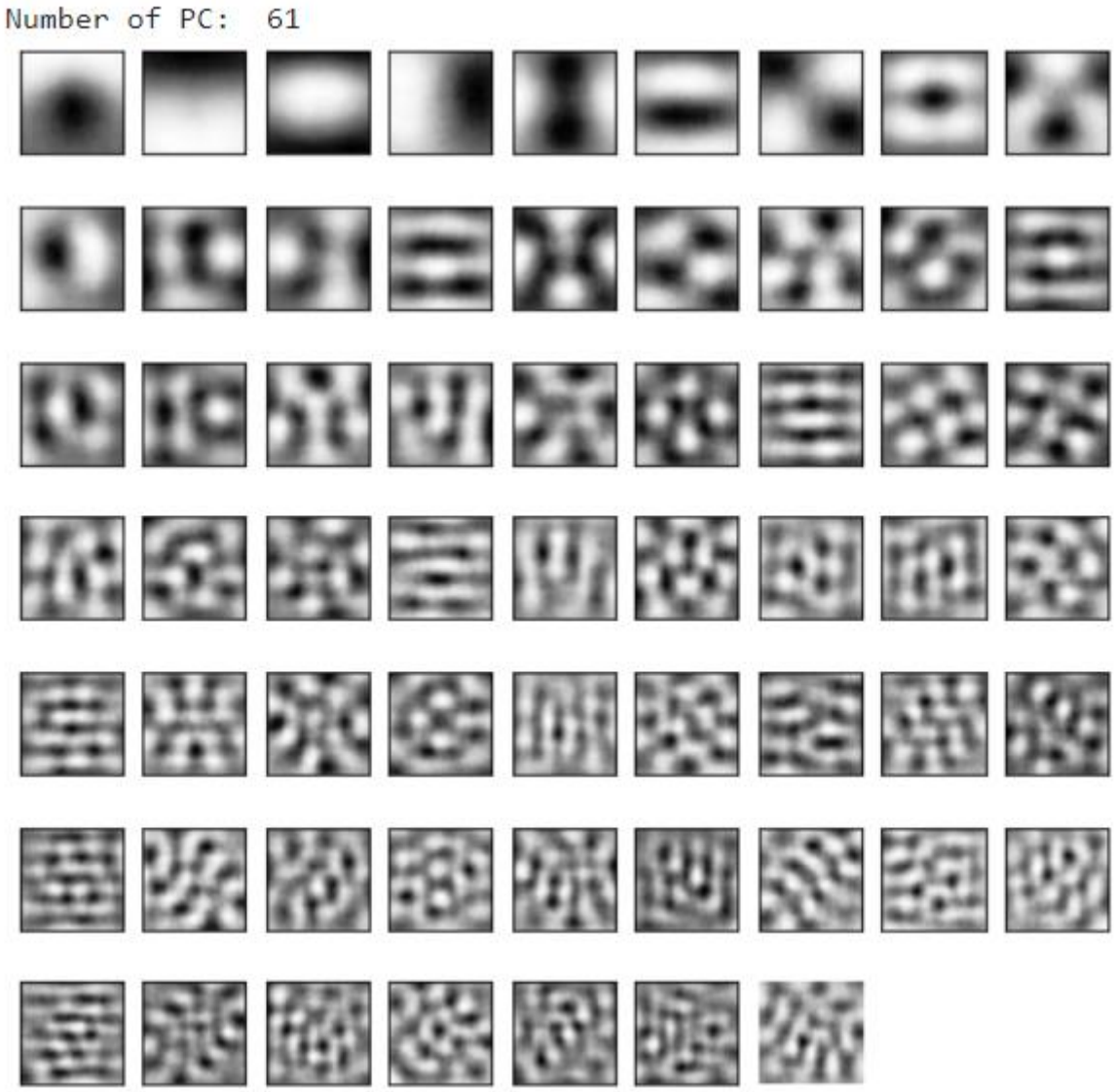


Figure 4.5 Eigen Images for Real Images

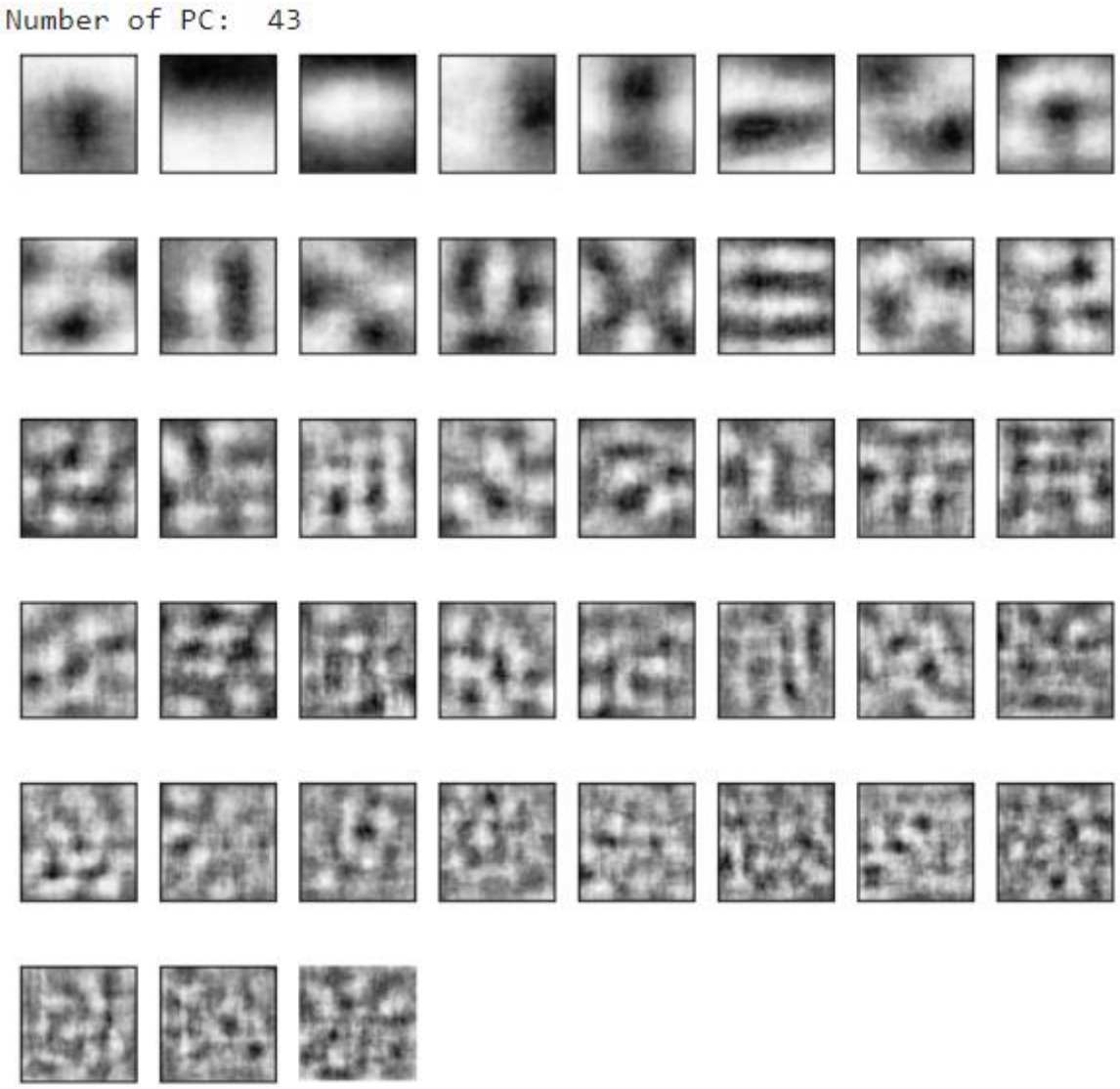


Figure 4.6 Eigen Images for Fake Images

4.3 Pre-Processing

In the architectural design of any model building, there are two major building blocks, one is a data processing and another is model building. In this research study very important data processing step was converting images into ELA images and then the images are resized to 128*128 shape. The next step after image resizing was normalization that we achieved by dividing each rgb value of an image by 255. Later we created labels for the dataset by

marking all the images from the Au folder as 1 and all the images from the Tp folder as 0 indicating real and fake images respectively. Finally, we divided the data into train and test with 80% data for training and 20% for test purpose.

4.3.1 Image Augmentation

Image data generator:

Data Augmentation is a great technique that can be used to increase dataset size. By using ImageDataGenerator we can create multiple images. This helps in building a model more robust and also reduces chances of overfitting.

ImageDataGenerator () in Keras is used to convert image files into preprocessed tensors that can be fed directly into models during training. To create new versions of existing images, image transformation operations such as rotation, translation, and zooming are applied to the training dataset. During training, we added these additional photos to our model.

Few options that are available for DataImageGenerator () are as follows:

- Rotation_range is a degree range (0-180) within which to rotate photos at random.
- Width_shift and height_shift are ranges (as a percentage of total width or height) within which images can be translated vertically or horizontally at random.
- Rescale is like we scaled them down to values between 0 and 1.
- Shear_range is used to apply shearing transformations at random.
- Zoom_range is used to zoom in and out of images at random.
- Horizontal_flip is used to flip half of the photos horizontally at random, which is useful when there are no horizontal asymmetry assumptions (e.g., real-world pictures).
- fill_mode method is used to fill in newly produced pixels after a rotation or a width/height change

Implementation

Figure 4.7 shows a code snippet of ImageDataGenerator of how it is implemented.

```
#Initializing Data Generators
train_datagen = ImageDataGenerator(rotation_range=40,
                                   width_shift_range=0.2,
                                   height_shift_range=0.2,
                                   rescale = 1./255,
                                  shear_range = 0.2,
                                  zoom_range = 0.2,
                                  horizontal_flip = True)
validation_datagen = ImageDataGenerator(rescale = 1./255)
```

Figure 4.7 Code snippet of Data generator used

The above function performs the following task:

1. Decode image to RGB pixels.
2. Converting them to floating point tensors.
3. Normalizing the image by rescaling.
4. Apply sheer transformation.
5. Zooming the image by the factor of 0.2.
6. Apply horizontal flip using horizontal-flip

Results

Image populated by applying to perform various operations using the ImageDataGenerator function is shown in Figure 4.8.

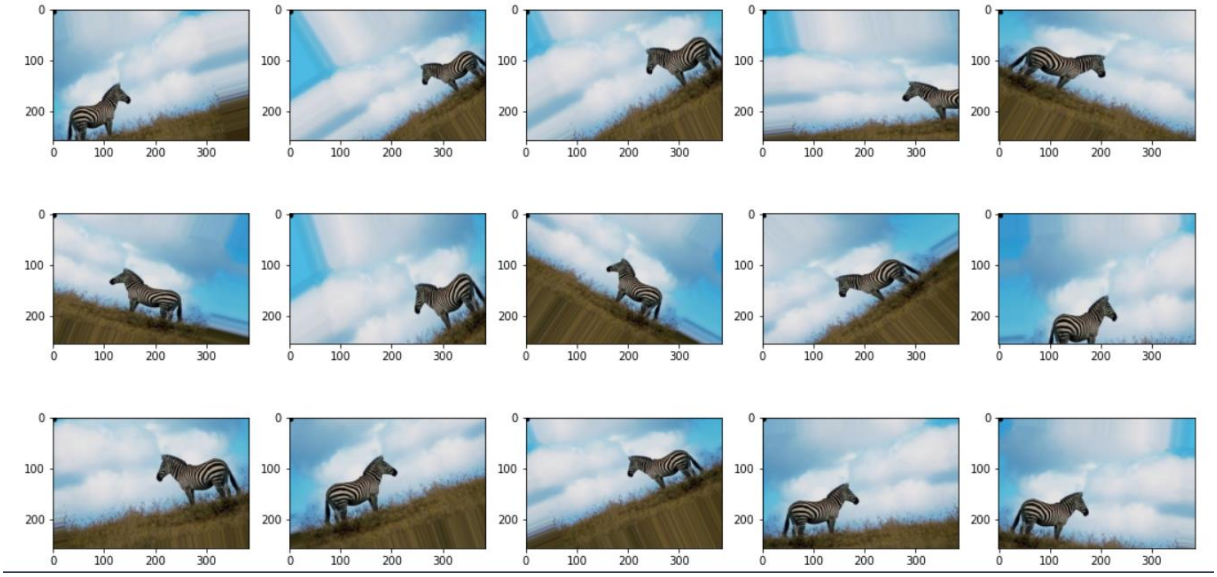


Figure 4.8 Result of Data Augmentation performed on one sample image

Model built using CNN starts to show overfitting after few epochs only, so to avoid overfitting we use data augmentation. This makes the model built robust and reliable. Data augmentation is really helpful when we are dealing with a small dataset.

4.3.2 Denoising

Image Noise: It is undesired by-product of an image that hides the relevant information of the image that is obtained by the random changes in brightness or color.

Image noise is generally introduced during capturing of image, sharing or during coding or processing.

