Media Forgery Detection

A major project report submitted in partial fulfillment of the requirement for the award of degree of

Bachelor of Technology

in

Computer Science & Engineering / Information Technology

Submitted by

Anubhav (201241)

Abhinav Nayak (201240)

Under the guidance & supervision of

Dr. Ruchi Verma

Assistant Professor (SG)

Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology, Waknaghat, Solan - 173234 (India)

CERTIFICATE

This is to certify that the work which is being presented in the project report titled **“Media Forgery Detection”** in partial fulfillment of the requirements for the award of the degree of B.Tech in Computer Science And Engineering and submitted to the Department of ComputerScience And Engineering, Jaypee University of Information Technology, Waknaghat is an authentic record of work carried out by **“Anubhav (201241)”**, **“Abhinav Nayak (201240)”** during the period from January 2024 to May 2024 under the supervision of **Dr.Ruchi Verma**, Assistant Professor (SG), Department of Computer Science and Engineering, Jaypee University of Information Technology, Waknaghat.

Submitted by:

Anubhav	**Abhinav Nayak**
(201241)	**(201240)**

The above statement made is correct to the best of my knowledge.

Dr. Ruchi Verma
Assistant Professor(SG)
Department of Computer Science & Engineering and Information Technology Jaypee University Of Information Technology

Candidate's Declaration

I hereby declare that the work presented in this report entitled **'Media Forgery Detection'** in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science & Engineering / Information Technology** submitted in the Department of Computer Science & Engineering and Information Technology**,** Jaypee University of Information Technology, Waknaghat is an authentic record of my own work carried out over a period from January 2024 to May 2024 under the supervision of **Dr. Ruchi Verma** (Assistant Professor (SG), Department of Computer Science & Engineering and Information Technology).

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

(Student Signature with Date)
Student Name: Anubhav
Roll No.: 201241

(Student Signature with Date)
Student Name: Abhinav Nayak
Roll No.: 201240

This is to certify that the above statement made by the candidates is true to the best of our knowledge.

(Supervisor Signature with Date)
Supervisor Name: Dr. Ruchi Verma
Designation: Assistant Professor (SG)
Department: Department of Computer Science & Engineering and Information Technology
Dated:

Acknowledgement

Firstly, we express our heartiest thanks and gratefulness to almighty God for His divine blessing makes it possible for us to complete the project work successfully. We are really grateful and wish to be profoundly indebted to Supervisor Dr. Ruchi Verma, Assistant Professor(SG), Department of CSE Jaypee University of Information Technology, Waknaghat. Deep Knowledge & keen interest of our supervisor in the field of "Artificial Intelligence /Information Security" to carry out this project. His endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, valuable advice, reading many inferior drafts and correcting them at all stages have made it possible to complete this project. We would like to express our heartiest gratitude to Dr. Ruchi Verma Department of CSE, for his kind help to finish our project. We would also generously welcome each one of those individuals who have helped us straightforwardly or in a roundabout way in making this project a win. In this unique situation, we might want to thank the various staff individuals, both educating and non-instructing, which have developed their convenient help and facilitated our undertaking. Finally, we must acknowledge with due respect the constant support and patients of our parents.

Anubhav
(201241)
Abhinav Nayak
(201240)

TABLE OF CONTENT

LIST OF FIGURES

List Of Abbreviations

1. CNN: Convolutional Neural Network
2. ELA: Error Level Analysis
3. DTCWS: Dual-Tree Complex Wavelet Transform and Singular Value Decomposition
4. PLA: Patch Level Analysis
5. GAP: Global Average Pooling
6. JPEG: Joint Photographic Experts Group
7. PNG: Portable Network Graphics
8. RGB: Red Green Blue
9. ReLU: Rectified Linear Unit
10. GPU: Graphic Processing Unit
11. ADAM: Adaptive Moment Estimation
12. CMFD: Copy Move Forgery Detection
13. SLIC: Simple Linear Iterative Clustering
14. CSS: Cascading Style-Sheets
15. HTML:Hypertext Markup Language
16. JS:JavaScript

Abstract

Throughout the last few decades, there has been a sharp rise in the use of digital images. Virtually every facet of life now makes use of digital photos. Many documents that must be filedonline require images in the form of soft copies, and a lot of images are shared on social mediaevery day. Editing or altering a digital image has become simpler thanks to the rapid advancement of digital image processing technology and the prevalence of digital cameras, evenfor a novice forger. It is conceivable for any user to alter digital images in such a way that it would be difficult to tell one from the other visually. The methods are commonly referred to asImage forgery detection for the purpose of identifying these modifications in images. The detection of fake images is a difficult field of study. It is clear that in the past decade, high- quality work has been done in the area of detecting image forgeries. Yet, given the sophisticationof image editing technologies, there is still a need to pay close attention to this subject. The primary objective of our project is to observe and work on the different techniques for detectingmedia forgery as concentrating on the copy move and the splicing assaults which are widely used.

Chapter 1: Introduction

1.1 Introduction

As widespread use of the desktop computers, proliferation among the smart devices with ever-improving cameras and image processing apps, as of the devices which are virtually always combined to the Internet with distant data servers have made it possible for regular people to gather, store, and process vast amounts of digital data that was previously unimaginable, but because there are sophisticated tools for picture editing, anyone with even a basic understanding of computers may easily alter the image. Media forgery was described "falsely and fraudulently manipulating a digital image" in the Merriam-Webster dictionary. The idea of image falsification is not new, it first appeared around 1840. The earliest altered photograph was made by the French photographer Hippolyte Bayard which was pronounced as "Self Portrait as a Drowned Man" with which Bayard claims to have committed himself. .Recently, in July 2017, a false photo of Russian President Vladimir Putin meeting with American President Donald Trump at the G20 summit was shared on social media. This bogus image earned several thousand likes and retweets[16]. Sports, legal services, medical imaging, insurance claims, andjournalism are just a few of the fields where images play an impressive role. Other fields includeforensic investigation, criminal investigation, surveillance systems, intelligence systems, and journalism. In the past ten years, significant study has been done in the area of image forgery detection. There are two different types of detection methods: active forgery detection and passive forgery detection. The digital image undergoes some sort of pre-processing in an activemethod, such as a watermark [17] or signatures [14] produced at the moment the image was created. A significantly difficult in image processing approaches is the passive approach , because the image is the only accessible information in this case. Copying and pasting (cloning)
[2] can be used to alter a picture in dependent forging. Independent forgery, on the other hand, is a type of fraud in which certain aspects of the same image are changed.

1.2 Problem Statement

The integrity of photos and films is becoming harder to guarantee in the age of digital modification. It is becoming increasingly difficult to recognize altered media information due to the development of sophisticated forgery techniques, which has serious repercussions for professions including journalism, forensics, and security. There has never been a more urgent demand for reliable forgery detection and classification systems. This project, "Media Forgery Detection Application," addresses this pressing issue by outlining a thorough solution that makes use of deep learning methods and a recently developed dataset.

The main goal of the project is to create a cutting-edge Convolutional Neural Network (CNN) model that is capable of reliably identifying and categorizing picture and video forgeries. The recently developed Forged Images Detection and Classification dataset, that contains pairs of original camera-captured pictures and the tampered pictures, that is used to train the suggested algorithm. Error Level Analysis, a novel preprocessing step, is used to improve the model's detection skills. This phase gives the CNN useful features for precise classification.

The model's design and training procedure are covered in the project detail. A thorough experimental analysis results from comparing the CNN model's performance against well-known pre-defined models utilizing a variety of dataset combinations.

With the creation of user-friendly online applications, the initiative goes beyond research and into real-world applications. The framework-based web application enables users to check for forgery detection in a simple and user-friendly manner. This program fills the gap between recently produced high-tech products and the end users who need a solid solution to protect the integrity of digital media material.

This project is an important initiative in the fight against the deepfake media. This is an age where authenticity of visual media is crucial, this project tackles the important requirement for effective counterfeit detection and classification by integrating cutting-edge deep learning techniques, a novel dataset, and a user-friendly interface.

1.3 Objectives

- **Image and Video Forgery Detection:**

  To create a reliable system that can identify and categorize many kinds of video and image forgeries, such as splicing, copy-move, and other tampering methods.

- **Enhancing Media Authentication:**

  For enhancing the steps for media content, we need authentication and verification, which can guarantee the authenticity and reliability of the pictures, that can be in the situations where we keep authenticity a crucial part in our system.

- **Model Development:**

  Our goal is to develop the model to detect the picture forgeries using Convolutional Neural Networks, and we need to preprocessed our images that we clicked with the help of Error Level Analysis. As we developed our system's so that it can be easy to evaluate the integrity of the photographs and we can categorize them as manipulated or forged images.

1.4 Significance and Motivation of the Project Work

The significance of our project work is that even if there are several ways to detect fake images, it could be difficult for us to decide on which ones are the most useful and effective, as people can trust it also.

With the help of the techniques of detection, we try to build the project so it can attempt to classify our algorithms into five different categories i.e. JPEG Compression Quantization, Edge Detection, Clone Detection, Resampling Detection, and Light Color Anomaly Detection. With the help of these groups technique that we will be studied to assess the effectiveness of the algorithms that can be established.

Our project's main work is to thoroughly investigate and work on to analyze the various techniques and algorithms for identifying fake images.
Our project will implement and test the most accurate and reliable algorithm, with a focus on detecting the different types of forgery.

We use extensive testing using a library of images that we will be performed to assess the success rate of our implemented algorithms, as we also take consideration of their false positive rates, and runtime. With the help of variance, we determined the same algorithm that we will test in determining the changes withing their parameters effect that will affect the performance on different types of images.

With the help of these results from this project research will be valuable for improving the credibility of images used in the media and in daily life. Our project aims to provide the insight of detecting the forgery and to identify the most effective algorithms for different types of images.

1.5 Organization of Project Report

The rest of this report is organized as follows:

Chapter 2 gives an overview of the literature study performed.

Chapter 3 discusses the system development and workflow.

Chapter 4 shows the testing phase

Chapter 5 Highlights the conclusion, future scope and application contributions

Chapter 2: Literature Survey

2.1 Overview of Relevant Literature

Detection of the image forgery has been the subject of numerous surveys over the decade. **Lanh et al.** [19] described numerous methods for image counterfeit clicked by a camera. The Author made a clear statement that, in terms of dependability, camera-based systems outperform other forgery detection methods.

Pixel-based approaches, format-based techniques, camera-based techniques, physically based techniques, and geometric based techniques are the five groups that Farid [16] divided under image counterfeiting tools. The author has thoroughly explained each technique.

Meena K.B, Tyagi V [20] reviewed four techniques that are used for tampering techniques. This paper conducts a survey for deep learning techniques for copy-move and splicing forgery. These methods actually date back to the pre-DL era, which is the one we are now in, and as a result, their eventual training phase requires little to no data. Old machine learning methods, like regression techniques, (SVM) etc. are often used in those that still need training data. While they rely on models with a manageable number of parameters and don't necessitate a large amount of training data and classify here as classic approaches. To speed up process of training and obtain a good performance, deep learning models can also be employed in conjunction with some of the fundamental concepts and principles on which these approaches are based.

Saboor Koul [1] With the advent of unique technologies and non-traditional image forging equipment and procedures, digital imaging has become basic. It is not even possible to use digital picture forgeries as proof anywhere because we are aware of their possibility. Analyzing this fact, we need to go into the incomprehensible problem in order to assist in reducing such neglect. In this digital kingdom, copy-move and image splicing to construct a fabricated one are dominant practices. In a copy-move, a portion of the image is copied and pasted to another portion of the image, whereas in a merger, two images are combined to drastically alter the original and produce a new forged image. This paper proposes a fresh method for automatic copy-move forgery detection using a convolutional neural network (CNN). For the experimental effort, 2000 images—1300 of which are authentic and 700 of which are fake—from the benchmark dataset MICC-F2000 are taken into consideration. According to the experimental findings, the suggested model works better at copy-move forgery detection than the other conventional techniques. Copy-move forgery produced extremely promising results, with an accuracy of 97.52%—2.52% greater than the methods now in use.

The paper's suggested approach centers on applying a three-layered Convolutional Neural Network (CNN) to identify copy-move forgeries in images. The first step in the process is to separate an image dataset into testing and training sets. The distinctive qualities of every image are then represented by the extraction of image features. Then, using the learned features, the CNN—a deep learning classifier—is used to automatically learn and categorize images. The novel aspect of the authors' approach—which uses deep learning to directly classify images and capture both tampered and untampered variations—is highlighted. The inclusion of a dropout layer reduces overfitting. In order to enhance its capacity to discriminate between real and altered images, the model uses the backpropagation algorithm to modify its weights during the training phase. The overall goal of the suggested method is to improve image forgery detection by utilizing deep neural networks' automated learning capabilities.

M. Elaskily [2] he only information source that can be widely shared and visualized in virtual conferences and social media during this pandemic is digital images, and sharing documents for the purpose of detecting forgeries is difficult. For the purpose of detecting such forgeries, particularly in light of the current pandemic, an effective novel Discrete-Time Cosine Wavelet and Spatial (DTCWS) Markov feature-based algorithm has been developed in this paper. In the DTCWS domain, high-dimensional Markov features were extracted for this work, and PCA was used in order to reduce the dimensions for these Markov attributes. there is now more correlation between the coefficients thanks to the co-occurrence matrix. Rather than using a support vector machine classifier, an optimized ensemble classifier is used to evaluate the results for classification. Owing to the time constraints associated with online activities, the suggested algorithm outperforms the current work in terms of accuracy, achieving 99.9%, while requiring less time and complex operations.

A feature extraction technique with three domains—spatial, DCT (Block Discrete Cosine Transform), and DWT (Dobchies wavelet transform)—is included in the framework. The procedure entails computing difference arrays for every domain after pre-processing to manage low-quality pixels produced during splicing. Markov modeling was used in DTCWS algorithm for emphasize splicing effect. Transition probability matrices are applied after thresholding to produce Markov features. Principal Component Analysis, or PCA, is used to perform feature reduction in order to lower computational complexity. The correlation matrix was created to analyze attribute correlation. Ultimately, the ensemble classifier is used to effectively assess the outcomes.
The ensemble classifier is appropriate for the COVID-19 scenario since it integrates choices made by base learners to improve classification speed and accuracy. The methodology offers promising results for image forgery detection and is tailored to handle high-dimensional features in the DTCWS domain.

M. Elaskily [3] Digital images are commonplace in today's technologically advanced world. They could be created as well as altered using various hardware and software. Copy move forgery can be regarded as a manipulation method for a image which produces tampered images by hiding undesirable material or duplicating it. As a result, verifying the content of images has become crucial. It proposes a design which is based on novel deep learning and is used for detecting image forgeries. CMFD is the exclusive use case for a Convolutional Neural Network (CNN). To distinguish between tampered and original images, CNN is used tolearn hierarchical feature representations from input images. The comprehensive tests show thaton the three publicly available datasets—MICC-F220, MICC-F2000, and MICC-F600—the suggested deep CMFD algorithm performs noticeably better than the traditional CMFD systems.In addition, new dataset combinations are created by integrating and joining the three datasets with the SATs-130 dataset.

This work aims to provide an effective way of deep CMFD algorithm with an emphasis on obtaining low computational cost and high performance. Based on the CNN architecture, algorithm emphasizes a non-block-based method that works with the entire image. Three stages make up the methodology: feature extraction, classification, and pre-processing. Pre-processing involves resizing input images to a target size without any part of them being cropped. Six convolution (CNV) layers are used to extract features, a max-pooling layer comes after each, and the GAP layer is being applied for reduction of data.

Convolutional layers perform the function of feature extractors, producing feature maps. By minimizing parameters, the GAP layer minimizes overfitting and promotes the similarity with the structure of convolution, which can be used as a structural regularizer. The dense layer has a soft-max activation function that acts as a distance metric for self-similarity between image patches. The methodology offers promising results for image forgery detection and is tailored to handle high-dimensional features in the DTCWS domain.

Verma et al[4] Image manipulation tools are so readily available, forged JPEG images are commonplace in today's world. This paper presents a deep learning-based system that, based on detecting single and double compressed blocks, distinguishes between original and forged regions in a JPEG image without fully decompressing the image. The system makes use of the inherent relationship between histograms of quantized DCT coefficients and corresponding quantization step sizes. To address the suggested input representation for single vs. double JPEG compression detection, we suggest a novel combination of raw histograms of the quantized DCT coefficients and matching quantization step sizes. This combination is then used as input to the corresponding standard CNN architectures.
It is demonstrated that the suggested input has negligible impact on performance using a range of conventional CNN architectures that are intended to manage the suggested input for the given task. More specifically, we have extracted the compression-specific artifacts from the suggested input for the extra experiments documented in this work using DenseNet.
Dataset that was made available to the public and was created using a variety of 1,120 quantization matrices. When learning the compression artifacts, using the suggested input performs better than the baseline methods for blocks with sizes of 256 × 256, 128 × 128, and 64 × 64. The suggested method performs better than the baseline methods when test blocks are compressed using quantization matrices that are entirely different from those used in training.

In order to successfully localize tampered regions, the paper presents a novel method for detecting single and double compressed blocks in JPEG images. It starts off by giving some background on image compression, with a focus on the popular lossy JPEG compression standard. An RGB image is converted to a YCbCr color space as part of the JPEG compression process. Down sampling and the 2D Discrete Cosine Transform (DCT) are then used to obtain coefficients. A quantization matrix is used to quantize these coefficients, resulting in a lossy compression process. Creating the best possible input representation for a Convolutional Neural Network (CNN) to differentiate between single and double compressed JPEG patches is the main goal of the suggested methodology. Quantization factors and DCT coefficient histograms for different frequencies are combined in the input representation.

Francesco Marra[5] Current deep learning models only accept relatively small images as input because there is a lack of memory resources, necessitating initial image resizing. High-level vision problems do not have this issue because resizing hardly affects discriminative features. Conversely, resizing negatively affects performance in image forensics because it frequently obliterates important high-frequency details. Patch-wise processing can be used to avoid resizing at the expense of forgoing whole-image analysis. In this study, we present a CNN-based framework for detecting image forgeries that bases its judgments on comprehensive, full-resolution data collected from the entire image. The framework is trainable end-to-end with limited memory resources and weak (image-level) supervision because of gradient checkpointing, which enables the joint optimization of all parameters.

