

Image Tampering Detection using Error Level Analysis and Convolutional Neural Network

A Project Report

Submitted by

Vardhaman Munot

Aniket Dubey

Jash Shah

Prince Tibadia

Under the Guidance of

Prof. Supriya Agarwal

in partial fulfilment for the award of the degree of

BACHELORS OF TECHNOLOGY

COMPUTER ENGINEERING

at



**MUKESH PATEL SCHOOL OF TECHNOLOGY
MANAGEMENT AND ENGINEERING**

April 2023

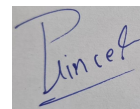
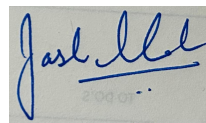
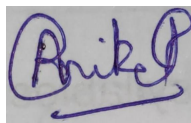
DECLARATION

We, Vardhaman Munot, Aniket Dubey, Jash Shah and Prince Tibadia, Roll No. B033, B064, B072, B079 B.Tech (Computer Engineering), VII-VIII semester understand that plagiarism is defined as anyone or combination of the following:

1. Un-credited verbatim copying of individual sentences, paragraphs or illustration (such as graphs, diagrams, etc.) from any source, published or unpublished, including the internet.
2. Un-credited improper paraphrasing of pages paragraphs (changing a few words phrases, or rearranging the original sentence order)
3. Credited verbatim copying of a major portion of a paper (or thesis chapter) without clear delineation of who did write what. (Source:IEEE, The institute, Dec. 2004)
4. I have made sure that all the ideas, expressions, graphs, diagrams, etc., that are not a result of my work, are properly credited. Long phrases or sentences that had to be used verbatim from published literature have been clearly identified using quotation marks.
5. I affirm that no portion of my work can be considered as plagiarism and I take full responsibility if such a complaint occurs. I understand fully well that the guide of the seminar/ project report may not be in a position to check for the possibility of such incidents of plagiarism in this body of work.

Signature:

Vardhaman Munot



Name: Vardhaman Munot, Aniket Dubey, Jash Shah, Prince Tibadia

Roll No. B033, B064, B072, B079

Place: Mumbai

Date: 25/03/23

CERTIFICATE

This is to certify that the project entitled “Image Tampering Detection using ELA and CNN” is the bonafide work carried out by Vardhaman Munot, Aniket Dubey, Jash Shah and Prince Tibadia of B.Tech (Computer Engineering), MPSTME (NMIMS), Mumbai, during the VIIth and VIIIth semester of the academic year 2022-2023, in partial fulfilment of the requirements for the award of the Degree of Bachelors of Engineering as per the norms prescribed by NMIMS. The project work has been assessed and found to be satisfactory.

<Ms. Supriya Agarwal>

Internal Mentor

Examiner 1

Examiner 2

HOD

Table of contents

CHAPTER NO.	TITLE	PAGE NO.
	Abstract	i
	List of Tables	ii
	List of Figures	iii
	Abbreviations	iv
1.	INTRODUCTION	1
	1.1 Problem Scope	2
	1.2 Project Objectives	2
2.	REVIEW OF LITERATURE	3
3.	ANALYSIS & DESIGN	7
	3.1 Proposed System Diagram	7
	3.2 Design Diagram for Model	8
	3.3 Proposed Methodology	9
	3.4 Datasets Used	11
4.	IMPLEMENTATION AND RESULT	15
	4.1 Implementation - CNN Model	15
	4.2 Implementation - WebApp	19
	4.3 Results and Discussions	24
5.	CONCLUSION AND FUTURE WORK	25
	REFERENCES	26

Abstract

Fake image detection has become an urgent task. Advanced technology benefits us but it also poses a threat to us when used in Cyber Crime. In reality, images are often considered as solid evidence to prove something concrete and thus image falsification or tampering in any way that benefits a party at fault, results in misleading. To detect this manipulation, it takes a substantial amount of image data and a model that can process every pixel and still provide results that are accurate and efficient enough to be used in real time. Therefore, a model that uses Convolutional Neural Networks in combination with Error Level Analysis has been developed that achieved an accuracy of 94% and convergence with 40 epochs. Using this model, a website has been developed that provides users with the facility to check if an image is original or has been tampered with and gives them the option to store authentic images in a secure environment.

List of Tables

CHAPTER NO.	TITLE	PAGE NO.
2.	REVIEW OF LITERATURE	
	Table-1: Comparison of Existing Methods	5
3.	ANALYSIS & DESIGN	
	Table-2: Functions of layers in CNN	11
	Table-3: Datasets Comparison	13
4.	IMPLEMENTATION & RESULT	
	Table-4: Quantitative Results	24

List of Figures

CHAPTER NO.	TITLE	PAGE NO.
3.	ANALYSIS & DESIGN	7
	Fig 3.1 Design Diagram of Model	8
4.	IMPLEMENTATION & RESULT	
	Fig 4.1 Code for modules & ELA	15
	Fig 4.2 Real Image & ELA Image	16
	Fig 4.3 Training the model	17
	Fig 4.4 Saving model & Summary	18
	Fig 4.5 Testing the model	19
	Fig 4.6 Index Page	20
	Fig 4.7 Login Page	20
	Fig 4.8 About Us	21
	Fig 4.9 Upload Image	21
	Fig 4.10 MyImages	22
	Fig 4.11 Flask Driver Code	23

Abbreviations

Abbreviation	Description
CASIA	Institute of Automation, Chinese Academy of Sciences (CASIA) dataset
CNN	Convolutional Neural Network
ELA	Error Level Analysis
GAN	Generative Adversarial Networks
PG-GANs	Progressive growing of GANs
RELU	Rectified Linear Unit

Chapter 1: Introduction

People frequently accept what they can see, which impairs their judgement and causes a number of undesirable reactions. As forgeries have grown, there is a far greater need than ever to identify them. The main motivations behind image fabrication are evil ones. Information is distorted, immorality and fake news are spread, money is fraudulently obtained from an unknowing audience, the reputation of a well-known celebrity or other public figure is ruined, and there is a spread of unfavourable political influence among users of a digital platform. Since digital image editing tools have become more widely available and simpler to use, there has been an increase in the use of altered photos. As a result, more phoney or altered photographs are being posted online, which can be utilised for a range of nefarious activities like disseminating propaganda or conveying false information. The issue has been made worse by the development of deepfake technology, which makes it possible to produce difficult-to-detect fake images and movies that are incredibly convincing. As it diminishes confidence in digital media and can be used to disseminate false information, damage reputations, and meddle in political processes, this tendency has important socioeconomic consequences. For the aforementioned reasons, it is essential to provide techniques for determining if an image is real or altered.

A photograph typically implies the reality of what occurred. However, due to the abundance of digital images in our daily lives and the ease and simplicity with which they can be altered using readily accessible digital processing tools, seeing is sometimes no longer believed in the digital era. Images that have been altered frequently appear in reports, experiments, and even in court documents. As a result, we can no longer assume that photos are legitimate. Image tampering detection tools and methods are greatly desired. The majority of photos taken nowadays do not have digital watermarks, despite the fact that they can be used as a tool to add legitimacy to images. And it's likely that things will stay this way for the foreseeable future. A third party is also required to licence watermarks, and the usefulness and reliability of digital watermarks for image manipulation detection have not yet been proven. Passive image manipulation detection is hence more useful and crucial. By embedding watermarks in the original photographs, it tries to instantly confirm the authenticity of digital photos. Researchers have been concentrating more and more on picture tampering detection in recent years, particularly on passive techniques.