There are four major types of image noises (Fan et al., n.d.):

1. **Gaussian Noise:** The probability Density Function (PDF) for this type of noise is equal to Normal Distribution. These noises are usually created during image acquisition due to sensor noise caused by high temperature and poor illumination and during transmission because of electronic circuit noise.
   For Gaussian random variable (Z), the Probability Density Function is defined as:

$$P_G(Z) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(z-\mu)^2}{2\sigma^2}}$$

   Where μ is the mean value and σ is the standard deviation

2. **Salt and pepper:** Also known as Data drop-out. It is fixed value impulse noise and has only two values

$$PDF_{\text{salt\&pepper}} = \begin{cases} A \; for \; g = a(''pepper'') \\ B \quad for \; q = b(''salt'') \end{cases}$$

Figure 4.9 shows the PDF for salt and pepper Noise.

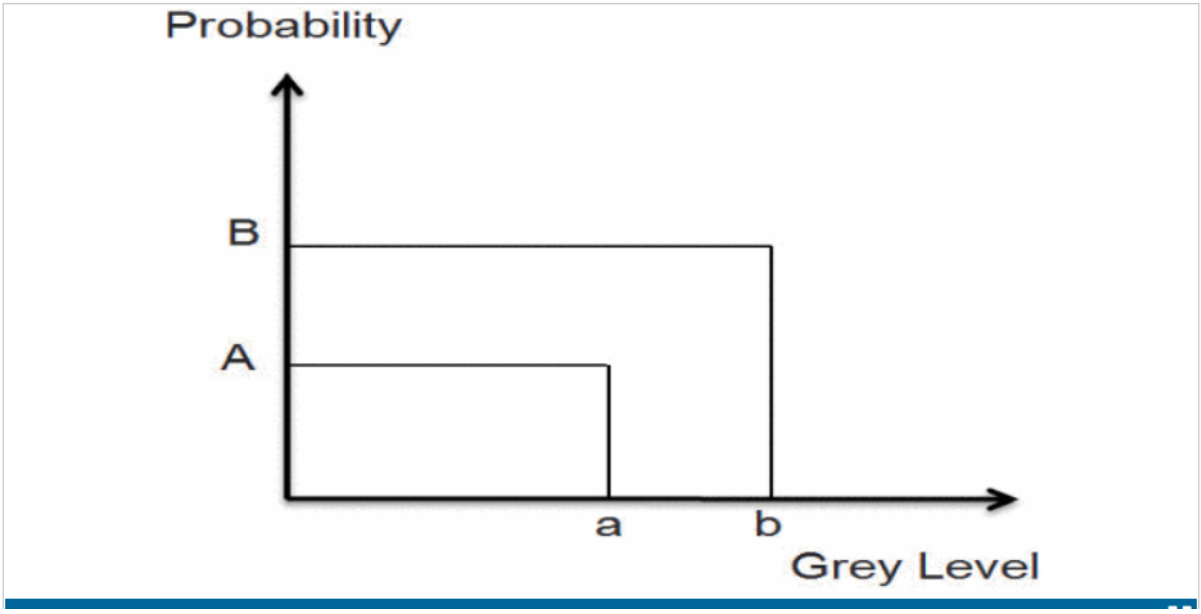


Figure 4.9 Probability Density Function for Salt and Pepper noise. (Ahmad Jawad Khan Muhammad Salah Ud Din Iqbal, 2019)

3. **Speckle Noise:** This noise occurs naturally and can be understood as random value getting multiplied to pixel values and hence corrupting the quality of an image.

4. **Poisson Noise:** Also known as shot noise. Here the Probability Density Function of such noise is same as the Poisson Distribution. It is usually caused by random fluctuations of photons.

What is image denoising?

It is a method of preserving image information while removing as much random noise as possible from the image.

We divide picture denoising filters into two categories:

1) **Traditional Filters**, which are filters that have been used to remove noise from images in the past. Spatial domain filters and Transform domain filters are subsets of these filters.

2). Filters that use fuzzy logic in their filtering operation are known as **fuzzy-based filters**.

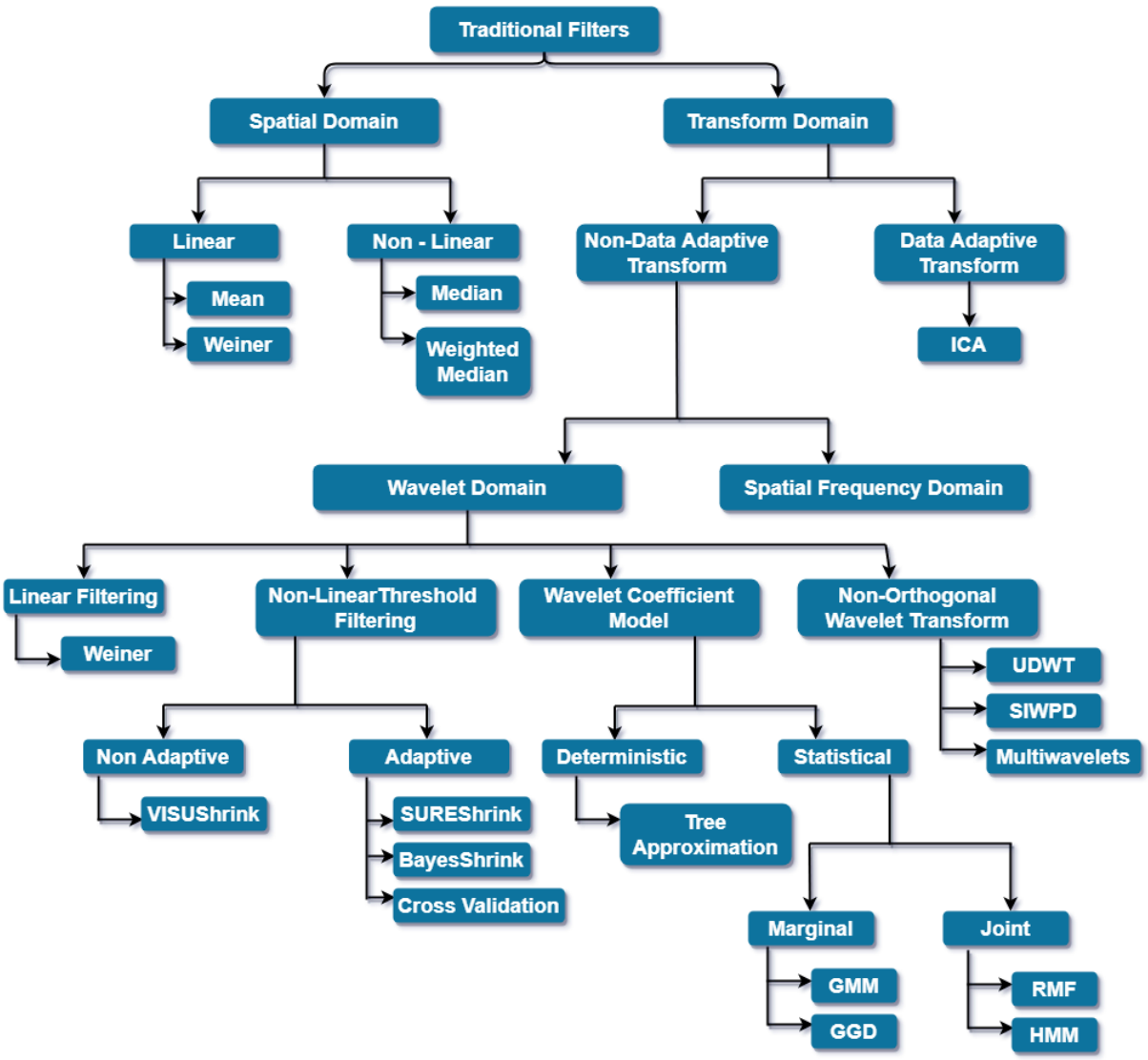


Figure 4.10 Traditional Filters for Denoising techniques (Nidhi Mantri, 2021)

Denoising Implementation:

Denoising with wavelets

The wavelet representation of the image is used in wavelet denoising. In the wavelet domain, Gaussian noise is represented by tiny values, which can be eliminated by setting coefficients below a specific threshold to zero (hard thresholding) or decreasing all coefficients by a given amount toward zero (soft thresholding).

The basic denoising methods use below steps (Bnou et al., 2020) as shown in Figure 4.11:

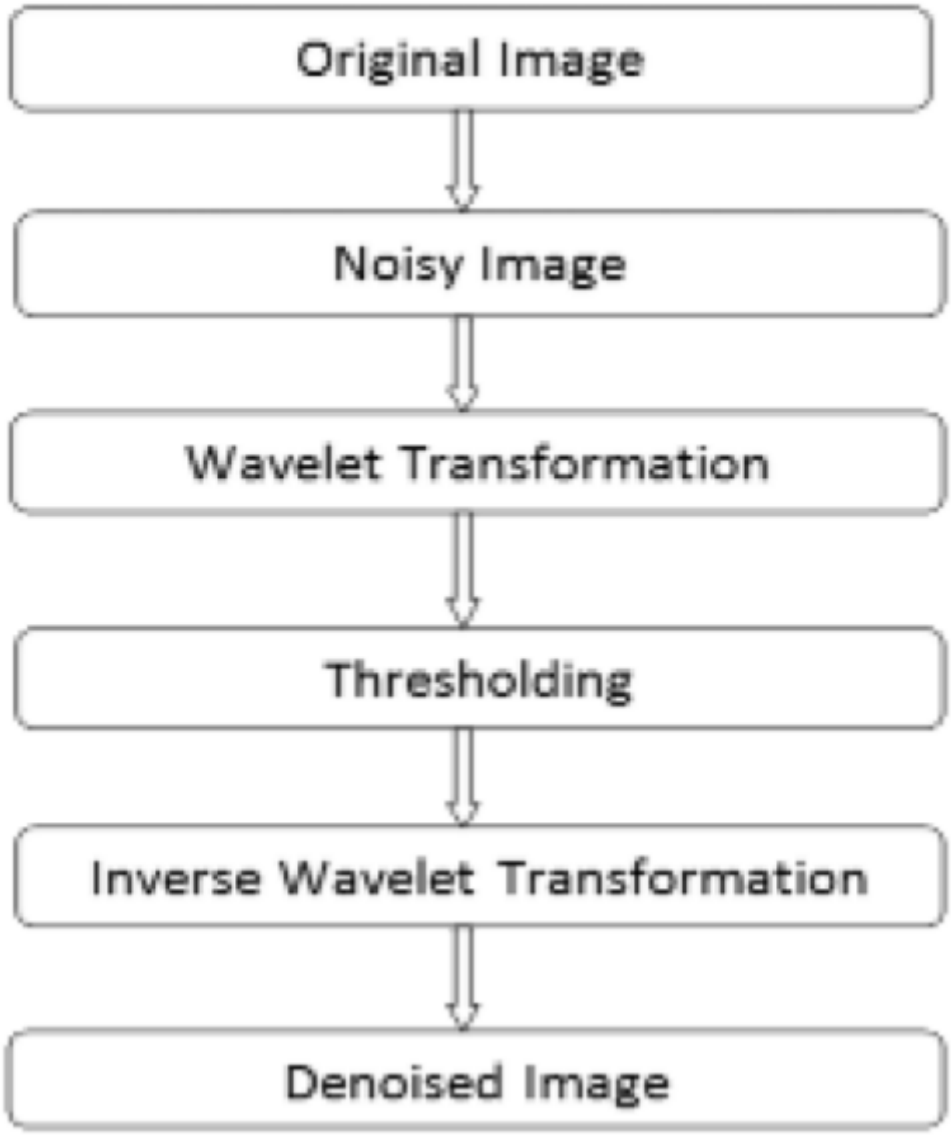


Figure 4.11 Block diagram for denoising approach (Bnou et al., 2020)

1. To decompose the noisy image and obtain the wavelet coefficients, the discrete wavelet transform is used.
2. Using the wavelet thresholding approach, the wavelet coefficients are denoised.
3. The changed coefficients are transformed using the inverse discrete wavelet transform the denoised image.

In this report, we show how to determine wavelet coefficient thresholds using two alternative methods:

1. Non-Adaptive: VisuShrink;
2. Adaptive - BayesShrink

VisuShrink

All wavelet detail coefficients are subjected to a single, universal threshold in the VisuShrink method. This threshold is intended to remove additive Gaussian noise with a high probability, which might cause a picture to appear too smooth. A more aesthetically appealing outcome can be obtained by specifying a sigma that is smaller than the genuine noise standard deviation.

BayesShrink

The BayesShrink algorithm is an adaptive wavelet soft thresholding method that estimates a unique threshold for each wavelet sub band. In most cases, this improves on what can be accomplished with a single threshold.

