

A Project Report on

DIGITAL IMAGE FORGERY DETECTION USING CONVOLUTIONAL NEURAL NETWORKS

*submitted in partial fulfillment of the requirement for the award of the Degree of
BACHELOR OF TECHNOLOGY*

to

G. PULLAIAH COLLEGE OF ENGINEERING AND TECHNOLOGY

(Autonomous)

Approved proved by AICTE | NAAC Accreditation with 'A' Grade
Accredited by NBA (CSE, ECE & EEE) | Permanently Affiliated to JNTUA

by

ADDULA JAHNAVI	(20AT1A0554)
POTHULA INDHU	(20AT1A0552)
MEKALA HEMALATHA	(20AT1A0548)

Under the Guidance of

Mr. D JAYANARAYANA REDDY_{M.Tech., (Ph.D.)}

Assistant Professor



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
G. PULLAIAH COLLEGE OF ENGINEERING AND TECHNOLOGY
(Autonomous)

(Approved by AICTE | NAAC Accreditation with 'A' Grade | Accredited by NBA (ECE, CSE & EEE) |
Permanently Affiliated to JNTUA)

2020-2024

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
G. PULLAIAH COLLEGE OF ENGINEERING AND TECHNOLOGY
(Autonomous)

Approved by AICTE | NAAC Accreditation with 'A' Grade
Accredited by NBA (CSE, ECE & EEE) | Permanently Affiliated to JNTUA
Nandikotkur Road, Venkayapalli (V), Kurnool - 518452, Andhra Pradesh



CERTIFICATE

This is to certify that the project report entitled **“DIGITAL IMAGE FORGERY DETECTION USING CONVOLUTIONAL NEURAL NETWORKS”** being submitted by **ADDULA JAHNAVI (20AT1A0554), POTHULA INDHU (20AT1A0552), MEKALA HEMALATHA (20AT1A0548)** in partial fulfillment of the requirement for the award of the degree of Bachelor of Technology in Computer Science and Engineering of G. Pullaiah College of Engineering and Technology, Kurnool is a record of bonafide work carried out by them under my guidance and supervision.

The results embodied in this project report have not been submitted to any other university or institute for the award of any Degree or Diploma.

Internal Guide

Head of the Department

Mr. D JAYANARAYANA REDDY M.Tech., (Ph.D.)

Assistant Professor

Dr. M. SRI LAKSHMI M.Tech., Ph.D.

Associate Professor

Date of Viva-Voce: _____

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

We thank our project supervisor, **Mr. D JAYANARAYANA REDDY**, M.Tech., (Ph.D.) for his guidance, valuable suggestions and support in the completion of the project.

We would like to express our deep sense of gratitude and our sincere thanks to HOD

Dr. M. Sri Lakshmi, M.Tech., Ph.D. Department of Computer Science and Engineering, G. Pullaiah College of Engineering and Technology, Kurnool for providing the necessary facilities and encouragement towards the project work.

We owe indebtedness to our Principal **Dr. C. Srinivasa Rao**, M.E., Ph.D. G. Pullaiah College of Engineering and Technology, Kurnool for providing us the required facilities.

We are extremely grateful to Chairman, **Sri G.V.M. Mohan Kumar**, of G. Pullaiah College of Engineering and Technology, Kurnool, Andhra Pradesh for their good blessings.

We gratefully acknowledge and express our thanks to teaching and non-teaching staff of CSE Department.

Project Associates

ADDULA JAHNAVI	(20AT1A0554)
POTHULA INDHU	(20AT1A0552)
MEKALA HEMALATHA	(20AT1A0548)

ABSTRACT

Digital image forgery, encompassing a wide range of manipulations and alterations to images, has become a pervasive issue in today's digital age. Because of the ever-evolving technology, creating fake images is no longer difficult. These manipulations can be used for various purposes, including misinformation, fraud, and privacy breaches. This project focuses on the application of Convolutional Neural Networks (CNNs) for classification after being fed with Error Level Analysis (ELA) preprocessed images to detect image forgery using FIDAC (Forged Images Detection And Classification), which consists of original camera clicked images along with their tampered version. CNNs are employed to analyze image features and patterns, enabling the identification of various types of image manipulations and alterations. The proposed system consists of two phases; detection phase which is used to detect whether the image is forged or not, localization phase which is used to locate the forged part on the image. This project aims to enhance the accuracy and efficiency of image forgery detection, contributing to the preservation of digital image integrity in an era of increasing visual content manipulation.

Keywords: Image forgery detection, Convolutional Neural Networks (CNN), Error Level Analysis (ELA), Authentic, Tampered, Image processing, Deep learning.

LIST OF CONTENTS

ABSTRACT.....	iv
CHAPTER 1.....	1
1. INTRODUCTION	1
CHAPTER 2.....	2
2. LITERATURE SURVEY.....	2
CHAPTER 3.....	5
3. SYSTEM ANALYSIS AND REQUIREMENTS.....	5
3.1 Objective of the project.....	5
3.2 Existing System.....	6
3.3 Proposed System.....	6
3.4 System Requirements.....	7
3.5 System Study.....	8
CHAPTER 4.....	9
4. SYSTEM DESIGN.....	9
4.1 System Architecture.....	9
4.2 Data Flow Diagrams.....	10
4.3 UML Diagrams.....	12
CHAPTER 5.....	18
5. ALGORITHMS.....	18
5.1 CNN Algorithm.....	18
CHAPTER 6.....	21
6. SOFTWARE ENVIRONMET.....	21
6.1 Python.....	21
6.2 Modules used in the project.....	21
6.3 Front end technologies.....	24
6.4 Back end Technologies.....	26
6.5 Implementation.....	27

LIST OF CONTENTS

CHAPTER 7.....	31
7. EXPERIMENT AND ANALYSIS.....	31
7.1 Image Dataset	31
7.2 Models used for comparison	31
7.3 Model Comparison and Analysis	32
CHAPTER 8.....	36
8. SCREENSHOTS.....	36
CHAPTER 9.....	41
9. CONCLUSION AND FUTURE SCOPE	41
CHAPTER 10.....	42
10. REFERENCES	42

LIST OF FIGURES

4.1 System Architecture.....	9
4.2 Data Flow Diagram.....	11
4.3.1 Use Case Diagram.....	13
4.3.2 State Diagram.....	14
4.3.3 Sequence Diagram.....	16
4.3.4 Collaboration Diagram.....	17
5.1 CNN Architecture.....	19
5.2 Covets.....	19
5.3 Patch in an image.....	20
6.3.1 HTML page structure.....	24
7.1 Confusion matrix.....	33
7.2 (a) The evolution of training loss (left).....	34
(b) accuracy (right) versus number of epoch.....	34
8.1 Starting the application.....	36
8.2 Home page of the webpage.....	36
8.3 Webpage for image forgery detection.....	37
8.4 Input an image.....	37
8.5 Output and ELA of predicted image.....	38
8.6 Output of an authentic image.....	39
8.7 Output of metadata.....	39

CHAPTER 1

INTRODUCTION

Images became an important part of our life in the current days with the common usage of smart acquisition devices like cameras and smartphones, then the ease of sharing over the internet. In parallel to the current fact, the number of image-processing software increased and are accessible, specified anyone may easily modify and share images online. Several techniques have emerged and one among them is image splicing which is foremost commonly used manipulations. In this technique of forgery an image is forged by implanting some other pixels of another image. Some of these manipulations are difficult to detect for non-expert user. Moreover, some manipulated images aim at delivering misleading information which could be a threat to society. Therefore, the forensic researcher community focused on developing tools which validate the integrity of a picture. Many approaches detecting the image authenticity and assessing the integrity of the photographs are proposed. There are two approaches in authentication of images active and passive approach. In the active approach, information is hidden inside the image. For an active method, prior information of the image is required, due to which it is not feasible to use. Digital watermarking and signature come under it. The passive method of authentication detects forgery by analyzing binary information in an image. Copy-move, splicing, compression and resampling fall under passive methods. Copy-move images are created by copying one area from within an image and pasting it to another region within the same image. Image splicing is done by cropping regions from an image and pasting it onto another. Digital image forgery detection finds applications across a diverse spectrum of fields and industries. It ensures the trustworthiness of images in journalism, guarding against the spread of fake news and misinformation. In law enforcement, it aids criminal investigations by identifying manipulated evidence and tackling cybercrimes. The healthcare sector relies on it to maintain the integrity of medical images, while scientific research benefits from accurate imagery for valid outcomes. It safeguards intellectual property in advertising and entertainment, upholds security in surveillance systems, and curbs manipulated content on social media. Moreover, it contributes to historical preservation, supports legal proceedings, and plays a vital role in forensic analyses of cybercrimes. By detecting deepfakes, it protects privacy, and in the art world, it ensures the authentication of valuable artworks. These applications underscore the indispensable role of digital image forgery detection in preserving credibility and integrity in our increasingly visual and digital world.