Numerous methods for detecting image manipulation have been proposed over time, including statistical, feature-based, and learning-based techniques. These techniques' sensitivity to particular tampering tactics and their applicability to novel tampering scenarios, however, are frequently their limitations. Additionally, the rapid development of image

manipulation tools has made it more challenging to distinguish between altered and authentic photographs.

Artificial intelligence is one of the most explored topics today, with a wide range of applications and outcomes. Using technologies like machine learning, computers can learn and develop without any explicit programming. In recent years, the data boom and affordable computing power have made AI algorithms easier to implement. Machine learning (ML), deep learning (DL), natural language processing (NLP), and computer vision (CV) are some of the subsets of artificial intelligence that are used to recognise patterns, explain, and forecast based on data with equal accuracy to that of humans. Today, digital images have completely replaced the conventional photographs from every sphere of life but unfortunately, they seldom enjoy the credibility of their conventional counterparts, thanks to the rapid advancements in the field of digital image processing.

1.1 Project Scope

To identify a tampered image is a challenging task. Reliability of an image being shared which is tampered is lost. This project aims to devise a method of detecting tampered images using modern AI based technologies like Error Level Analysis(ELA) and Convolutional Neural Networks(CNN) and integrate it in the back-end of a WebApp in order to protect against fake images and the impact that they may have on unsuspecting users.

1.2 Project Objectives

1. Users can input images that need to be tested.
2. The input image is resized, flattened and converted to an ela image.
3. This image is then tested against the CNN Model to detect any tampering.
4. As a result, the image will be classified as Authentic or Tampered.
5. Authentic Images are encrypted and stored in the cloud storage.
6. Tampered Images are rejected.

Chapter 2: Review of Literature

We have conducted literature surveys for various techniques for detecting image tampering. We tried to find research papers which worked on a wide variety of image tampering methods such as Image Retouching, Image Splicing, and Copy-Move Attack, Morphing. We needed a thorough understanding of ML and DL algorithms in order to conduct our research of the existing Image Tamper detection methods. As a result, we only searched credible, peer-reviewed journals. In order to conduct our research, we employed Google Scholar, IEEEExplore, Sciencedirect, and Springer.

We used phrases and keywords like "Fake Picture Detection using Deep Learning," "Detecting Fake Images using Error Level Analysis," "Image Forgery Detection using ML Approaches," and others as part of our search criteria. In a few review papers where it wasn't in the title or abstract, we had to manually look for the search phrase. The search terms "ML for Images" or "DL for Tamper" typically returned publications that were unrelated to our survey because ML and DL have applications in many other fields. To find relevant publications, we searched for "ML for Image Tamper detection" or "Image using DL."

There exist several techniques to detect tampered images using Deep Learning models. This section discusses a few of those techniques based on their working, dataset used and accuracy.

CNN Based Techniques

Rezende et al. [1] proposed using a pre-trained ResNet-50 and transfer learning to identify computer-generated images. The proposed method can classify images between computer generated or photo-generated directly from the raw image data with no need for any pre-processing or hand-crafted feature extraction. A fully connected layer is used instead of the top layer; a Support Vector Machines (SVM) classifier is used for classification purposes. According to the authors, the SVM classifier had an average accuracy of 94.05%, while a softmax layer had an accuracy of 92.28%.

Similar to this, Sengur et al. [2] identified bogus content evidence by extracting characteristics from faces using AlexNet and VGG16. The suggested method for determining if a face is fake or real replaces the dense layer with an SVM and forgoes the fine-tuning process by importing the training weights. The authors also suggested merging the data gleaned from the two networks, which would add more details and boost prediction accuracy. For the NUAA Photograph Imposter Dataset, the integrated features produced the best model performance, with an accuracy of 88.09%.

A straightforward CNN model based on three groups of convolutional and max-pooling layers was proposed by Mo et al. [3] to detect fraudulent faces. The authors made use of a series of spatial high-pass filters, which highlight minute details on images spatially while amplifying noise in the process. The input features for the proposed CNN architecture consist of residual noise. In legitimate photos from the CELEBA-HQ dataset that were supplemented with artificial faces created using a GAN-based method, the authors reported an accuracy of 99.4%.

GAN Based Techniques

A deep forgery discriminator (DeepFD) was proposed by Hsu et al. [4] in this study to quickly and accurately identify computer-generated photographs. The authors used CNNs with a contrastive loss function to deal with false detection. The authors suggested using a fully connected layer attached to the feature extraction network to combine features collected from actual and false images for the following prediction. When comparing the feature representations of the input pair of photos, the contrastive loss function assesses whether the images are comparable or not. The proposed deep learning model can handle the fake spots in the feature representation of the images once it has been trained. The authors' suggested DeepFD was able to identify 94.7% of fraudulent photos produced by five GANs architectures.

Furthermore, Korshunov et al. [5] showed that sophisticated face recognition systems based on deep learning are susceptible to deep fake movies. The VidTIMIT dataset, which was provided by the authors, was used to test a number of baseline approaches, and the authors found that methods based on visual quality criteria, which are frequently employed in presentation attack detection, perform the best. They also demonstrate how difficult it is for conventional facial recognition algorithms to recognize deep fake videos produced by GANs. Additionally, they claim that when new deep fake technologies are developed, the situation tends to get worse.

In a study on the image-to-image translation problem, Marra et al. [6] compared a variety of deep learning algorithms using pairs of real and fake images produced using a Generative Adversarial Network architecture. The goal of the study was to evaluate deep learning architectures and steganalysis features for image-to-image translation recognition. The authors also considered how well the models performed when the image was compressed before being posted on any social network, particularly when employing the Twitter-based compression technique. The study demonstrated an accuracy of 95% attained by the XceptionNet architecture in recognizing fraudulent images while taking into account compressed circumstances, thereby attesting to the stability and viability of such models.

Using Flood Fill Algorithm

Birunda et al. [7] uses a different technique that uses the Flood Fill Algorithm to highlight the forged object and a Deep Learning-based solution to detect if the image is fake or genuine. The authors collected publicly available images from Twitter as their dataset which consisted of 800 samples. In the Flood Fill stemmer, they used two variations - 8 way and 4 way. The authors attained an accuracy of 96%.

Using Error Level Analysis

Chakraborty et al. [8] proposed using a dual-branch Convolutional Neural Network in conjunction with Error Level Analysis. The authors' methodology consisted of 4 layers - a convolutional layer, a max pooling layer, a concatenation layer, and a fully connected layer. The authors used RMSprop to constantly update parameters and reduce loss. The model was trained on the Casia 2.0 Dataset consisting of 12616 fake and authentic images, which converged at 16 epochs. The authors reported an accuracy of 98.55%.

Similarly, Qurat-ul-ain et. al [9] proposed using Convolutional Neural Networks to detect fake images while using Error Level Analysis as a pre-processing step. The authors' methodology uses a pretrained VGG-19 model with an addition of sequential model layers as output layers. The authors used RMSprop as an optimising function with 0.0001 learning rate and 5 batch size per epoch. The model was trained on the Real and Fake Face Detection Dataset by Computational Intelligence Photography Lab, Yonsei University consisting of 2041 fake and authentic images. The authors reported an accuracy of 92.09%.