Code Implementation:

We have used both BayesShrink and VisuShrink as denoising methods. Since VisuShrink is designed in such a way that it reduces high probability noises and this generally results in a very smooth image. Thus, we have also tried reducing the threshold by factors 2 and 4 also to observe the impact on an image with factor reduction. Figures 4.12 and 4.13 show code snippets of denoising implementation.

```
im_bayes = denoise_wavelet(noisy, convert2ycbcr=True,multichannel=True,
                           method='BayesShrink', mode='soft',
                           rescale_sigma=True)
im_visushrink = denoise_wavelet(noisy, convert2ycbcr=True,multichannel=True,
                                method='VisuShrink', mode='soft',
                                sigma=sigma_est, rescale_sigma=True)
```

Figure 4.12 Code Snippet of denoising implementation

```
im_visushrink2 = denoise_wavelet(noisy, convert2ycbcr=True,multichannel=True,
                                 method='VisuShrink', mode='soft',
                                 sigma=sigma_est/2, rescale_sigma=True)
im_visushrink4 = denoise_wavelet(noisy, convert2ycbcr=True,multichannel=True,
                                 method='VisuShrink', mode='soft',
                                 sigma=sigma_est/4, rescale_sigma=True)
```

Figure 4.13 Code Snippet of denoising implementation with reduced threshold

4.3.3 K Fold Cross Validation

When random shuffling and splitting the data isn't enough, and we want the data to be distributed correctly in each fold, we use Stratified K Fold. In the case of a regression problem, folds are chosen so that the mean response value is almost identical across all of them. Folds are chosen to have the same proportion of class labels in the case of a classification challenge. The Stratified K Fold is more beneficial in classification problems when having the same percentage of labels in each fold is important.

Implementation:

The feasible choice for the value of K lies between 5 to 10. Any value of k greater than 10 will increase the computational time without adding any performance improvement. We found that 10-fold is the optimum option for splitting data into training and testing because it has lower sensitivity and is less biased. For finding the optimum value for K, there is no specific rule. Model's bias toward the dataset will increase for a smaller value of K. Though a higher value of K reduced the bias, it could be led to a lot of variation. By using k = 10, the photos in the dataset are divided into ten equal groups. The training dataset is made up of nine of these partitions, with the remaining partition being used for test data. Every time, a different partition was used as a test group, while the remaining nine partitions were used as the training dataset. Finally, the mean result is used to determine the model's final evaluation. Figure 4.14 shows K-fold implementation.

```
IMG_SIZE = 128
BATCH_SIZE = 16
EPOCHS = 1
N_SPLIT = 10
#Initializing Data Generators
train_datagen = ImageDataGenerator(rotation_range=40,
                                   width_shift_range=0.2,
                                   height_shift_range=0.2,
                                   rescale = 1./255,
                                  shear_range = 0.2,
                                  zoom_range = 0.2,
                                  horizontal_flip = True)
validation_datagen = ImageDataGenerator(rescale = 1./255)

# k-fold
kfold = StratifiedKFold(n_splits=N_SPLIT,shuffle=True,random_state=42)
```

Figure 4.14 Code snippet of K fold implementation in Keras

4.3.4 ELA

Lossy compression is normally applied uniformly across the image. Whenever an image is forged a part of an image is copy pasted from another source may result in different compression levels. Therefore, a difference in compression level in different parts of an image may indicate that the image has been edited

Additionally, metadata identifying the precise lossy compression utilized is sometimes included in digital data formats such as JPEG. If the observed compression artefacts in such data differ from those predicted by the metadata description, the metadata may not accurately represent the compressed data, implying that the data has been changed.

ELA Implementation:

```
def convert_to_ela_image(path, quality):
    temp_filename = 'temp_file_name.jpg'
    ela_filename = 'temp_ela.png'
    
    image = Image.open(path).convert('RGB')
    image.save(temp_filename, 'JPEG', quality = quality)
    temp_image = Image.open(temp_filename)

    ela_image = ImageChops.difference(image, temp_image)

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

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

    return ela_image
```

4.15 Code snippet of the method used to apply ELA on all images

We have applied on all the images present in the Au and the Tp folder of the dataset and have created X variable that contains ELA images.

```
import random
path = '/kaggle/input/casia-dataset/CASIA2/Au/'
for dirname, _, filenames in os.walk(path):
    for filename in filenames:
        if filename.endswith('jpg') or filename.endswith('png'):
            full_path = os.path.join(dirname, filename)
            X.append(prepare_image(full_path))
            Y.append(1)
            if len(Y) % 500 == 0:
                print(f'Processing {len(Y)} images')

random.shuffle(X)
# X = X[:2100]
# Y = Y[:2100]
print(len(X), len(Y))
```

```
def prepare_image(image_path):
    return np.array(convert_to_ela_image(image_path, 90).resize(image_size)).flatten() / 255.0
```

Figure 4.16 Code snippet of ELA conversion on all the Au folder images

Figure 4.16 shows the code snippet that is used to apply ELA on all the images present in the Au folder and similarly, it is applied on all the images of the Tp folder and stored in list X that is later converted to an array.

Figure 4.7 shows the result of ELA performed on one sampled real and fake image.

We have taken samples of real and tampered images to view what is the output of ELA on these images.

Figure 4.17 ELA output on Real and Fake image

From Figure 4.17 we can see that zebra in fake image ELA is highlighted as there is a difference in their compression levels.

4.3.5 Class Imbalance

Since there are 7491 original images and 5123 tampered images, this results in class imbalance. The presence of class imbalance creates a bias towards the majority class and hence we get imperfect results. Thus, to avoid any class bias we have taken equal samples of both real and fake images.

Implementations:

Taken 2100 samples of real images and 2064 samples of fake images to avoid class imbalance.

4.4 Model Building

CNN model generally contains a rectified linear unit (ReLU) layer that introduces non-linearity, two pooling layers that allow us to reduce the size of the input and make feature detection more robust by making it more impervious to scale and orientation changes, and one pooling layer stride in this experiment. Flatten layer is a simple layer that takes data from the pooling layer, or in the case of no pooling layer, data from the convolution layer, and turns it into a 1D feature vector that is used as the input to the final layer. The fully connected layer accepts inputs from the preceding layer (Flatten layer) and connects them to each and every neuron in the system. The drop out layer prevents over-fitting of parameters. It is a regularisation method that stochastically sets the activations of hidden units for each training case to zero for each training instance.

The reason for this is that the number of parameters in the layers is enormous, and it takes a long time to train and may result in model overfitting.

Some open-source libraries, like Keras, are also used. Keras is a well-known and simple deep learning library created in Python that focuses on facilitating rapid experimentation. This project uses many backends, like TensorFlow, and CNTK.

Keras is used in conjunction with TensorFlow backends.

4.4.1 VGG16

VGG-16 architecture can be used until the block 5 pool layer, and that can then be connected to fully-connected layers with an outputs layer with two outputs. The proposed architecture will include pre-trained (frozen) parameters for the layers 1 to layer 5 pools, that will later connect to a new FC layer.

Implementation:

Figures 4.18 and 4.19 shows code implementation and model summary of the model

```
from tensorflow.keras.applications import MobileNetV2,VGG16
from tensorflow.keras.models import Model
from tensorflow.keras.layers import GlobalAveragePooling2D,BatchNormalization,Dropout,MaxPooling2D
from tensorflow.keras.regularizers import l1,l2,l1_l2
from keras.models import Sequential
from keras.layers import Dense, Embedding, LSTM, GRU, Flatten, Dropout, Lambda
from keras.layers.embeddings import Embedding
import tensorflow as tf
from tensorflow.keras import layers
image_size = [128, 128]
model=VGG16(input_shape=image_size + [3],include_top=False,weights='imagenet')
for layer in model.layers:
    layer.trainable=False

model.summary()
def layer_adder(bottom_model, num_classes):
    top_model= bottom_model.output
    top_model= GlobalAveragePooling2D()(top_model)
    top_model = layers.Dense(4096, activation = 'relu')(top_model)# we can add a new fully connected layer but it wi
    top_model = layers.Dense(4096, activation = 'relu')(top_model)
    top_model = layers.Dense(1000, activation = 'relu')(top_model)
    #x=Dropout(0.25)(x)
    top_model = layers.Dense(num_classes,activation = 'softmax')(top_model) # adding the output layer with softmax fu
    return top_model
```

Figure 4.18 Code snippet of the model used for transfer learning using VGG16

```
Model: "functional_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 128, 128, 3)]     0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 128, 128, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 128, 128, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 64, 64, 64)        0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 64, 64, 128)       73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 64, 64, 128)       147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 32, 32, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 32, 32, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 32, 32, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 32, 32, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 16, 16, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 16, 16, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 16, 16, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 16, 16, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 8, 8, 512)         0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 8, 8, 512)         2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 8, 8, 512)         2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 8, 8, 512)         2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 4, 4, 512)         0
_________________________________________________________________
global_average_pooling2d (Gl (None, 512)               0
_________________________________________________________________
dense_2 (Dense)              (None, 4096)              2101248
_________________________________________________________________
dense_3 (Dense)              (None, 4096)              16781312
_________________________________________________________________
dense_4 (Dense)              (None, 1000)              4097000
_________________________________________________________________
dense_5 (Dense)              (None, 2)                 2002
=================================================================
Total params: 37,696,250
Trainable params: 22,981,562
Non-trainable params: 14,714,688
_________________________________________________________________
None
```

Figure 4.19 Architecture of CNN with transfer learning via VGG16 model used

4.4.2 RESNET50

RESNET stands for Residual Network. This pre-trained model used the skip connection strategy and was fine-tuned with the same parameters as the others. Due to concerns about the time required to train the layers, the building block was redesigned into a bottleneck design. Instead of the previous two layers, this used a three-layer stack. As a result, each of the Resnet34's 2-layer bottleneck blocks was replaced with a 3-layer bottleneck block, resulting in the Resnet 50 architecture. The accuracy of this model is substantially higher than the 34-layer ResNet model.

Implementation

Figures 4.18 and 4.19 shows code implementation and model summary of the model

```
from tensorflow.keras.applications import MobileNetV2,VGG16
from tensorflow.keras.applications import ResNet50
from tensorflow.keras.models import Model
from tensorflow.keras.layers import GlobalAveragePooling2D,BatchNormalization,Dropout,MaxPooling2D
from tensorflow.keras.regularizers import l1,l2,l1_l2
from keras.models import Sequential
from keras.layers import Dense, Embedding, LSTM, GRU, Flatten, Dropout, Lambda
from keras.layers.embeddings import Embedding
IMAGE_RESIZE = 224
RESNET50_POOLING_AVERAGE = 'avg'
DENSE_LAYER_ACTIVATION = 'softmax'
OBJECTIVE_FUNCTION = 'categorical_crossentropy'
import tensorflow as tf
from tensorflow.keras import layers
resnet_weights_path = '../input/resnet50/resnet50_weights_tf_dim_ordering_tf_kernels_notop.h5'
def buil_model():
    model = Sequential()

    # 1st layer as the lumpsum weights from resnet50_weights_tf_dim_ordering_tf_kernels_notop.h5
    # NOTE that this layer will be set below as NOT TRAINABLE, i.e., use it as is
    model.add(ResNet50(include_top = False, pooling = RESNET50_POOLING_AVERAGE, weights = resnet_weights_path))

    # 2nd layer as Dense for 2-class classification, i.e., dog or cat using SoftMax activation
    model.add(Dense(2, activation = DENSE_LAYER_ACTIVATION))

    # Say not to train first layer (ResNet) model as it is already trained
    model.layers[0].trainable = False
    return model
```

Figure 4.20 Code snippet of the model used for transfer learning using RESNET50

```
model = build_model()
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_2 (Conv2D)            (None, 124, 124, 32)      2432
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 120, 120, 32)      25632
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 60, 60, 32)        0
_________________________________________________________________
dropout_2 (Dropout)          (None, 60, 60, 32)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 115200)            0
_________________________________________________________________
dense_6 (Dense)              (None, 256)               29491456
_________________________________________________________________
dropout_3 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_7 (Dense)              (None, 2)                 514
=================================================================
Total params: 29,520,034
Trainable params: 29,520,034
Non-trainable params: 0
_________________________________________________________________
```

Figure 4.21 Architecture of CNN with transfer learning via RESNET50 model used

4.4.3 Custom CNN Model

In this deep-learning custom model, the very first layer consists of a convolutional layer with an input shape of 128*128, a filter of 32 and a kernel size of (5.5). Second layer is similar to the first layer followed by max pooling layer and then by a dropout layer of 0.25. The next layer is a flatten layer that convert input into 1 D array and the last layer is the output layer that produces two output namely real and fake and for this activation function used is SoftMax.

Implementation

Figures 4.22 and 4.23 shows code implementation and model summary of the model

```
def build_model():
    model = Sequential()
    model.add(Conv2D(filters = 32, kernel_size = (5, 5), padding = 'valid', activation = 'relu', input_shape = (128, 128, 3)))
    model.add(Conv2D(filters = 32, kernel_size = (5, 5), padding = 'valid', activation = 'relu', input_shape = (128, 128, 3)))
    model.add(MaxPool2D(pool_size = (2, 2)))
    model.add(Dropout(0.25))
    model.add(Flatten())
    model.add(Dense(256, activation = 'relu'))
    model.add(Dropout(0.5))
    model.add(Dense(2, activation = 'softmax'))
    return model
```

Figure 4.22 Code snippet of the model used for transfer learning using RESET50

```
[38]: model = build_model()
      model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 124, 124, 32)      2432
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 120, 120, 32)      25632
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 60, 60, 32)        0
_________________________________________________________________
dropout (Dropout)            (None, 60, 60, 32)        0
_________________________________________________________________
flatten (Flatten)            (None, 115200)            0
_________________________________________________________________
dense (Dense)                (None, 256)               29491456
_________________________________________________________________
dropout_1 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_1 (Dense)              (None, 2)                 514
=================================================================
Total params: 29,520,034
Trainable params: 29,520,034
Non-trainable params: 0
_________________________________________________________________
```

Figure 4.23 Architecture of custom CNN model

4.5 Summary

This chapter was mainly focused on model building for image forgery detection. We looked into various EDA techniques on an image dataset and their implementation and outputs. We also looked into various pre-processing steps like image augmentation, denoising, k fold CV, class imbalance, and ELA. Later we focussed on the final model building where we have used 2 state of art models and 1 custom model.