Regardless of the size of the forgery and the image, researchers have developed a deep learning-based method to identify localized forgeries in images. The challenge of handling images with different resolutions is what inspired this work, especially since deep networks frequently require smaller input sizes and advances in camera technologies may capture high- resolution images. Forgery detection, it is considered impractical to use the traditional methodsof resizing images to fit the network input or processing images patch-wise. Important details may be lost due to resizing, particularly in the case of minor forgeries.
An end-to-end trainable deep neural network that can handle full-size images is used in the suggested framework. The three primary building blocks of architecture are decision, feature aggregation, and patch-level feature extraction. To extract discriminative features, such as image noise prints that highlight spatial anomalies, overlapping patches are processed as part of the patch-level feature extraction process. Using a variety of pooling techniques, feature aggregation combines these features into a single descriptor for the classification task. The gradient checkpointing strategy is utilized by the authors to address the difficulty of processing large images with constrained memory. This approach entails computing activations only from the last checkpoint onwards during the backward pass, and storing only a small number of "checkpoint" layer activations during the forward pass.

Ritu Agarwal[6] The proliferation of media tampering techniques has led to a rapid increase in image forgery activities. Attackers carry out such actions with the goal of slandering individuals and websites, obtaining financial gain, extortion, etc. There are several ways to carry out image forgeries, and copy-move forgery is one of them. Media tampering is necessary for safeguarding authenticity. In the system to identify tampered region, it is suggested for the algo to uses tampered image as input. Segmentation and feature extraction are all part of our system. The suggested deep learning-based system can identify duplicate regions more accurately and with less computational effort.

The copy-move forgery, a common manipulation made possible by a variety of image editing tools, is the focus of the suggested image forgery detection methodology. Multi-step procedures are involved in the process. First, a corrupted image is preprocessed by segmenting it using color similarity SLIC. Using the deep learning technique called VGGNet, which can identify replicated regions even when they are undergoing changes like rotation, scaling, or compression, the segmented patches are subjected to feature extraction. An adaptive patch matching technique is used to reconstruct each patch's depth for block comparison. By comparing segmented patches tampered parts are found, with matching crucial points. Input images manipulated areas are revealed by this process. SLIC, a clustering algorithm that makes use of color values and pixel positions, is the foundation for image segmentation. VGGNet is used in feature extraction to extract multi-scale features that are resilient to different types of attacks. Accurately identifying tampered regions is made easier by dense depth reconstruction, and the detection process is completed by adaptive patch matching. This all-encompassing method improves the effectiveness of forgery detection, especially when copy-move manipulation is involved.

2.2 Key Gaps in the Literature

Restricted Assessment of Real-World Datasets: A thorough evaluation on a variety of real-world datasets is lacking in many papers. To show the robustness and generalizability of the suggested techniques, testing them on a larger set of images with different content and quality is crucial.

Scalability Difficulties: It's possible that some papers don't discuss how scalable the suggested techniques are. For practical applications, handling large-scale datasets and high-resolution images is essential, and this area needs more research.
Insufficient Comparative Analysis

Selected Analysis of Attack Situations: A more thorough investigation of potential attack scenarios ought to be part of the evaluation of forgery detection techniques. Determining the effectiveness of the suggested methods against more complex adversarial attacks than copy-move forgeries is essential to determining their practicality.

Efficiency of Computation: There isn't always a full discussion of the suggested methods' computational efficiency. When it comes to the practical implementation of forgery detection in large-scale or real-time applications, processing speed optimization of algorithms ought to be the top concern.

Restricted Investigation of Deep Learning Frameworks: Some papers employ deep learning techniques such as VGGNet, there may be a lack of research comparing and examining different deep learning architectures. Examining the efficacy of more recent architectures or model combinations may offer ways to increase the accuracy of forgery detection.

Instantaneous Applicability: Not every time is the real-time applicability of the suggested methods discussed. Practical utility requires an understanding of the computational requirements and deployment feasibility in real-time scenarios, like online image sharing platforms.

2.3 Proposed System

Deep learning techniques have become incredibly popular and have been used to solve a wide range of scientific issues. This is because it has been demonstrated that they excel in classification difficulties as well as regression and segmentation problems. Some techniques can even be executed more accurately and precisely than humans for some tasks. CNN has already proven to have exceptional potential in several computer vision applications, such as object detection and image segmentation. Due to the diverse origins of the images, a range ofdistortions appear when a piece of an image is transferred from one to another. CNNs can spot these distortions in falsified images even though they may be invisible to the unaided eye.

An internal dataset called FIDAC adds another level of analysis by including images that were originally taken with a camera and then edited using widely used programs such as Adobe Photoshop, B612, Picsarts, and Canva. The collection purposefully contains a variety of forgeries, such as splicing, copy-move, retouching, recoloring, rotating, scaling, and filtering. By exposing the trained model to a wider variety of forgery scenarios, FIDAC is intended to improve the model's accuracy and realism.

For Preprocessing the data we used Error Level Analysis (ELA) is a critical step in preparing the input data for the suggested Convolutional Neural Network (CNN) architecture. This method increases the model's sensitivity to changes and irregularities brought about by the forging process. The first step in the ELA process is to divide the image into smaller pieces. Each of these pieces is then separately recompressed using error level analysis, with a predetermined error rate of 95%. The primary calculation for ELA is figuring out the absolute difference, over all color channels, between the recompressed image (Xrc) and the suspected forged image (X). The calculation is expressed mathematically as:

$$\boldsymbol{ELA(p_1, p_2) = \frac{1}{3}\sum_{i=1}^{3} |X(p_1, p_2, i) - X_{rc}(p_1, p_2, i)|}$$

The sum of all error levels for each color channel in an RGB image is used to calculate the total error levels:

$$\boldsymbol{ELA(p_1, p_2) = |\ X(p_1, p_2) - X_{rc}(p_1, p_2)|}$$

Here, p_1 and $p_2$represent row and column indices the color channel is represented by i, and the suspected forged image and recompressed image are denoted by X and Xrc, respectively. This ELA preprocessing step makes sure that the model learns to comprehend the image in a more sophisticated way, especially in areas where manipulation has taken place.

Suggested CNN Architecture: The Convolutional Neural Network (CNN) architecture, painstakingly crafted to extract minute features and patterns suggestive of image forgery, is the brains behind the suggested system. Each of the 15 layers in the feature extraction and classification process has a distinct function. Images with 128x128x3 (RGB) dimensions can be accommodated in the first layer, known as the input layer. Convolutional and pooling operations are used in subsequent layers to gradually reduce the image's dimensions while maintaining its essential features. Rectified Linear Unit (ReLU) activation functions are a useful tool for adding non-linearity to convolutional layers of a model. This helps the model better capture the intricate relationships present in the data. Each fully connected layer in the architecture adds to the input data's hierarchical understanding.

The last layer, which has a sigmoid activation function, makes sure that binary classification is maintained by identifying whether or not the image has been altered.

To sum up, the suggested system combines careful dataset selection, efficient ELA preprocessing, and an advanced CNN architecture to move through an extensive procedure. The combination of the intricate ELA preprocessing and the diversity provided by the CASIA and FIDAC datasets enables the CNN to identify a broad range of forgery techniques, making it a reliable solution for image tampering detection in practical applications.

Now for our addition of web application in our project:-

1. **Designing User Interfaces:**

   Specify the web application's user interface (UI), including its design, functionality, and interactive components.
   Analysis: Examine user preferences and expectations for utilizing the web interface to communicate with the forgery detection system. Think about the principles of user experience (UX), accessibility, and usability.

2. **Managing Input and Output:**

   Describe the process by which users will submit photographs for the purpose of detecting forgeries and how the results will be displayed.

   Analysis: Examine the various ways to upload photos (such as file uploads and URL inputs) and exhibit the results of image detection (such as confidence scores and displayed alterations). Take into account security protocols for managing user input.

3. **Django Framework Integration:**

   Requirement: Specify Django integration needs, including how to build up routes, views, and web application templates.

   Analysis: Examine Django's web application development capabilities and consider how to best utilize its features to construct a forgery detection system. Think about the database administration, authentication, and authorization capabilities of Django.

4. **Scalability and Deployment:**

   Requirement: Indicate the web application's scalability needs and deployment environment.

   Analysis: Examine several deployment alternatives, such as dedicated servers or cloud platforms like AWS and Google Cloud, for hosting the Django application. Think about scalability issues like managing several users at once and speed optimization.

5. **Privacy and Security:**

   Establish security protocols to safeguard user information and stop illegal access to the program.
   Analysis: Examine security vulnerabilities related to picture uploads and findings of forgery detection. To protect user privacy and stop criminal activity, put in place safeguards like input validation, secure authentication, and encryption.

Chapter 3: System Development

3.1 Requirements and Analysis

Our project goal was to developed the media forgery for the system which uses CNN to detect image category and Error Level Analysis (ELA) as a preprocessing technique. By precisely locating altered areas in digital photos, the main objective is to improve digital forensics and content integrity verification. Now, we have extended our project and created a web application for image forgery detection.

3.1.1 Functional Requirements

- ELA image preprocessing to find possibly tampered regions.
- CNN architecture put into practice for image classification.
- Integration for testing and training with external datasets, such as FIDAC and CASIA.
- Reaching a rate of accuracy in forgery detection that is higher than predetermined thresholds.
- Uploading photos should be possible for users to use in forgery detection.
- Popular picture formats including JPEG, PNG, and BMP ought to be supported.
- The forgery detection model should be used by the system to process uploaded photos.
- Images that have been uploaded with alterations (such as splicing or copy-paste) detected should be marked or commented.
- Confidence scores and details regarding the kind of forgery found should be included in the detection findings.

3.1.2 Non Functional Requirements

- Response time of no more than [insert time] for forgery detection.
- Scalability of the system to support a wide variety of images.
- strong dependability and resilience in the detection of forgeries.
- The forgery detection model should be used by the system to process uploaded photos. Images that have been uploaded with alterations (such as splicing or copy-paste) detected should be marked or commented.
- Confidence scores and details regarding the kind of forgery found should be included in the detection findings.
- To avoid unwanted access, safe authentication methods should be implemented by the system.
- To safeguard user privacy, uploaded photographs and detection results should be encrypted both during transmission and storage.
- It is important to take precautions against security flaws like SQL injection and cross-site scripting (XSS) attacks.

3.1.3 Requirements for Data

- Using the FIDAC and CASIA datasets for testing and training.
- Steps in the preprocessing stage, such as augmentation and normalization, to guarantee data quality.

Analysis

3.1.4 Methodology of Analysis

- Justification for selecting ELA as a preprocessing method, focusing on how well it can draw attention to differences in error levels.
- Explanation of why a CNN was used, pointing out that it works well at learning hierarchical features for image classification.
- Verify that the web application's features—such as picture forgery detection, authentication, and user registration—all function as expected.
- To ensure that the program is acting correctly and consistently, test various scenarios and user interactions.
- Assess the speed of your web application by monitoring server-side processing and HTTP request response times.
- Track how much time is spent on CPU, memory, and disk input/output to spot slow spots and improve system efficiency.

3.1.5 Evaluation of Feasibility

- **Technical viability:** CNN and ELA are well-known and frequently utilized in image processing and machine learning.
- **Economic viability:** Considering that open-source tools and datasets are readily available, the project is financially feasible.
- Ascertain whether the hardware, software, and human resources needed to create, implement, and maintain the web application are readily available.
- Examine the viability and associated costs of hosting the Django application on appropriate infrastructure, such as dedicated servers or cloud platforms.
- Take into account the accessibility of knowledgeable experts who can help the project

and have experience with Django development and image processing methods.

3.1.6 Examination of Risks

- Identification of possible hazards, such as restricted datasets, complex algorithms, and computational resource limitations.
- Strategies for risk mitigation for each identified threat.

WebApp(Risks):-

- Data Security: Dangers pertaining to user data security, such as submitted photos and detection outcomes. Unauthorized access, data breaches, and data loss are examples of potential hazards.
- Vulnerabilities: Dangers arising from online application security flaws including SQL injection, cross-site scripting (XSS), and bypassing authentication. If these vulnerabilities are not fixed, bad actors may take advantage of them.
- Scalability: The potential for increasing user traffic or concurrent requests to cause downtime or performance deterioration. A large number of picture uploads and processing requests may be too much for the program to manage, which could cause server crashes or sluggish response times.
- Resource Utilization: This section discusses the risks of inefficient resource use, including high CPU consumption, memory leakage, and database bottlenecks. Ineffective methods or poorly optimized code may put a burden on server resources and reduce the performance of the program.
- Integration Challenges: Dangers involved in incorporating the model for detecting picture forgeries into the Django program. Project deadlines may be impacted by compatibility problems, version inconsistencies, or API modifications that interfere with the integration process.

- Technology Stack: This section discusses potential hazards associated with the Django, Python, and third-party library technology stack utilized in development.

3.2 Project Design and Architecture

- **Dataset Preparation:**

    - Gathering and preparing camera clicked picture and sample videos for creating the necessary datasets.
    - Split data into training, validation, and test sets

- **Model Development:**

    - Implement proposed CNN architecture
    - Include ELA preprocessing for image and video conversion

- **Testing & Evaluation:**

    - Compare proposed model with VGG16, VGG19, Inception and evaluating accuracy.
    - Test model on mixed CASIA-FIDAC test dataset and Visualizing results with confusion matrices and loss curves

DFD Diagram

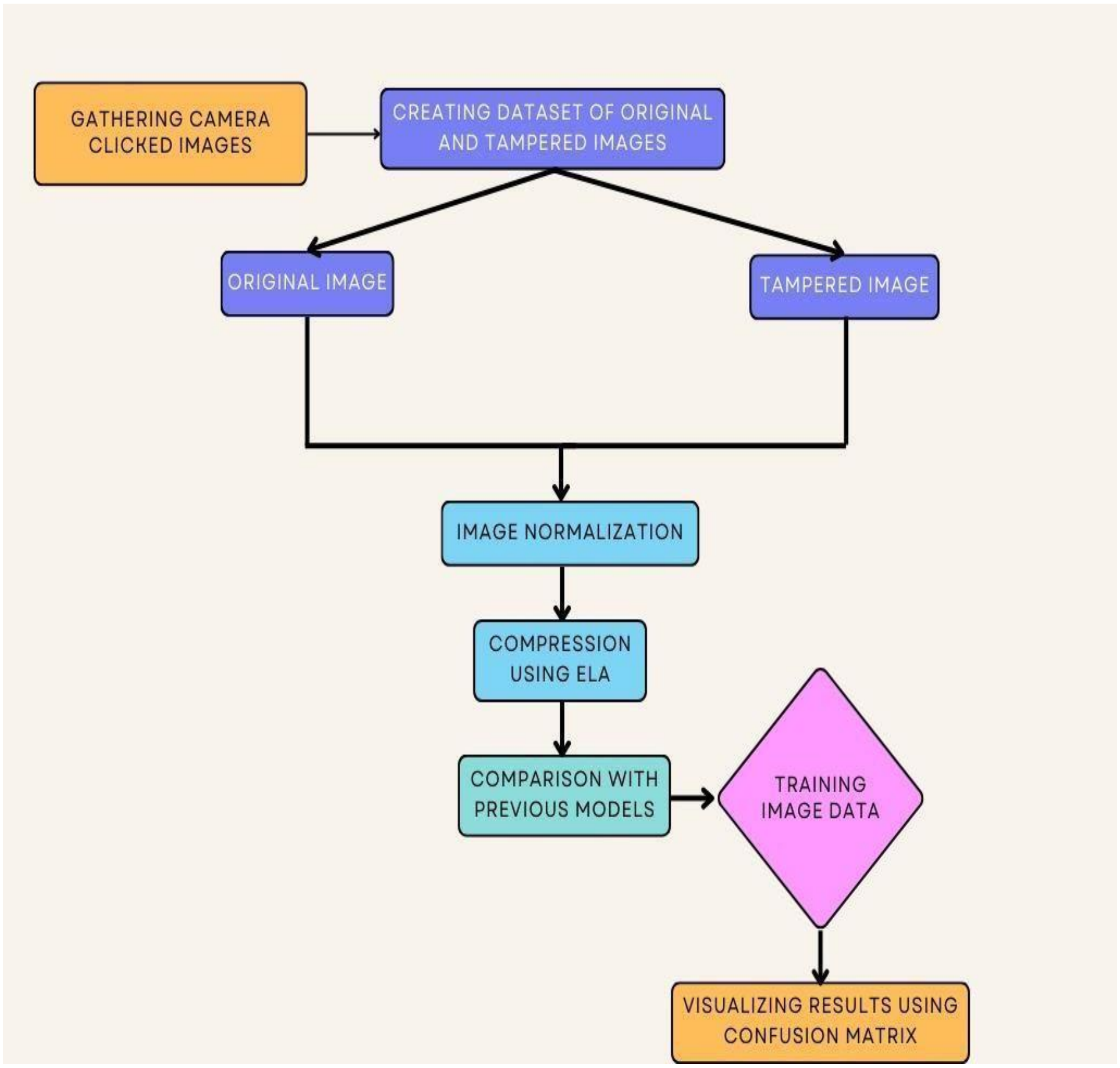

GATHERING CAMERA CLICKED IMAGES
CREATING DATASET OF ORIGINAL AND TAMPERED IMAGES
ORIGINAL IMAGE
TAMPERED IMAGE
IMAGE NORMALIZATION
COMPRESSION USING ELA
COMPARISON WITH PREVIOUS MODELS
TRAINING IMAGE DATA
VISUALIZING RESULTS USING CONFUSION MATRIX

Fig.1.3 Data Flow Diagram to represent Image Forgery Detection

<u>ER Diagram</u>

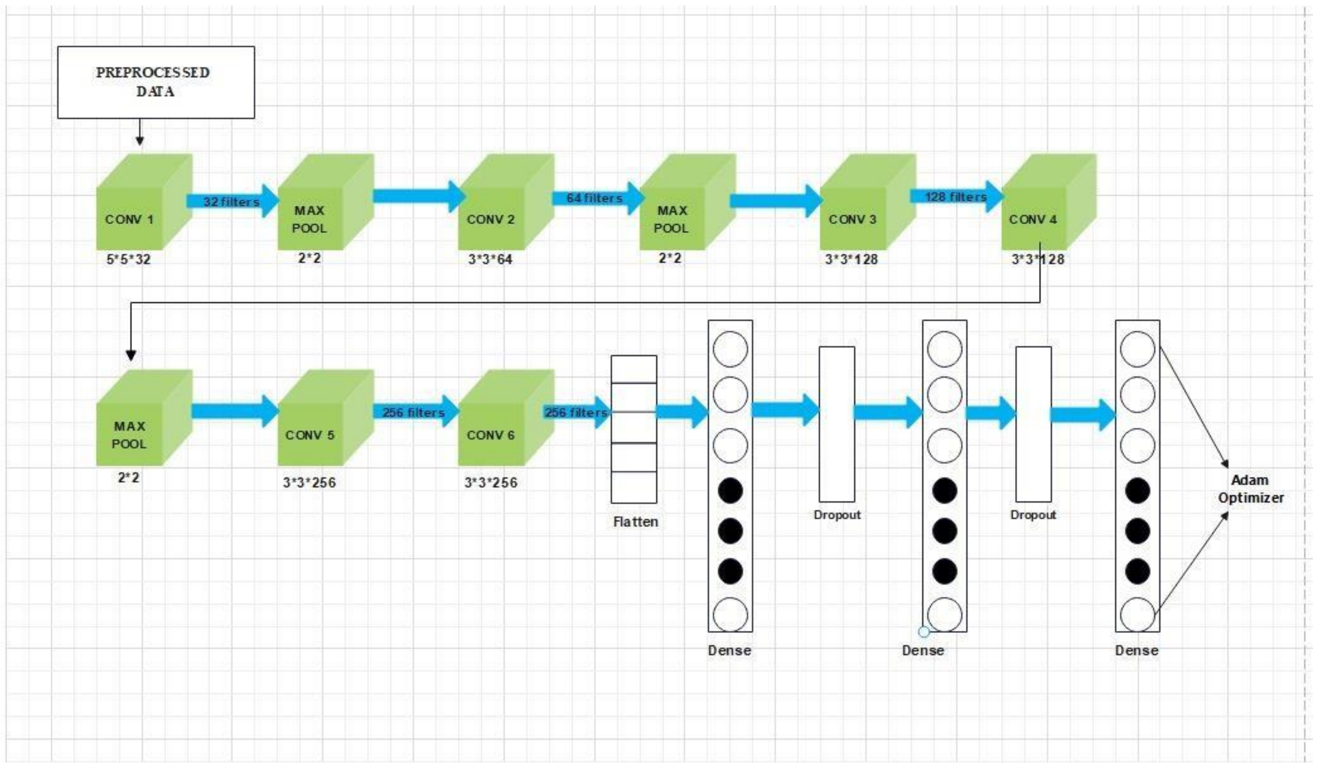


Fig.1.4 Block diagram of proposed Image Forgery Detection model

3.3 Data Preparation

1. Data Collection:

Firstly, we Collected photos taken with a camera click for the FIDAC dataset, used a variety of programs, including Canva, B612, Picsarts, and Adobe Photoshop, to alter images. To produce a heterogeneous dataset using a range of forgeries methods such as splicing, rotation, scaling, filtering, retouching, and recoloring.