CHAPTER 2

LITERATURE SURVEY

2.1 Image Forgery Detection: Survey and Future Directions

In this age of digitization, digital images are used as a prominent carrier of visual information. Images are becoming increasingly ubiquitous in everyday life. Unprecedented involvement of digital images can be seen in various paramount fields like medical science, journalism, sports, criminal investigation, image forensic, etc., where authenticity of image is of vital importance. Various tools are available free of cost or with a negligible amount of cost for manipulating images. Some tools can manipulate images to such an extent that it becomes impossible to discriminate by human visual system that image is forged or genuine. Hence, image forgery detection is a challenging area of research. It is evident that good quality work has been carried out in the past decade in the field of image forgery detection. However, there is still a need to pay much attention in this field, as image manipulation tools are becoming more and more sophisticated. The main purpose of this paper is to review the various existing methods developed for detecting the image forgery. A categorization of various forgery detection techniques has been presented in the paper.

2.2 A Deep Learning Approach to Universal Image Manipulation Detection Using a New Convolutional Layer

When creating a forgery, a forger can modify an image using many different image editing operations. Since a forensic examiner must test for each of these, significant interest has arisen in the development of universal forensic algorithms capable of detecting many different image editing operations and manipulations. In this paper, we propose a universal forensic approach to performing manipulation detection using deep learning. Specifically, we propose a new convolutional network architecture capable of automatically learning manipulation detection features directly from training data. In their current form, convolutional neural networks will learn features that capture an image's content as opposed to manipulation detection features. To overcome this issue, we develop a new form of convolutional layer that is specifically designed to suppress an image's content and adaptively learn manipulation detection features. Through a series of experiments, we demonstrate that our proposed approach can automatically learn how to detect multiple image manipulations without relying on pre-selected features or any preprocessing. The results of these experiments show that our proposed approach can automatically detect several different manipulations with an average accuracy of 99.10%.

2.3 Image forgery detection using error level analysis and deep learning

Many images are spread in the virtual world of social media. With the many editing software that allows so there is no doubt that many forgery images. By forensic the image using Error Level Analysis to find out the compression ratio between the original image and the fake image, because the original image compression and fake images are different. In addition to knowing whether the image is genuine or fake can analyze the metadata of the image, but the possibility of metadata can be changed. In this case the authors apply Deep Learning to recognize images of manipulations through the dataset of a fake image and original images via Error Level Analysis on each image and supporting parameters for error rate analysis. The result of our experiment is that we get the best accuracy of training 92.2% and 88.46% validation by going through 100 epoch.

2.4 Forged Face Detection using ELA and Deep Learning Techniques

The paper presents real and fake facial image recognition using Deep learning and CNN. Image forgery recognition becomes a difficult task to find the authenticity of an image with the naked eye. This research aims to evaluate the working of different deep learning techniques in the novel “Real and Fake Face detection” dataset by Computational Intelligence Photography Lab, Yonsei University. For the detection of forged faces, the first step of the proposed method is image normalization for real and fake image recognition. Normalized images are then preprocessed using Error Level Analysis (ELA) and train to different pre-trained deep learning models. We finetune these models for categorization of 2 classes that are forged and real to evaluate these models' performance. From all tested models, VGG models give the best training accuracy of 91.97% and 92.09% on VGG-16 and VGG-19, whereas VGG-16 shows the good test set Accuracy using a smaller number of epochs, which is competitively better than all other techniques. Results of these models were evaluated using confusion matrix evaluation measures and compared with state-of-the-art techniques.

2.5 Digital image forgery detection using deep learning approach

This paper presents an algorithm for detecting one of the most commonly used types of digital image forgeries - splicing. The algorithm is based on the use of the VGG-16 convolutional neural network. The proposed network architecture takes image patches as input and obtains classification results for a patch: original or forgery. On the training stage we select patches from original image regions and on the borders of embedded splicing. The obtained results

demonstrate high classification accuracy (97.8% accuracy for fine-tuned model and 96.4% accuracy for the zero-stage trained) for a set of images containing artificial distortions in comparison with existing solutions. Experimental research was conducted using CASIA dataset.

2.6 An evaluation of Error Level Analysis in image forensics

The advancement in digital image tampering has encouraged studies in the image forensics fields. The image tampering can be found over various image formats such as Joint Photographic Experts Group (JPEG). JPEG is the most common format that supported by devices and applications. Therefore, researchers have been studying the implementation of JPEG algorithm in the image forensics. In this paper, the Error Level Analysis (ELA) technique was evaluated with different types of image tampering, including JPEG compression, image splicing, copy-move and image retouching. From the experiment, the ELA showed reliability with JPEG compression, image splicing and image retouching forgery.

CHAPTER 3

SYSTEM ANALYSIS AND REQUIREMENTS

3.1 OBJECTIVE OF THE PROJECT

In order to test the application an image is uploaded to the system and applying machine learning algorithm to check whether the image is tampered or not.

This project extends to the meticulous analysis of image properties and anomalies, authentication of image integrity, detection of steganographic data, and ensuring data integrity in applications like medical imaging and legal evidence. It also embraces the emerging challenges and solutions involving machine learning which automate and secure the forgery detection process. Specifically, the project focuses on utilizing Convolutional Neural Networks (CNNs) for classification after preprocessing images using Error Level Analysis (ELA). ELA helps highlight areas of the image that may have been tampered with, providing valuable input to the CNN. The proposed system is designed to analyze image features and patterns to identify forged images accurately. By enhancing the accuracy and efficiency of image forgery detection, the project aims to contribute to the preservation of digital image integrity in an era of increasing visual content manipulation.

After the execution, results show if the uploaded image is forged or not and to give metadata of the image as output.

3.2 EXISTING SYSTEM

In the existing system it is only able to detect whether the image is forged or not. It does not show or localize the part of forgery on the image. And also, the accuracy levels of the existing system are too low to determine.

Furthermore, the existing system lacks the capability to provide detailed information regarding the specific regions of an image that have been forged or manipulated. This limitation hinders forensic analysis efforts, as investigators often require precise localization of forgery to assess the authenticity of visual content effectively. Additionally, the accuracy levels attained by the current system fall below the desired threshold, undermining its practical utility in real-world applications where reliable forgery detection is paramount.

3.3 PROPOSED SYSTEM

This proposed system is mainly focusing on image splicing technique where we project a technique with two phases. In the first phase the project is the detection phase which is used to detect whether the image is forged or not, and the second phase is the localization phase which is used to locate the forged part on the image. Here are two approaches in authentication of images: active and passive approach. In the active approach, information is hidden inside the image. For an active method, prior information of the image is required, due to which it is not feasible to use. Digital watermarking and signature come under it. The passive method of authentication detects forgery by analyzing binary information in an image. Copy-move, splicing, compression and resampling fall under passive methods. Copy-move images are created by copying one area from within an image and pasting it to another region within the same image. Image splicing is done by cropping regions from an image and pasting it onto another.

3.4 SYSTEM REQUIREMENTS:

3.4.1 HARDWARE REQUIREMENTS:

- System : 11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz.
- Hard Disk : 218 GB.
- Ram : 8.00 GB.