CHAPTER 5: RESULTS AND DISCUSSION

5.1 Introduction

In this chapter, we will discuss all the model and pre-processing steps taken by us and their results. Firstly, we will discuss the result of pre-processing steps that we have taken and then we will discuss the results of state of art models that we have used. Finally, we will discuss results achieved by the custom model that help us in the achieving highest accuracy.

5.2 Image Pre-processing Results

There are multiple steps for image preprocessing we have taken before

5.2.1 Wavelet Denoising Results:

Noise removal using wavelet-based algorithms has been proved to be effective. The discrete wavelet transforms coefficients, which have been modified by additive white Gaussian noise, are thresholder in these methods. For evaluation of denoised images, we are using the PSNR index.

$$MSE = \frac{1}{N_1 N_2} \sum_{i=1}^{N1} \sum_{j=1}^{N2} [I_1(i,j) - I_2(i,j)]^2$$

PSNR is based on Mean Squared Error.

$$PSNR = 10 * log\left(\frac{255^2}{MSE}\right)$$

First, we applied denoising on real images and fake images and we got the following results.

Table 5.1 PSNR comparison of wavelet denoising

	Real Image	Fake Image
Gaussian noise standard deviation	0.000419	0.000363
PSNR [Original vs. Denoised (VisuShrink,$\sigma = \sigma_{est}$]	56.24	57.08
PSNR [Original vs. Denoised (Bayes)]	65.8	59.09

PSNR [Original vs. Denoised (VisuShrink,$\sigma = \sigma_{est}/2$)]	61.86	62.69
PSNR [Original vs. Denoised (VisuShrink,$\sigma = \sigma_{est}/4$)]	67.6	68.45

Figure 5.1 shows the denoising impact on the real image and also gaussian noise standard deviation.

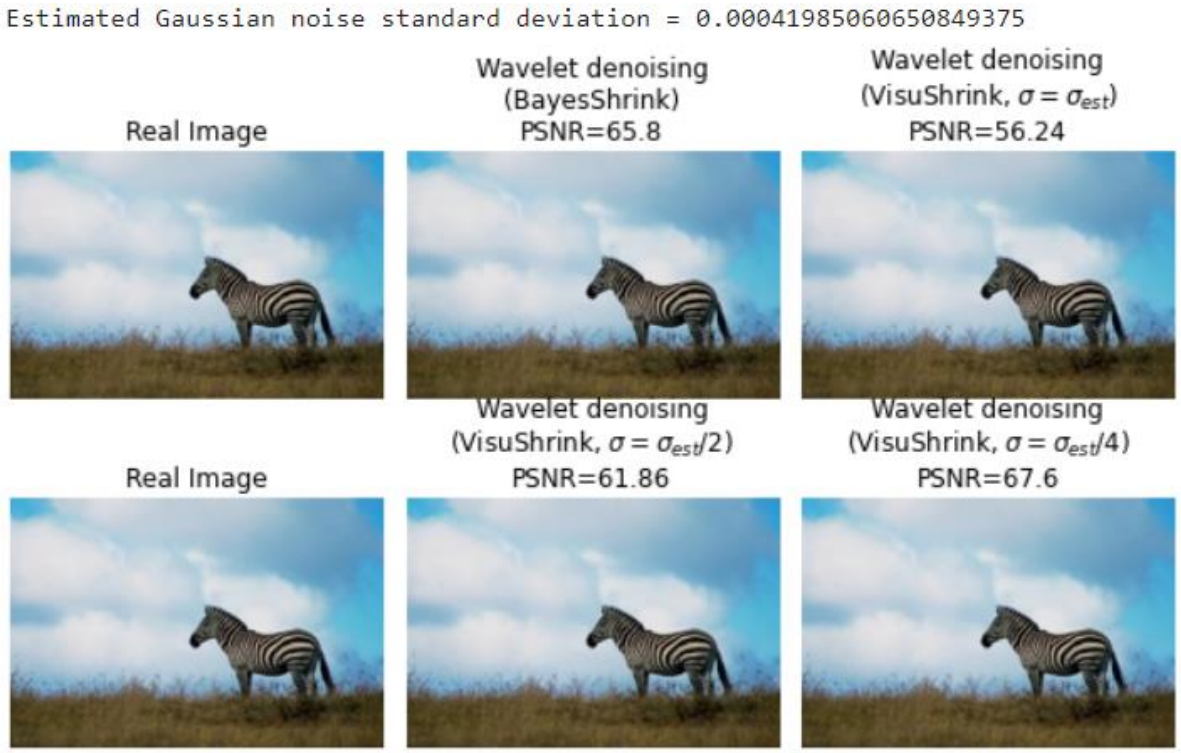


Figure 5.1 Denoising impact on the real image

Figure 5.2 show Denoising impact on the fake image and also gaussian noise standard deviation.

Figure 5.2 Denoising impact on the Fake Image

5.3 Model Building

We have tried using two state of art models and one custom CNN model. Let's discuss their outputs.

5.3.1 Model Building using transfer learning via VGG16

We have utilized state of art model VGG 16 as our first trial to achieve the desired results of identifying image forgery. The output of the ELA images is passed to the model. We have used 20 epochs. Below is the result of 20 epochs.

Table 5.2 Training and validation accuracy for 20 epochs

EPOCH	Time	Train Loss	Train Accuracy	Validation Loss	Validation Accuracy
Epoch 1/20	12s 31ms/step	0.3582	0.8264	0.2562	0.8827
Epoch 2/20	10s 27ms/step	0.283	0.8669	0.2409	0.8885
Epoch 3/20	10s 27ms/step	0.2692	0.8735	0.2669	0.8678
Epoch 4/20	10s 28ms/step	0.2549	0.8801	0.2281	0.8938
Epoch 5/20	10s 28ms/step	0.2538	0.882	0.2865	0.8673
Epoch 6/20	10s 28ms/step	0.2402	0.8888	0.2369	0.8854
Epoch 7/20	10s 28ms/step	0.2371	0.8896	0.2122	0.8997
Epoch 8/20	10s 28ms/step	0.2378	0.8869	0.2348	0.896
Epoch 9/20	10s 27ms/step	0.2298	0.8942	0.2348	0.8986
Epoch 10/20	10s 28ms/step	0.228	0.8967	0.2235	0.8986
Epoch 11/20	11s 28ms/step	0.2291	0.8909	0.2564	0.8148
Epoch 12/20	10s 27ms/step	0.2226	0.8937	0.2156	0.8912
Epoch 13/20	10s 28ms/step	0.2223	0.8971	0.2067	0.9013
Epoch 14/20	10s 28ms/step	0.2143	0.903	0.2198	0.8954
Epoch 15/20	10s 27ms/step	0.2177	0.8979	0.2211	0.897
Epoch 16/20	10s 28ms/step	0.2099	0.907	0.2014	0.9029
Epoch 17/20	10s 28ms/step	0.2097	0.905	0.2098	0.9007
Epoch 18/20	10s 28ms/step	0.2113	0.904	0.213	0.8949
Epoch 19/20	10s 27ms/step	0.2058	0.907	0.2083	0.897
Epoch 20/20	10s 28ms/step	0.2042	0.9044	0.2211	0.8976

We have achieved training accuracy of 90.44% and validation accuracy of 89.76%, with training and validation loss as 0.2042 and 0.2211 respectively. Figure5.3 shows a graph for Training loss Vs Validation loss and Training accuracy Vs Validation Accuracy

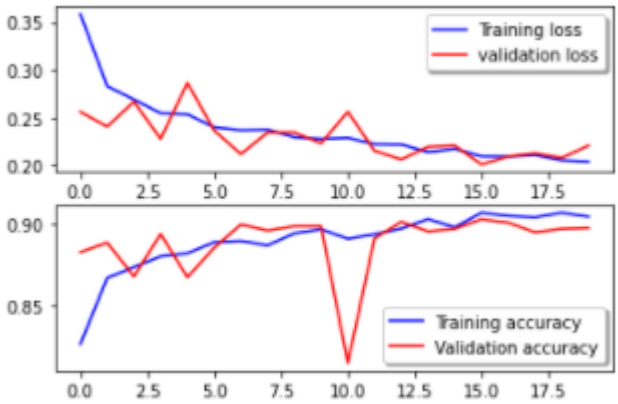


Figure 5.3 Training loss Vs Validation loss and Training accuracy Vs Validation Accuracy

We have also used a confusion matrix to calculate the performance of the model. Figure 5.4 shows a confusion matrix. Since diagonal elements have higher values, we can say that model is better performing.345 fake images are correctly identified out of a total of 386 fake images and 1346 real images are correctly identified as real out of 1489 images. Thus, giving total accuracy of 90.93%.

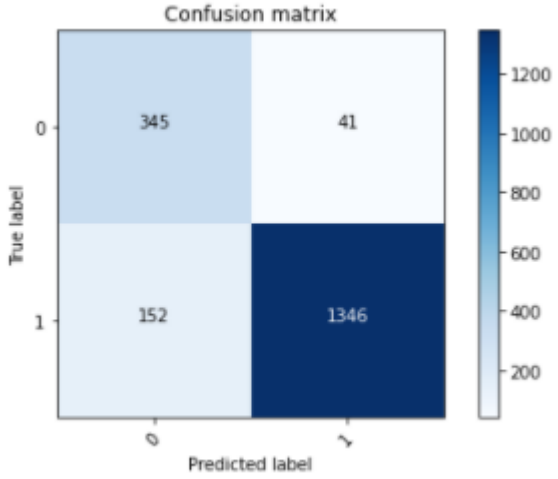


Figure 5.4 Confusion Matrix for VGG 16

5.3.2 Model Building using transfer learning via RESNET50

We have utilized state of art model RESNET50 as our second trial to achieve the desired results of identifying image forgery. The output of the ELA images is passed to the model. We have used 20 epochs. Below is the result of 20 epochs.

Table 5.3 Training and validation accuracy for 20 epochs

EPOCH	Time	Train Loss	Train Accuracy	Validation Loss	Validation Accuracy
Epoch 1/20	7s 17ms/step	0.3181	0.8411	0.2517	0.8912
Epoch 2/20	6s 15ms/step	0.2213	0.8967	0.1657	0.9257
Epoch 3/20	6s 16ms/step	0.1675	0.9238	0.1357	0.9379
Epoch 4/20	6s 15ms/step	0.1388	0.9395	0.1213	0.9379
Epoch 5/20	6s 16ms/step	0.1157	0.9505	0.1418	0.9321
Epoch 6/20	6s 15ms/step	0.101	0.9587	0.1221	0.9421
Epoch 7/20	6s 16ms/step	0.0893	0.9654	0.1204	0.9411

Epoch 8/20	6s 15ms/step	0.0752	0.9719	0.1143	0.9443
Epoch 9/20	6s 15ms/step	0.065	0.9744	0.1227	0.9464
Epoch 10/20	6s 15ms/step	0.0587	0.9793	0.1279	0.9453
Epoch 11/20	6s 15ms/step	0.0487	0.9831	0.1268	0.9453
Epoch 12/20	6s 16ms/step	0.0471	0.9819	0.1301	0.9453
Epoch 13/20	6s 16ms/step	0.0397	0.9857	0.1356	0.949
Epoch 14/20	6s 15ms/step	0.0378	0.9866	0.1453	0.9432
Epoch 15/20	6s 15ms/step	0.0324	0.9882	0.1443	0.9469
Epoch 16/20	6s 15ms/step	0.034	0.989	0.1503	0.9475
Epoch 17/20	6s 15ms/step	0.0259	0.9912	0.1665	0.949
Epoch 18/20	6s 16ms/step	0.024	0.9923	0.15	0.9443
Epoch 19/20	6s 15ms/step	0.0257	0.9915	0.1652	0.9496
Epoch 20/20	6s 16ms/step	0.0216	0.9926	0.1629	0.9501

We have achieved training accuracy of 99.26% and validation accuracy of 95.01 %, with training and validation loss as 0.0216 and 0.1629 respectively. Figure 5.5 shows a graph for Training loss Vs Validation loss and Training accuracy Vs Validation Accuracy.

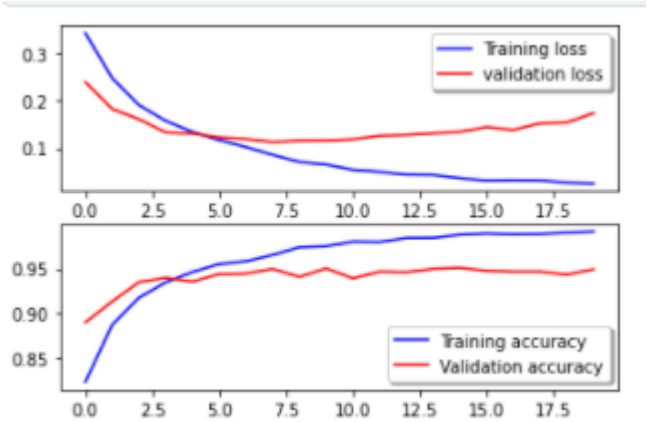


Figure 5.5 Training loss Vs Validation loss and Training accuracy Vs Validation Accuracy

We have also used a confusion matrix to calculate the performance of the model. Figure 5.6 shows a confusion matrix. Since diagonal elements have higher values, we can say that model is better performing. 317 fake images are correctly identified out of a total of 386 fake images and 1471 real images are correctly identified as real out of 1498 images. Thus, giving total accuracy of 94.28 %.