2. Dataset Selection:

Then we take two significant datasets were selected for algorithm analysis and testing.
A popular database for identifying visual media forgeries is the CASIA Dataset.
includes pictures divided into eight groups, some of which have had parts altered by pasting, cutting, and other techniques, Then we use our own custom dataset FIDAC Dataset: An original dataset of photos taken with our camera. With the help of applications we used it to further alter images in order to simulate real-world scenarios.

3. Preprocessing Method:

We used Error Level Analysis (ELA) to implement the preprocessing method that divided pictures into the more manageable sections of chunks of images were individually recompressed using ELA with a predetermined 95% error rate.
In this we determined the exact difference between the re-compressed and analyzed images.

4. Suggested CNN Architecture:

Then we use Convolutional neural networks(CNN), were created with the purpose of detecting image forgeries. The 15 layers are configured with dropout, dense, convolution, pooling, and flattening operations.
We used Rectified Linear Unit (ReLU) was employed as the non-linearity activation function. To determine whether photos are authentic or manipulated, then Sigmoid activation function was applied to the final output layer.

5. Phase of Implementation

Firstly we do Data Preprocessing to Enhanced forgery detection by applying ELA representations to the images datasets that are ready to be used with the suggested CNN architecture.
Then we perform ELA Implementation to the pipeline for detecting image forgeries to make sure that the possible tampering regions were properly highlighted by the ELA.

6. CNN Model Development

We used TensorFlow or a comparable deep learning framework to implement the planned CNN architecture. Model layers, activation functions, and loss functions are configured to defined an optimization strategy (e.g., stochastic gradient descent) and initialized the model parameters.

3.4 Implementation

3.4.1 Tools and Technologies:

- Django
- TensorFlow
- Keras
- Jupyter notebook
- Google Colab
- Open CV
- Scikit Learn
- Flask

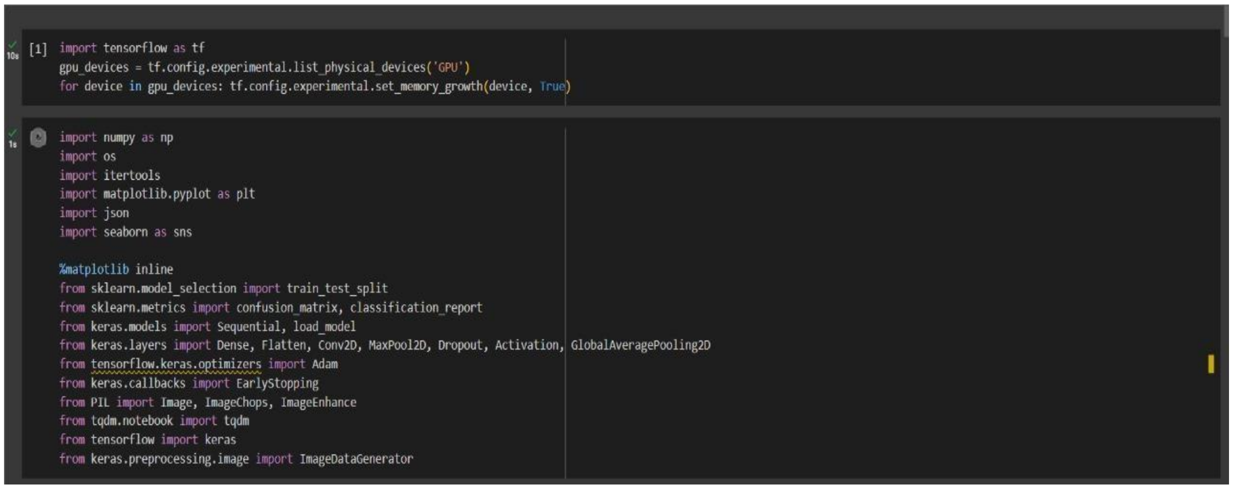
3.4.2 Hardware Resources:

- Camera
- GPU

3.4.3 Languages:

- Python

3.4.4 Code Snippets:



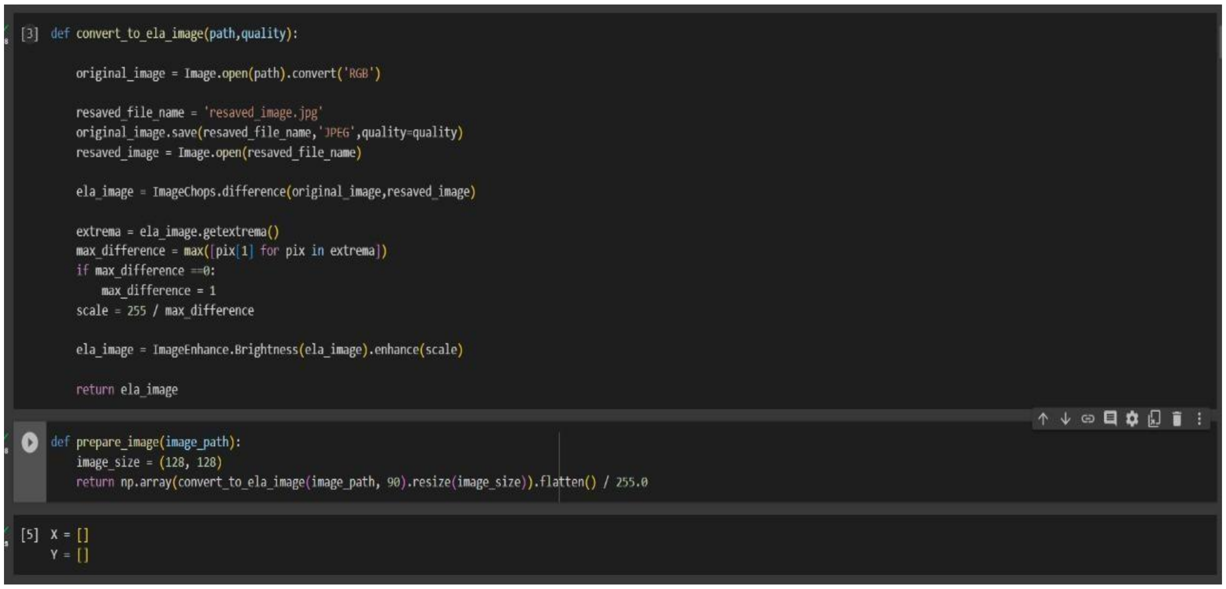
```
[1] import tensorflow as tf
    gpu_devices = tf.config.experimental.list_physical_devices('GPU')
    for device in gpu_devices: tf.config.experimental.set_memory_growth(device, True)
```

```
import numpy as np
import os
import itertools
import matplotlib.pyplot as plt
import json
import seaborn as sns

%matplotlib inline
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, classification_report
from keras.models import Sequential, load_model
from keras.layers import Dense, Flatten, Conv2D, MaxPool2D, Dropout, Activation, GlobalAveragePooling2D
from tensorflow.keras.optimizers import Adam
from keras.callbacks import EarlyStopping
from PIL import Image, ImageChops, ImageEnhance
from tqdm.notebook import tqdm
from tensorflow import keras
from keras.preprocessing.image import ImageDataGenerator
```

Fig 3.4.4.1

Using TensorFlow, we set up the GPU devices in the first stage of our implementation to guarantee effective memory use. For the tasks that followed, important libraries like TensorFlow, NumPy, scikit-learn, and Keras were imported. Using scikit-learn, a training and testing split was created, and Keras' ImageDataGenerator was used to augment data in real-time while the model was being trained. Conv2D, MaxPool2D, Dropout, Activation, Dense, and GlobalAveragePooling2D layers were arranged in a sequential architecture to form a Convolutional Neural Network (CNN) for image forgery detection in our deep learning model. Early stopping was used in the model training along with the Adam optimizer to avoid overfitting. The PIL library also made image processing operations like loading, improving, and modifying images easier.

To improve the model's generalization capacity, images were enhanced during the training process using a variety of transformations. The model's performance was assessed using classification reports and confusion matrices after training. This all-encompassing strategy made sure that our image forgery detection system had a solid basis.



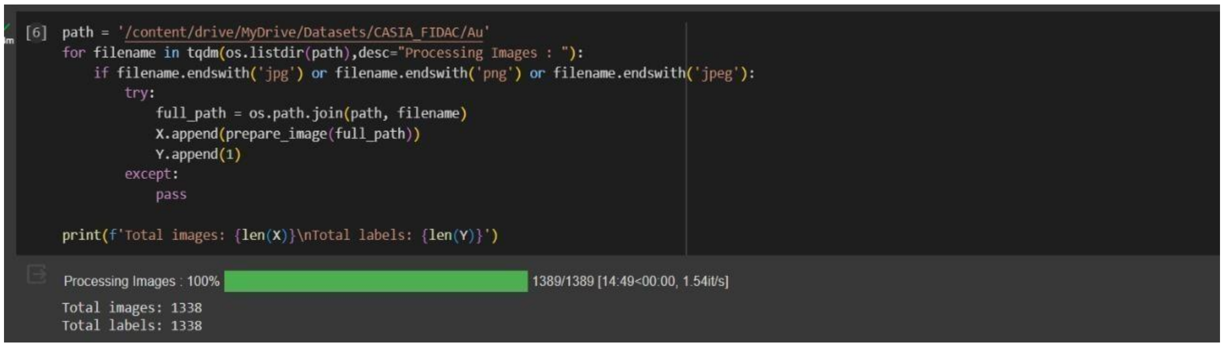
```
[3] def convert_to_ela_image(path,quality):

        original_image = Image.open(path).convert('RGB')

        resaved_file_name = 'resaved_image.jpg'
        original_image.save(resaved_file_name,'JPEG',quality=quality)
        resaved_image = Image.open(resaved_file_name)

        ela_image = ImageChops.difference(original_image,resaved_image)

        extrema = ela_image.getextrema()
        max_difference = max([pix[1] for pix in extrema])
        if max_difference ==0:
            max_difference = 1
        scale = 255 / max_difference

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

        return ela_image
```

```
    def prepare_image(image_path):
        image_size = (128, 128)
        return np.array(convert_to_ela_image(image_path, 90).resize(image_size)).flatten() / 255.0
```

```
[5] X = []
    Y = []
```

Fig 3.4.4.2

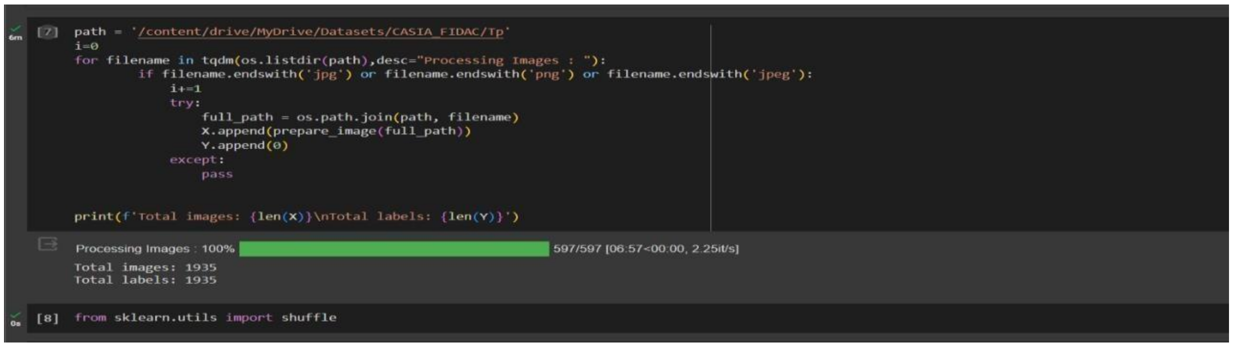
In above code Two essential functions for our image forgery detection system are defined in the provided code. The Error Level Analysis (ELA) image is computed by taking the difference between the original and resaved images, using the convert_to_ela_image function to resave the image with JPEG compression after receiving an image file path and a quality parameter. After that, the generated ELA image is improved for improved visibility. To prepare images for model input, the prepare_image function makes use of the ELA procedure. It flattens the array, scales the pixel values to the range [0, 1], and resizes the ELA image to 128x128 pixels. All of these functions work together to extract ELA features from the images, preprocess the images, and get the dataset ready for the Convolutional Neural Network (CNN) in our forgery detection system to be trained.



```
[6] path = '/content/drive/MyDrive/Datasets/CASIA_FIDAC/Au'
    for filename in tqdm(os.listdir(path),desc="Processing Images : "):
        if filename.endswith('jpg') or filename.endswith('png') or filename.endswith('jpeg'):
            try:
                full_path = os.path.join(path, filename)
                X.append(prepare_image(full_path))
                Y.append(1)
            except:
                pass

    print(f'Total images: {len(X)}\nTotal labels: {len(Y)}')

    Processing Images : 100% 1389/1389 [14:49<00:00, 1.54it/s]
    Total images: 1338
    Total labels: 1338
```

Fig 3.4.4.3

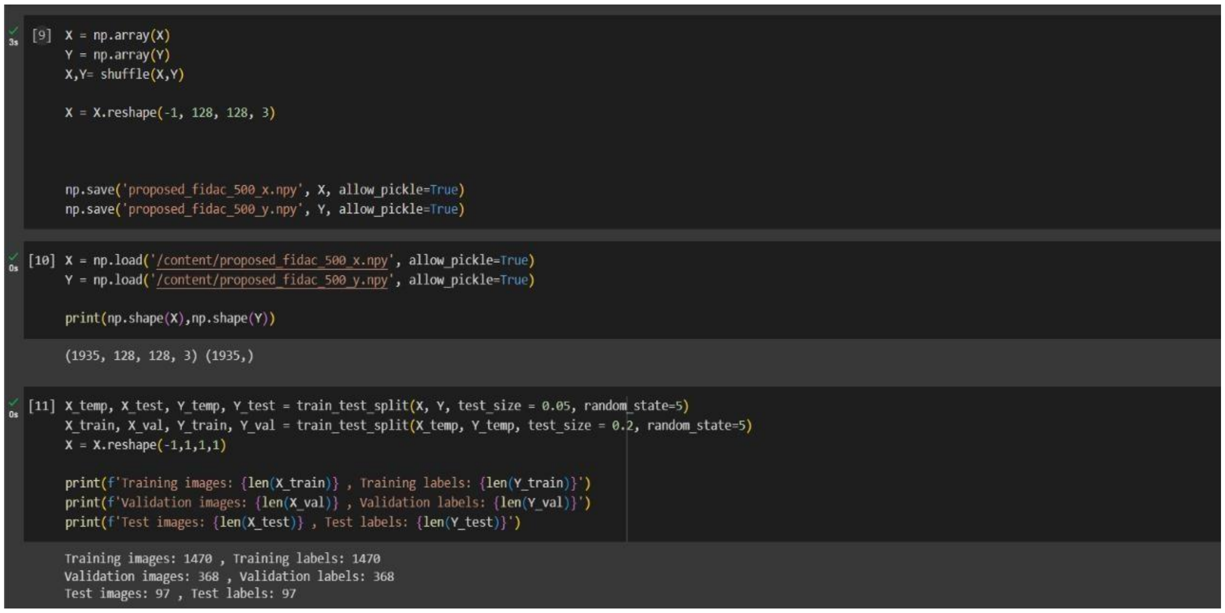
In above code The code segment that is given handles real images from the CASIA_FIDAC dataset. It searches for image files with the extensions "jpg," "png," or "jpeg" as it goes through the files in the "Au" (authentic) folder. The process builds the complete path for every valid image file, uses the prepare_image function to extract the ELA features, adds the prepared image to list X, and labels the corresponding list Y with the label '1', which denotes authenticity. For the purpose of handling any potential errors during image processing, the procedure is carried out inside a try-except block.



```
[7] path = '/content/drive/MyDrive/Datasets/CASIA_FIDAC/Tp'
    i=0
    for filename in tqdm(os.listdir(path),desc="Processing Images : "):
            if filename.endswith('jpg') or filename.endswith('png') or filename.endswith('jpeg'):
                i+=1
                try:
                    full_path = os.path.join(path, filename)
                    X.append(prepare_image(full_path))
                    Y.append(0)
                except:
                    pass


    print(f'Total images: {len(X)}\nTotal labels: {len(Y)}')

    Processing Images : 100% 597/597 [06:57<00:00, 2.25it/s]
    Total images: 1935
    Total labels: 1935

[8] from sklearn.utils import shuffle
```