Referenc e	Technique	Dataset	Input	Accura cy(%)
Rezende [1]	CNN and Transfer Learning	Public Dataset	9700 RAW Images	94%
Sengur [2]	CNN and Transfer Learning	NUAA Photograph Imposter Dataset	5761 for Testing, 3491 for Training	88.09%
Mo [3]	CNN	CelebA HQ Dataset	24000 for Training, 6000 for Validation, 30000 for Testing	99.40%
Hsu [4]	GAN	CelebA Dataset	202599 Images	94.70%
Korshunov [5]	GAN	VidTIMIT Database	320 High Quality Videos and 320 Low Quality Videos	91.03%
Marra [6]	GAN and Deep Learning	Public Dataset	36302 Images	95%

Birunda [7]	Flood Fill Algorithm	Public Images from Twitter	800 Images	96%
Chakraborty [8]	CNN and ELA	Casia 2.0	12616 Images	98.55%
Qurat-ul-ain [9]	CNN and ELA	Real and Fake Face Detection	2041 Images	92.09%

Table-I: Comparison of Existing Methods for Image Tamper Detection

Chapter 3: Analysis and Design

3.1 Proposed System Diagram

The above diagram represents our proposed system. The user would be given an option to upload an image which can either be authentic or tampered. The provided image would be passed through our pre-trained model which has been trained on CASIA 2 dataset. The model will analyse the image and give a confidence score. The tampered image would be rejected and the authentic image would be encrypted and stored into the database. The user can then access stored images through the account which the user creates.

An image to be operated on is acquired from the user. The image undergoes some processes like flattening, brightening and splicing before it can be passed through the model.

ELA is then applied to the image. Image is once reverified and converted into a pixel array for the image to be passed through the neural network. Before all of this the model is already trained and retrained as and when new images are passed through it. After processing, if the image is detected as tampered/fake, a report is generated automatically. The user has to choose from deciding whether to print the report or not.

3.2 Design Diagram of Model

An image to be operated on is acquired from the user. The image undergoes some processes like flattening, brightening and splicing before it can be passed through the model.

ELA is then applied to the image. Image is once reverified and converted into a pixel array for the image to be passed through the neural network. Before all of this the model is already trained and retrained as and when new images are passed through it. After processing, if the image is detected as tampered/fake, a report is generated automatically. The user has to choose from deciding whether to print the report or not.

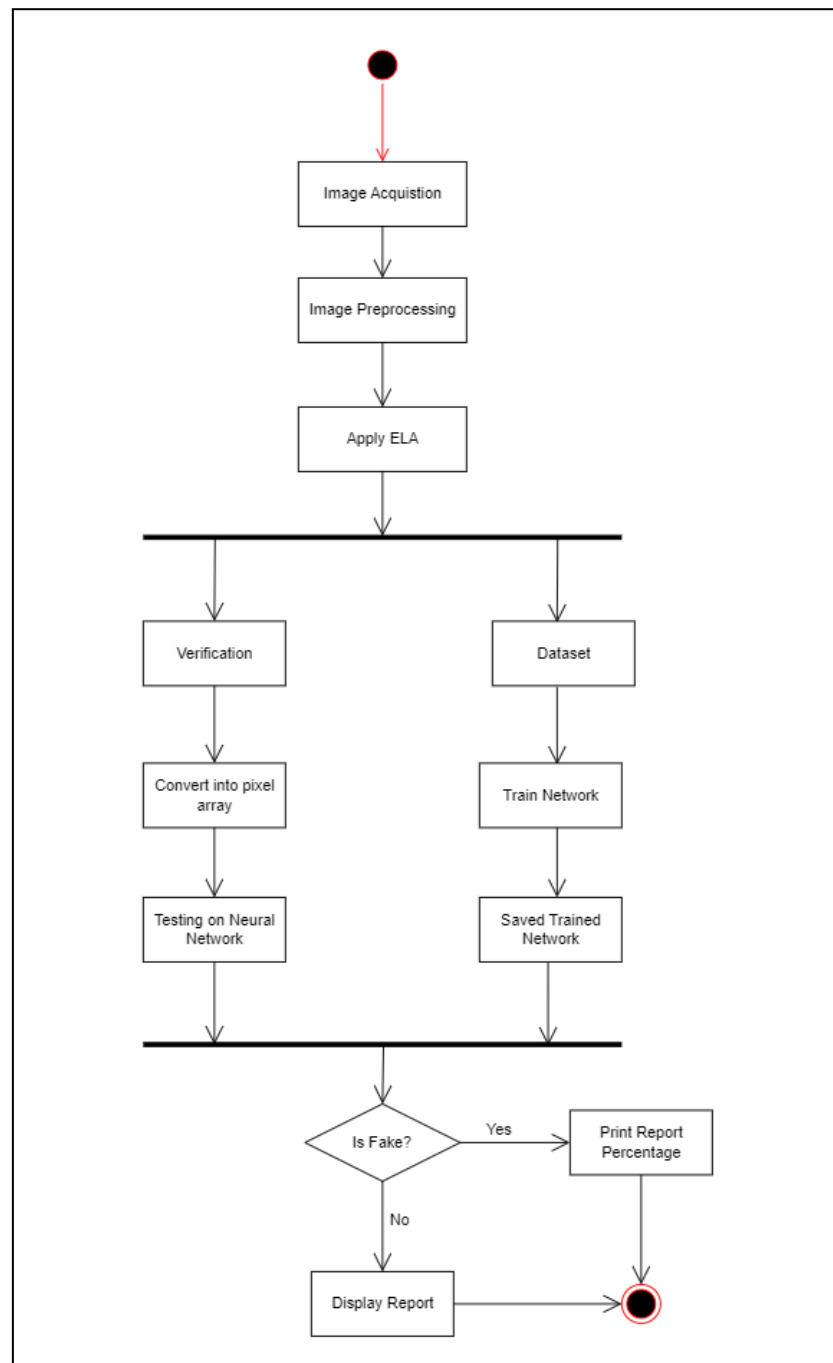


Figure 3.1: Design Diagram of Model

3.3 Proposed Methodology

For a number of reasons, it is crucial to identify altered images. Images that have been altered can be used to promote propaganda or false information. Photo editing software makes it extremely simple to change images in the modern digital era, which can be exploited to create false information or fake news. Serious repercussions may result from this, including influencing public sentiment, harming reputations, or inciting violence. Images that have been altered may be used as evidence in court. An altered photograph might not be allowed to be used as evidence in court. As a result, it's crucial to be able to see any symptoms of fabrication to make sure the proof being offered is solid and trustworthy. Moreover, spotting altered images can aid in preserving people's privacy and preventing identity theft. As social media and internet platforms have grown in popularity, images are frequently posted publicly and there is a chance that private information could be compromised. It is simpler to spot any unlawful adjustments or distortions that might be applied for malicious intentions by spotting tampered photos.

Authentic images are used as a standard against allegedly altered images. It is simpler to spot any alterations or manipulations that may have been applied to an image when there is a repository of real images. It's critical to save authentic images in their original form since they could have monetary, sentimental, or historical worth. For instance, to ensure that an image is accepted as evidence in a court proceeding, it must be authenticated and kept in its original form. Similar to this, sentimental or historical images, such images, may have cultural or emotional value that has to be preserved for future generations. Last but not least, genuine images may include important or personal information that needs to be kept private. These images can be protected and kept from illegal access or exposure by using encryption techniques.