3.4.2 SOFTWARE REQUIREMENTS:

- Operating System : Windows
- Coding Language : Python 3.6
- Front end : HTML, CSS, Javascript
- Back end : Django Framework
- Code Editor : Visual Studio Code

3.5 SYSTEM STUDY

3.5.1 FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

- ECONOMICAL FEASIBILITY
- TECHNICAL FEASIBILITY
- SOCIAL FEASIBILITY

3.5.2 ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus, the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

3.5.3 TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

3.5.4 SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

CHAPTER 4

SYSTEM DESIGN

4.1 SYSTEM ARCHITECTURE:

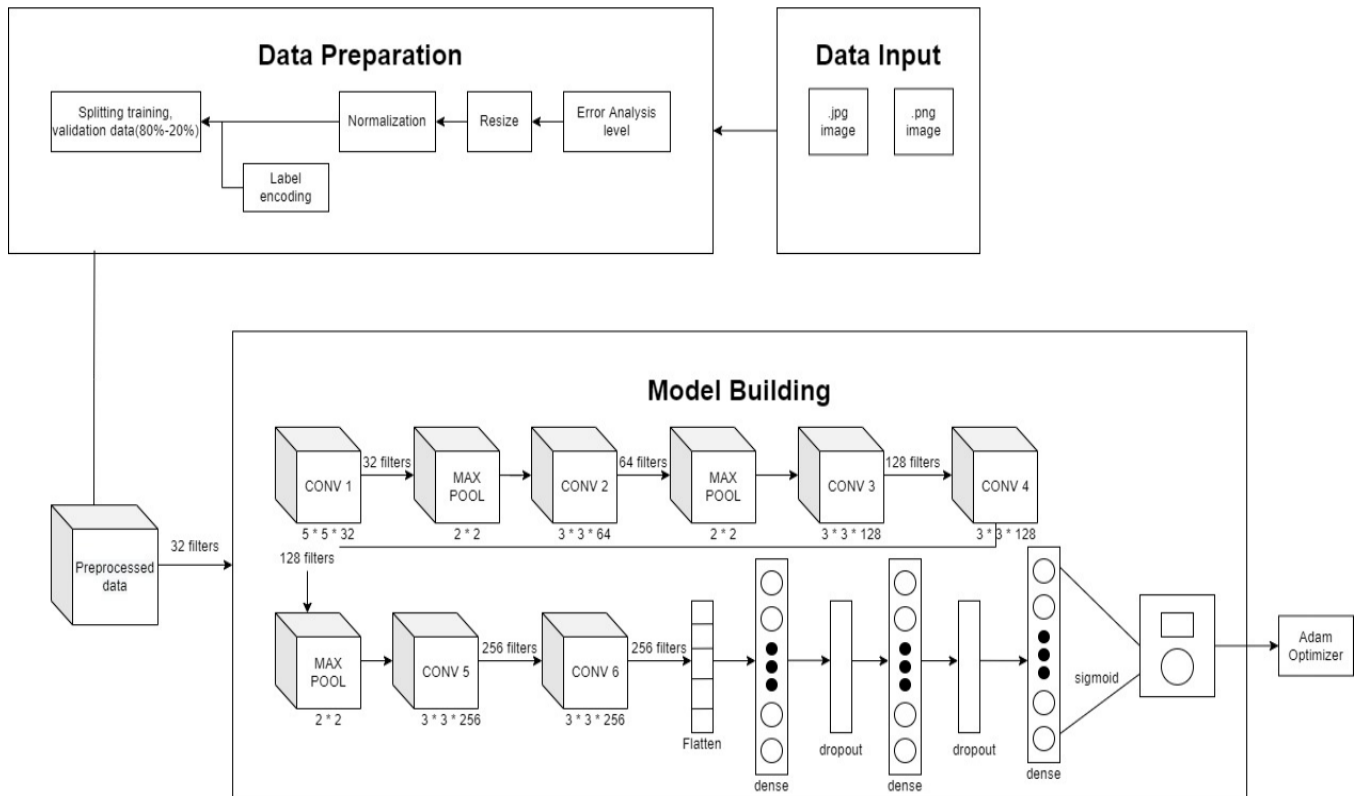


Fig 4.1 System Architecture

System Architecture shows the detailed information of the machine learning algorithm where we give some records of images for data preprocessing, which is achieved by Error Analysis Level, resizing and normalizing the data. Then we input the preprocessed data for training the data and for testing the data and we train the machine learning algorithms using Convolutional Neural Networks. As a result, we give a new input image for prediction. If we got false results, we train the algorithm to get correct output results.

4.2 DATA FLOW DIAGRAM:

The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system. The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system. DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into levels that represent increasing information flow and functional detail.

The process begins with the input image, which may contain one or more digital forgeries such as splicing, or tampering. The input image undergoes preprocessing steps to enhance its quality and prepare it for CNN analysis. Preprocessing may include operations like resizing, normalization, and noise reduction. The preprocessed image is then fed into the CNN model for feature extraction. The CNN architecture consists of multiple layers, including convolutional layers, pooling layers, and fully connected layers, which extract relevant features from the input image. The feature-extracted data is compared with a labeled dataset of authentic and forged images. This dataset is used for training the CNN model. It contains images with known ground truth labels indicating whether they are authentic or forged. During the training process, the CNN model learns to distinguish between authentic and forged images by adjusting its internal parameters based on the training data. This process involves forward and backward propagation of data through the network to minimize the loss function.

Once training is complete, the CNN model is fully trained and ready for inference. It has learned to identify patterns and features indicative of digital forgeries. When presented with a new, unseen image, the trained CNN model performs inference to predict whether the image is authentic or forged. The input image undergoes the same preprocessing steps as during training before being passed through the trained model. The CNN model generates a prediction or confidence score indicating the likelihood that the input image is authentic or forged. This output can be used for decision-making or further analysis. Finally, the detection result, along with any relevant metadata or additional information, is provided as output. This information can be used for forensic analysis, legal proceedings, or other purposes.

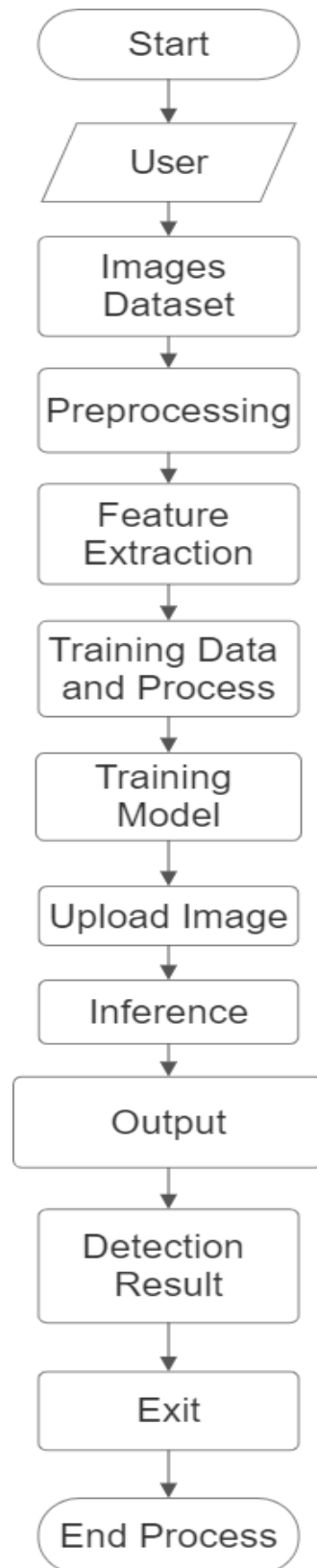


Fig 4.2 Data Flow Diagram

4.3 UML DIAGRAMS:

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object-oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing object-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

GOALS:

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.

4.3.1 Use Case Diagram:

The use case diagram is used to identify the primary elements and processes that form the system. The primary elements are termed as "actors" and the processes are called "use cases."

The use case diagram shows which actors interact with each use case.