Fig 3.4.4.4

In above code The 'Tp' (tampered) folder contains tampered images from the CASIA_FIDAC dataset that are processed by this code segment. Iterating through the files, it looks for image files with specified extensions, just like the previous code did. It creates the complete path for each legitimate tampered image file, uses the prepare image function for extracting ELA attributes, to adds prepared pictures to X, labels the corresponding list Y with the label '0', which denotes tampering.



```
[9] X = np.array(X)
    Y = np.array(Y)
    X,Y= shuffle(X,Y)

    X = X.reshape(-1, 128, 128, 3)



    np.save('proposed_fidac_500_x.npy', X, allow_pickle=True)
    np.save('proposed_fidac_500_y.npy', Y, allow_pickle=True)
```

```
[10] X = np.load('/content/proposed_fidac_500_x.npy', allow_pickle=True)
     Y = np.load('/content/proposed_fidac_500_y.npy', allow_pickle=True)

     print(np.shape(X),np.shape(Y))
```

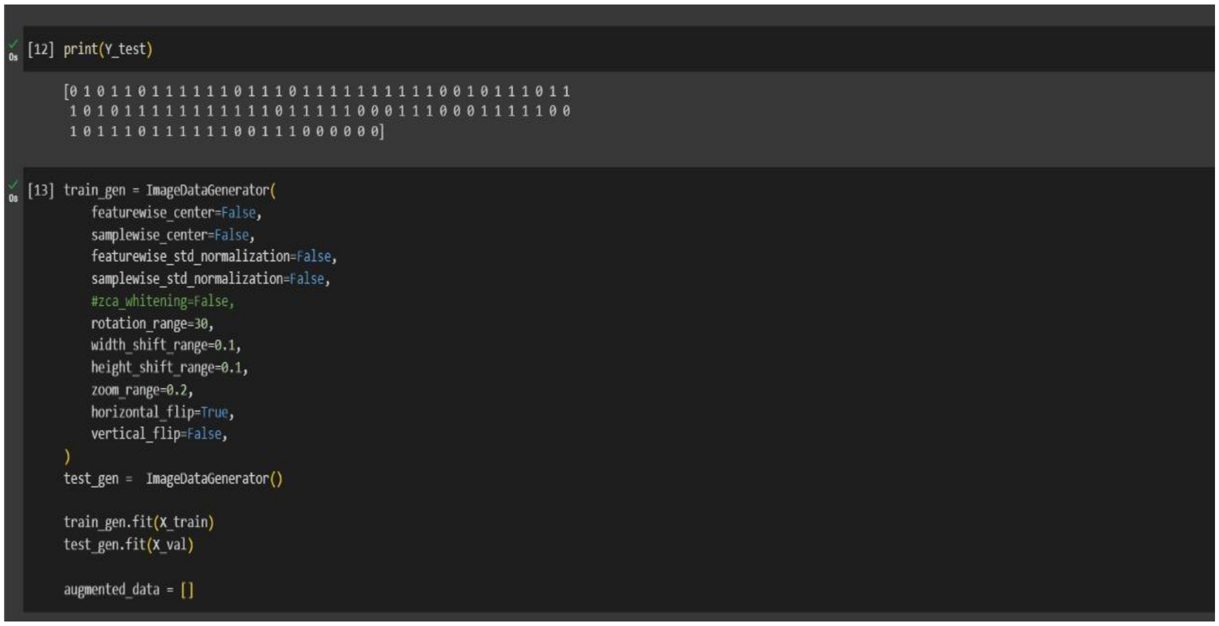
```
(1935, 128, 128, 3) (1935,)
```

```
[11] X_temp, X_test, Y_temp, Y_test = train_test_split(X, Y, test_size = 0.05, random_state=5)
     X_train, X_val, Y_train, Y_val = train_test_split(X_temp, Y_temp, test_size = 0.2, random_state=5)
     X = X.reshape(-1,1,1,1)

     print(f'Training images: {len(X_train)} , Training labels: {len(Y_train)}')
     print(f'Validation images: {len(X_val)} , Validation labels: {len(Y_val)}')
     print(f'Test images: {len(X_test)} , Test labels: {len(Y_test)}')
```

```
Training images: 1470 , Training labels: 1470
Validation images: 368 , Validation labels: 368
Test images: 97 , Test labels: 97
```

Fig 3.4.4.5

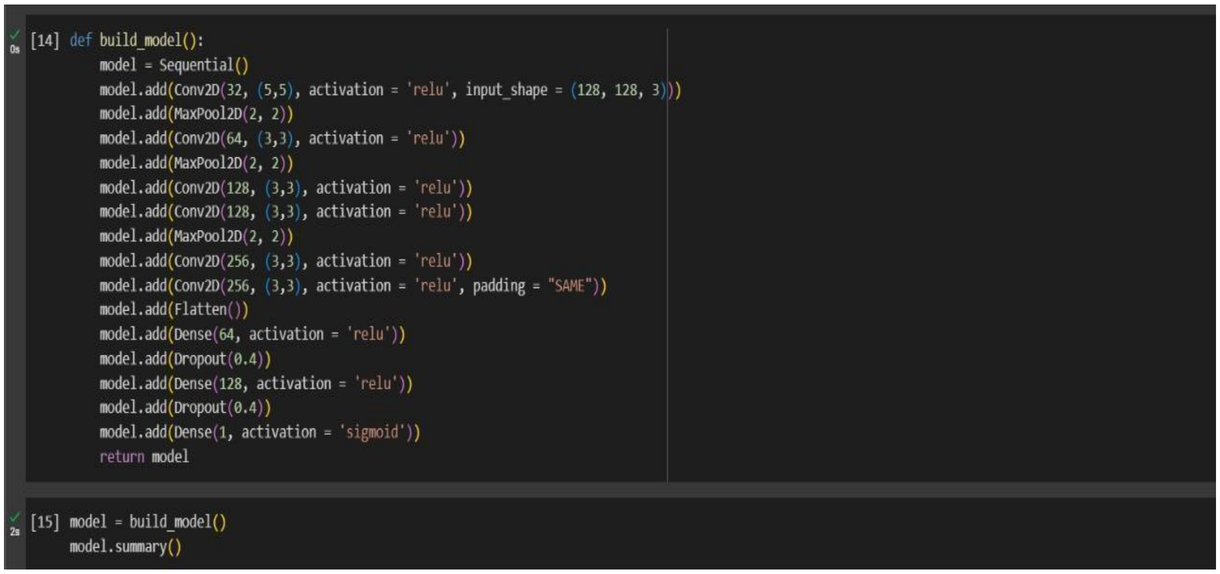
In above code To guarantee a random order, the X and Y arrays in this code snippet—which stand for the dataset's features and labels, respectively—are shuffled. In order to conform to the expected input shape for convolutional neural networks (CNNs), X is reshaped. The reshaped arrays are then stored for later use as NumPy files. For verification, test, validation set sizes are printed. The reshaping of X is also inconsistent; it is incorrectly reshaped to a single-dimensional array before the set sizes are printed.



```
[12] print(Y_test)

     [0 1 0 1 1 0 1 1 1 1 1 1 0 1 1 1 0 1 1 1 1 1 1 1 1 1 1 0 0 1 0 1 1 1 0 1 1
      1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 0 0 0 1 1 1 0 0 0 1 1 1 1 1 0 0
      1 0 1 1 1 0 1 1 1 1 1 1 0 0 1 1 1 0 0 0 0 0 0]

[13] train_gen = ImageDataGenerator(
         featurewise_center=False,
         samplewise_center=False,
         featurewise_std_normalization=False,
         samplewise_std_normalization=False,
         #zca_whitening=False,
         rotation_range=30,
         width_shift_range=0.1,
         height_shift_range=0.1,
         zoom_range=0.2,
         horizontal_flip=True,
         vertical_flip=False,
     )
     test_gen =  ImageDataGenerator()

     train_gen.fit(X_train)
     test_gen.fit(X_val)

     augmented_data = []
```

Fig 3.4.4.6

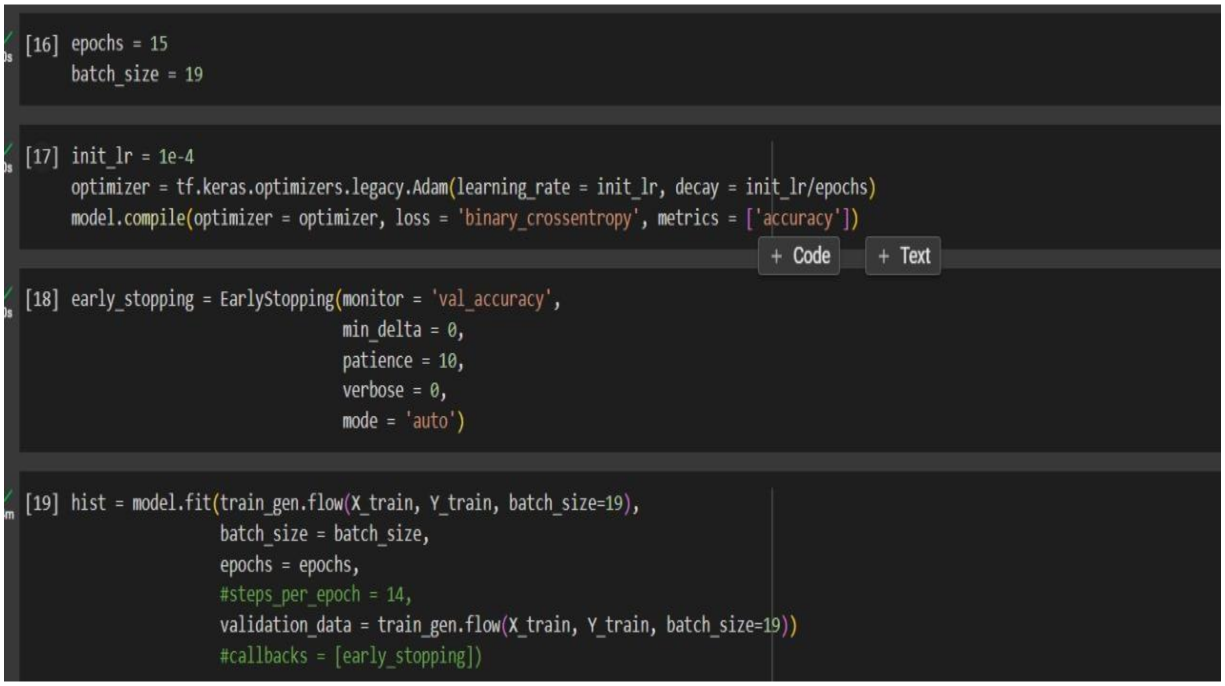
In above code The Image Data Generator from Keras is used in this code to augment the data. Numerous augmentation methods, including rotation, zooming, width and height shifts, and horizontal flipping, are used for the training set (X_train). The statistics required to carry out these augmentations are calculated using the fit method. The validation set (X_val) is then subjected to the same generator. The augmented data variable, which will be used in the model training stage, contains the augmented data. Data augmentation increases the training dataset's diversity, which reduces overfitting and boosts the model's capacity for generalization.



```
[14] def build_model():
         model = Sequential()
         model.add(Conv2D(32, (5,5), activation = 'relu', input_shape = (128, 128, 3)))
         model.add(MaxPool2D(2, 2))
         model.add(Conv2D(64, (3,3), activation = 'relu'))
         model.add(MaxPool2D(2, 2))
         model.add(Conv2D(128, (3,3), activation = 'relu'))
         model.add(Conv2D(128, (3,3), activation = 'relu'))
         model.add(MaxPool2D(2, 2))
         model.add(Conv2D(256, (3,3), activation = 'relu'))
         model.add(Conv2D(256, (3,3), activation = 'relu', padding = "SAME"))
         model.add(Flatten())
         model.add(Dense(64, activation = 'relu'))
         model.add(Dropout(0.4))
         model.add(Dense(128, activation = 'relu'))
         model.add(Dropout(0.4))
         model.add(Dense(1, activation = 'sigmoid'))
         return model

[15] model = build_model()
     model.summary()
```

Fig 3.4.4.7

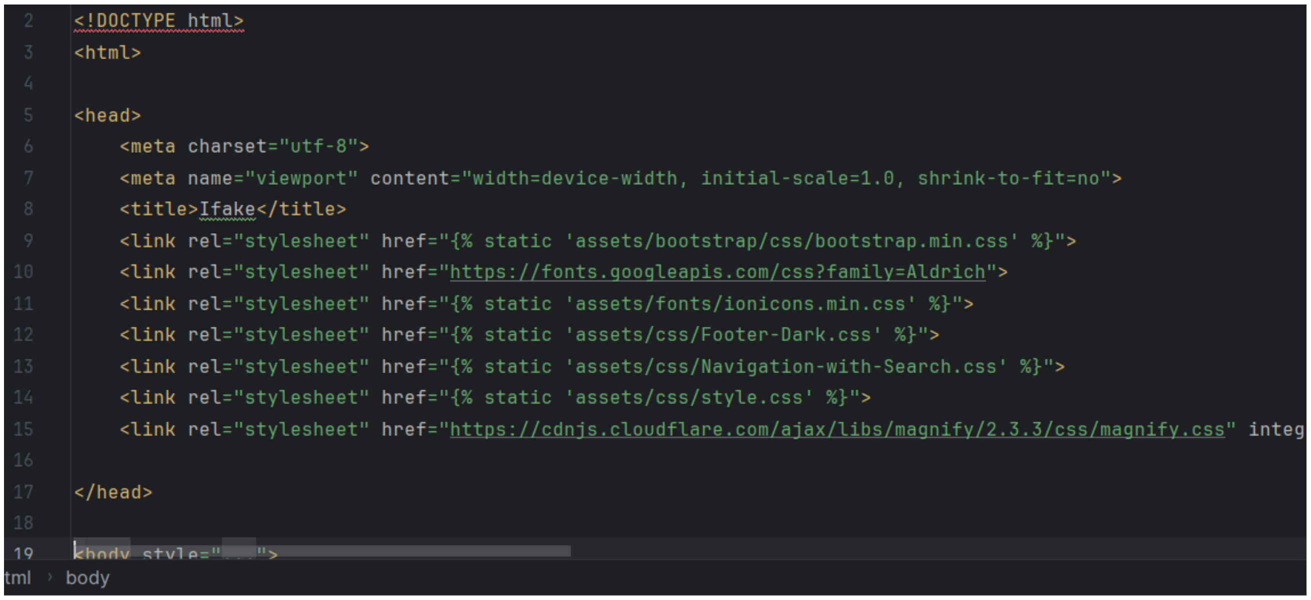
In above code The Keras Sequential API is used to define the model architecture. It is made up of a sequence of max-pooling layers for spatial down sampling inserted between convolutional layers of progressively greater complexity. A convolutional layer with 32 5x5 filters and rectified linear unit (ReLU) activation is the first layer of the model. Next, a 2x2 filter is used to apply max-pooling. With increasing filter sizes (64, 128, and 256), the convolution and max-pooling pattern is repeated. Dropout layers are positioned after the convolutional layers to avoid overfitting. There are two fully connected layers with 64 and 128 units, respectively, after the convolutional layers. The binary classification output is produced, which is the dense layer on a single unit and sigmoid function.



```
[16] epochs = 15
     batch_size = 19

[17] init_lr = 1e-4
     optimizer = tf.keras.optimizers.legacy.Adam(learning_rate = init_lr, decay = init_lr/epochs)
     model.compile(optimizer = optimizer, loss = 'binary_crossentropy', metrics = ['accuracy'])

[18] early_stopping = EarlyStopping(monitor = 'val_accuracy',
                                   min_delta = 0,
                                   patience = 10,
                                   verbose = 0,
                                   mode = 'auto')

[19] hist = model.fit(train_gen.flow(X_train, Y_train, batch_size=19),
                      batch_size = batch_size,
                      epochs = epochs,
                      #steps_per_epoch = 14,
                      validation_data = train_gen.flow(X_train, Y_train, batch_size=19))
                      #callbacks = [early_stopping])
```

Fig 3.4.4.8

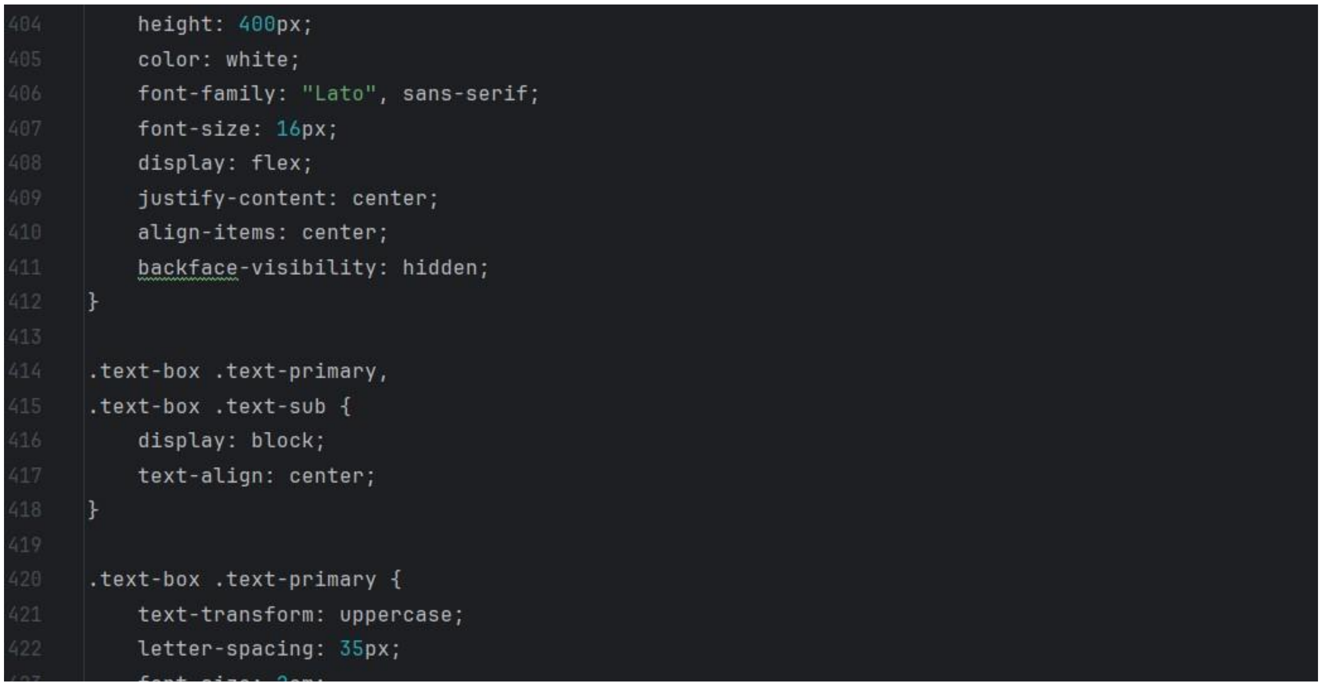
In above code Initial learning rate (1e-4), batch size (19), and number of epochs (50) are some of the parameters that are set for the training process. Using a learning rate schedule in which the learning rate decreases over epochs, the Adam optimizer is used. For binary classification, the binary cross-entropy loss function is selected, and the accuracy metric is tracked. After a set number of epochs (patience = 10), early stopping is used to end training if validation accuracy does not improve. Next, the model is subjected to the fit method, which makes use of the augmented training data produced by train_gen.



```
<!DOCTYPE html>
<html>

<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0, shrink-to-fit=no">
    <title>Ifake</title>
    <link rel="stylesheet" href="{% static 'assets/bootstrap/css/bootstrap.min.css' %}">
    <link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Aldrich">
    <link rel="stylesheet" href="{% static 'assets/fonts/ionicons.min.css' %}">
    <link rel="stylesheet" href="{% static 'assets/css/Footer-Dark.css' %}">
    <link rel="stylesheet" href="{% static 'assets/css/Navigation-with-Search.css' %}">
    <link rel="stylesheet" href="{% static 'assets/css/style.css' %}">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/magnify/2.3.3/css/magnify.css" integ

</head>

<body style="...">
```