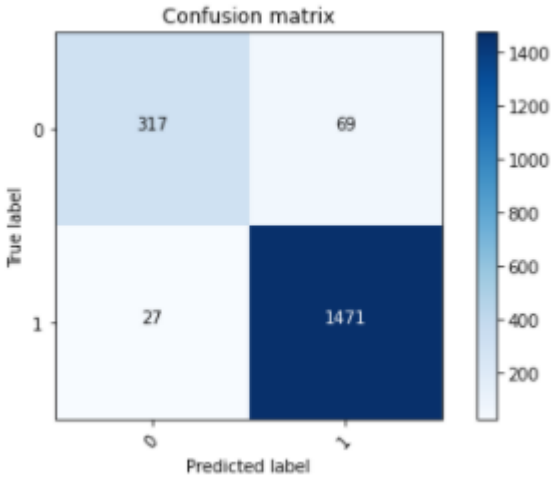


Figure 5.6 Confusion Matrix for RESNET50

5.3.3 Model Building using transfer learning using a custom model

We have utilized a custom model for our final trial to achieve the desired results of identifying image forgery. The output of the ELA images is passed to the model. We have used 30 epochs. Below is the result of 30 epochs.

Table 5.4 Training and validation accuracy for 30 epochs

EPOCH	Time	Train Loss	Train Accuracy	Validation Loss	Validation Accuracy
Epoch 1/30	5s 23ms/step	0.3325	0.828	0.2312	0.8896
Epoch 2/30	5s 21ms/step	0.2454	0.8872	0.1984	0.8981
Epoch 3/30	5s 21ms/step	0.1997	0.9064	0.1531	0.9262
Epoch 4/30	5s 20ms/step	0.1663	0.9246	0.1389	0.94
Epoch 5/30	5s 20ms/step	0.145	0.9367	0.1316	0.9358
Epoch 6/30	5s 21ms/step	0.1276	0.9439	0.1237	0.94
Epoch 7/30	5s 20ms/step	0.1139	0.9522	0.132	0.9453
Epoch 8/30	5s 20ms/step	0.1037	0.9589	0.122	0.9421
Epoch 9/30	5s 21ms/step	0.0892	0.9662	0.1226	0.9416
Epoch 10/30	5s 21ms/step	0.0806	0.9697	0.1233	0.9374
Epoch 11/30	5s 21ms/step	0.0741	0.9715	0.1167	0.9437
Epoch 12/30	5s 20ms/step	0.0696	0.9735	0.1218	0.9432
Epoch 13/30	5s 20ms/step	0.0591	0.9777	0.1211	0.948
Epoch 14/30	5s 20ms/step	0.0543	0.9808	0.121	0.9464
Epoch 15/30	5s 20ms/step	0.0478	0.9823	0.1255	0.9448

Epoch 16/30	5s 21ms/step	0.0453	0.9834	0.1256	0.9464
Epoch 17/30	5s 20ms/step	0.0382	0.9873	0.1362	0.9517
Epoch 18/30	5s 21ms/step	0.0343	0.9875	0.1403	0.9448
Epoch 19/30	5s 20ms/step	0.0346	0.9891	0.1463	0.9522
Epoch 20/30	5s 20ms/step	0.0289	0.9918	0.1509	0.9459
Epoch 21/30	5s 20ms/step	0.0256	0.9915	0.1467	0.9506
Epoch 22/30	5s 21ms/step	0.0307	0.989	0.1424	0.9496
Epoch 23/30	5s 21ms/step	0.0281	0.9915	0.1513	0.9485
Epoch 24/30	5s 21ms/step	0.0231	0.9935	0.1597	0.9485
Epoch 25/30	5s 21ms/step	0.0205	0.9947	0.1688	0.9522
Epoch 26/30	5s 20ms/step	0.0222	0.9927	0.1553	0.9517
Epoch 27/30	5s 20ms/step	0.0226	0.9931	0.1697	0.9522
Epoch 28/30	5s 20ms/step	0.0167	0.995	0.1847	0.9459
Epoch 29/30	5s 22ms/step	0.0193	0.9935	0.2029	0.9363
Epoch 30/30	5s 20ms/step	0.0176	0.9943	0.1732	0.9469

We have achieved training accuracy of 99.43% and validation accuracy of 94.69 %, with training and validation loss as 0.0176 and 0.1732 respectively. Figure 5.7 shows a graph for Training loss Vs Validation loss and Training accuracy Vs Validation Accuracy.

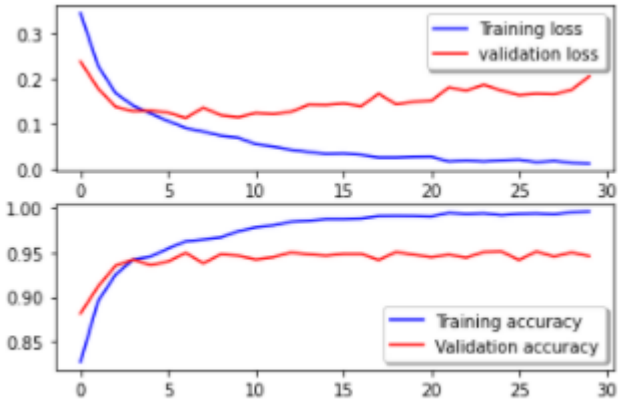


Figure 5.7 Training loss Vs Validation loss and Training accuracy Vs Validation Accuracy

We have also used a confusion matrix to calculate the performance of the model. Figure 5.8 shows confusion matrix. Since diagonal elements have higher values, we can say that model is better performing. 359 fake images are correctly identified out of a total of 386 fake images and 1423 real images are correctly identified as real out of 1498 images. Thus, giving total accuracy of 98.74 %.

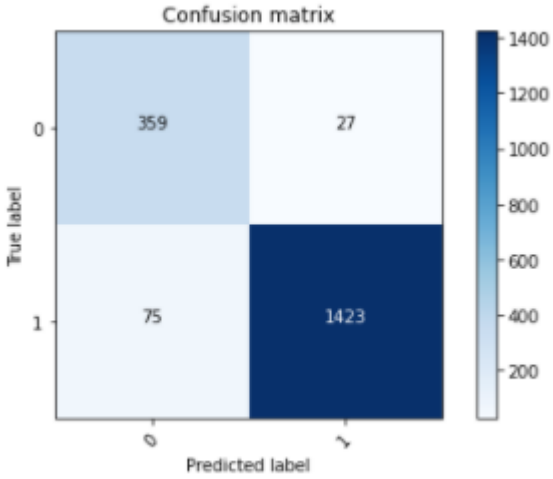


Figure 5.8 Confusion Matrix for custom CNN model

5.**4 Summary**

In this chapter, we discussed the outcome/results of various pre=processing techniques used. We later in section 5.3 discussed the results of various models based on evaluation metrics discussed in chapter3. In comparison it has been observed that a custom CNN model is giving better accuracy in our case and we consider this custom CNN model as our final model.

CHAPTER 6: CONCLUSION AND FUTURE RECOMMENDATION

6.1 Introduction

In this chapter, we will discuss the research by summarizing the major findings in connection to the research objectives and questions, as well as discussing their values and contribution. It will also go through the study's limitations and make recommendations for further work.

6.2 Conclusion

6.2.1 Conclusion

With the advancement of technology, image editing tools are also advancing and it is getting very difficult to identify image forgery via naked eyes. Since with hand-held devices the flow of information is extremely fast, these forged images are transferred easily and quickly to multiple users and thus cause harm in very different ways like provoking riots, etc. Thus, it is very important to detect forged images effectively and efficiently and this is the main objective of this study.

In chapter 1 we have discussed the aims and objectives of this study and also discussed the approach and scope of this study. Chapter 2 covers details about digital photography and its pipeline, different image formats and image compressions. We also discussed various literature that covers different image forgeries detection techniques studied so far. Chapter 3 covers the dataset description, CASIA V2.0 that we have used for the training of model. It also discussed various pre-processing techniques, usage and implementation and also discussed various CNN models. We have also studied the importance of transfer learning using state of art models.

Chapter 4 was mostly about implementation, where we listed down various EDA applied on the image dataset and their results to draw some conclusion about dataset beforehand. We also discussed 3 models, 2 state of art models (VGG16 and RESNET50) and 1 custom model.

In chapter 5, we discussed outcomes of various pre-processing we applied on the dataset and also the result of final model building. Table 6.1 shows comparisons of different models used in this study based on evaluation metrics discussed in chapter 3.

Table 6.1 Comparison of performance of various models implemented

	K fold + Custom Model	RESNET50	VGG16	Custom CNN Model
Accuracy	0.592595529	0.949044586	0.900212314	0.945859873
Precision	0.515105561	0.948293687	0.915365502	0.94978829
F1 Score	0.446456835	0.947930972	0.904405928	0.947003701
Confusion Matrix	[[7443 48] [5091 32]]	[[317 69] [27 1471]]	[[345 41] [147 1351]]	[[359 27] [75 1423]]

From table 6.1, it can be concluded that the custom model is giving better results, but the same model without ELA is not giving good results. The custom model is the final model selected with a validation accuracy of 94.58%.

In general CNN models require large amount of training data and huge computational time and hardware. But CNN is a promising technology in field of image forensics. Though these results are not giving the desired accuracy with further studies in this field we can hopefully expect better results.

6.2.2 Limitations of the model

Though we can achieve an accuracy of 98.98%, there are still some limitations with of the data.

CNN models require a lot of data for training purpose

- ELA method used for feature extraction has the limitation that it only helps in identifying tampering in images with lossy compression. ELA fails to identify tampering in case of lossless compression e.g., in the case of png format image ELA is of no use. Apart from this ELA also has limitations in working well with grayscale images
- CNN models require a lot of training data and training with a small dataset majorly results in overfitting. CNN models also require powerful computational hardware. Even after heavy hardware requirements it takes a lot of time to train the model.

6.2.3 Learnings

This research has added a lot to my learning. I have understood the background of digital images and their different formats. And technically I got an opportunity to learn a few algorithms of deep learning that can be beneficial to me in the future.

Below are a few major takeaways for me from this research:

1. ELA is a very important pre-processing step. This is justified by the use of a custom model with ELA which gave us far better performing results than model without ELA and with K fold.

2. Though the model performance of both RESNET50 and custom model are comparable but we choose the custom model as the final model for us because the custom model accurately identified 359 fake images out of a total 398 in comparison to RESNET 50 that identified only 317 out 398 images correctly. For detecting image forgeries, it is important that the accuracy of identifying fake images is better than the accuracy of identifying real images accurately.

6.3 Future Recommendation

We have tried to achieve maximum accuracy using deep learning approaches to detect image forgeries. But there is still some scope of improvement, there are few points that we want to highlight that can be studied further and may help in achieving far better results. Below are a few future recommendations.

- Creating deep learning models that can be trained from a small dataset. Generally, deep learning algorithms are used in applications that have a huge amount of unsupervised data. With large quantities of unlabelled training datasets, deep learning has a better chance of succeeding. With a limited training dataset, robust models are required to boost learning potential. Thus, it is strongly advised that researchers investigate how to create a deep model learning from a tiny training dataset.
- Using optimization techniques to tweak the parameters of the model. The number of parameters that must be modified in deep learning CNN models is enormous. Furthermore, due to a large number of hidden units, the model is more likely to become ensnared in the local peak ideal. To tackle this problem, optimization techniques such as PSO are required. As a result, the proposed model should be capable of automatically extracting the features and needs fewer human interventions.
- CASIA V2.0 is a slightly imbalanced dataset and to remove imbalance we just took an equal number of samples of real and fake images. Despite recent breakthroughs in deep learning and its growing popularity, there is virtually little empirical work on deep learning with class imbalance.

6.4 Summary

As a final conclusion for this report, we can say that transfer learning provides lots of benefits like saving the resources and computational time when training new models. It can also help in applications where a huge unlabelled dataset is available. We have used ELA and various other pre-processing (rescaling, normalization) that helped in improving the accuracy. We have also listed down all the future work in this field that can prove helpful in better accuracy.

REFERENCE

Abdalla, Y., (2019) Convolutional Neural Network for Copy-Move. *Symmetry*, pp.1–17.

Afsal Villan, M., Kuruvilla, K., Paul, J. and Elias, E.P., (2017) Fake Image Detection Using Machine Learning. *IRACST -International Journal of Computer Science and Information Technology & Security*, 72, pp.2249–9555.

Ahmad Jawad Khan Muhammad Salah Ud Din Iqbal, K., (2019) *A comparative study of Different Denoising Techniques in Digital Image Processing; A comparative study of Different Denoising Techniques in Digital Image Processing*. *2019 8th International Conference on Modeling Simulation and Applied Optimization (ICMSAO)*.