The use case diagram for the project illustrated in the fig 4.3.1 the interactions between the user and the system components. The primary actor, the User, initiates various actions within the system. These actions are represented by the use cases, including "Upload Dataset," where the user uploads datasets of original and forged images for training the CNN model, and "Input Image," where the user provides new images for forgery detection. Additionally, the system encompasses several internal processes, such as "Preprocessing" to enhance image quality and

prepare them for analysis, "Feature Extraction" using the CNN model to identify relevant features, and "Training Data and Process" to train the model using labeled datasets.

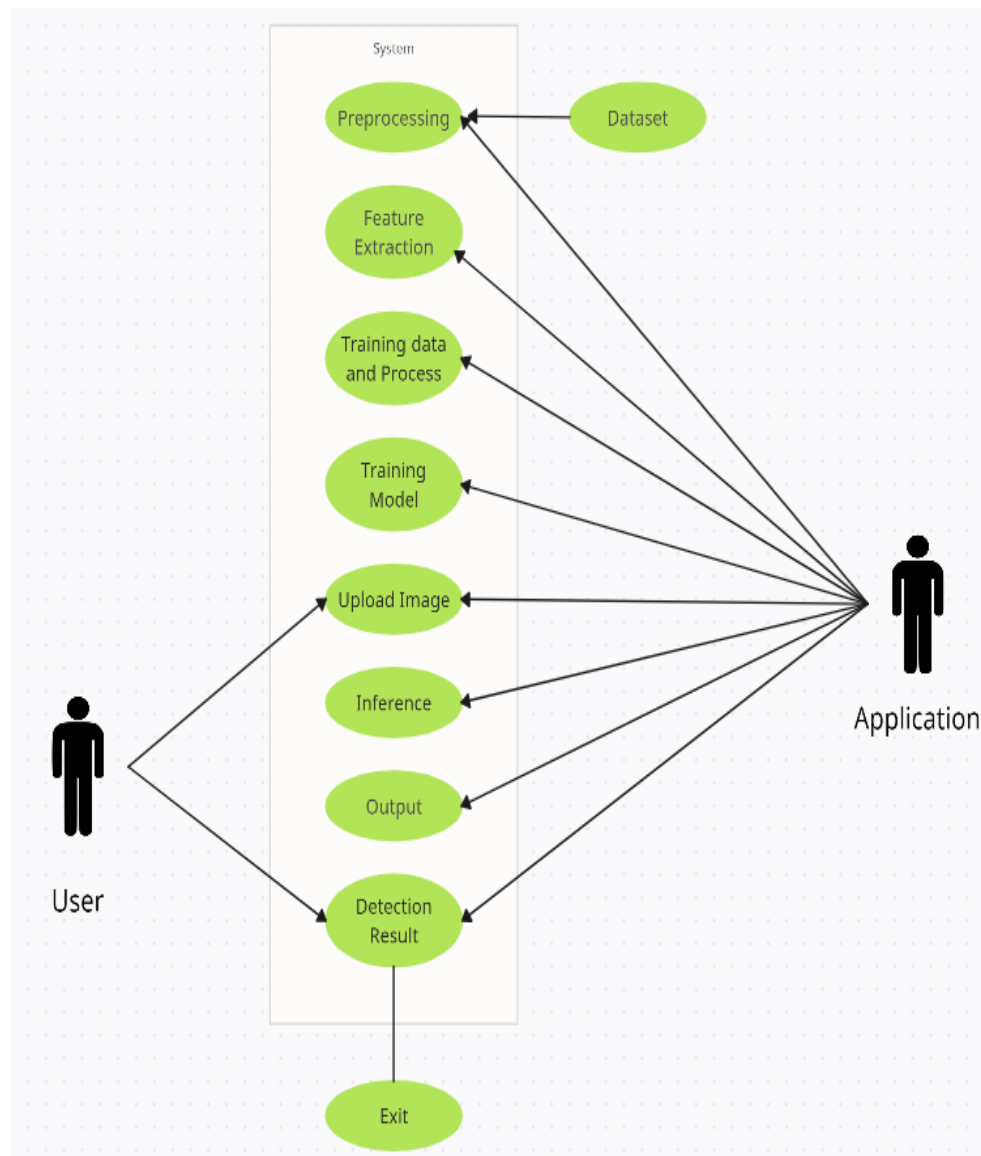


Fig 4.3.1 Use Case Diagram

Once trained, the "Trained Model" is capable of inference, predicting whether input images are authentic or forged. The detection result, along with any relevant metadata, is presented to the user through the "Detection Result" use case. Associations between actors and use cases depict their involvement, while associations between use cases illustrate the flow of events within the system. This diagram provides a comprehensive overview of the functionalities and interactions involved in digital image forgery detection using CNN.

4.3.2 State diagram:

A state diagram, as the name suggests, represents the different states that objects in the system undergo during their life cycle. Objects in the system change states in response to events. In addition to this, a state diagram also captures the transition of the object's state from an initial state to final state in response to events affecting the system.