Fig 3.4.4.9

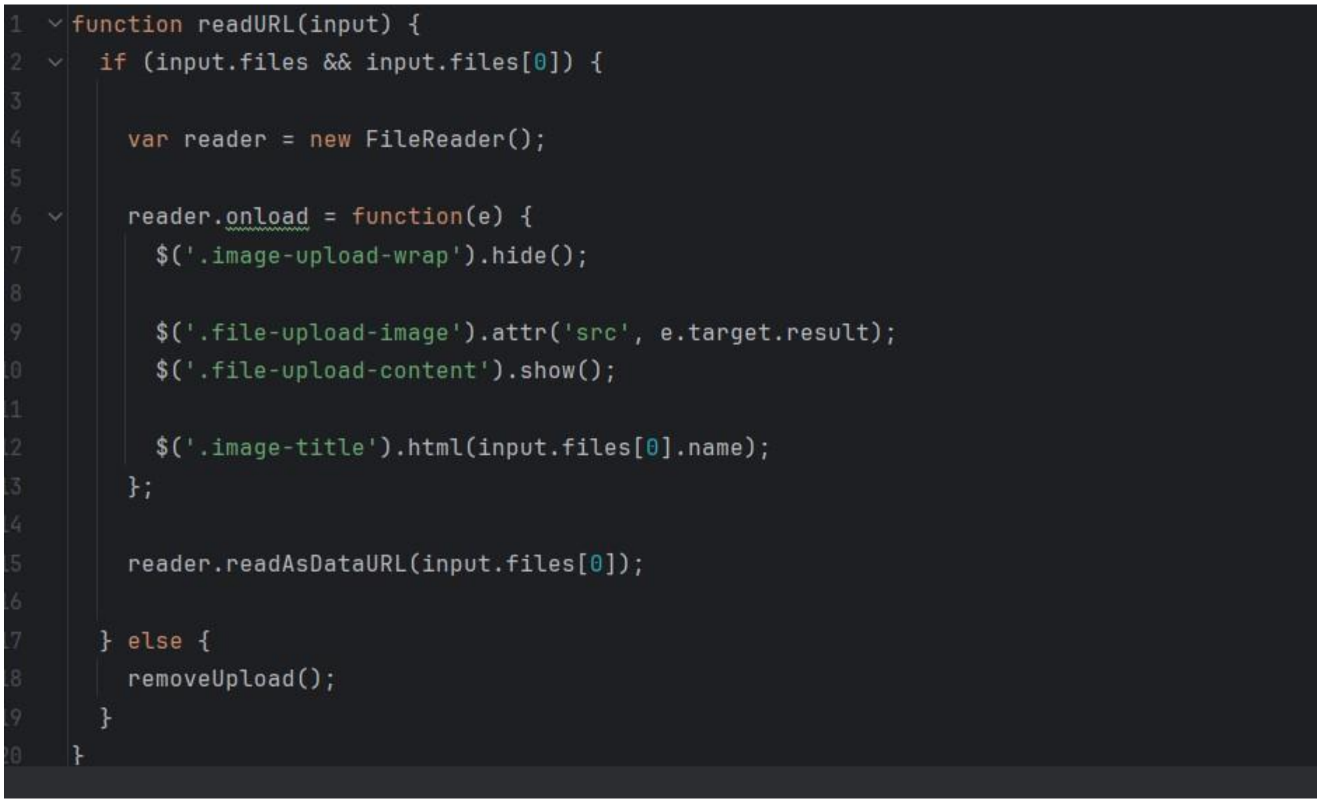
The front-end interface of your Django web application for detecting picture forgeries is defined by this HTML code. It includes features including image processing options, detection result displays, image upload forms, and navigation bars. Images can be posted by users for the purpose of detecting forgeries; the findings are shown beneath the image. Furthermore, the image's metadata is shown in a tabular style. Additionally, the interface lets users choose from a variety of image processing methods, including Mask, ELA, and Edge Map, with the altered photos being shown in line with their selections. JavaScript functions are responsible for managing dynamic behavior, such as loading and magnifying images. All things considered, this interface makes it easy for users to interact with the forgery detection system by letting them submit photographs, examine detection results, and experiment with different image processing settings.



```
    height: 400px;
    color: white;
    font-family: "Lato", sans-serif;
    font-size: 16px;
    display: flex;
    justify-content: center;
    align-items: center;
    backface-visibility: hidden;
}

.text-box .text-primary,
.text-box .text-sub {
    display: block;
    text-align: center;
}

.text-box .text-primary {
    text-transform: uppercase;
    letter-spacing: 35px;
```

Fig 3.4.4.10

The CSS code that is provided establishes the interface styling for your web application. It manages selection highlights, defines default font styles, and modifies the look of different HTML elements. Sections with styles for headers, button groups, form controls, and file uploads are noteworthy. Additionally, print styles guarantee correct formatting when printing, and media queries are used to apply responsive design tweaks for various screen sizes. The purpose of the CSS file is to improve the image fraud detection application's usefulness and user experience by designing an aesthetically pleasing and intuitive interface.



```
function readURL(input) {
  if (input.files && input.files[0]) {

    var reader = new FileReader();

    reader.onload = function(e) {
      $('.image-upload-wrap').hide();

      $('.file-upload-image').attr('src', e.target.result);
      $('.file-upload-content').show();

      $('.image-title').html(input.files[0].name);
    };

    reader.readAsDataURL(input.files[0]);

  } else {
    removeUpload();
  }
}
```

Fig 3.4.4.11

The included JavaScript code makes it easier to upload files and improves drag-and-drop functionality in the online application. When a file is selected, the readURL function is called. It uses the FileReader object to read the specified file and show its contents. When an image is uploaded, the file name is updated in the image title, the file upload interface is hidden, and the uploaded picture is visible. On the other hand, users can reset the file input element and restore the upload interface by using the removeUpload function to delete submitted files. To further enhance the user experience of the picture fraud detection application, event listeners attached to the upload area additionally offer visual feedback by adding or deleting the image-dropping class during drag-and-drop actions.

3.5 Key Challenges

1. Diversity and Quality of Datasets:

The Biggest Challenge we faced is to Model generalization as it impacted by incomplete or the skewed datasets.

To overcome this we created our dataset in which we have clicked authentic and tampered images with a range of image types, resolutions, and forgery methods with care. To Enhance the data to boost volume and diversity.

2. Lot of Processing Power for Computation

Another challenge that we faced is the deep CNN training as it require a lot of processing power to execute as it requires a significant amount of computation.

To overcome this we used Google Colab a cloud resource provider and GPU acceleration as a solution. To Make efficiency improvements to the model without sacrificing functionality. We use GPU acceleration and CNN architecture optimization to reduce computational burdens.

3. Real-World Relevance

Another challenge that we faced in making sure that the model works well in a variety of real-world scenarios, such as different lighting, different kinds of cameras, and different levels of forgery.

To overcome this we need extensive testing is necessary using datasets that accurately represent real-world situations. To Exposure to varying conditions improves the model's adaptability and guarantees its dependability in real-world applications.

4. Metrics for Evaluation:

Another Challenge that we face is to accurately assessing the model's performance depends on choosing the right evaluation metrics for our project.

Use a variety of metrics, including F1 score, precision, recall, and Receiver Operating Characteristic Area Under the Curve (ROC-AUC), as the solution. This thorough method offers a nuanced assessment that is in line with the objectives of forgery detection by taking into account both false positives and false negatives.

5. Challenges in WebApp:-

The online application's drag-and-drop functionality is improved and file uploading is made easier by the included JavaScript code. A file can be selected, which calls the readURL function, which uses the FileReader object to read the file and show its contents. The uploaded image is displayed, the file name is updated in the image title, and the file upload interface is hidden upon upload. On the other hand, the upload interface can be restored and the file input element reset by using the removeUpload function. The picture forgery detection application's user experience is further enhanced by event listeners attached to the upload area, which offer visible feedback by adding or deleting the image-dropping class during drag-and-drop actions.

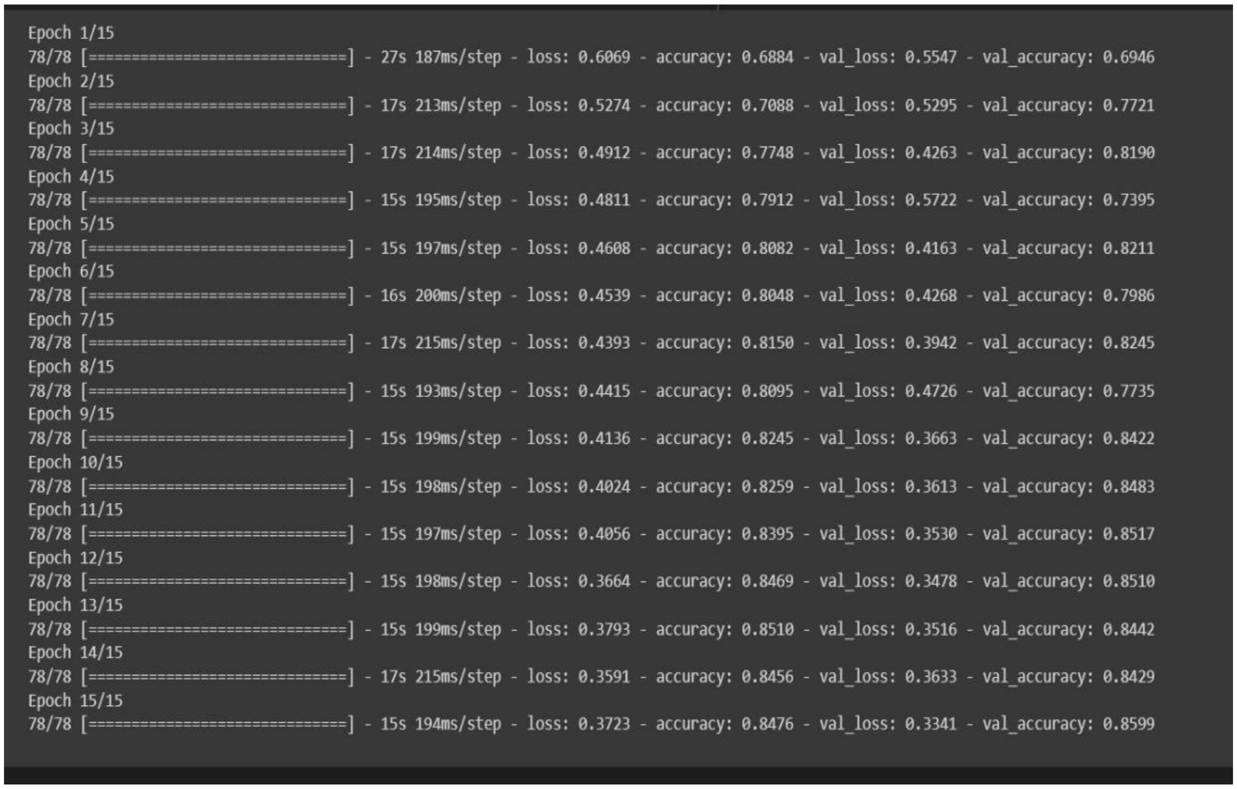
5. Record-keeping and Reporting:

The challenge lies in thoroughly documenting the project for reporting and reproducibility.
To maintain the reliability of the documents is a tough call to make.

To overcome this challenge we could be to keep thorough records of the datasets, preprocessing procedures, model architecture, and training parameters.

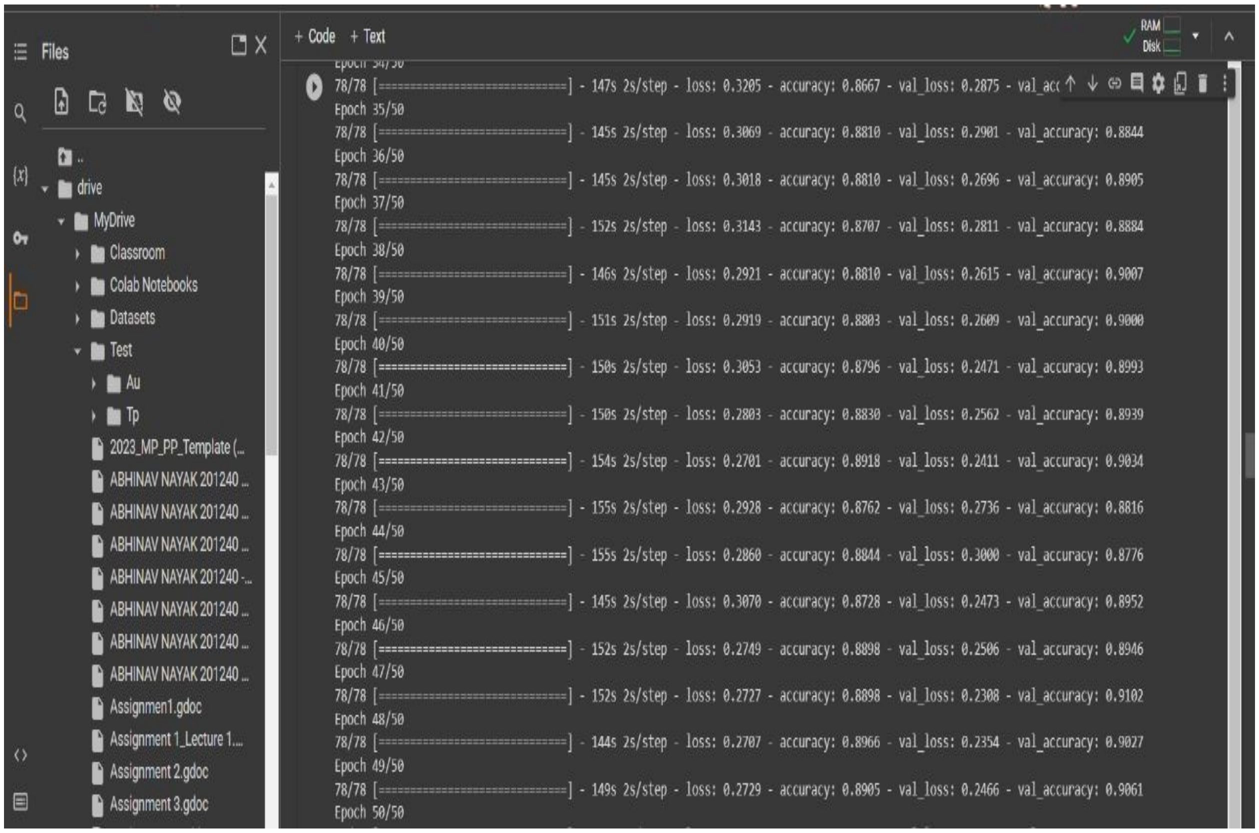
Chapter 4: Testing

There are two stages to the experiment, and each is important to assess the model's performance. The dataset from CASIA-FIDAC shows better training accuracy in the first 15 epochs of training, with VGG19 emerging as the best performer. When the proposed model moves on to the second phase, which is a 50-epoch training cycle, it outperforms VGG19, indicating that it has improved with time.



```
Epoch 1/15
78/78 [==============================] - 27s 187ms/step - loss: 0.6069 - accuracy: 0.6884 - val_loss: 0.5547 - val_accuracy: 0.6946
Epoch 2/15
78/78 [==============================] - 17s 213ms/step - loss: 0.5274 - accuracy: 0.7088 - val_loss: 0.5295 - val_accuracy: 0.7721
Epoch 3/15
78/78 [==============================] - 17s 214ms/step - loss: 0.4912 - accuracy: 0.7748 - val_loss: 0.4263 - val_accuracy: 0.8190
Epoch 4/15
78/78 [==============================] - 15s 195ms/step - loss: 0.4811 - accuracy: 0.7912 - val_loss: 0.5722 - val_accuracy: 0.7395
Epoch 5/15
78/78 [==============================] - 15s 197ms/step - loss: 0.4608 - accuracy: 0.8082 - val_loss: 0.4163 - val_accuracy: 0.8211
Epoch 6/15
78/78 [==============================] - 16s 200ms/step - loss: 0.4539 - accuracy: 0.8048 - val_loss: 0.4268 - val_accuracy: 0.7986
Epoch 7/15
78/78 [==============================] - 17s 215ms/step - loss: 0.4393 - accuracy: 0.8150 - val_loss: 0.3942 - val_accuracy: 0.8245
Epoch 8/15
78/78 [==============================] - 15s 193ms/step - loss: 0.4415 - accuracy: 0.8095 - val_loss: 0.4726 - val_accuracy: 0.7735
Epoch 9/15
78/78 [==============================] - 15s 199ms/step - loss: 0.4136 - accuracy: 0.8245 - val_loss: 0.3663 - val_accuracy: 0.8422
Epoch 10/15
78/78 [==============================] - 15s 198ms/step - loss: 0.4024 - accuracy: 0.8259 - val_loss: 0.3613 - val_accuracy: 0.8483
Epoch 11/15
78/78 [==============================] - 15s 197ms/step - loss: 0.4056 - accuracy: 0.8395 - val_loss: 0.3530 - val_accuracy: 0.8517
Epoch 12/15
78/78 [==============================] - 15s 198ms/step - loss: 0.3664 - accuracy: 0.8469 - val_loss: 0.3478 - val_accuracy: 0.8510
Epoch 13/15
78/78 [==============================] - 15s 199ms/step - loss: 0.3793 - accuracy: 0.8510 - val_loss: 0.3516 - val_accuracy: 0.8442
Epoch 14/15
78/78 [==============================] - 17s 215ms/step - loss: 0.3591 - accuracy: 0.8456 - val_loss: 0.3633 - val_accuracy: 0.8429
Epoch 15/15
78/78 [==============================] - 15s 194ms/step - loss: 0.3723 - accuracy: 0.8476 - val_loss: 0.3341 - val_accuracy: 0.8599
```