Al-qershi, O.M. and Khoo, B.E., (2018) Evaluation of copy-move forgery detection : datasets and evaluation metrics. pp.31807–31833.

Alahmadi, A., Hussain, M., Aboalsamh, H., Muhammad, G., Bebis, G. and Mathkour, H., (2017) Passive detection of image forgery using DCT and local binary pattern. *Signal, Image and Video Processing*, 111, pp.81–88.

Amerini, I., Uricchio, T., Ballan, L. and Caldelli, R., (2017) Localization of JPEG double compression through multi-domain convolutional neural networks.

Anderson, S.D., (2011) *Digital image analysis: Analytical framework for authenticating digital images.* University of Colorado at Denver.

Armas Vega, E.A., González Fernández, E., Sandoval Orozco, A.L. and García Villalba, L.J., (2020) Passive Image Forgery Detection Based on the Demosaicing Algorithm and JPEG Compression. *IEEE Access*, 8, pp.11815–11823.

Bakiah, N., Warif, A., Yamani, M., Idris, I., Wahid, A., Wahab, A. and Salleh, R., (2015) *An evaluation of Error Level Analysis in image forensics; An evaluation of Error Level Analysis in image forensics*. [online] *2015 5th IEEE International Conference on System Engineering and Technology (ICSET)*. Available at: http://fotoforensics.com/.

Bayar, B. and Stamm, M.C., (2016) A deep learning approach to universal image manipulation detection using a new convolutional layer. *IH and MMSec 2016 - Proceedings of the 2016 ACM Information Hiding and Multimedia Security Workshop*, pp.5–10.

Bnou, K., Raghay, S. and Hakim, A., (2020) A wavelet denoising approach based on unsupervised learning model. *Eurasip Journal on Advances in Signal Processing*, 20201.

Bondi, L., Lameri, S., Guera, D., Bestagini, P., Delp, E.J. and Tubaro, S., (2017) Tampering Detection and Localization Through Clustering of Camera-Based CNN Features. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 2017-July, pp.1855–1864.

Chen, J., Kang, X., Liu, Y. and Wang, Z.J., (2015) Median Filtering Forensics Based on

Convolutional Neural Networks. *IEEE Signal Processing Letters*, 2211, pp.1849–1853.

Christlein, V., Riess, C., Jordan, J., Riess, C. and Angelopoulou, E., (2012) An evaluation of popular copy-move forgery detection approaches. *IEEE Transactions on Information Forensics and Security*, 76, pp.1841–1854.

Dennis P Curtin, (2011) *An Extension to PhotoCourse Digital Photography Textbooks : Sensors, pixels and image sizes*.

Dirik, A.E. and Memon, N., (2009) *Image tamper detection based on demosaicing artifacts; Image tamper detection based on demosaicing artifacts*. *2009 16th IEEE International Conference on Image Processing (ICIP)*.

Fan, L., Zhang, F., Fan, H. and Zhang, C., (n.d.) Brief review of image denoising techniques. [online] Available at: https://doi.org/10.1186/s42492-019-0016-7.

Farid, H., (2009) Image forgery detection. March, pp.16–25.

Gholamalinezhad, H. and Khosravi, H., (2020) Pooling Methods in Deep Neural Networks, a Review. [online] Available at: http://arxiv.org/abs/2009.07485.

Gitbook, (n.d.) *Artificial Intelligence*. [online] Available at: https://leonardoaraujosantos.gitbook.io/artificial-inteligence/.

Hsu, C.C., Zhuang, Y.X. and Lee, C.Y., (2020) Deep fake image detection based on pairwise learning. *Applied Sciences (Switzerland)*, 101.

Hsu, Y.F. and Chang, S.F., (2010) Camera response functions for image forensics: An automatic algorithm for splicing detection. *IEEE Transactions on Information Forensics and Security*, 54, pp.816–825.

Huang, Y., Lu, W., Sun, W. and Long, D., (2011) Improved DCT-based detection of copy-move forgery in images. *Forensic Science International*, 2061–3, pp.178–184.

Jaiswal, A.K. and Srivastava, R., (2019) Image Splicing Detection using Deep Residual Network. *SSRN Electronic Journal*.

Jalab, H.A., Subramaniam, T., Ibrahim, R.W., Kahtan, H. and Noor, N.F.M., (2019) New texture descriptor based on modified fractional entropy for digital image splicing forgery detection. *Entropy*, 214, pp.1–9.

Jeronymo, D.C., Borges, Y.C.C. and Coelho, L. dos S., (2017) Image forgery detection by semi-automatic wavelet soft-Thresholding with error level analysis. *Expert Systems with Applications*, 85, pp.348–356.

Jing Dong, W. and T.T., (2013) CASIA IMAGE TAMPERING DETECTION EVALUATION DATABASE Jing Dong , Wei Wang and Tieniu Tan Institute of Automation , Chinese Academy of Sciences. pp.422–426.

Johnson, M.K., (2005) Exposing Digital Forgeries by Detecting Inconsistencies in Lighting.

Karam, L.J., (n.d.) *Chapter 16 - Lossless Image Compression*. *The Essential Guide to Image Processing*.

Kashyap, A., Parmar, R.S., Agarwal, M. and Gupta, H., (n.d.) An Evaluation of Digital Image Forgery Detection Approaches.

Kimmel, R., (1998) Demosaicing: Image reconstruction from color CCD samples. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 14069, pp.610–622.

Kuznetsov, A., (2019) Digital image forgery detection using deep learning approach. *Journal of Physics: Conference Series*, 13683.

Lin, H.J., Wang, C.W. and Kao, Y.T., (2009a) Fast copy-move forgery detection. *WSEAS Transactions on Signal Processing*, 55, pp.188–197.

Lin, Z., He, J., Tang, X. and Tang, C.K., (2009b) Fast, automatic and fine-grained tampered JPEG image detection via DCT coefficient analysis. *Pattern Recognition*, 4211, pp.2492–2501.

Lukac, R. and Plataniotis, K.N., (2005) *Color filter arrays: design and performance analysis; Color filter arrays: design and performance analysis*. [online] *IEEE Transactions on Consumer Electronics*, Available at: http://www.dsp.utoronto.ca/~lukacr.

Moynihan, T., (2011) *CMOS Is Winning the Camera Sensor Battle, and Here's Why | TechHive*. [online] techhive.com. Available at: https://www.techhive.com/article/246931/cmos-is-winning-the-camera-sensor-battle-and-heres-why.html [Accessed 2 Oct. 2021].

N.Krawetz, (2007) A Picture ' s Worth ... Digital Image Analysis and Forensics Table of Contents. pp.1–31.

Nidhi Mantri, (2021) *https://iq.opengenus.org/image-denoising-and-image-processing-techniques/*. iq.opengenus.org.

Nielsen, M., (2015) *Neural Networks and Deep Learning*.

Popescu, A.C. and Farid, H., (2004) Exposing Digital Forgeries by Detecting Duplicated Image Regions. *Technical Report, TR2004-515, Department of Computer Science, Dartmouth College, Hanover, New Hampshire*, [online] 2000, pp.1–11. Available at: http://os2.zemris.fer.hr/ostalo/2010_marceta/Diplomski_files/102.pdf.

Qurat-Ul-Ain, Nida, N., Irtaza, A. and Ilyas, N., (2021) Forged Face Detection using ELA and Deep Learning Techniques. *Proceedings of 18th International Bhurban Conference on Applied Sciences and Technologies, IBCAST 2021*, pp.271–275.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C. and Fei-Fei, L., (2015) ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, [online] 1153,

pp.211–252. Available at: http://dx.doi.org/10.1007/s11263-015-0816-y.

Salloum, R., Ren, Y. and Jay Kuo, C.C., (2018) Image Splicing Localization using a Multi-task Fully Convolutional Network (MFCN). *Journal of Visual Communication and Image Representation*, 51, pp.201–209.

Samir, S., Emary, E., El-Sayed, K. and Onsi, H., (2020) Optimization of a pre-trained AlexNet model for detecting and localizing image forgeries. *Information (Switzerland)*, 115.

Simonyan, K. and Zisserman, A., (2015) Very deep convolutional networks for large-scale image recognition. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, pp.1–14.

Sudiatmika, I.B.K., Rahman, F., Trisno and Suyoto, (2019) Image forgery detection using error level analysis and deep learning. *Telkomnika (Telecommunication Computing Electronics and Control)*, 172, pp.653–659.

Tammina, S., (2019) Transfer learning using VGG-16 with Deep Convolutional Neural Network for Classifying Images. *International Journal of Scientific and Research Publications (IJSRP)*, 910, p.p9420.

Tan, L.K., (2006) Image file formats. *Biomedical Imaging and Intervention Journal*, 21, pp.1–7.

Walia, S., (2021) Fusion of Handcrafted and Deep Features for Forgery Detection in Digital Images. *IEEE Access*, 9, pp.99742–99755.

Wang, W., Dong, J. and Tan, T., (2014) Exploring DCT coefficient quantization effects for local tampering detection. *IEEE Transactions on Information Forensics and Security*, 910, pp.1653–1666.

Wiggins, R.H., Davidson, H.C., Harnsberger, H.R., Lauman, J.R. and Goede, P.A., (2001) Image file formats: past, present, and future. *Radiographics : a review publication of the Radiological Society of North America, Inc*, 213, pp.789–798.

Wikipedia, (2021) *Telephoto lens - Wikipedia*. [online] Available at: https://en.wikipedia.org/wiki/Telephoto_lens [Accessed 2 Oct. 2021].

Xiaobing, K. and Shengmin, W., (2008) Identifying tampered regions using singular value decomposition in digital image forensics. *Proceedings - International Conference on Computer Science and Software Engineering, CSSE 2008*, 3, pp.926–930.

Zhang, Y., Goh, J., Win, L.L. and Thing, V., (2016) Image region forgery detection: A deep learning approach. *Cryptology and Information Security Series*, 14, pp.1–11.

APPENDIX A – RESEARCH PROPOSAL

FAKE IMAGE FORGERY DETECTION USING DEEP LEARNING

SHUBHANGI TRIVEDI

Research Proposal

AUGUST 2021

Table of Contents

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

CASIA - CHINESE ACADEMY OF SCIENCES' INSTITUTE OF AUTOMATION.
CFA – COLOR FILTER ARRAY
CMFD - COPY MOVE FORGERY DETECTION
CNN- CONVOLUTIONAL NEURAL NETWORK
DCT - DIGITAL COSINE TRANSFORM
ELA – ERROR LEVEL ANALYSIS
FPR - FALSE POSITIVE RATE
FCN – FULLY CONVOLUTIONAL NETWORK
MFCN - MULTI-TASK FULLY CONVOLUTIONAL NETWORK
PCA - PRINCIPAL COMPONENT ANALYSIS
PCT - PRINCIPAL COMPONENT TRANSFORMATION
SIFT - SCALE-INVARIANT FEATURE TRANSFORM
SVM - SUPPORT VECTOR MACHINE

ABSTRACT

Now we depend a lot on multimedia content and with the availability of increased need of content, there is a lot of intentional manipulation is happening. With advancement, we have multiple options available in the market for image editing. Image tempering has become very easy and detecting these forgeries through naked eyes has become a difficult task. Thus, with such advancements in technology, it gets very difficult to judge the credibility of any image available, and such images are accepted by the public without questioning their integrity. So, in this dissertation, we will try to detect real and forged images using deep learning models. For training purposes, we have used the publicly available dataset CASIA V2.0. We used the passive image forgery detectors that would use ELA information of the image to classify an image into fake or real. When an image is forged it is generally the metadata information of the image that gets manipulated and on a portion of the image, an error is introduced due to recompression that happens on each image save.

By using error level analysis, we will calculate the difference between compression levels between different images. We will be using different state of Art models for feature extraction we used ELA technique. We have also used various pre-processing techniques like denoising, image augmentation that helped in achieving maximum accuracy.

We have used custom CNN model also along with state of art models like VGG16and RESNET50, but we achieved the maximum validation accuracy of 94% with custom CNN model and that mode we have considered the final model for image forgery detection.

CHAPTER 1: INTRODUCTION

1.1 Background

The use of digital media has become a part of our daily life with the advent of technology-handheld devices and fast flow of information. Internet and connectivity have exposed us to Exabyte of data especially graphical data such as images, videos, etc. Mobile sharing of images has become so common throughout the world that people have by default exposed themselves to cyber predators who keep a close watch on a typical user's activity. A typical mobile/computer user is not aware of what these predators can do with the images they are sharing. These predators use different types of techniques of changing the data within the media and use the exploited media for malicious purposes.

This situation portrays a definitive need for a comprehensive solution to protect the end user from such digital predators. For providing a flawless solution to the problem, developers/security experts need to have some sort of basis to find out whether the given sample media (photo or a video) is a tampered one or the original one.

The developer may simply ask two questions to determine the authenticity of a given sample image. Firstly, a developer may ask whether the image still depicts the captured original scene. Secondly, he/she may ask whether the image was captured from the device from which it is claimed to be generated. The second question is of prime importance as it allows determining definitive information (user/device which generated the media) about the source of the image. Since it is never easy to determine the actual source of the image, it becomes a great challenge to know anything about the original image.