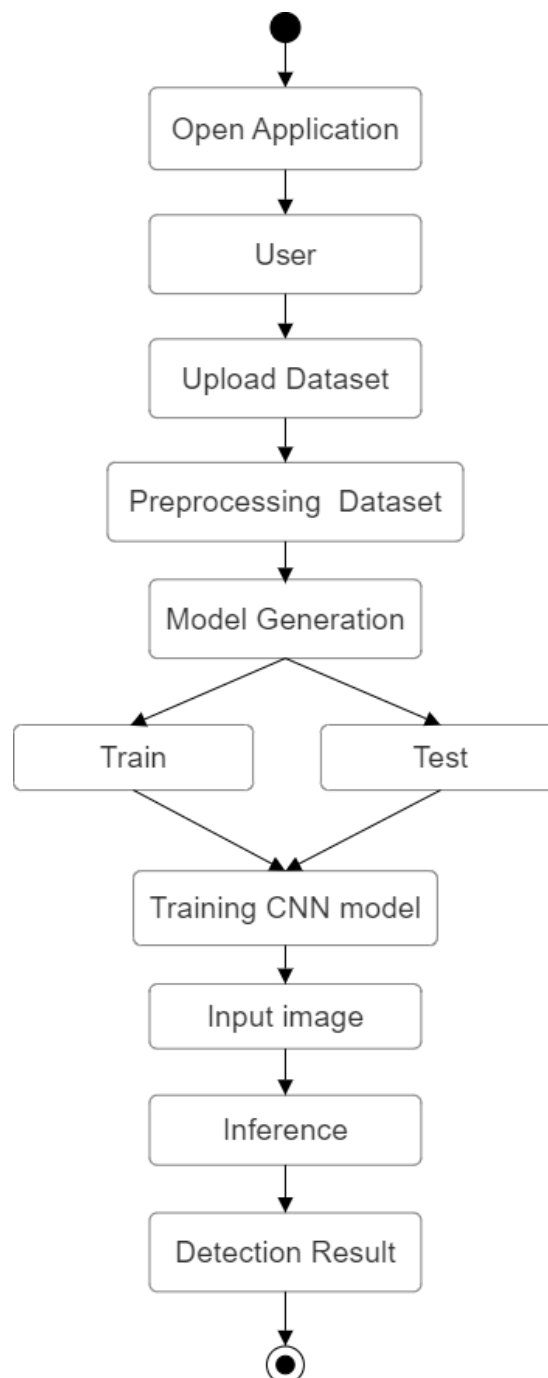


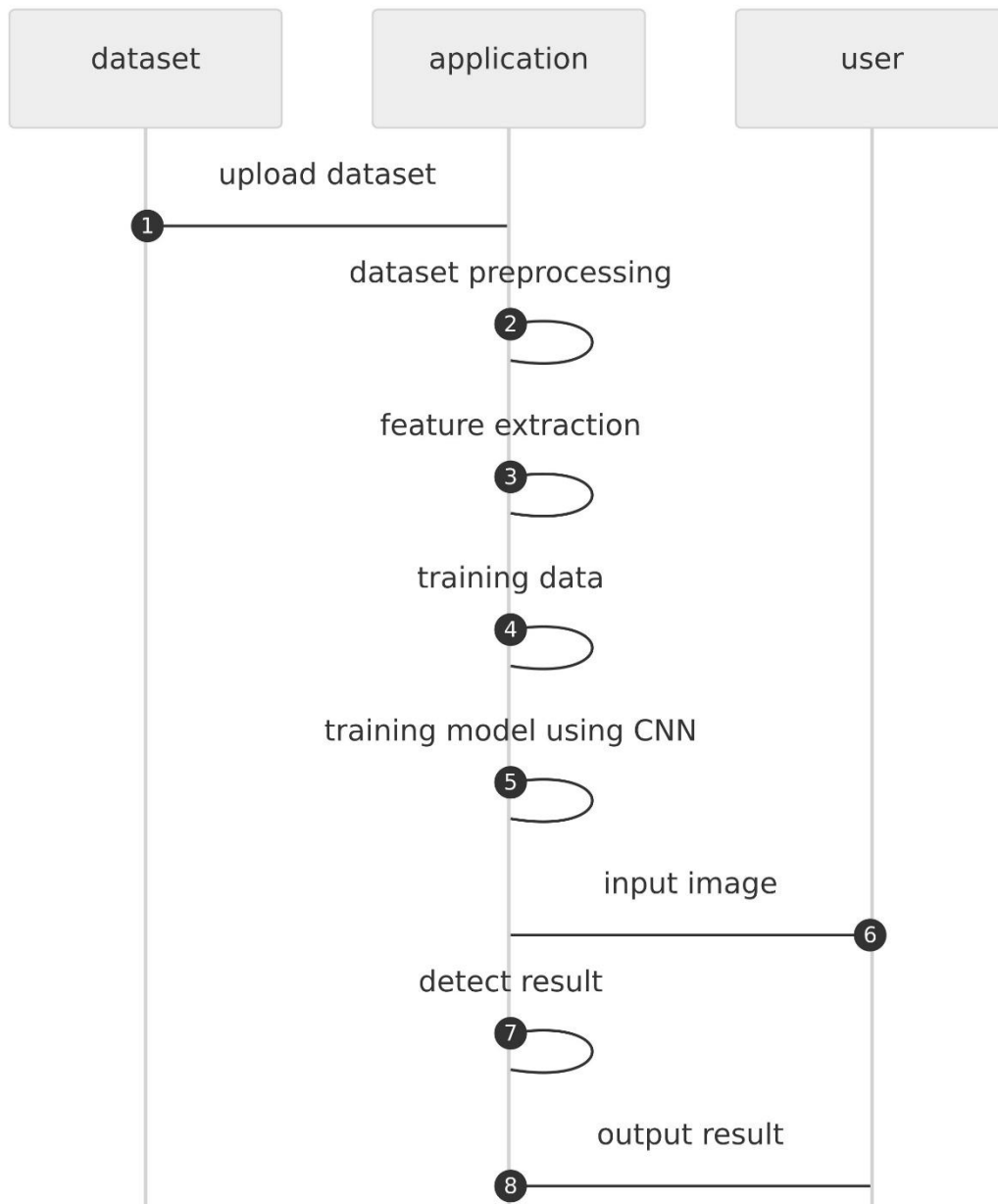
Fig 4.3.2 State Diagram

The state diagram for the project shown in the Fig 4.3.2 encapsulates the system's behavior and transitions are depicted through different states and the transitions between them. Initially, the system resides in an Idle state, awaiting user interaction. Upon the user's action of uploading datasets of original and forged images, the system transitions to the Dataset Uploaded state, indicating the availability of training data. Subsequently, the system moves to the Model Training state, where it trains the CNN model using the uploaded datasets. Upon completion of training, the system transitions to the Model Trained state, signaling readiness for inference. When the user provides a new image for forgery detection, the system enters the Image Input state. The trained model then performs inference on the input image, transitioning to the Inference state. Finally, upon completion of the inference process, the system moves to the Result Displayed state, where it presents the forgery detection results to the user. Through these states and transitions, the state diagram provides a comprehensive overview of the system's behavior and progression through various stages of forgery detection.

4.3.3 Sequence diagram:

A sequence diagram represents the interaction between different objects in the system. The important aspect of a sequence diagram is that it is time-ordered. This means that the exact sequence of the interactions between the objects is represented step by step. The sequence diagram for the project portrays the chronological flow of interactions between the user and the forgery detection system is visualized in fig 4.3.3.

The diagram involves two primary participants: the User and the system components responsible for forgery detection. Initially, the User uploads datasets of original and forged images, triggering the preprocessing and training processes within the system. The preprocessing module applies necessary operations to enhance image quality, while the training module trains the CNN model using the uploaded datasets. Upon completion of training, the system is ready for inference. When the User provides a new image for forgery detection, the system processes it through preprocessing and performs inference using the trained CNN model. The result presentation module then generates and displays the forgery detection results to the User. Through this sequence of interactions, the sequence diagram provides a comprehensive overview of the forgery detection process, from dataset upload to result presentation.

Sequence Diagram**Fig 4.3.3 Sequence Diagram**

4.3.6 Collaboration diagram:

A collaboration diagram groups together the interactions between different objects. The interactions are listed as numbered interactions that help to trace the sequence of the interactions. The collaboration diagram helps to identify all the possible interactions that each object has with other objects.

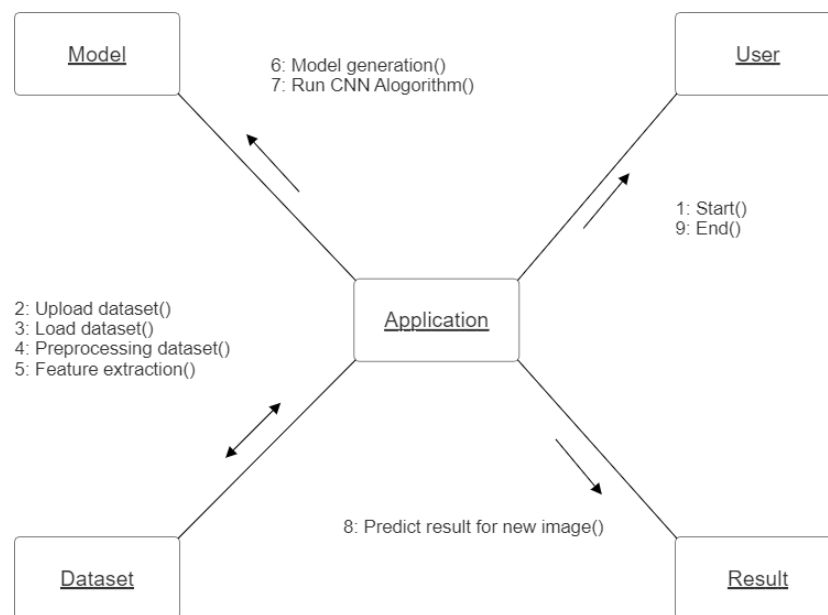


Fig 4.3.6 Collaboration diagram

In a collaboration diagram for the project, the interactions between objects representing the user and the forgery detection system are illustrated in fig 4.3.6. the interactions and relationships between various system components and the user are illustrated. The diagram depicts how these entities collaborate to achieve the task of forgery detection. The primary participants include the User and different system components responsible for preprocessing, training, inference, and result presentation. Through collaboration links, the flow of communication and interactions between these entities is visualized. For instance, the User interacts with the preprocessing module to provide input images, which are then processed and passed to the training module for CNN model training. Once trained, the model collaborates with the inference module to analyze input images and generate forgery detection results. Finally, the result presentation module collaborates with the User to display the detection outcomes. Overall, the collaboration diagram offers a clear representation of how different system components collaborate to accomplish forgery detection tasks, facilitating understanding of the system's overall behavior and interactions.

CHAPTER 5

ALGORITHMS

5.1 CNN algorithm:

A Convolutional Neural Network (CNN) is a type of Deep Learning neural network architecture commonly used in Computer Vision. Computer vision is a field of Artificial Intelligence that enables a computer to understand and interpret the image or visual data. When it comes to Machine Learning, Artificial Neural Networks perform really well. Neural Networks are used in various datasets like images, audio, and text. Different types of Neural Networks are used for different purposes, for example for predicting the sequence of words we use Recurrent Neural Networks more precisely an LSTM, similarly for image classification we use Convolution Neural networks. In this blog, we are going to build a basic building block for CNN.