In this research paper, we propose a website that identifies tampered images using Convolutional Neural Networks (CNN) and Error Level Analysis (ELA). The technology will use an intuitive user experience to allow users to post images, and CNN will review the uploaded images for any indications of manipulation, such as irregularities in image texture, shadows, or lighting. The ELA method will also reveal changes in picture compression between the original and modified versions. The suggested system will use encryption techniques to store genuine images safely, ensuring their security and secrecy. Real images can be uploaded by users and encrypted so that only authorised users can access and view them. Overall, the suggested approach will provide a trustworthy and simple method for spotting altered images and protecting authentic images from illegal access.

A convolutional neural network (CNN) is a type of artificial neural network which is capable of extracting features from images and doing image analysis. The lines, curves, and textures in the images can be recognized by these layers as patterns. The network can recognize more intricate patterns and structures in the image as it gains knowledge. In computer vision applications including object recognition, picture segmentation, and face detection, CNNs are frequently used. Our model consists of CNN which has 8 layers (refer Table-2). It has 2 convolutional layers which are used to find patterns and attributes in an image that might point to manipulation. These layers are intended to recognize particular visual characteristics, such as edges, corners, and textures, which can aid in differentiating between genuine and altered images. Convolutional layers are used to examine the image at various scales and orientations in the context of identifying manipulated images. This enables the CNN to recognize texture, brightness, or shadow variations that might point to tampering. The CNN may be able to identify differences in texture or illumination, for instance, if a portion of the image has been copied and pasted from another image.

The max pooling layer is typically used to reduce the dimensionality of the features detected by the convolutional layers. By minimising the quantity of data that needs to be evaluated, max pooling layers can help the CNN be more successful and efficient while detecting manipulated images. Max pooling works by choosing the maximum value within each non-overlapping sub-region of the feature map created by the convolutional layer. By repeating the features found in the convolutional layer, this lowers the number of parameters in the network and aids in preventing overfitting. Max pooling also helps the network become more resilient to slight changes in the input image because it only takes the most crucial features in each sub-region into account. Max pooling layers in the context of tampered image identification can assist in locating the most essential features for differentiating between genuine and manipulated images. The network can concentrate on the most crucial patterns and textures in the image while disregarding unimportant or distracting aspects by choosing the maximum value inside each sub-region. This can increase the CNN's precision and effectiveness, making it better at spotting altered images.

Furthermore, our CNN model has a fully-connected layer which consists of two dense and dropout layers along with one flatten layer. All of the neurons from the previous layer are connected to the current layer using dense layers. To perform classification tasks, these layers are often applied as the CNN's last layer. Dense layers can be employed in the context of tampered image detection to categorise whether an image is genuine or tampered with depending on the features collected by the convolutional layers. A probability distribution over the possible classifications, such as authentic or tampered, can be created by feeding the output of the dense layer into a softmax layer. In the CNN, dropout layers are utilised to prevent overfitting. When the network gets overly specialised to the training data and is unable to generalise to new data, overfitting takes place. Dropout layers randomly remove a

portion of the network's neurons during training, forcing the neurons that remain to pick up more durable and universal properties. By doing so, the network can avoid becoming overly dependent on the training set, which will increase its capacity to generalise to brand-new datasets. The combination of dense, dropout, and flatten layers can be utilised in tampered image detection to determine if an image is genuine or tampered with based on the features recovered by the convolutional layers.

Layer (Type)	Description
Conv2D	Find patterns and attributes in an image that might point to manipulation
Conv2D_1	Identify differences in texture or illumination
Max_Pooling2D	Reduce the dimensionality of the features detected by the convolutional layers
Dropout	Prevents Overfitting
Flatten	Converts output of Conv2D into a format that can be used for classification
Dense	Categorises whether an image is genuine or tampered with depending on the features
Dropout_1	Randomly removes a portion of the network's neurons during training
Dense_1	Classifies images based on output from convolutional layers

Table-2: Functions of each layer of CNN Model

3.4 Datasets Used

The ability to detect tampered images that could differ from the training data can be improved by employing dropout layers to train the network to generalise to new, unseen data. Put simply, dense, dropout, and flatten layers are significant parts of CNN that can be used in tampered image detection to classify images based on the features extracted by the convolutional layers, while avoiding overfitting and enhancing the network's capacity to generalise to new, unseen data.

Currently, several datasets are available for the purpose of research. Few of them include: CASIA 2.0, Columbia uncompressed image splicing detection evaluation dataset, Columbia Image Manipulation Dataset, UCID and many more (refer Table-3). All the datasets differ from each other in terms of the number of manipulated and authentic/untampered images and the manipulation techniques used. Few of the manipulation techniques appearing in these datasets include copy-move attack, image splicing, resampling, and retouching. The frequent use of these techniques in this dataset implies the frequent use of such methods in the real

world. Thus, the proposed model aims to target similar kinds of manipulation techniques to maximise the accuracy in real life applications.

The proposed model has been trained and validation using subsets of two of the datasets namely:

1) CASIA 2.0

CASIA 2.0 (CASIA WebFace Database 2.0) is a large-scale face image dataset that was collected and annotated by the Institute of Automation, Chinese Academy of Sciences. It contains approximately 10,000 subjects and over 500,000 images, making it one of the largest publicly available face recognition datasets. The images in CASIA 2.0 were obtained from the web and cover a wide range of variations in pose, expression, illumination, and occlusion. The dataset is commonly used for research and development in the field of computer vision, particularly for face recognition and face detection. It contains both original and fake (manipulated) face images. The original images are real photos of individuals, while the fake images are created by manipulating or combining existing images, for example, by changing the facial expression or swapping the face of one person with another. The fake images in CASIA 2.0 are meant to simulate real-world scenarios where face images may be deliberately manipulated to deceive face recognition systems. The dataset provides a challenging test bed for researchers to evaluate the robustness and generalisation ability of face recognition algorithms to these types of manipulations. However, it is important to note that the percentage of fake images in CASIA 2.0 may vary, and the specific details regarding the distribution and types of fake images are not publicly disclosed.

2) CG-1050

CG-1050 is a dataset collected and annotated by Maikol Castro, Dora M. Ballesteros and Diego Renza. The CG-1050 dataset, made up of 100 original images, 1050 altered images, and their related masks, is presented in this study. Four directories—original photos, modified images, mask images, and a description file—make up the dataset's structure. 85 grayscale photos and 15 colour ones can be found in the directory of original images. There are 1050 photos in the directory of tampered images that were altered by one of the following methods: copy-move, cut-paste, retouching, or colourizing. The mask directory contains the genuine mask for each set of the original and its altered image. The description file lists the image names (original, tampered, and mask), the image description, the location of the photo, the type of tampering, and the object that has been altered in the image. The dataset is useful for cases in which the dataset needs to be classified into tampered and untampered images along with labelling the manipulated pixels.