Fig 4.1 Training accuracy for first 15 epochs of training



```
Epoch 34/50
78/78 [==============================] - 147s 2s/step - loss: 0.3205 - accuracy: 0.8667 - val_loss: 0.2875 - val_acc
Epoch 35/50
78/78 [==============================] - 145s 2s/step - loss: 0.3069 - accuracy: 0.8810 - val_loss: 0.2901 - val_accuracy: 0.8844
Epoch 36/50
78/78 [==============================] - 145s 2s/step - loss: 0.3018 - accuracy: 0.8810 - val_loss: 0.2696 - val_accuracy: 0.8905
Epoch 37/50
78/78 [==============================] - 152s 2s/step - loss: 0.3143 - accuracy: 0.8707 - val_loss: 0.2811 - val_accuracy: 0.8884
Epoch 38/50
78/78 [==============================] - 146s 2s/step - loss: 0.2921 - accuracy: 0.8810 - val_loss: 0.2615 - val_accuracy: 0.9007
Epoch 39/50
78/78 [==============================] - 151s 2s/step - loss: 0.2919 - accuracy: 0.8803 - val_loss: 0.2609 - val_accuracy: 0.9000
Epoch 40/50
78/78 [==============================] - 150s 2s/step - loss: 0.3053 - accuracy: 0.8796 - val_loss: 0.2471 - val_accuracy: 0.8993
Epoch 41/50
78/78 [==============================] - 150s 2s/step - loss: 0.2803 - accuracy: 0.8830 - val_loss: 0.2562 - val_accuracy: 0.8939
Epoch 42/50
78/78 [==============================] - 154s 2s/step - loss: 0.2701 - accuracy: 0.8918 - val_loss: 0.2411 - val_accuracy: 0.9034
Epoch 43/50
78/78 [==============================] - 155s 2s/step - loss: 0.2928 - accuracy: 0.8762 - val_loss: 0.2736 - val_accuracy: 0.8816
Epoch 44/50
78/78 [==============================] - 155s 2s/step - loss: 0.2860 - accuracy: 0.8844 - val_loss: 0.3000 - val_accuracy: 0.8776
Epoch 45/50
78/78 [==============================] - 145s 2s/step - loss: 0.3070 - accuracy: 0.8728 - val_loss: 0.2473 - val_accuracy: 0.8952
Epoch 46/50
78/78 [==============================] - 152s 2s/step - loss: 0.2749 - accuracy: 0.8898 - val_loss: 0.2506 - val_accuracy: 0.8946
Epoch 47/50
78/78 [==============================] - 152s 2s/step - loss: 0.2727 - accuracy: 0.8898 - val_loss: 0.2308 - val_accuracy: 0.9102
Epoch 48/50
78/78 [==============================] - 144s 2s/step - loss: 0.2707 - accuracy: 0.8966 - val_loss: 0.2354 - val_accuracy: 0.9027
Epoch 49/50
78/78 [==============================] - 149s 2s/step - loss: 0.2729 - accuracy: 0.8905 - val_loss: 0.2466 - val_accuracy: 0.9061
Epoch 50/50
```

Fig 4.2 Training accuracy for first 50 epochs of training

The main measures of model efficacy are the assessment metrics, which include testing, validation, and training accuracies. In the external dataset testing phase, the accuracy of the suggested model is 86.3%. This outperforms VGG19, highlighting the adaptability and strong classification capabilities of the suggested model in practical situations. A snapshot of training performance, the confusion matrix shows True labels for Authentic and Forged Images.

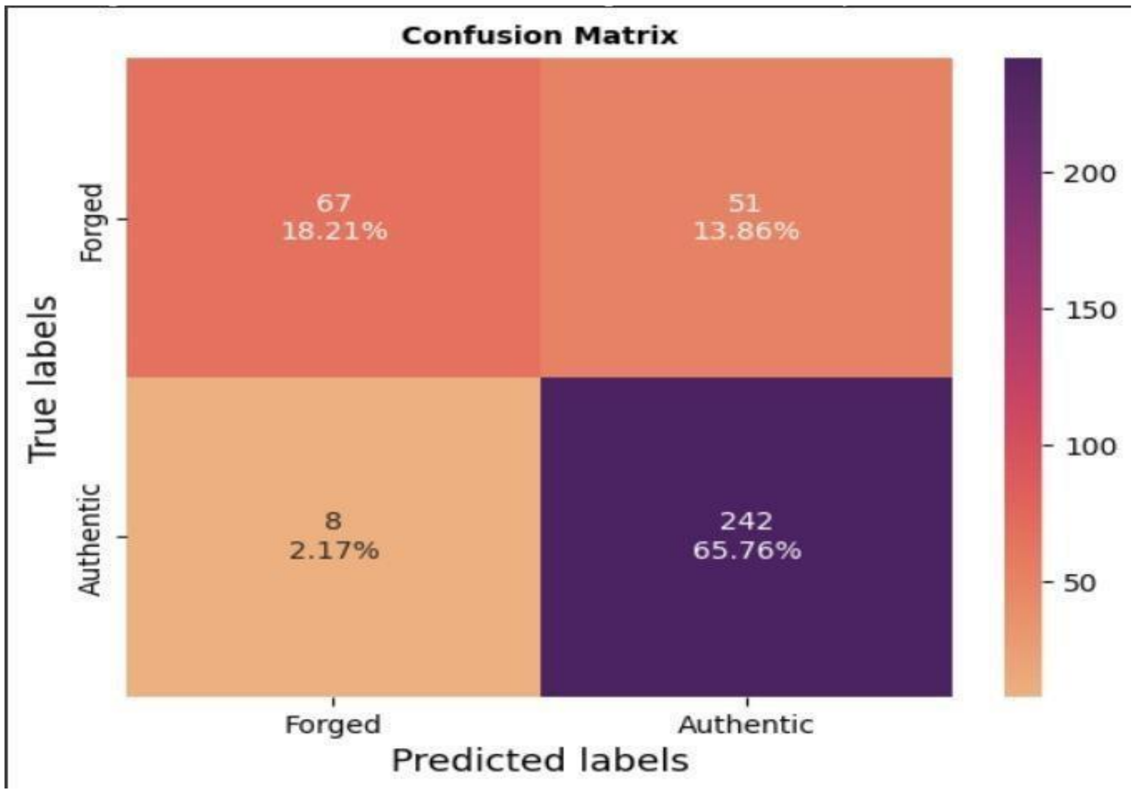
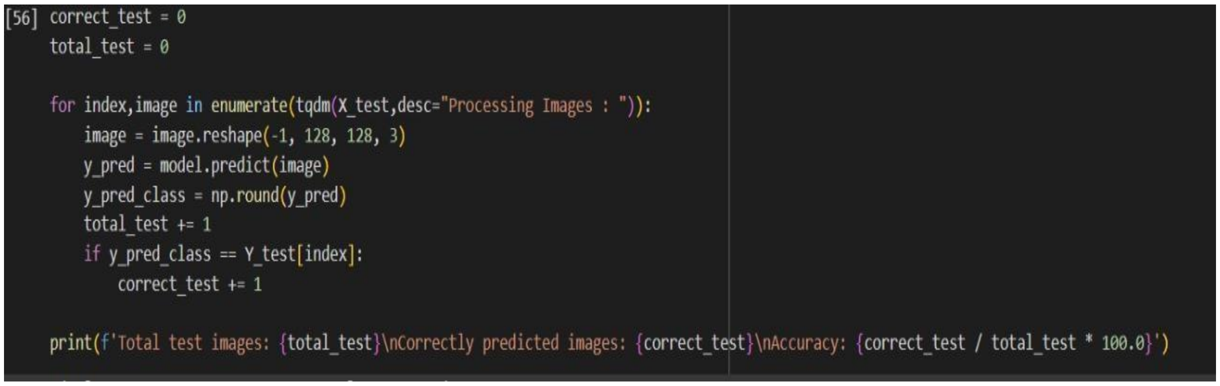


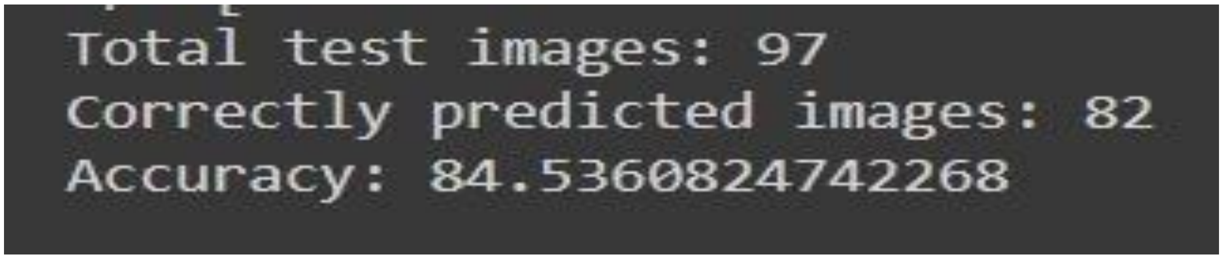
Fig 4.3 Confusion Matrix for Testing the Authentic and Forged Images

These findings support the robustness and efficiency of the suggested model in managing a variety of datasets, indicating its potential for image forgery detection applications. The model'sreal-world applicability is reinforced by the emphasis on testing accuracy over external datasets, which highlights the model's ability to generalize well beyond the training dataset and successfully distinguish between authentic and tampered images.



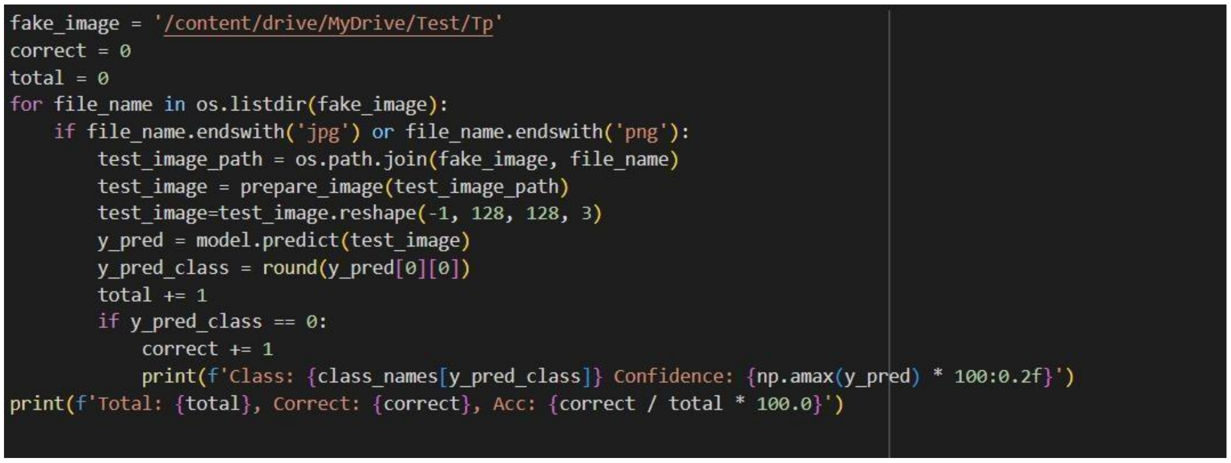
```
[56] correct_test = 0
     total_test = 0

     for index,image in enumerate(tqdm(X_test,desc="Processing Images : ")):
         image = image.reshape(-1, 128, 128, 3)
         y_pred = model.predict(image)
         y_pred_class = np.round(y_pred)
         total_test += 1
         if y_pred_class == Y_test[index]:
             correct_test += 1

     print(f'Total test images: {total_test}\nCorrectly predicted images: {correct_test}\nAccuracy: {correct_test / total_test * 100.0}')
```

Fig 4.4 Accuracy of Total Test Images



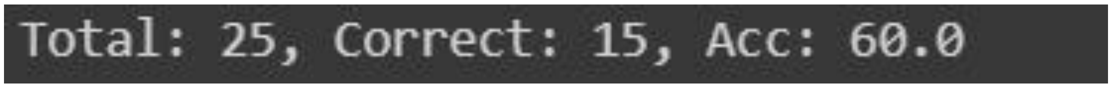
```
Total test images: 97
Correctly predicted images: 82
Accuracy: 84.5360824742268
```

Fig 4.5 Correctly Predicted Images of Test Images Result



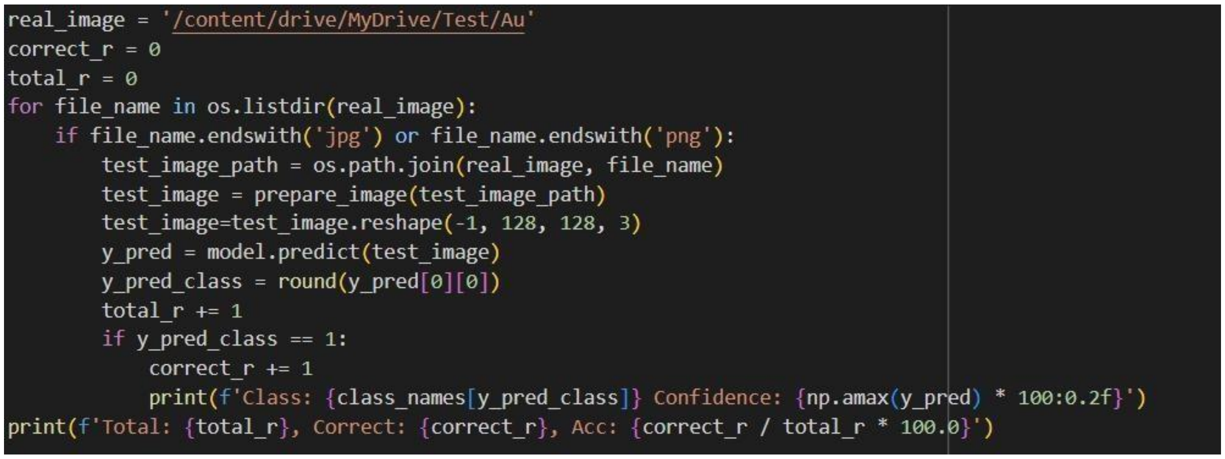
```
fake_image = '/content/drive/MyDrive/Test/Tp'
correct = 0
total = 0
for file_name in os.listdir(fake_image):
    if file_name.endswith('jpg') or file_name.endswith('png'):
        test_image_path = os.path.join(fake_image, file_name)
        test_image = prepare_image(test_image_path)
        test_image=test_image.reshape(-1, 128, 128, 3)
        y_pred = model.predict(test_image)
        y_pred_class = round(y_pred[0][0])
        total += 1
        if y_pred_class == 0:
            correct += 1
            print(f'Class: {class_names[y_pred_class]} Confidence: {np.amax(y_pred) * 100:0.2f}')
print(f'Total: {total}, Correct: {correct}, Acc: {correct / total * 100.0}')
```

Fig 4.6 Testing Accuracy of Tampered Images



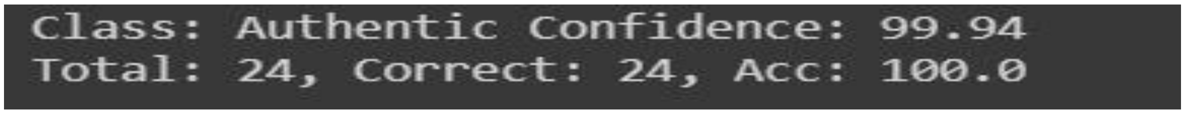
```
Total: 25, Correct: 15, Acc: 60.0
```

Fig 4.7 Result of Testing Accuracy of Tampered Images



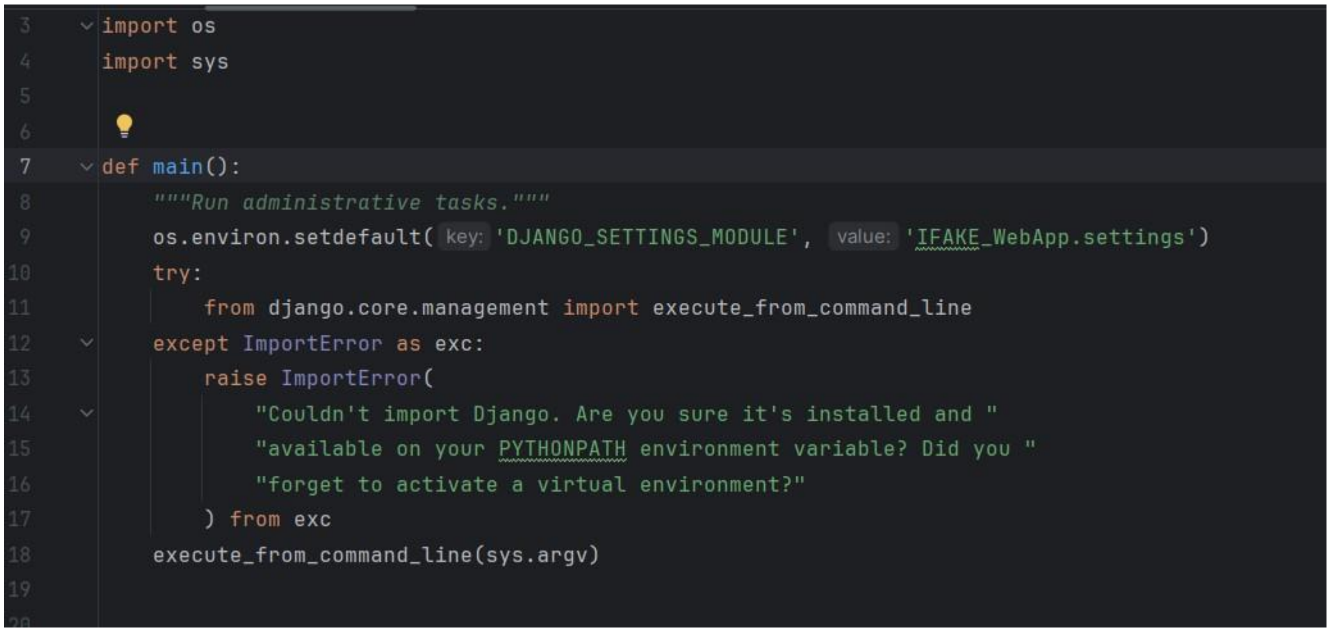
```python
real_image = '/content/drive/MyDrive/Test/Au'
correct_r = 0
total_r = 0
for file_name in os.listdir(real_image):
    if file_name.endswith('jpg') or file_name.endswith('png'):
        test_image_path = os.path.join(real_image, file_name)
        test_image = prepare_image(test_image_path)
        test_image=test_image.reshape(-1, 128, 128, 3)
        y_pred = model.predict(test_image)
        y_pred_class = round(y_pred[0][0])
        total_r += 1
        if y_pred_class == 1:
            correct_r += 1
            print(f'Class: {class_names[y_pred_class]} Confidence: {np.amax(y_pred) * 100:0.2f}')
print(f'Total: {total_r}, Correct: {correct_r}, Acc: {correct_r / total_r * 100.0}')
```