In a regular Neural Network there are three types of layers:

Input Layers: It's the layer in which we give input to our model. The number of neurons in this layer is equal to the total number of features in our data (number of pixels in the case of an image).

Hidden Layer: The input from the Input layer is then fed into the hidden layer. There can be many hidden layers depending on our model and data size. Each hidden layer can have different numbers of neurons which are generally greater than the number of features. The output from each layer is computed by matrix multiplication of the output of the previous layer with learnable weights of that layer and then by the addition of learnable biases followed by activation function which makes the network nonlinear.

Output Layer: The output from the hidden layer is then fed into a logistic function like sigmoid or softmax which converts the output of each class into the probability score of each class.

The data is fed into the model and output from each layer is obtained from the above step is called feedforward, we then calculate the error using an error function, some common error functions are cross-entropy, square loss error, etc. The error function measures how well the network is performing. After that, we backpropagate into the model by calculating the derivatives. This step is called Backpropagation which basically is used to minimize the loss.

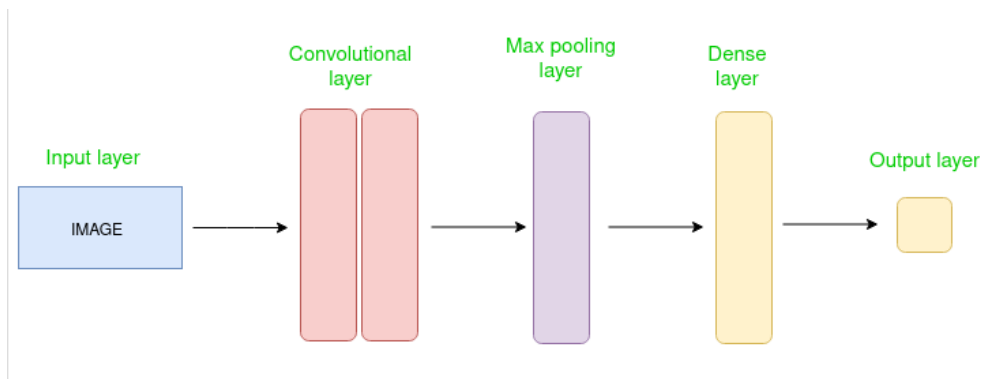


Fig 5.1 CNN Architecture

The Convolutional layer applies filters to the input image to extract features, the Pooling layer down samples the image to reduce computation, and the fully connected layer makes the final prediction. The network learns the optimal filters through backpropagation and gradient descent.

Convolution Neural Networks or convnets are neural networks that share their parameters. Imagine you have an image. It can be represented as a cuboid having its length, width (dimension of the image), and height (i.e the channel as images generally have red, green, and blue channels).

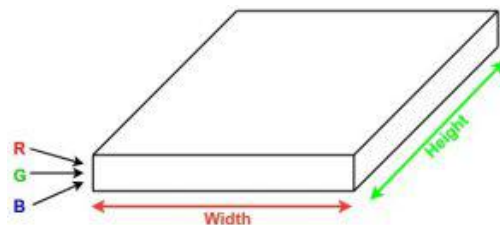


Fig 5.2 Convets

Now imagine taking a small patch of this image and running a small neural network, called a filter or kernel on it, with say, K outputs and representing them vertically. Now slide that neural network across the whole image, as a result, we will get another image with different widths, heights, and depths. Instead of just R, G, and B channels now we have more channels but lesser width and height. This operation is called Convolution. If the patch size is the same as that of the image it will be a regular neural network. Because of this small patch, we have fewer weights.

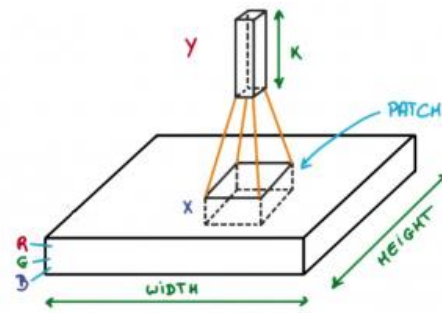


Fig 5.3 Patch in an image

Now let's talk about a bit of mathematics that is involved in the whole convolution process.

- Convolution layers consist of a set of learnable filters (or kernels) having small widths and heights and the same depth as that of input volume (3 if the input layer is image input).
- For example, if we have to run convolution on an image with dimensions $34 \times 34 \times 3$. The possible size of filters can be $a \times a \times 3$, where 'a' can be anything like 3, 5, or 7 but smaller as compared to the image dimension.
- During the forward pass, we slide each filter across the whole input volume step by step where each step is called stride (which can have a value of 2, 3, or even 4 for high-dimensional images) and compute the dot product between the kernel weights and patch from input volume.
- As we slide our filters we'll get a 2-D output for each filter and we'll stack them together as a result, we'll get output volume having a depth equal to the number of filters. The network will learn all the filters.

CHAPTER 6

SOFTWARE ENVIRONMENT

6.1 Python:

Python is a set of instructions that we give in the form of a Programme to our computer to perform any specific task. It is a Programming language having properties like it is interpreted, object-oriented and it is high-level too. Due to its beginner-friendly syntax, it became a clear choice for beginners to start their programming journey. The major focus behind creating it is making it easier for developers to read and understand, also reducing the lines of code. Python was created in 1980s by Guido van Rossum. During his research at the National Research Institute for Mathematics and Computer Science in the Netherlands, he created Python – a super easy programming language in terms of reading and usage. The first ever version was released in the year 1991 which had only a few built-in data types and basic functionality.

6.2 Modules Used in the Project:

1. Django

Django is a Python-based web application framework that is free and open source. A framework is simply a collection of modules that facilitate development. They're grouped together and allow you to build apps or websites from scratch rather than starting from scratch. "Rapid development and clean, pragmatic design" are key benefits of Django. When installed on a web server, the Django web framework can assist developers in quickly creating a feature-rich, secure, and scalable web frontend.

2. Keras

Keras is a high-level, deep learning API developed by Google for implementing neural networks. It is written in Python and is used to make the implementation of neural networks easy. It also supports multiple backend neural network computation. Keras is relatively easy to learn and work with because it provides a python frontend with a high level of abstraction while having the option of multiple back-ends for computation purposes. This makes Keras slower than other deep learning frameworks, but extremely beginner-friendly.

3. Matplotlib

Matplotlib is a powerful plotting library in Python used for creating static, animated, and interactive visualizations. Its primary purpose is to provide users with the tools and

functionality to represent data graphically, making it easier to analyze and understand. Matplotlib is popular due to its ease of use, extensive documentation, and wide range of plotting capabilities. It offers flexibility in customization, supports various plot types, and integrates well with other Python libraries like NumPy and Pandas.

4. Numpy

Numpy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

- A powerful N-dimensional array object
- Sophisticated (broadcasting) functions
- Tools for integrating C/C++ and Fortran code
- Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, Numpy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined using Numpy which allows Numpy to seamlessly and speedily integrate with a wide variety of databases.

5. Opencv

OpenCV, short for Open-Source Computer Vision Library, is an open-source computer vision and machine learning software library. Originally developed by Intel, it is now maintained by a community of developers under the OpenCV Foundation. Opencv is a huge open-source library for computer vision, machine learning, and image processing. Now, it plays a major role in real-time operation which is very important in today's systems. By using it, one can process images and videos to identify objects, faces, or even the handwriting of a human.

6. Pandas

Pandas is a powerful and open-source Python library. The Pandas library is used for data manipulation and analysis. Pandas consist of data structures and functions to perform efficient operations on data. Here is a list of things that we can do using Pandas.

- Easy handling of missing data (represented as NaN) in floating point as well as non-floating point data.

- Data set cleaning, merging, and joining.
- Columns can be inserted and deleted from DataFrame and higher-dimensional objects.
- Data Visualization.