Dataset name	Dataset Description	Release Year	Tampering methods	Authentic/Tampered	Format
CASIA 2.0	CASIA 2.0 is a large-scale benchmark dataset of face images, consisting of over 12000 images, captured under different conditions and containing variations in expression, illumination, and pose.	2011	Splicing, copy-move	7491/5123	TIFF, JPEG
CG 1050	The CG-1050 dataset, made up of 100 original images, 1050 altered images, and their related masks, is presented in this study. Four directories—original photos, modified images, mask images, and a description file—make up the dataset's structure	2019	Splicing, copy-move, retouching, colourizing	1050/1050	JPEG
COVERAGE	Copymove forged (CMFD) photos and their originals with comparable but real objects(SGOs) are included in the COVERAGE dataset. The purpose of COVERAGE is to draw attention to and address the ambiguity in common tamper detection techniques brought on by self-similarity in real-world photographs.	2016	Copy-move	100/100	TIFF
Wild web	The Wild web dataset is a large-scale benchmark dataset of images which is commonly used for evaluating and developing algorithms for image classification, object detection, and other related tasks.	2015	Splicing, Copy-move, Erase-fill	0/10646	Various
CoMoFoD	The CoMoFoD (Complex Motion Dataset for Optical Flow) dataset provides ground truth optical flow data for each frame, which enables the quantitative evaluation of optical flow algorithms. The CoMoFoD dataset is widely used for developing and benchmarking state-of-the-art optical flow methods in computer vision research.	2013	Copy-Move	5200/5200	JPEG, PNG
Columbia gray	The Columbia gray dataset is a collection of 933 grayscale images of indoor and outdoor scenes with various complexities, such as clutter, occlusions, and changes in lighting conditions. The images are taken from different viewpoints, and some images contain multiple objects or regions of interest	2004	Splicing	933/912	BMP

Columbia color	The Columbia color dataset is a collection of 183 color images of varying complexities. The dataset provides ground truth annotations for several computer vision tasks, including image classification, object detection, and semantic segmentation, making it useful for developing and evaluating algorithms in these areas.	2006	Splicing	183/180	TIFF
CASIA 1.0	The CASIA 1.0 dataset is widely used for evaluating and developing algorithms for face recognition, facial expression analysis, and other related tasks in computer vision. The large number of images and individuals in the dataset, along with the variability in conditions and expressions, make it a challenging benchmark for testing the robustness and accuracy of face recognition algorithms.	2009	Splicing	800/921	JPEG

Table-3: Image Tampering Datasets Comparison

Chapter 4: Implementation and Result Discussion

4.1 Implementation - CNN model

i)

```
In [1]: #import necessary libraries
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
np.random.seed(2)
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from keras.utils.np_utils import to_categorical
from keras.models import Sequential
from keras.layers import Dense, Flatten, Conv2D, MaxPool2D, Dropout
from keras.optimizers import Adam
from keras.preprocessing.image import ImageDataGenerator
from keras.callbacks import EarlyStopping
from numpy import loadtxt
from tensorflow.keras.models import load_model

In [2]: from PIL import Image, ImageChops, ImageEnhance
import os
import itertools

In [3]: 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
```

Figure 4.1: Code for modules & ELA

First we have imported all the necessary modules which we would be using to create our model. We then define a function called “convert_to_ela_image” which would take the image

provided by the user and then convert it to an ELA image. Error Level Analysis is one of the techniques used to detect image manipulation by re-saving the image at a certain quality level and calculating the ratio between the compression levels. In general, this technique is performed on images that have a lossy format (lossy compression).

ii)

```
In [4]: real_image_path = 'CASIA2/Au/Au_an1_00001.jpg'  
Image.open(real_image_path)
```

Out[4]:



After converting to ELA image

```
In [5]: convert_to_ela_image(real_image_path, 90)
```

Out[5]:



Figure 4.2: Real image and ELA Image

The above image is an example of how an image looks after being converted to an ELA image.

iii)

```
In [8]: image_size = (128, 128)
```

```
In [9]: def prepare_image(image_path):  
        return np.array(convert_to_ela_image(image_path, 90).resize(image_size)).flatten() / 255.0
```

```
In [10]: X = [] # ELA converted images  
        Y = [] # 0 for fake, 1 for real
```

Au => Total Images 7354, Take 2100 random images from the list Tp => Total Images 2064

```
In [11]: import random  
        path = '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))
```

```
In [84]: X = np.array(X)  
        Y = to_categorical(Y, 2)  
        X = X.reshape(-1, 128, 128, 3)
```

Train Test split with 80:20 ratio

```
In [85]: X_train, X_val, Y_train, Y_val = train_test_split(X, Y, test_size = 0.2, random_state=5)  
        X = X.reshape(-1,1,1,1)  
        print(len(X_train), len(Y_train))  
        print(len(X_val), len(Y_val))  
  
3331 3331  
833 833
```

Figure 4.3: Training the model

We then proceed to train our model after reshaping the provided image into 128x128 pixels. From the dataset, the images are trained and tested with a 80:20 split ratio. Image flattening is done before feeding the images to the model. Image flattening merges all the layers of the image into a single layer. We then store the ela converted images into a list and if the images are authentic or tampered into another list while training.

iv)

```
In [86]: 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

In [87]: model = build_model()
        model.summary()
```

Model: "sequential_2"

Layer (type)	Output Shape	Param #
conv2d_4 (Conv2D)	(None, 124, 124, 32)	2432
conv2d_5 (Conv2D)	(None, 120, 120, 32)	25632
max_pooling2d_2 (MaxPooling 2D)	(None, 60, 60, 32)	0
dropout_4 (Dropout)	(None, 60, 60, 32)	0
flatten_2 (Flatten)	(None, 115200)	0
dense_4 (Dense)	(None, 256)	29491456
dropout_5 (Dropout)	(None, 256)	0
dense_5 (Dense)	(None, 2)	514

=====
Total params: 29,520,034
Trainable params: 29,520,034
Non-trainable params: 0

Figure 4.4: Saving model & Summary

The first layer of CNN consists of a convolutional layer with a kernel size of 5x5 and a number of filters 32. The second layer of CNN consists of a convolutional layer with a kernel size of 5x5 and a number of filters 32, and a Max Pooling layer with a size of 2x2. The two convolutional layers use the ReLU activation function to make the neurons in the convolutional layer select so that they can receive useful signals from the input data. After that, the MaxPooling layer added a dropout of 0.25 to prevent overfitting. Next layer is a fully connected layer with 256 neurons and a ReLU activation function. After a fully connected layer, a dropout of 0.5 will be added to prevent overfitting. The output layer used has a softmax activation function.

v)

```
In [14]: def model_predict(x,model):  
        class_names = ['fake', 'real']  
        image = prepare_image(x)  
        image = image.reshape(-1, 128, 128, 3)  
        y_pred = model.predict(image)  
        y_pred_class = np.argmax(y_pred, axis = 1)[0]  
        print(f'Class: {class_names[y_pred_class]} Confidence: {np.amax(y_pred) * 100:0.2f}')  
  
In [19]: model_predict('C:\Users\jashs\FYP\Model\CASIA2\Au\fake.webp',model)  
1/1 [=====] - 0s 31ms/step  
Class: real Confidence: 100.00
```

Figure 4.5: Testing the model

The trained model is then stored as a .h5 file and loaded when the function `model_predict` is called. The `model_predict` function contains the image which needs to be verified from the user as a parameter and in return it prints the confidence score stating if the image is authentic or tampered.