The developer, therefore, needs to blindly follow what is available in his/her hands i.e. Tampered/original Image. Security experts have classified two methods to overcome challenges related to source determination, first is active in which information acquired from the source (camera, computer, etc.) is exploited to determine the authenticity of the available information. The second is passive in which no information of source is available for exploitation only the media available at disposal has to be checked directly.

An active approach is dependent upon a trustworthy camera i.e. A device that somehow grants a digital signature/watermark to the image automatically. If any cyber predator tries to tamper with the image, the act can easily be traced as the watermark/signature gets tampered with. But the active method bears a major drawback as this method requires a trustworthy camera with a standard protocol to provide a watermark to every image which is captured by it; however, such a requirement is next to impossible as providing a fixed watermark as a

standard practice by every manufacturer is not feasible. Hence, this method is limited to a very scarce number of scenarios.

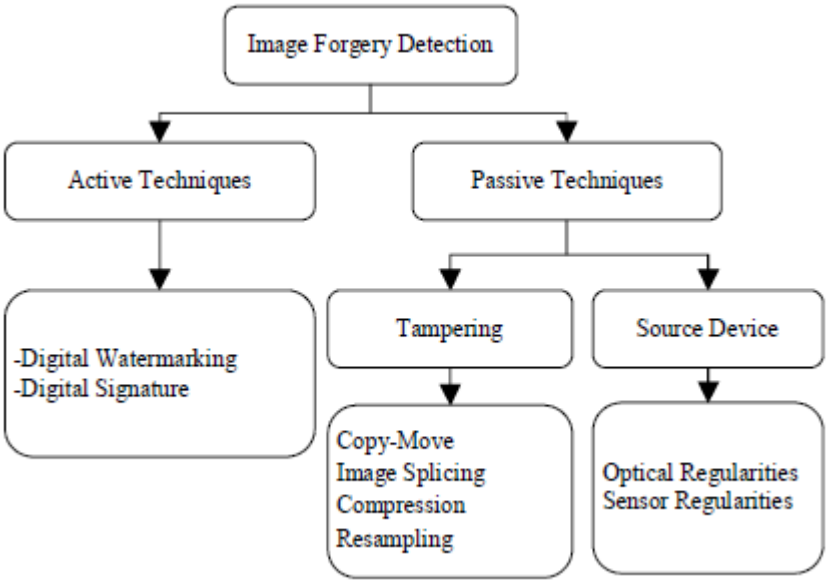


Figure 1.9 Image forgery detection techniques

Such problems can easily be overcome by the use of techniques that do not require any prior information about the image and are hence called passive techniques. These techniques work on the principle of determining traces of data left whenever an image is acquired, compressed, stored, or modified. When such features of data are analyzed it becomes easy to check for any kind of incongruence in the available data.

A typical image can be edited in a variety of ways and therefore, poses a challenge for developers/security experts. An image can be edited for various reasons such as for the improvement of image quality etc. such editing is known as innocent editing, in which the available information becomes more streamlined. On the contrary, when the information is changed by hiding or adding something then such editing is known as malicious editing which is of prime interest for security experts.

Figure 1.10 Forged Images Examples, the first example shows a change of person(image splicing), the second image shows the addition of fountain (Copy-move) and the third image shows the removal of a person(removal) (Diallo et al., 2020)

Image editing is broadly classified into three major categories, these categories comprise both operators i.e. innocent and malicious. The categories are Enhancement, Geometric and Content Modification. Here enhancement operator is primarily used for innocent editing, while other 02 categories (Geometric and Content modification) are intended for malicious attacks.

Enhancement techniques involve basic features such as color modification, contrast adjustments, etc. Geometric modifications involve zoom, rotation, cropping, etc. finally content modification involves copy-move, cut and paste, etc. From the above explanation, it can easily be sought that the latter two techniques will one way or the other involve malicious editing.

Most common malicious attacks involve cropping, cut-paste, etc. as these techniques allow the forger to remove the content of an image and paste it somewhere else, be it any media. The most common categories of image forgeries are

4) **Copy-Move Forgery:** This kind of forgery involves transforming an original image by tampering with its contents by the use of editing features such as moving and

copying. This kind of forgery becomes handy for forgers who want to make use of images in carrying out their political agendas, spreading misinformation, or creating illusions among the audience. This kind of forgery involves no change in the color and background of the original image. Many tools have been developed to detect such a forgery.

5) **Image Splicing:** This kind of forgery mainly involves merging the contents of the same image or different images to compose a forged image. This technique enables the forger to change the backgrounds of the image or use a background of the different images in the subject image. This technique also allows enhancing the field of vision of any particular image. This technique provides greater flexibility to the forger and allows the forger to make multiple changes in a single image. The detection of this technique is a cumbersome task but can be identified if the forger has left any shadows, reflections in a given image.

(a) *Image Original "donor"* (b) *Image Original "recipient"* (c) *Image Result*

Figure 1.11 Splicing

6) **Image Retouching:** This method allows altering common features of any image such as changing the color of the image, changing the scale of a given image, stretching, or squeezing a particular part of an image. Detection of this technique is very difficult as it involves multiple changes in multiple areas of a given image. Image retouching can be detected by detecting different lighting conditions in a spliced image.

Figure 1.12 Example of Image retouching (Armas Vega et al., 2020)

1.2 Problem statement

When talked about image forgery using copy-move techniques, then splicing is considered difficult to be detected. Spicing is an operation in which a portion of one image is copied by an attacker and pasted on another image thus altering the image. What further makes its recognition difficult is that this copy-move is also followed by some level of compression, with some blur effect and then smoothening of boundaries. Thus detecting forgery in case of splicing is comparatively tricky than forgery techniques.

ELA is the best technique that uses difference between compression levels among different regions of image. Similarly deep learning models are considered best for image recognition. Hence in this study we will try to use CNN for identification of forgery.

1.3 Aims and Objectives

The main aim of the research is to identify tampered images correctly using the CASIA V 2.0 dataset which involves the use of a Convolutional Neural Network for training purposes.

The objectives of the study based on the aims are following:

- To identify the based normalization technique to be applied on images so that we get the same range of pixel intensity per image.
- To identify the technique that can be used for feature reduction and feature extraction.

- To identify the best classifier that can be used for image classification into real and fake images.

1.4 Significance of the study

The result of this study will help in identifying the image forgery or tampering with high accuracy and precision. With the increased usage of social media, incidences of image manipulation are increasing and with advancement, in editing technology, these manipulations have become very hard to detect via the traditional approach and have started to erode the trust in digital content. This phenomenon demands the need for a way to help us verify the truthfulness and credibility of the image. Though color modification, contrast adjustment, and filtering are part of innocent image editing, geometric modification and content modification lead to malicious image editing.

If a forged image is spliced or copy-move, then interpolation is a necessary step. When a forged image contains areas from different sources, or another part of the same image, rescaling and /or rotation are often involved. In general, these changes are mostly done on a part of the image, not on the entire image. Thus, dividing the image into multiple blocks, and then detecting the changes compared to neighboring blocks will help us to detect the manipulated area. Apart from this whenever an image has been tampered the tampering process introduces a mild level of compression and hence compression can also help in identifying tampering. There is a significant level of change that may occur to image metadata also where the source of images is different. Hence by using this understanding of image forgery, we can develop a model that can help us identify image forgery.

In many cases, images are proof of medical reports and crime scene evidence and such forged image can result in loss of life or escape of convict. Thus by this paper, we will try to create a model that could detect image forgery effectively and efficiently and thus we can easily verify the authenticity of an image.

1.5 Scope of Study

The scope of this work focuses on proposing the use of a pre-trained model and its performance will be decided then based on the accuracy of detection of fake images.

Preprocessed images with labels real and fake will be fed to the pre-trained models for feature extraction using ELA and then PCA would be used for feature reduction. Later the extracted

features will be used for the classification of images using different classifiers. The proposed work will be using python and the CASIA V2.0 dataset.

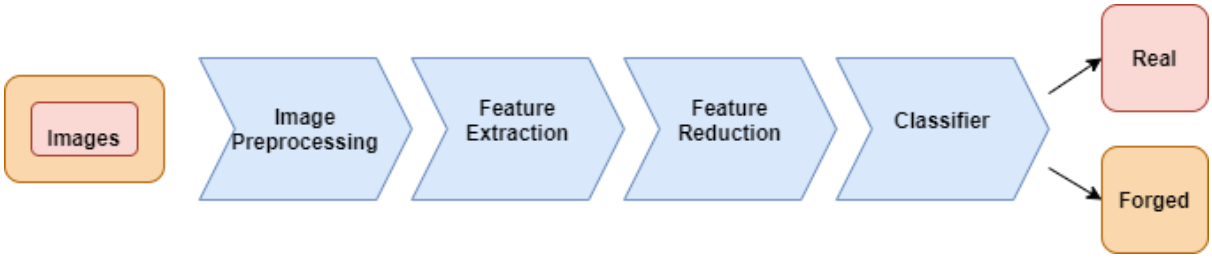


Figure 1.13 Flow Chart

1.6 Structure of Study

This work is majorly classified into four sections. First section is all about the introduction, which includes the background and objectives of this work. Second section covers all the related work done in the field of image forensics. Third Section discusses the proposed research methodology, research plan, and required resources. Fourth section covers the references.

CHAPTER 2: LITERATURE REVIEW

2.1 Introduction

This chapter covers a survey of different techniques used by different researchers that has been studied or implemented so far in field of image forensics. It also discusses results obtained and the dataset used.

2.2 Related Works

The field of digital forgery had drawn the attention of researchers worldwide. And researchers are now working on to find the image forgeries have social impacts, the methods used for such forgeries, how to detect them and how such incidences be prevented. Among which detection holds the highest importance.

There are a lot of research methodologies on which research has been done and researchers are still contributing like in paper (Afsal Villan et al., 2017) authors proposed the usage of a metadata analyser to extract features from the image but this approach didn't work well on images shared online.

(Salloum et al., 2018) proposed the use of Multi-task Fully Convolutional Network (MFCN) for image splicing localization, it uses 2 branches of multitask learning. One branch is used to learn the surface label, while the other branch is used to learn the edge or boundary of the spliced region. The base network architecture is the FCN VGG-16 architecture with skip connections.

(Abdalla, 2019) discussed a copy-move forgery detection using CNN architecture that employs a pre-processing layer but the results were only satisfactory on the passive forged image.

(Jalab et al., 2019) focussed on increasing the accuracy of image splicing detection as well as also on the reduction of feature vector dimensionality by designing a new fractional texture descriptor using DWT. DWT breaks down an input image into multiple sub-images by applying a low and high pass filter. SVM was finally used as a classifier.

(Jaiswal and Srivastava, 2019) used a pre-trained state of art model RESNET-50 for training the model and then further used Multiclass Model using SVM Learner, K-NN, and Naïve Bayes as a classifier which produced accuracy as 70.26%, 59.91%, and 59.91% respectively.

(Sudiatmika et al., 2019) proposed the use of the system that integrate ELA (Error Level Analysis) method and pre-trained model VGG-16, that produced 92.2% training accuracy after using 100 epoch. ELA (Error Level Analysis) uses the difference in compression level

between different subsections of an image to identify forgery. Similarly (Qurat-Ul-Ain et al., 2021) proposed the use of ELA along with multiple state of art models and found out that VGG 16 yields the best accuracy among all state of art models with a training accuracy of 91.97%.

(Kuznetsov, 2019) proposed the use of VGG 16 Convolution Neural Network for identifying image forgery-splicing. The author suggested feeding network architecture patches of an image from the original image and on the border of image splicing, obtaining results with training accuracy of 97.8 % and test accuracy of 96.8%.

(Hsu et al., 2020) proposed the use of CFFN using a network-based pairwise learning model to detect fake and real images. The pairwise learning approach helped in improving the generalization property of DeepFD. The dataset was generated using state of art GAN model. The model was able to detect fake images generated by new GAN models as well.

(Armas Vega et al., 2020) proposed use of two algorithms for image forgery detection. Error Level Analysis (ELA) algorithm, was used to detect splicing in an image which highlighted those pixels that have a different level of compression whereas another algorithm was based on the quadratic mean error of the Colour Filter Array (CFA) interpolation pattern which determined the level of interpolation in the manipulated image. They used CASIA V1.0 dataset and got an accuracy of 73.3% with a high-resolution image. The second algorithm didn't prove useful in the case of low-resolution images.

(Walia, 2021) proposed use of two streams one handcrafted and other deep features.

One stream uses discrete cosine transform of the image to compute Markov-based features. Another stream uses the luminance channel of YCbCr colorspace for feature extraction. CASIA v1 and CASIA v2 datasets were used. The accuracy achieved is 99.3% using the proposed fusion-based approach.

(Samir et al., 2020) proposed the use of the AlexNet model for the classification of image tampering. The model was modified by using batch normalization instead of local response normalization and a maxout activation function instead of rectified linear unit and softmax was used as a classifier. Dataset used were CASIA v2.0, CASIA v1.0, DVMM, and NIST Nimble Challenge 2017 datasets. K folds classification for dividing the dataset into test and train.