7. Pillow

Python Pillow module is built on top of PIL (Python Image Library). It is the essential modules for image processing in Python. But it is not supported by Python 3. But we can use this module with the Python 3.x versions as PIL. It supports the variability of images such as jpeg, png, bmp, gif, ppm, and tiff. Python pillow library is used to image class within it to show the image. The image modules that belong to the pillow package have a few inbuilt functions such as load images or create new images, etc.

8. Scikit

Scikit-learn has emerged as a powerful and user-friendly Python library. Scikit-learn is an open-source Python library that implements a range of machine learning, pre-processing, cross-validation, and visualization algorithms using a unified interface. It is an open-source machine-learning library that provides a plethora of tools for various machine-learning tasks such as Classification, Regression, Clustering, and many more.

9. Tensorflow

TensorFlow is an open-source library developed by Google primarily for deep learning applications. It also supports traditional machine learning. TensorFlow was originally developed for large numerical computations without keeping deep learning in mind. However, it proved to be very useful for deep learning development as well, and therefore Google open-sourced it. TensorFlow accepts data in the form of multi-dimensional arrays of higher dimensions called tensors. Multi-dimensional arrays are very handy in handling large amounts of data.

10. Streamlit

Streamlit is an open-source app framework in python language. It helps us create beautiful web apps for data science and machine learning in a little time. It is compatible with major python libraries such as scikit-learn, keras, PyTorch, latex, numpy, pandas, matplotlib, etc.

6.3 Front end Technologies

1. HTML

HTML stands for Hyper Text Markup Language. It is the standard markup language used to create web pages. HTML is a combination of Hypertext and Markup language. Hypertext defines the link between web pages. A markup language is used to define the text document within the tag to define the structure of web pages. It uses HTML tags and attributes to describe the structure and formatting of a web page. HTML consists of various elements, that are responsible for telling search engines how to display page content. For example, headings, lists, images, links, and more. Some of the features of HTML are

- It is easy to learn and easy to use.
- It is platform-independent.
- Images, videos, and audio can be added to a web page.
- Hypertext can be added to the text.
- It is a markup language.

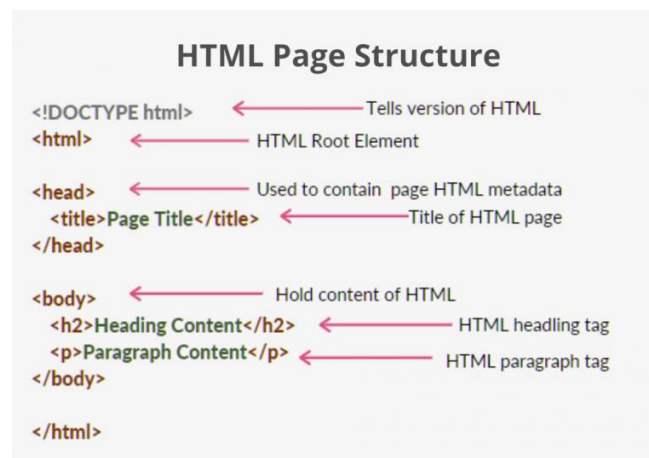


Fig 6.3.1 HTML Page Structure

2. CSS

Cascading Style Sheets (CSS) is a language used to style web pages that contain HTML elements, defining how elements are displayed on webpages, including layout, colors, fonts, and other properties of the elements on a web page. CSS works by targeting HTML elements and applying style rules to define how they should be displayed, including properties like color, size, layout, and positioning.

There are three types of CSS which are given below:

Inline CSS:

Inline CSS is a method of applying styling directly to individual HTML elements using the “style” attribute within the HTML tag, allowing for specific styling of individual elements within the HTML document, overriding any external or internal styles.

Internal or Embedded CSS:

Internal or Embedded CSS is defined within the HTML document’s <style> element. It applies styles to specified HTML elements, The CSS rule set should be within the HTML file in the head section i.e. the CSS is embedded within the <style> tag inside the head section of the HTML file.

External CSS:

External CSS contains separate CSS files that contain only style properties with the help of tag attributes (For example class, id, heading, ... etc). CSS property is written in a separate file with a .css extension and should be linked to the HTML document using a link tag. It means that, for each element, style can be set only once and will be applied across web pages.

3. JavaScript

JavaScript is a lightweight, cross-platform, single-threaded, and interpreted compiled programming language. It is also known as the scripting language for webpages. It is well-known for the development of web pages, and many non-browser environments also use it. JavaScript is a weakly typed language (dynamically typed). JavaScript can be used for Client-side developments as well as Server-side developments. JavaScript is both an imperative and declarative type of language. JavaScript contains a standard library of objects, like Array, Date, and Math, and a core set of language elements like operators, control structures, and statements.

JavaScript can be added to HTML file in two ways:

Internal JS: We can add JavaScript directly to our HTML file by writing the code inside the <script> tag. The <script> tag can either be placed inside the <head> or the <body> tag according to the requirement.

External JS: We can write JavaScript code in other files having an extension.js and then link this file inside the <head> tag of the HTML file in which we want to add this code.

6.4 Back end Technologies

1. Django:

Python-based web framework Django allows you to create efficient web applications quickly. It is also called batteries included web framework Django because It provides built-in features for everything including Django Admin Interface, default database – SQLite3, etc. When you're building a website, you always need a similar set of components: a way to handle user authentication (signing up, signing in, signing out), a management panel for your website, forms, a way to upload files, etc. Django Python gives you ready-made components to use for rapid development. There are many more benefits of using the Django framework. Django follows the MVT design pattern (Model View Template).

- Model - The data you want to present, usually data from a database.
- View - A request handler that returns the relevant template and content - based on the request from the user.
- Template - A text file (like an HTML file) containing the layout of the web page, with logic on how to display the data.

6.5 IMPLEMENTATION:

MODULES:

1. Selection of Datasets

We have majorly considered the two below datasets for our proposed algorithm and experimental analysis.

CASIA Dataset

CASIA's image tampering detection evaluation database, is a popular, freely available database for the field of visual media forgery identification. It contains photos with resolution of 384x256 or 256x384 in eight categories (animals, architecture, items, characters, nature, plants, scenery, and textures). Tampered photos are made by cutting and pasting (splicing, copy-move) operations in Adobe Photoshop on authentic photographs, and tampered portions may have random shape and size, rotation, or distortion.

FIDAC Dataset

The FIDAC dataset, which stands for forged images detection and classification, is a self-created dataset, made up of photos clicked by a camera device. Applications like Adobe Photoshop, B612, Picsarts, and Canva are used to further alter these captured images. On these photos, forgeries such as copy-move, splicing, retouching, recoloring, rotation, scaling, and filtering have been performed. This dataset is created with the goal of improving the trained model's accuracy and realism.

2. Preprocessing technique

We feed our proposed CNN architecture with images converted to their respective ELA representations. The image has been broken into small chunks and is further re-compressed individually using error level analysis (ELA) at a predefined error rate of 95%. By taking a lossy image and re-compressing it with a known error rate, ELA calculates the absolute difference between the analyzed and re-compressed images. The definition of the computation is given as: ELA: Error levels analysis, $ELA(p_1, p_2)$ where p_1, p_2 are row and column indices, can be represented by

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

for each color channel, where X is the image suspected to be forged image and X_{rc} is the re-compressed image. Total error levels are defined as the aggregate value of all error levels for

all color levels, as in

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

where, $i = 1, 2, 3$, for a RGB image.

3. Proposed CNN architecture

There are 15 layers in our proposed CNN architecture. Figure 4.1 shows the block diagram to represent a flow of information and steps taken for the final outcome. The first layer is made up of an input image with the dimensions 128x128x3 (128 for height and width, 3 for depth) (RGB). It is convolved with 32 filters of size 5x5 resulting in dimension of 124x124x32. The second layer is a Pooling operation which filters size 2x2 and stride of 2. Hence the resulting image dimension will be 62x62x32.