4.2 Implementation - WebApp

For the purpose of creating a WebApp, we have used modern technologies like HTML5, CSS3, Javascript for the front-end and Flask i.e a Python based framework for the back-end.

index.html: Landing page for our website

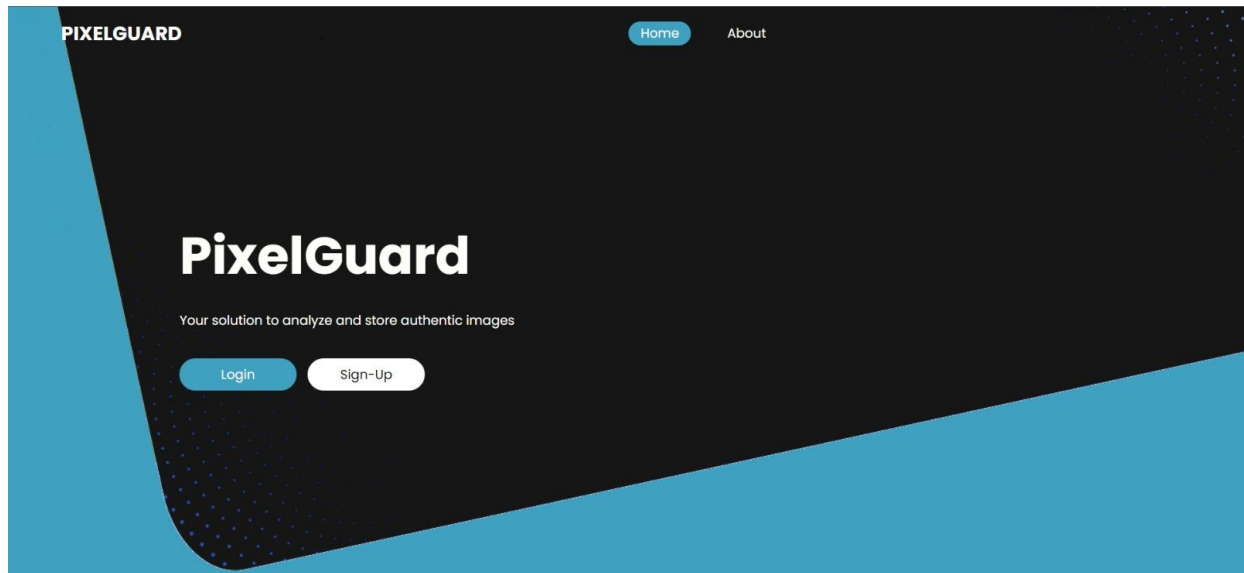


Figure 4.6: Index Page UI

login.html: login page for users to interact (code + UI)

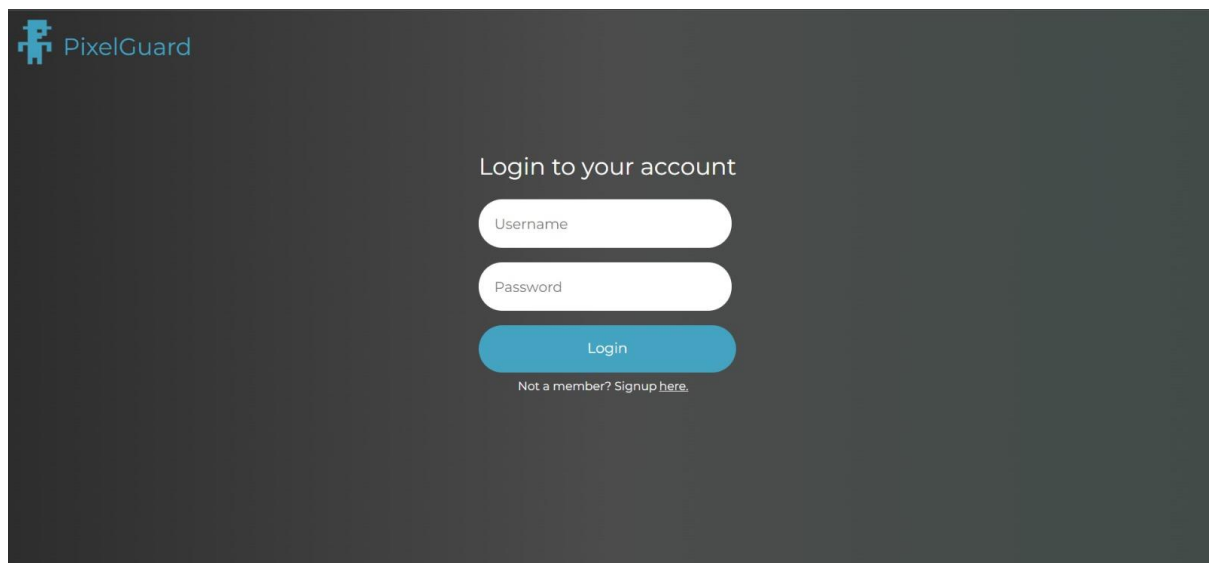


Figure 4.7: Login Page

aboutus.html: This page showcases our missions and values.

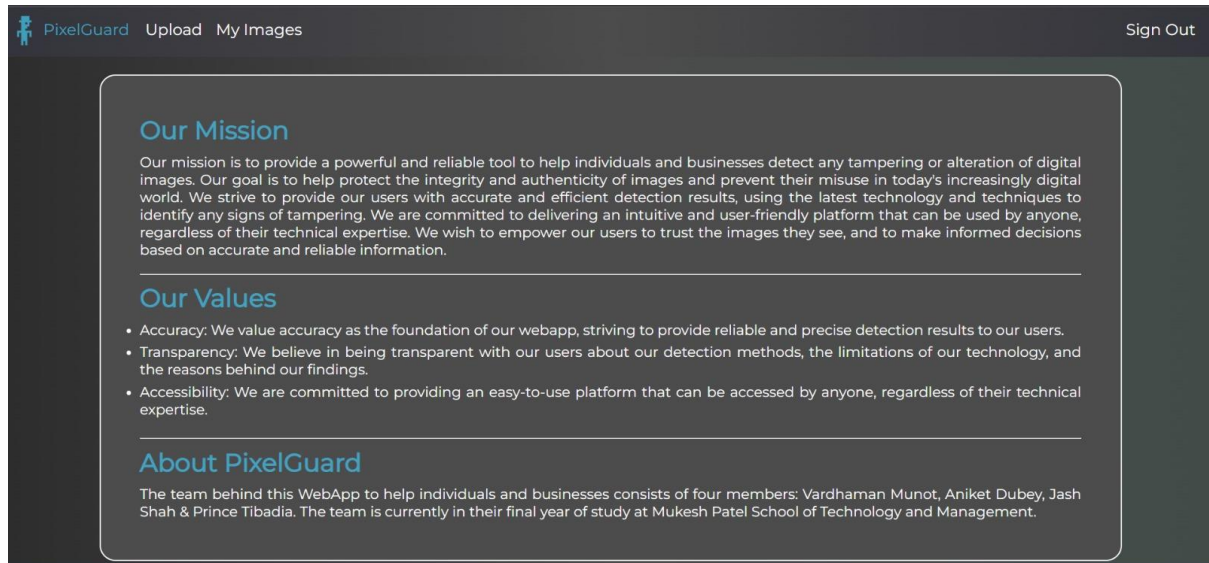


Figure 4.8: AboutUs Page

upload.html: Here the user uploads the image that is to be tested.