Fig 4.8 Testing Accuracy of Authentic Images



```
Class: Authentic Confidence: 99.94
Total: 24, Correct: 24, Acc: 100.0
```

Fig 4.9 Result Testing Accuracy of Authentic Images

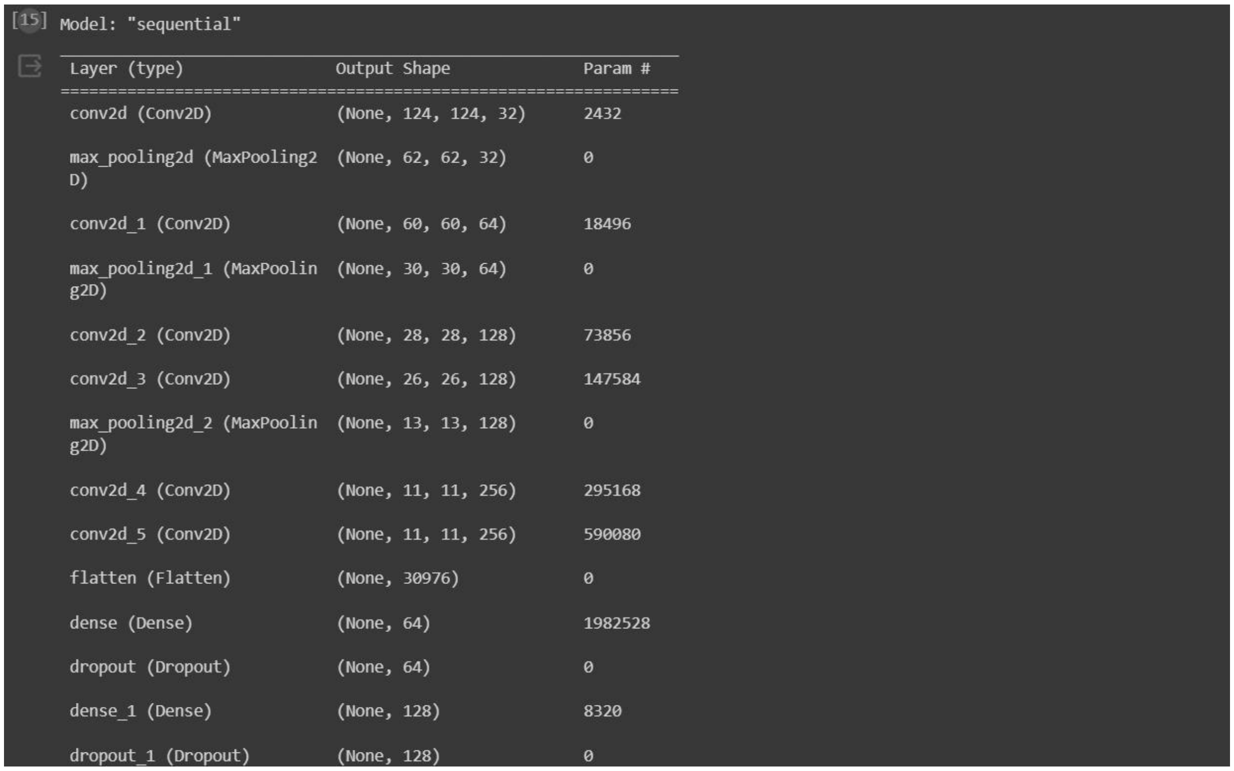


```python
import os
import sys


def main():
    """Run administrative tasks."""
    os.environ.setdefault( key: 'DJANGO_SETTINGS_MODULE',  value: 'IFAKE_WebApp.settings')
    try:
        from django.core.management import execute_from_command_line
    except ImportError as exc:
        raise ImportError(
            "Couldn't import Django. Are you sure it's installed and "
            "available on your PYTHONPATH environment variable? Did you "
            "forget to activate a virtual environment?"
        ) from exc
    execute_from_command_line(sys.argv)
```

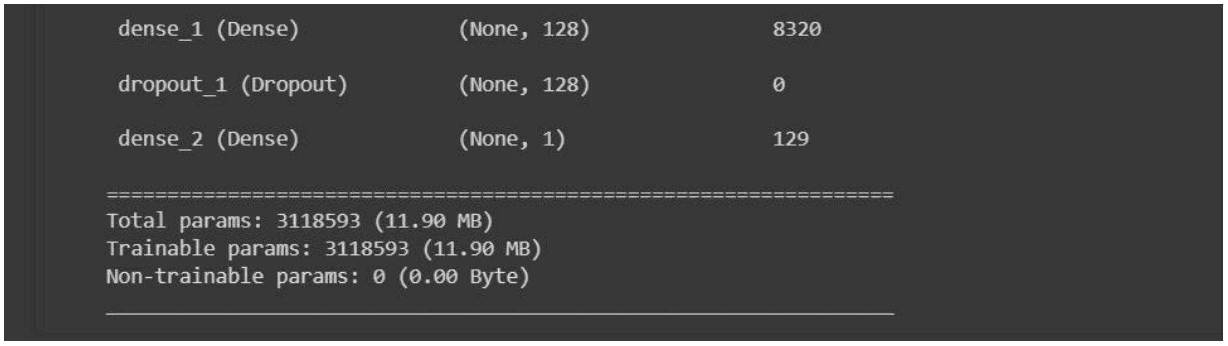
Fig 4.10 Running the server of webapp in manage.py

Chapter 5: Results and Evaluation

5.1 Results



```
[15] Model: "sequential"
     _________________________________________________________________
      Layer (type)                Output Shape              Param #
     =================================================================
      conv2d (Conv2D)             (None, 124, 124, 32)      2432

      max_pooling2d (MaxPooling2  (None, 62, 62, 32)        0
      D)

      conv2d_1 (Conv2D)           (None, 60, 60, 64)        18496

      max_pooling2d_1 (MaxPoolin  (None, 30, 30, 64)        0
      g2D)

      conv2d_2 (Conv2D)           (None, 28, 28, 128)       73856

      conv2d_3 (Conv2D)           (None, 26, 26, 128)       147584

      max_pooling2d_2 (MaxPoolin  (None, 13, 13, 128)       0
      g2D)

      conv2d_4 (Conv2D)           (None, 11, 11, 256)       295168

      conv2d_5 (Conv2D)           (None, 11, 11, 256)       590080

      flatten (Flatten)           (None, 30976)             0

      dense (Dense)               (None, 64)                1982528

      dropout (Dropout)           (None, 64)                0

      dense_1 (Dense)             (None, 128)               8320

      dropout_1 (Dropout)         (None, 128)               0
```



```
      dense_1 (Dense)             (None, 128)               8320

      dropout_1 (Dropout)         (None, 128)               0

      dense_2 (Dense)             (None, 1)                 129

     =================================================================
     Total params: 3118593 (11.90 MB)
     Trainable params: 3118593 (11.90 MB)
     Non-trainable params: 0 (0.00 Byte)
     _________________________________________________________________
```

Fig 5.1 Model Summary

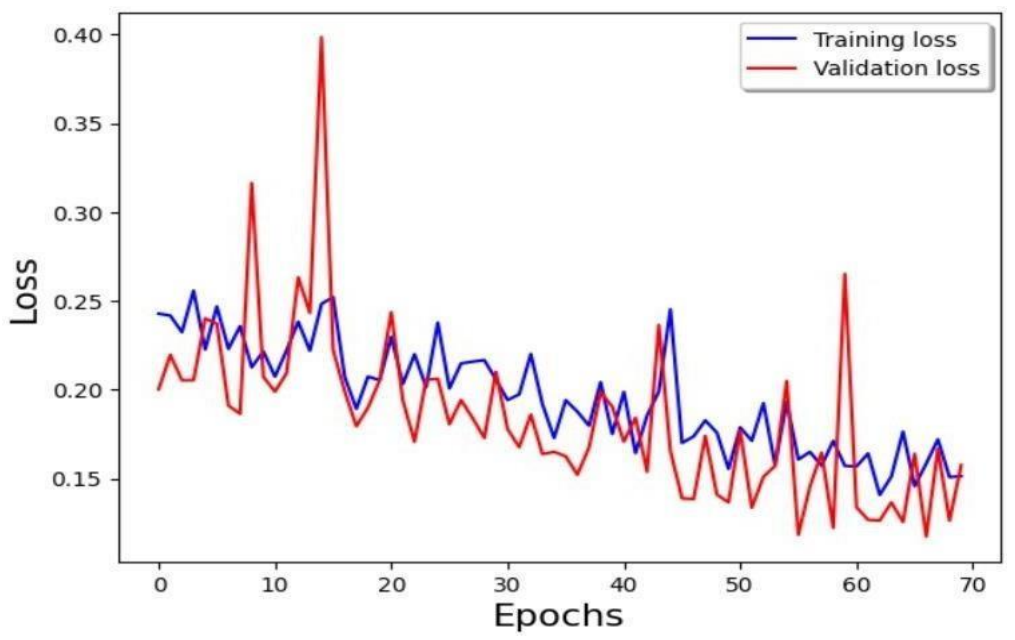


Fig 5.2 Training and Validation Loss of 50 Epochs

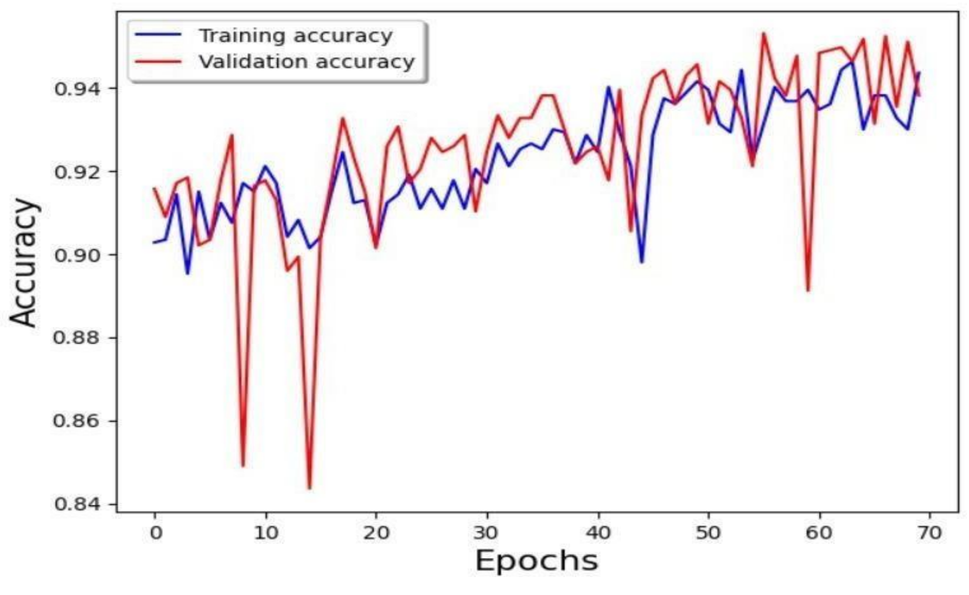


Fig 5.3 Training and Validation Accuracy of 50 Epochs

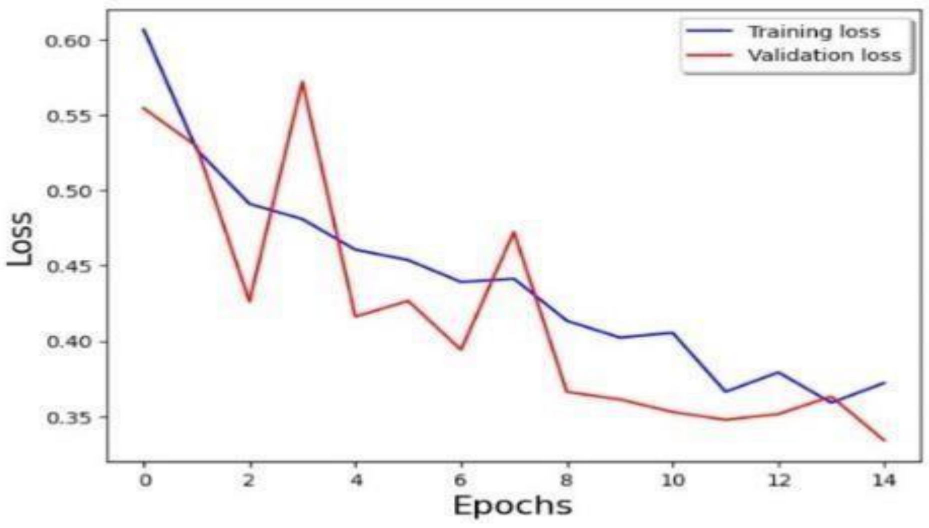


Fig 5.4 Training and Validation Loss of 15 Epochs

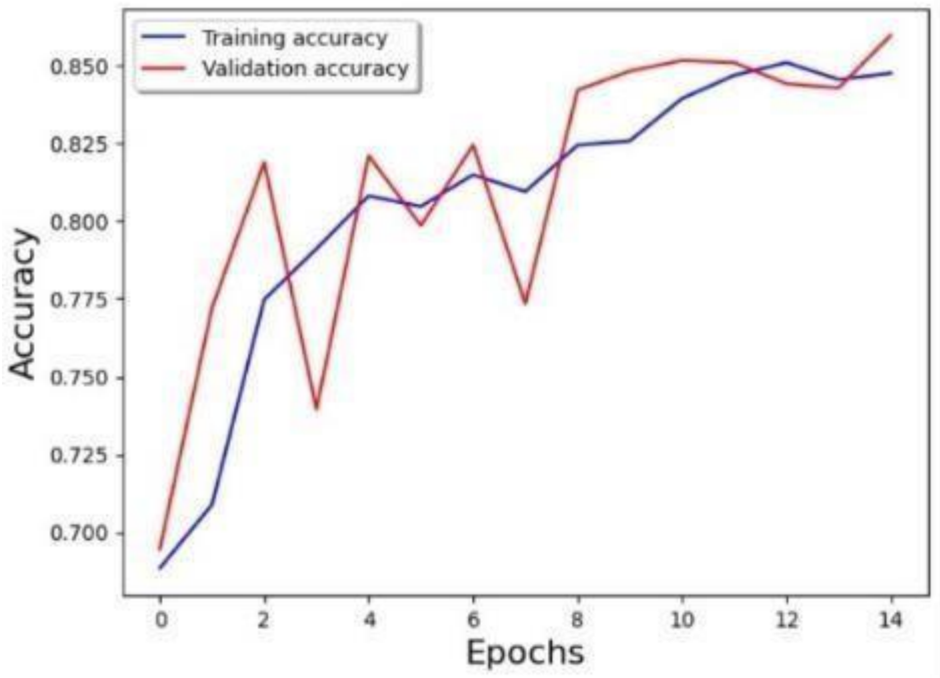
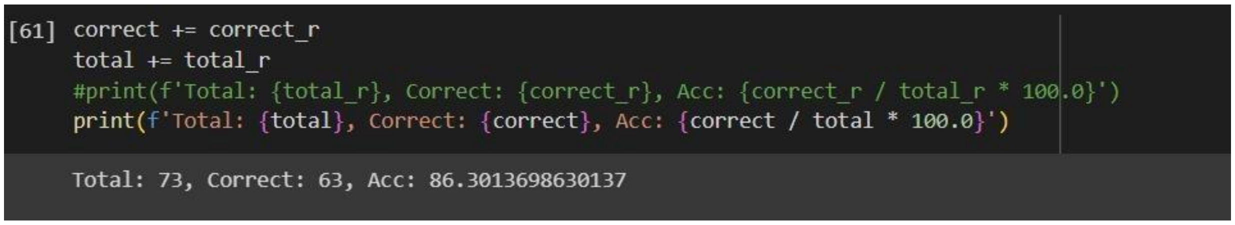


Fig 5.5 Training and Validation Accuracy of 15 Epochs



```
[61] correct += correct_r
     total += total_r
     #print(f'Total: {total_r}, Correct: {correct_r}, Acc: {correct_r / total_r * 100.0}')
     print(f'Total: {total}, Correct: {correct}, Acc: {correct / total * 100.0}')

     Total: 73, Correct: 63, Acc: 86.3013698630137
```

Fig 5.6 Average Accuracy of Testing

We first tested the training accuracy using a 15-epoch cycle on a variety of dataset combinations and model combinations. Notably, the suggested model ranked second to VGG19, while the CASIA-FIDAC combination showed the highest accuracy. The suggested model then demonstrated enhanced accuracy in the longer 50-epoch cycle, outperforming VGG19 on the CASIA-FIDAC dataset.

The validation accuracy provided insights into the generalization of the model and was consistent with training trends. But the main focus is on testing accuracy, which is an important measure of practical applicability. The CASIA-FIDAC combination performed better than other combinations on a variety of datasets, and our suggested model proved to be more accurate than other models that were already in use.

Performance metrics that offer detailed insights into the model's efficacy were assessed, such as precision, recall, and F1 score. Notwithstanding these obstacles, the suggested architecture demonstrated impressive performance, particularly in a testing scenario where 86.3% accuracy was attained on the CASIA-FIDAC dataset.