2.3 Summary

In order to identify image forgery correctly, it is important that we select most recent and effective technique, that give us most accurate results. In this chapter we have discussed about all the research that has been done so far in field of image forensics.

CHAPTER 3: RESEARCH METHODOLOGY

3.1 Introduction

Deep learning is widely used in the field of image classification and recognition. For this work, images need to be classified as real and fake. CNN will be taking images as input and processed through various layers for feature extraction. Technically, images are passed through multiple CNN layers and filters. Thus, features extracted via the CNN network are used for training and testing purposes. Test data is used to predict the classes based on the model build using train data.

3.2 Data Set Description

CASIA V2 is a dataset for forgery classification. It contains a total of 12323 colored images which are divided into 7491 original images and 5123 tampered images. It contains images from 320X240 to 800X600 pixel images. It contains images with JPEG, TIFF, and BMP formats. Tempered images are mostly splicing using Photoshop as well as blurred images, are also introduced. Original images contain an image of the following 9 categories: scene, animal, architecture, character, plant, article, nature, indoor, and texture.

Figure 14 Example of images captured from CASIA v 2.0 dataset

Tampered images are created by using an original image or multiple original images. Region cropped from original is subjected to rotation, resize, or other distortions before pasting on another image.

Below is the statistical information about the forged/tampered image.

.

Table 2 Statistical information about the Tampered image(Jing Dong, 2013)

Category		No of Images
JPEG Format		2064
TIFF Format		3059
Source of tampered region	Same Image	3274
	Different Image	1849
	Rotation	568
	Resize	1648
	Distortion	196
	Rotation and Resize	532
	Resize and Distortion	211
	Rotation and Distortion	42
	Rotation, Distortion, and Resize	83
Manipulation without pre-processing		1843
Manipulation with post-processing	Blurring along spliced edges	848
	Blurring on other regions	131
Manipulation without post-processing (Blurring)		4144
Size of Tampered Region	Small	3358
	Medium	819
	Large	946

3.3 Proposed Methodology

3.3.1 Pre Processing

Preprocessing of image: This is a very important step when dealing with image classification problems. So first of all we will be applying normalization on the image because CNN learns by back-propagating through various weight matrices throughout the network, thus when the learning rate is applied on all the images then each correction will different each image proportionately.

Another important thing that needs to be kept in mind is the removal of noise. Denoising is a very important step before any further action is taken.

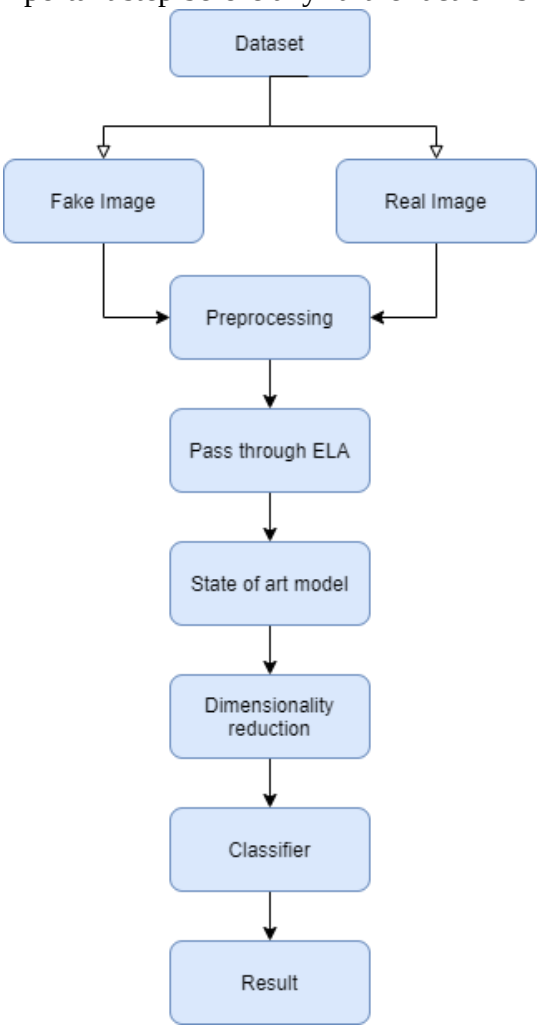


Figure 15 Proposed Approach

3.3.2 Feature extraction

For feature extraction we will be using ELA. Error Level Analysis will highlight the area where there is difference in level of compression. This ELA generated image will be saved in form of Numpy array. Then this data will be passed through pre trained models for feature extraction.

3.3.3 Feature reduction

For dimensionality reduction we will use PCA. So far dimensionality reduction to 60 features had lowest accuracy but we will try to maintain dimensionality up to 100 features.

3.3.4 Classification

We will try using different classifiers such as SVM, XGBoost, Random forest and Naïve Bayes. The classifier that yields max accuracy will be considered for final model.

3.3.5 Result

We will check accuracy of predictions on test dataset.

3.4 Required Resources

Dataset requirements are as follows:

Table 3 Dataset Requirements

Dataset	Format	Operation Methods	Size
CASIA V2.0	JPEG, TIFF, BMP	Splicing using photoshop, compression, Blurring	5123 tampered and 7200 Authentic

Hardware requirements are as follows:

Table 4 Hardware requirement

Resource	Features
Operating System	Window 10 Pro
Memory	16 GB DDR4-2933 SDRAM
Process	Intel® Evo™ platform feat 11th Generation Intel® Core™ i7 processor
Graphics	Xe Graphics
HDD	100 GB
GPU - External	GTX1060 6 GB

Software requirements are as follows:

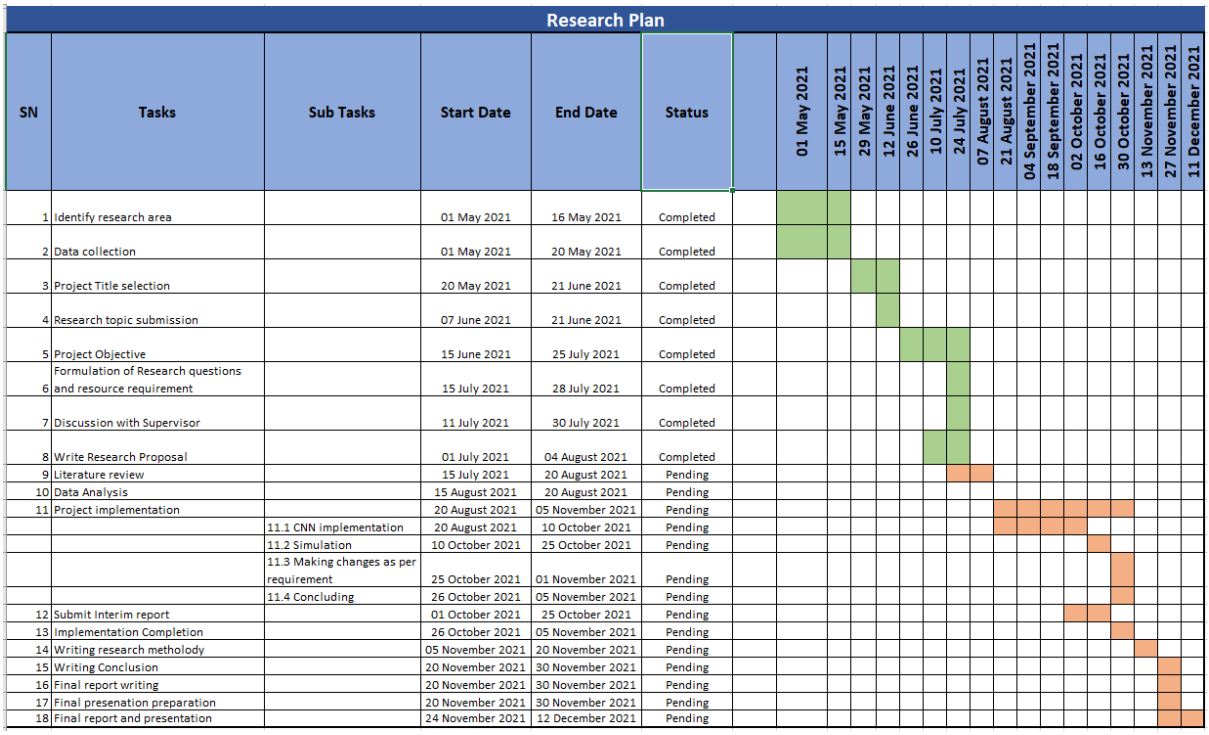
Table 5 Software Requirements

Resources	Features
Software	Jupyter Notebook, Pycharm
Libraries	TensorFlow, Pytorch, OpenCV Keras, Scikit-learn, Numpy, Pandas, Scipy, matplotlib, seaborn

3.5 Research Plan

Research Plan

SN	Tasks	Sub Tasks	Start Date	End Date	Status
1	Identify research area		01 May 2021	16 May 2021	Completed
2	Data collection		01 May 2021	20 May 2021	Completed
3	Project Title selection		20 May 2021	21 June 2021	Completed
4	Research topic submission		07 June 2021	21 June 2021	Completed
5	Project Objective		15 June 2021	25 July 2021	Completed
6	Formulation of Research questions and resource requirement		15 July 2021	28 July 2021	Completed
7	Discussion with Supervisor		11 July 2021	30 July 2021	Completed
8	Write Research Proposal		01 July 2021	04 August 2021	Completed
9	Literature review		15 July 2021	20 August 2021	Pending
10	Data Analysis		15 August 2021	20 August 2021	Pending
11	Project implementation		20 August 2021	05 November 2021	Pending
		11.1 CNN implementation	20 August 2021	10 October 2021	Pending
		11.2 Simulation	10 October 2021	25 October 2021	Pending
		11.3 Making changes as per requirement	25 October 2021	01 November 2021	Pending
		11.4 Concluding	26 October 2021	05 November 2021	Pending
12	Submit Interim report		01 October 2021	25 October 2021	Pending
13	Implementation Completion		26 October 2021	05 November 2021	Pending
14	Writing research metholody		05 November 2021	20 November 2021	Pending
15	Writing Conclusion		20 November 2021	30 November 2021	Pending
16	Final report writing		20 November 2021	30 November 2021	Pending
17	Final presenation preparation		20 November 2021	30 November 2021	Pending
18	Final report and presentation		24 November 2021	12 December 2021	Pending



CHAPTER 4: REFERENCES

Abdalla, Y., (2019) Convolutional Neural Network for Copy-Move. *Symmetry*, pp.1–17.

Afsal Villan, M., Kuruvilla, K., Paul, J. and Elias, E.P., (2017) Fake Image Detection Using Machine Learning. *IRACST -International Journal of Computer Science and Information Technology & Security*, 72, pp.2249–9555.

Armas Vega, E.A., González Fernández, E., Sandoval Orozco, A.L. and García Villalba, L.J., (2020) Passive Image Forgery Detection Based on the Demosaicing Algorithm and JPEG Compression. *IEEE Access*, 8, pp.11815–11823.

Diallo, B., Urruty, T., Bourdon, P. and Fernandez-Maloigne, C., (2020) Robust forgery detection for compressed images using CNN supervision. *Forensic Science International: Reports*, [online] 2September 2019, p.100112. Available at: https://doi.org/10.1016/j.fsir.2020.100112.

Hsu, C.C., Zhuang, Y.X. and Lee, C.Y., (2020) Deep fake image detection based on pairwise learning. *Applied Sciences (Switzerland)*, 101.

Jaiswal, A.K. and Srivastava, R., (2019) Image Splicing Detection using Deep Residual Network. *SSRN Electronic Journal*.

Jalab, H.A., Subramaniam, T., Ibrahim, R.W., Kahtan, H. and Noor, N.F.M., (2019) New texture descriptor based on modified fractional entropy for digital image splicing forgery detection. *Entropy*, 214, pp.1–9.

Jing Dong, W. and T.T., (2013) CASIA IMAGE TAMPERING DETECTION EVALUATION DATABASE Jing Dong , Wei Wang and Tieniu Tan Institute of Automation , Chinese Academy of Sciences. pp.422–426.

Kuznetsov, A., (2019) Digital image forgery detection using deep learning approach. *Journal of Physics: Conference Series*, 13683.

Qurat-Ul-Ain, Nida, N., Irtaza, A. and Ilyas, N., (2021) Forged Face Detection using ELA and Deep Learning Techniques. *Proceedings of 18th International Bhurban Conference on Applied Sciences and Technologies, IBCAST 2021*, pp.271–275.

Salloum, R., Ren, Y. and Jay Kuo, C.C., (2018) Image Splicing Localization using a Multi-task Fully Convolutional Network (MFCN). *Journal of Visual Communication and Image Representation*, 51, pp.201–209.

Samir, S., Emary, E., El-Sayed, K. and Onsi, H., (2020) Optimization of a pre-trained AlexNet model for detecting and localizing image forgeries. *Information (Switzerland)*, 115.

Sudiatmika, I.B.K., Rahman, F., Trisno and Suyoto, (2019) Image forgery detection using error level analysis and deep learning. *Telkomnika (Telecommunication Computing Electronics and Control)*, 172, pp.653–659.

Walia, S., (2021) Fusion of Handcrafted and Deep Features for Forgery Detection in Digital Images. *IEEE Access*, 9, pp.99742–99755.