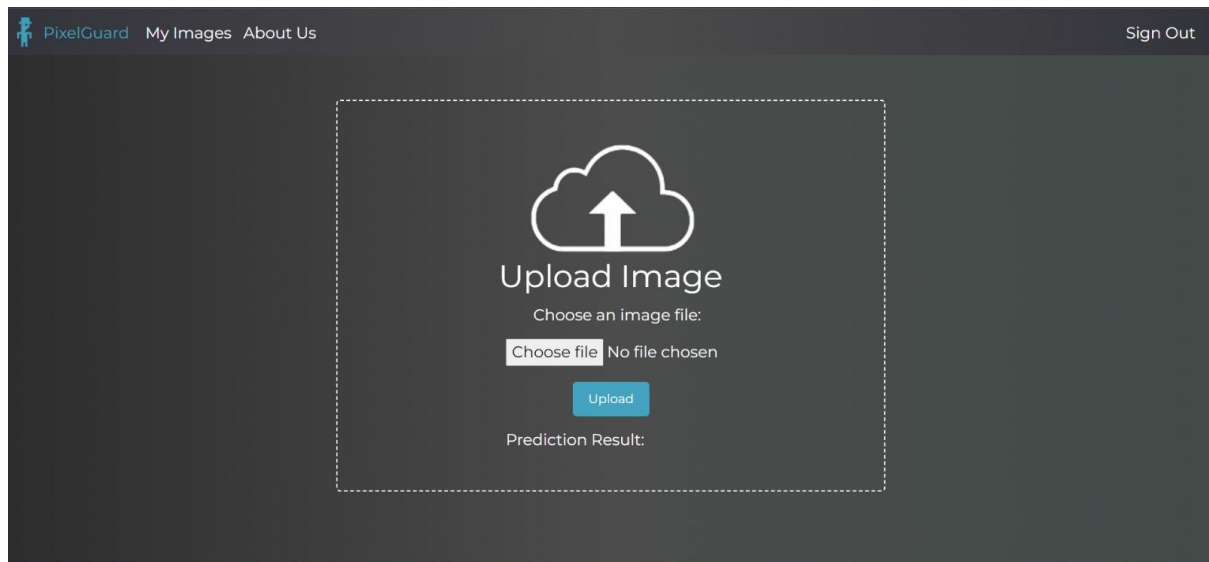


Figure 4.9: Upload Image

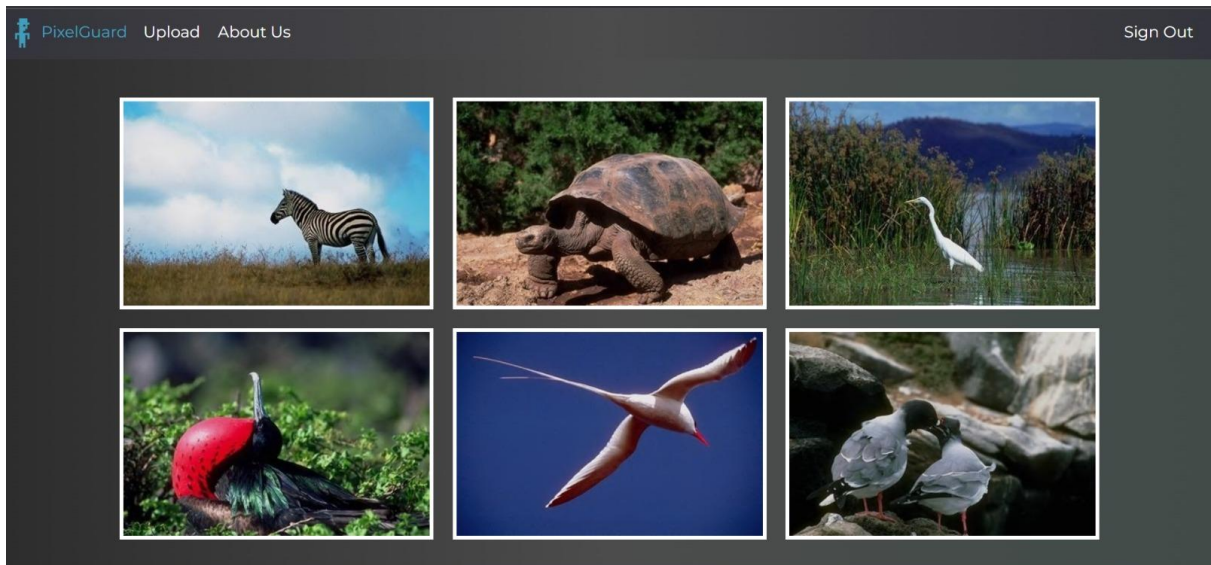


Figure 4.10: MyImages

app.py: this file is the main driver of the application and connects the front end with the back-end

```
from flask import Flask, render_template, request, redirect, url_for
from flask_sqlalchemy import SQLAlchemy
from keras.models import load_model
import numpy as np
import os
np.random.seed(2)
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from keras.utils.np_utils import to_categorical
from keras.models import Sequential
from keras.layers import Dense, Flatten, Conv2D, MaxPool2D, Dropout
from keras.optimizers import Adam
from keras.preprocessing.image import ImageDataGenerator
from keras.callbacks import EarlyStopping
from numpy import loadtxt
from PIL import Image, ImageChops, ImageEnhance
import itertools

app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///users.db'
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
app.config['SECRET_KEY'] = 'secret-key'
app.run(debug=True)
model = load_model(r'model_casia_run1.h5')

db = SQLAlchemy(app)

class User(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(50), unique=True, nullable=False)
    password = db.Column(db.String(255), nullable=False)

@app.route('/signup', methods=['GET', 'POST'])
def signup():
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']
        confirm_password = request.form['confirm_password']
        if password != confirm_password:
            error = 'Passwords do not match.'
            return render_template('signup.html', error=error)
        new_user = User(username=username, password=password)
        db.session.add(new_user)
        db.session.commit()
        return redirect(url_for('login'))
    return render_template('signup.html')

@app.route('/', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']
        user = User.query.filter_by(username=username).first()
        if not user or user.password != password:
            error = 'Invalid username or password.'
            return render_template('login.html', error=error)

        return render_template('home.html')
    return render_template('login.html')
```

Figure 4.11: Flask Driver Code

4.3 Results and Discussion

In its current state, the model delivers an accuracy of 94% and converges at 40 epochs. The model has been trained and tested with CASIA-2 Dataset. It contains 4795 images, 1701 authentic images and 3274 forged images. We have tested the model against custom unseen inputs which included both authentic and tampered images. The model has managed to detect the tampered image with a confidence score of 99.5+. The model is also tested against CG 1050 dataset which has 1050 authentic and tampered images each. The model has classified the images successfully with an average accuracy of 84%.

Dataset name	Authentic/ Tampered	Image size	Accuracy Obtained on Our Model
CASIA 2.0	7491/5123	$240 \times 160, 900 \times 600$	94%
CG 1050	1050/1050	$3456 \times 4608, 4608 \times 3456$	84%

Table-4: Quantitative Results

Chapter 5: Conclusion and Future Work

After carrying out this project on detecting image tampering using a combination of Error Level Analysis (ELA) and Convolutional Neural Networks (CNN), it can be inferred that the combined use of these techniques can be very successful in identifying tampered images. ELA is a quick and easy approach that may be used to identify portions of an image that have been compressed or edited. It compares the variations in colour values between the original and compressed images to determine how well it performs. The areas where there have been the biggest changes in colour values probably underwent manipulation. ELA has its limitations, though, as it is unable to definitively state whether or not an image has been altered. It can only give a possible location where the tampering took place. The limitations of ELA are addressed by combining it with CNNs, which can produce findings that are more precise. Deep learning models like CNNs are capable of learning patterns and characteristics from big datasets. These models can be trained to spot particular kinds of image manipulation, including copy-move or splicing. The CNN can train to detect tampering with greater accuracy by using the data supplied by ELA as input.