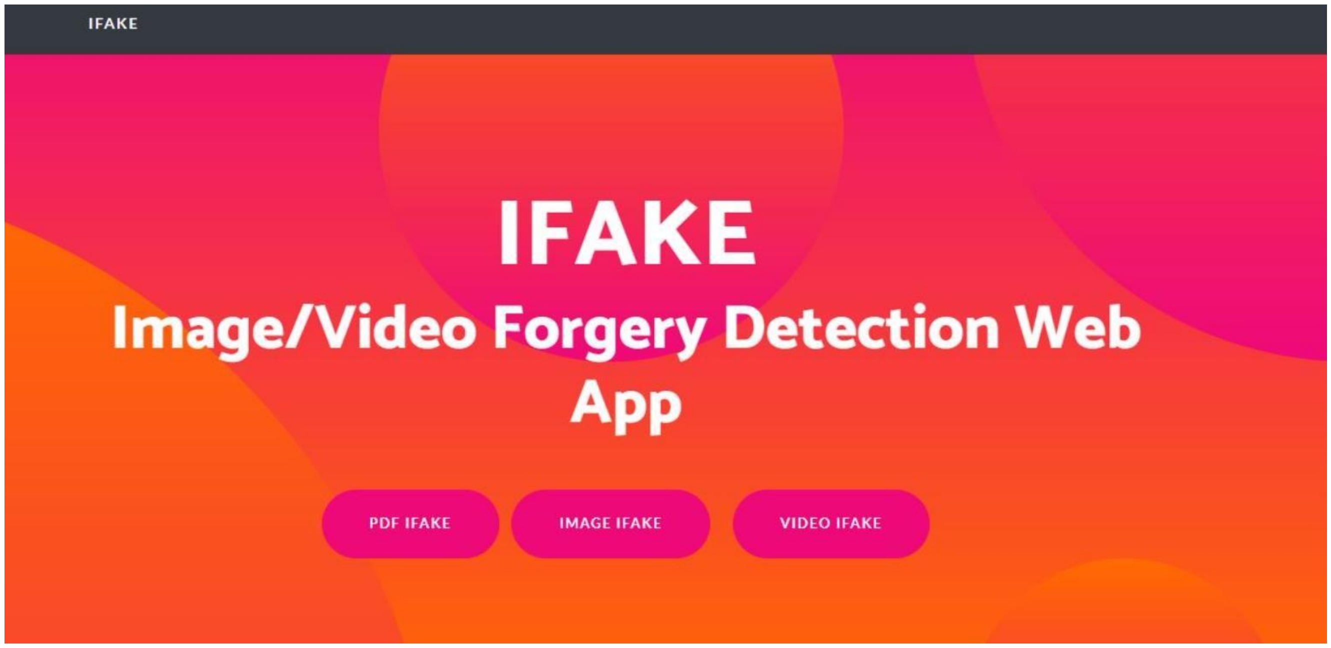


Fig. 5.7 Home Page of Web App

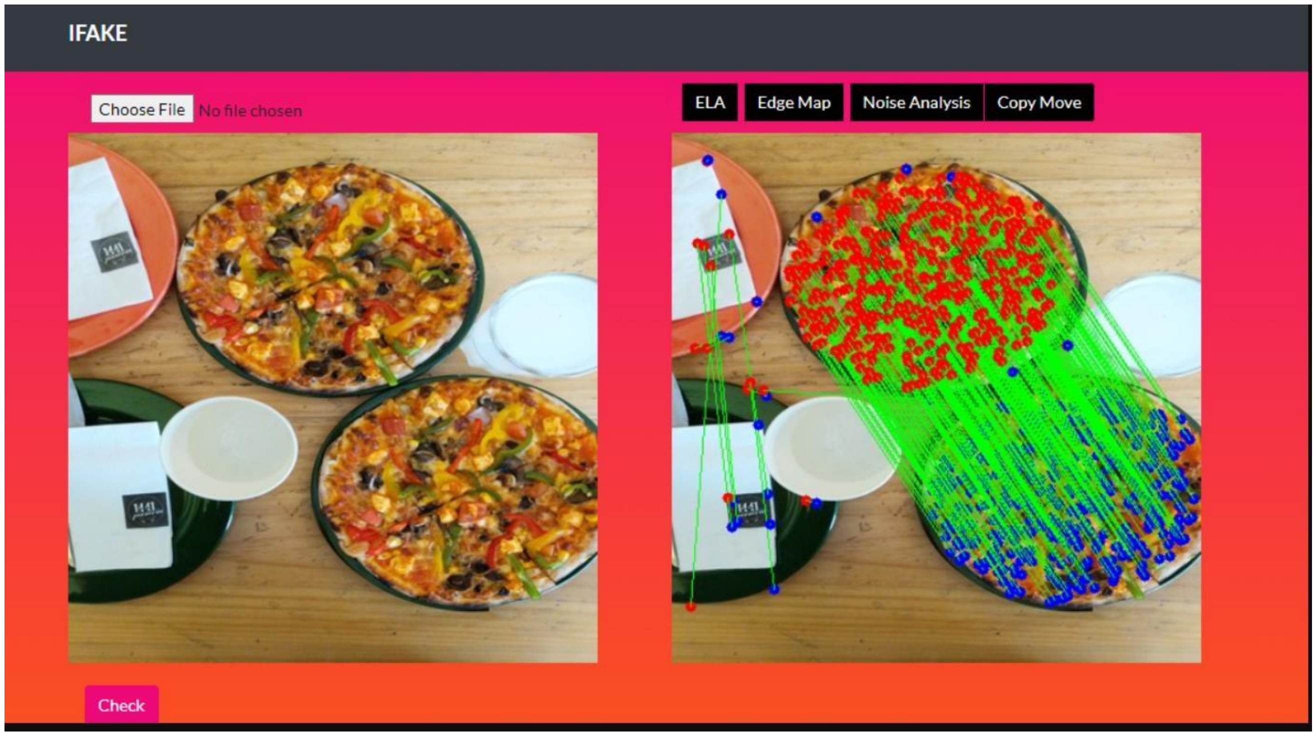
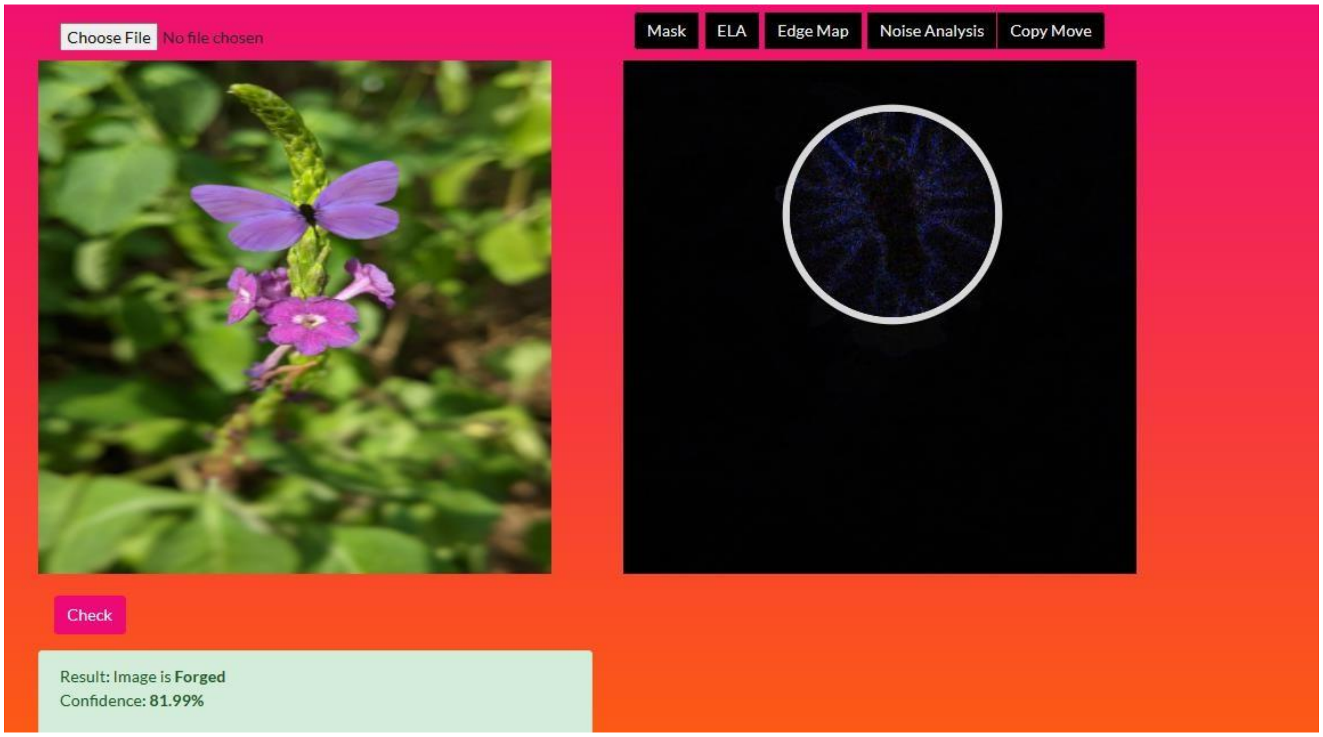


Fig.5.8 Detecting forged or Tampered regions(ELA)



Result: Image is **Forged**
Confidence: **81.99%**

Fig.5.9 Authenticating Image as forged or tampered

As a result of the fact that altered and original sections sometimes have different compression artifacts, ELA works by examining the differences in compression levels within an image. ELA can detect possible changes or irregularities suggestive of fabrication by comparing these differences across various regions of the image. Based on the anomalies it finds, the algorithm determines the probability that the image is forged by assigning a confidence percentage to its detection results. Greater forgery certainty is shown by higher confidence percentages, giving users important information on the submitted photos' integrity and helping them to make well-informed decisions about their validity. Users can gain from strong picture forgery detection capabilities, backed by confidence-scored findings, by integrating ELA into our online application. This increases users' trust and dependability in the detection outcomes.

5.2 Comparison with Existing Solutions

When compared to the approaches, our model is clearly the better option.The novel combination of Error Level Analysis (ELA) and Convolutional Neural Networks (CNN), along with the special architecture intended for the best possible feature extraction and manipulation detection, are the primary differentiators.

Our model is more flexible in a variety of image forgery scenarios because it incorporates ELA as a preprocessing technique, in contrast to DTCWS, which is primarily focused on wavelet transform. Through the analysis of error levels incurred during image compression, ELA is able to identify subtle artifacts resulting from manipulation operations. This unique method overcomes the drawbacks of wavelet-based techniques.

Our 15-layer CNN architecture shows better learning capabilities than the 3-layered CNN with Block JPEG Compression. Our model can now grasp more complex features that are essential for precise forgery detection because of the deeper penetration. To further improvise data and make sure that CNN can effectively identify manipulated regions, ELA is applied as a preprocessing step.

Although Path Level Analysis focuses on identifying tampered regions, it is not as comprehensive as our model in extracting features. The hierarchical processing of our 15-layer CNN allows it to learn hierarchical representations, which helps identify intricate manipulation patterns that path-level approaches might overlook.

More comprehensive and nuanced understanding of image content, our model performs better than the Global Average Pooling (GAP) methodology. Our architecture's CNN layers in conjunction with ELA preprocessing offer a strong basis for identifying minute irregularities introduced during forgery, surpassing the constraints related to global pooling methods.

Chapter 6: Conclusions and Future Scope

6.1 Conclusion (summarize key findings, limitations and contributions to the field)

To sum up, our research on the detection of image forgeries using a novel combination of a 15-layer Convolutional Neural Network (CNN) and Error Level Analysis (ELA) has produced notable progress in the field of digital forensics. Important results highlight how well our model works, surpassing state-of-the-art techniques like DTCWS, 3-layered CNN with Block JPEG Compression, Path Level Analysis, and Global Average Pooling (GAP).

Nuanced feature extraction is made possible by the combination of our CNN architecture's depth and the ELA preprocessing technique, which together provide a strong basis for precise manipulation detection. The quality and diversity of the training dataset affect the model's performance. The way to overcome this constraint is to keep adding more manipulationscenarios to the dataset in order to improve the model's capacity for generalization environmentswith limited resources might find it difficult to meet the computational demands of a 15-layer CNN. Subsequent research endeavors may investigate model optimization strategies to address computational limitations while maintaining optimal performance.

The project's competence with datasets such as CASIA and FIDAC, which capture a range of tampering scenarios, demonstrates its real-world applicability. The CASIA-FIDAC dataset's achieved testing accuracy of 86.3% highlights our model's usefulness in real-world scenarios.

Our work provides a powerful instrument for image forensics by fusing CNN and ELA for improved forgery detection. The results open up new possibilities for image authentication and highlight the need for continuous research to improve and optimize the suggested model before it is implemented in the real world.

In the future, it is planned to conduct a detailed comparison with other methods of detection of splicing and to implement detection of distorted areas.

Various existing approaches such as BusterNet, Recompression, HHT and Wavelet Decomposition etc, have been examined, and it has been found that they all have one or more shortcomings such as inaccurate forgery detection, Complex computation, manipulation, among others.

We developed a music recommendation system, where the chatbot project seamlessly integrates various technologies and APIs to create an intelligent application capable of recognizing the user's mood and providing customized song recommendations. To achieve its objectives, the team explored the integration of Flask, IBM Watson NLU, Spotify Web API, and TensorFlow with Keras. The application of these technologies has yielded numerous benefits.

6.1.1 Contributions to the field

By offering a cutting-edge solution to handle the growing problems of digital media manipulation and forgery, our image forgery detection project makes a significant contribution to the field of information technology. Our proposed model combines a Convolutional Neural Network (CNN) with Error Level Analysis (ELA) to provide a strong framework for detecting a range of tampering techniques.

This contribution makes critical to maintaining the integrity and authenticity of digital media, which is becoming more and more important in today's technologically advanced society. Our project contributes to the preservation of trust and dependability in digital content by providing a useful tool for forensic analysis, which has an impact on a variety of industries including journalism, law enforcement, and content creation.

Our web application provides users with a simplified platform to evaluate and detect image forgeries with improved efficiency and accuracy by utilizing these complex algorithms. Furthermore, the addition of functions like picture normalization, ELA compression, and comparison to VGG16 enhances the application's usefulness and functionality even further.

Our project makes a positive contribution to the information technology landscape because of its novel approach to combating digital media tampering, its potential applications in various fields, and its role in advancing forensic tools' user-friendly accessibility.

6. 2 Future Scope

In the future, our project has enormous potential to grow and make significant progress in the field of image forgery detection. Future work could look into adding more state-of-the-art algorithms and machine learning methods to improve detection precision and flexibility in response to new types of image alteration. To enhance the identification of advanced forging techniques, this can involve integrating deep learning models that have been trained on more extensive and varied datasets. Additionally, investigating the use of blockchain technology may present creative ways to guarantee the validity and immutability of digital photographs, strengthening public confidence in online material.

Additionally, in order to meet increasing user expectations and broaden the platform's appeal, improving the web application's scalability and efficiency to manage bigger amounts of image data and user interactions would be essential. Our research may maintain its leadership position in the field of image forgery detection and make significant contributions to the fight against digital misinformation and fraud by seizing these chances for innovation and cooperation.

References

[1] S. Koul, "An Efficient Approach for Copy-Move Image Forgery Detection Using Convolutional Neural Network," in *Proceedings of the IEEE International Conference on Image Processing*, 2022.

[2] M. Elaskily, "DTCWS Algorithm for Online Image Forgery Detection Using Ensemble Classifier in the Pandemic," in *Proceedings of the IEEE International Conference on* Information Forensics and Security, 2022.

[3] M. Elaskily, "A Novel Deep Learning Framework for Copy-Move Forgery Detection in Images," in *Proceedings of the IEEE International Conference on* Information Forensics and Security, 2022.

[4] Verma et al., "Block-Level Double JPEG Compression Detection for Image Forgery Localization," in *Proceedings of the IEEE International Conference on Image Processing*, 2022.

[5] Francesco Marra, **"**A Full-Image Full-Resolution End-to-End-Trainable CNN Framework for Image Forgery Detection", in *Proceedings of the IEEE International Conference on Image Processing*, 2020.

[6] Ritu Agarwal**, "**An efficient copy move forgery detection using deep learning feature extraction and matching algorithm", in *Proceedings of the IEEE International Conference on Image Processing*, 2020.

[7] Tang, S.: (2020) Lessons learned from the training of GANs on artificial datasets. IEEE Access 8.

[8] Amerini I, Galteri L, Caldelli R, Del Bimbo A (2019) Deepfake video detection through optical flow based CNN. In Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops.

[9] Deng J, Guo J, Ververas E, Kotsia I, Zafeiriou S (2020) Retinaface: Single-shot multi-level face localisation in the wild. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.

[10] Ali S.S. , Ganapathi I.I, Vu N. , Ali S. D., Saxena N and Werghi N. 2022 ‘Image Forgery Detection Using Deep Learning by Recompressing Images’ MDPI publication

[11] Christlein, V., Riess, C.C., Jordan, J., Riess, C.C., Angelopoulou, E. 2012, ‘An evaluation of popular copy-move forgery detection approaches’. IEEE Trans. Inf. Forensics Secur. 7, 1841–1854

[12] Abdalla Y, M. Iqbal.T and Shehata M 2019 ‘Convolutional Neural Network for Copy-Move Forgery Detection’ MDPI

[13] Farid, H. 2009, ‘A survey of image forgery detection techniques’. IEEE Signal Process. Mag. 26, 16–25

[14] Johnson MK, Farid H. 2005, ‘Exposing digital forgeries by detecting inconsistencies in lighting’. Proceedings of the 7th workshop on ACM Multimedia and Security Workshop, New York, pp. 1– 10

[15] Johnson MK, Farid H.2006, ‘Exposing digital forgeries through chromatic aberration’. In Proceedings of the 8th workshop on ACM Multimedia and Security Workshop, Geneva, Switzerland, pp. 48–55

[16] Lanh, T.V.L.T., Van Chong, K.-S., Chong, K.-S., Emmanuel, S., Kankanhalli, M.S. 2007, ‘A survey on digital camera image forensic methods’. In: 2007 IEEE International Conference on Multimedia and Expo, pp. 16–19

[17] Meena K.B, Tyagi V : 2019, ‘Image Forgery Detection: Survey and Future Directions’ Data, Engineering and Applications pp 163–194

[18] Ng, T., Chang, S., Sun, Q. 2004 ‘Blind detection of photomontage using higher order statistics’. In: IEEE International Symposium on Circuits System, pp. 7–10

[19] Popescu AC, Farid H. 2004 ‘Statistical tools for digital forensics. 6th International Workshop on Information Hiding, Toronto’ , pp. 128–147

[20] Popescu AC, Farid H. 2005 ‘Exposing digital forgeries in colour filter array interpolated images’. IEEE Trans Signal Process 53(10):3948–3959:

[21] Popescu AC, Farid H. 2005, ‘Exposing digital forgeries by detecting traces of resampling’. IEEE Trans Signal Process 53(2):758–767:(2005)

[22] Schneider, M., Chang, S. 1996, ‘A robust content based digital signature for image authentication’. In: IEEE International Conference on Image Processing. pp. 227–230