This project has various potential future focuses like Increasing the detection accuracy. Although the joint use of ELA and CNNs in this study produced accurate findings, the mechanism's accuracy might still be raised. To increase the accuracy of the CNN's capacity to identify tampering, future research might concentrate on optimising the model's hyperparameters, expanding the training data, and enhancing the CNN's architecture. Another future scope is adapting the method to detect video tampering: Although the focus of this study was on picture manipulation, there is an increasing need to detect video tampering, particularly in the age of deepfakes. Future studies can concentrate on improving the system for detecting video manipulation with CNNs and ELA.

Overall, there is a lot of room for further research and advancement in the area of picture tampering detection utilising ELA and CNNs, and this project has the potential to be applied to many different fields where such detection is required. However, more investigation is required to improve the model's functionality and increase its capacity to recognise more complex tampering.

References

1. Yang, Jiachen, et al. "Detecting fake images by identifying potential texture differences." *Future Generation Computer Systems* 125 (2021): 127-135.
2. Birunda, S. Selva, et al. "Fake Image Detection in Twitter using Flood Fill Algorithm and Deep Neural Networks." *2022 12th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*. IEEE, 2022.
3. Kumar, Gaayan, Sunil Kumar Chowdhary, and Abhishek Srivastava. "Intelligent Morphed Image Identification using Error Level Analysis and Deep learning." *Elementary Education Online* 20.5 (2021): 7181-7181.
4. Sudiatmika, Ida Bagus Kresna, et al. "Image forgery detection using error level analysis and deep learning." *TELKOMNIKA (Telecommunication Computing Electronics and Control)* 17.2 (2019): 653-659.
5. Tanaka, Miki, and Hitoshi Kiya. "Fake-image detection with Robust Hashing." *2021 IEEE 3rd Global Conference on Life Sciences and Technologies (LifeTech)*. IEEE, 2021.
6. Zheng, Lilei, Ying Zhang, and Vrizlynn LL Thing. "A survey on image tampering and its detection in real-world photos." *Journal of Visual Communication and Image Representation* 58 (2019): 380-399.
7. He, Peisong, Haoliang Li, and Hongxia Wang. "Detection of fake images via the ensemble of deep representations from multi-color spaces." *2019 IEEE international conference on image processing (ICIP)*. IEEE, 2019.
8. Gunawan, Agus, Holy Lovenia, and A. Pramudita. "Deteksi Pemalsuan Gambar dengan ELA dan Deep Learning." (2018).
9. Zhang, Kejun, et al. "No one can escape: A general approach to detect tampered and generated image." *IEEE Access* 7 (2019): 129494-129503.
10. Alzamil, Lubna. "Image Forgery Detection with Machine Learning." (2020).
11. Passos, Leandro A., et al. "A Review of Deep Learning-based Approaches for Deepfake Content Detection." *arXiv preprint arXiv:2202.06095* (2022).
12. Rao, Yuan, and Jiangqun Ni. "A deep learning approach to detection of splicing and copy-move forgeries in images." *2016 IEEE international workshop on information forensics and security (WIFS)*. IEEE, 2016.

13. Hamid, Yasir, et al. "An improvised CNN model for fake image detection." *International Journal of Information Technology* 15.1 (2023): 5-15.
14. Chakraborty, Sunen, Kingshuk Chatterjee, and Paramita Dey. "Detection of Image Tampering Using Deep Learning, Error Levels & Noise Residuals." (2022).
15. Hosny, Khalid M., et al. "A New Method to Detect Splicing Image Forgery Using Convolutional Neural Network." *Applied Sciences* 13.3 (2023): 1272.
16. Ali, Syed Sadaf, et al. "Image forgery detection using deep learning by recompressing images." *Electronics* 11.3 (2022): 403.
17. Cozzolino, Davide, et al. "Forensictransfer: Weakly-supervised domain adaptation for forgery detection." *arXiv preprint arXiv:1812.02510* (2018).
18. Jeronymo, Daniel Cavalcanti, Yuri Cassio Campbell Borges, and Leandro dos Santos Coelho. "Image forgery detection by semi-automatic wavelet soft-thresholding with error level analysis." *Expert Systems with Applications* 85 (2017): 348-356.
19. Sari, Wina Permana, and Hisyam Fahmi. "The effect of error level analysis on the image forgery detection using deep learning." *Kinetik: Game Technology, Information System, Computer Network, Computing, Electronics, and Control* (2021).
20. Cohen, Assaf, Aviad Cohen, and Nir Nissim. "ASSAF: Advanced and Slim StegAnalysis Detection Framework for JPEG images based on deep convolutional denoising autoencoder and Siamese networks." *Neural Networks* 131 (2020): 64-77.
21. Nida, Nudrat, Aun Irtaza, and Nouman Ilyas. "Forged face detection using ELA and Deep Learning Techniques." *2021 International Bhurban Conference on Applied Sciences and Technologies (IBCAST)*. IEEE, 2021.
22. Zhou, Peng, et al. "Learning rich features for image manipulation detection." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018.
23. Alhaidery, Manaf Mohammed Ali, Amir Hossein Taherinia, and Haider Ismael Shahadi. "A robust detection and localization technique for copy-move forgery in digital images." *Journal of King Saud University-Computer and Information Sciences* 35.1 (2023): 449-461.
24. Singh, Bhuvanesh, and Dilip Kumar Sharma. "Predicting image credibility in fake news over social media using multi-modal approach." *Neural Computing and Applications* 34.24 (2022): 21503-21517.

25. Zhang, Weiguo, and Chenggang Zhao. "Exposing face-swap images based on deep learning and ELA detection." *Proceedings*. Vol. 46. No. 1. MDPI, 2019.
26. Xue, Junxiao, et al. "Detecting fake news by exploring the consistency of multimodal data." *Information Processing & Management* 58.5 (2021): 102610.
27. Zeng, Pingping, et al. "Multitask Image Splicing Tampering Detection Based on Attention Mechanism." *Mathematics* 10.20 (2022): 3852.
28. Gideon, S. Jerome, et al. "Handwritten signature forgery detection using convolutional neural networks." *Procedia computer science* 143 (2018): 978-987.
29. Sari, Wina Permana, and Hisyam Fahmi. "The effect of error level analysis on the image forgery detection using deep learning." *Kinetik: Game Technology, Information System, Computer Network, Computing, Electronics, and Control* (2021).
30. Abd Warif, Nor Bakiah, et al. "An evaluation of Error Level Analysis in image forensics." 2015 5th IEEE international conference on system engineering and technology (ICSET). IEEE, 2015.
31. Zeng, Pingping, et al. "Multitask Image Splicing Tampering Detection Based on Attention Mechanism." *Mathematics* 10.20 (2022): 3852.
32. Bondi, Luca, et al. "Tampering Detection and Localization Through Clustering of Camera-Based CNN Features." *CVPR Workshops*. Vol. 2. 2017.