<i>Layer</i>	<i>Output Shape</i>	<i>Parameter</i>
Conv2D	(None, 124, 124, 32)	2432
MaxPooling	(None, 62, 62, 32)	0
Conv2D	(None, 60, 60, 64)	18496
MaxPooling	(None, 30, 30, 64)	0
Conv2D	(None, 28, 28, 128)	73856
Conv2D	(None, 26, 26, 128)	147584
MaxPooling	(None, 13, 13, 128)	0
Conv2D	(None, 11, 11, 256)	295168
Conv2D	(None, 11, 11, 256)	590080
Flatten	(None, 30976)	0
Dense	(None, 64)	1982528
Dropout	(None, 64)	0
Dense	(None, 128)	8320
Dropout	(None, 128)	0
Dense	(None, 1)	129

Table 6.1 Proposed architecture summary

Similarly, the third layer has a convolution operation with 64 filters of size 5x5, followed by a fourth pooling layer with a similar filter size of 2x2 and stride of 2. As a result, the image size is decreased to 30x30x64 pixels. In each convolutional layer forward it to the activation function. The Rectified Linear Unit (ReLU) is used as an activation function that removes non-linear values from process data. In order to minimize the image dimension, this pattern is repeated two more times. The tenth layer is a fully linked convolutional layer with 64 filters that is applied after the image dimension is lowered. Each of the 64 units in this layer will be connected to the 30976 units from the preceding layers in this layer. With 128 units, the sixth layer is also a completely connected layer. The final layer processes the output using “sigmoid”

activation and only one neuron for the final output layer. Sigmoid activation classifies the image into one of two states: authentic or tampered. The summary of the model is displayed as in Table 6.1.

4. Forgery Detection

The application takes a new image as an input and performs an inference on image and outputs whether the image is authentic or forged, confidence level, metadata about the image along with the ELA, Edge map and Noise Analysis.

MODULE DESCRIPTION

To implement this project, we will design following modules

- 1) Upload dataset: using this module we will upload dataset of original and forged images to application
- 2) Preprocessing: The input image undergoes preprocessing steps to enhance its quality and prepare it for CNN analysis. Preprocessing may include operations like resizing, normalization, and noise reduction.
- 3) Feature Extraction: The preprocessed image is then fed into the CNN model for feature extraction. The CNN architecture consists of multiple layers, including convolutional layers, pooling layers, and fully connected layers, which extract relevant features from the input image.
- 4) Training Data and Process: The feature-extracted data is compared with a labeled dataset of authentic and forged images. This dataset is used for training the CNN model. It contains images with known ground truth labels indicating whether they are authentic or forged. During the training process, the CNN model learns to distinguish between authentic and forged images by adjusting its internal parameters based on the training data. This process involves forward and backward propagation of data through the network to minimize the loss function.
- 5) Trained Model: Once training is complete, the CNN model is fully trained and ready for inference. It has learned to identify patterns and features indicative of digital forgeries.
- 6) Input Image: A new input image is given to the application for detecting whether the image is forged or not.

- 7) Inference: When presented with a new, unseen image, the trained CNN model performs inference to predict whether the image is authentic or forged. The input image undergoes the same preprocessing steps as during training before being passed through the trained model.
- 8) Output: The CNN model generates a prediction or confidence score indicating the likelihood that the input image is authentic or forged. This output can be used for decision-making or further analysis.
- 9) Detection Result: Finally, the detection result, along with any relevant metadata or additional information, is provided as output. This information can be used for forensic analysis, legal proceedings, or other purposes.

CHAPTER 7

EXPERIMENT AND ANALYSIS

We performed an experiment on a system with configurations of 16 GB Ram with 2GB graphics card of Intel I5 processor. The experiment involves two datasets and four CNN architectures where in 3 are pre-defined models obtained from the Keras library. The details of the considered models are given in section III-B. These pre-defined architectures are the most popular CNN architectures used in the field of image classification. We have compared our proposed neural network with these trained models available. Also, we have created 3 sets of datasets whose details are in section III-A. This is done to analyze the efficiency of training in order to get accurate results.

7.1 Image Dataset

For performing the experiment, we have created 3 sets of datasets each of 500 images (250 per class). First set consists of all images from the CASIA dataset. Second set contains all images from FIDAC. The third set is a combination of CASIA and FIDAC datasets such that the tampered class consists of 125 forged images from CASIA FIDAC respectively and the authentic class consists of 125 authentic images from CASIA FIDAC respectively. The datasets are further splitted into 80%-20% such that 400 images are used for training the model and 100 images are used for validation purposes. The Test Dataset consists of 100 images (50 images per class) which is a combination of both CASIA and FIDAC dataset.

7.2 Models used for comparison

- VGG16 and VGG19: The VGG16 architecture is a convolutional neural network (CNN) that won the 2014 ILSVR(Imagenet) competition. VGG16 is made up of 16 layers of various weights. It is widely acknowledged as one of the most sophisticated vision model designs ever created. Rather than having a huge number of hyperparameters, VGG16 concentrates on having 3x3 filter convolution layers with a stride 1 and always using the same padding and maxpool layer of a 2x2 filter of stride 2. Throughout the architecture, the convolution and max pool layers are organised in the same way. At the conclusion, there are two Fully Connected layers, followed by a softmax for output. With an estimated 138 million parameters, this network is fairly large. The VGG19

model is a variant of the VGG model with a total of 19 layers (16 convolution layers, 3 Fully connected layers, 5 MaxPool layers and 1 SoftMax layer).

- Inception: Inception, a convolutional neural network architecture developed by Google, was named the winner of the ImageNet Large Scale Visual Recognition Challenge 2014. The key innovation on the inception model is the inception module. This is a concatenation of the outputs of a block set of parallel convolutional layers with varying filter sizes (e.g. 1x1, 3x3, 5x5) and a 3x3 maximum pooling layer.

Following Table 7.1 displays the total and trainable parameters for each model considered for performing the experiment.

<i>PARAMETERS/MODEL</i>	<i>VGG16</i>	<i>INCEPTION</i>	<i>VGG19</i>	<i>PROPOSED</i>
Total	27,560,769	40,679,201	32,870,465	3,118,593
Trainable	12,846,081	18,876,417	12,846,081	3,118,593

Table 7.1: Model Parameters

7.3 Model Comparison and Analysis

The Experiment was carried in two parts. First part is where we train all the considered models for 15 epoch cycles and in the second part, we train all models for 50 epoch cycles for each dataset combination. Table 7.2 and Table 7.3 show the respective results. Here, training accuracy is about how well is the model able to classify the two images over the training dataset (400 images) which was passed over the CNN. Validation accuracy is about how well is the model able to classify the two images over the validation dataset (100 images obtained during dataset splitting). Testing accuracy is about how well is the model able to classify the two images over the test dataset (100 images from external test directory). Hence, the testing accuracy is given the utmost weightage as it tests over images from the combination of different datasets.

Coming to the results obtained in the first part of the experiment whose results are in Table 7.2, we can see that the training accuracy of the CASIA-FIDAC combination dataset is the highest for all the algorithms when comparing their respective results with FIDAC and CASIA. Also, we can see that the accuracy of VGG19 is the highest and the proposed model is second highest for the CASIA-FIDAC combination. Thus, the performance of the two is the best this part.

Coming to the results obtained in the second part of the experiment whose results are in Table 7.3, we again observe that results for CASIA-FIDAC are better than the other two datasets for all the models. Also, we can see that the accuracy obtained this time by the proposed model has crossed the accuracy of VGG19. Thus, we can state that the proposed model has performed better compared to the other models.

The figures 7.1 and 7.2 depict the results obtained by the proposed model in the form of confusion matrix and graph. From the confusion matrix in figure 7.1, we can see that 80 images out of 94 are correctly classified. This confusion matrix is plotted on the basis of the training results received.

Table 7.4 states the performance measures of our proposed architecture when trained on CASIA-FIDAC dataset for 50 epochs. This performance is on the basis of the testing results received. When the model was tested on the Test dataset, 84 images out of 99 were correctly classified, giving an accuracy of 84.85%. In figure 7.2, we see that the loss decreases with that increase in epochs whereas the accuracy increases with increase of epochs.

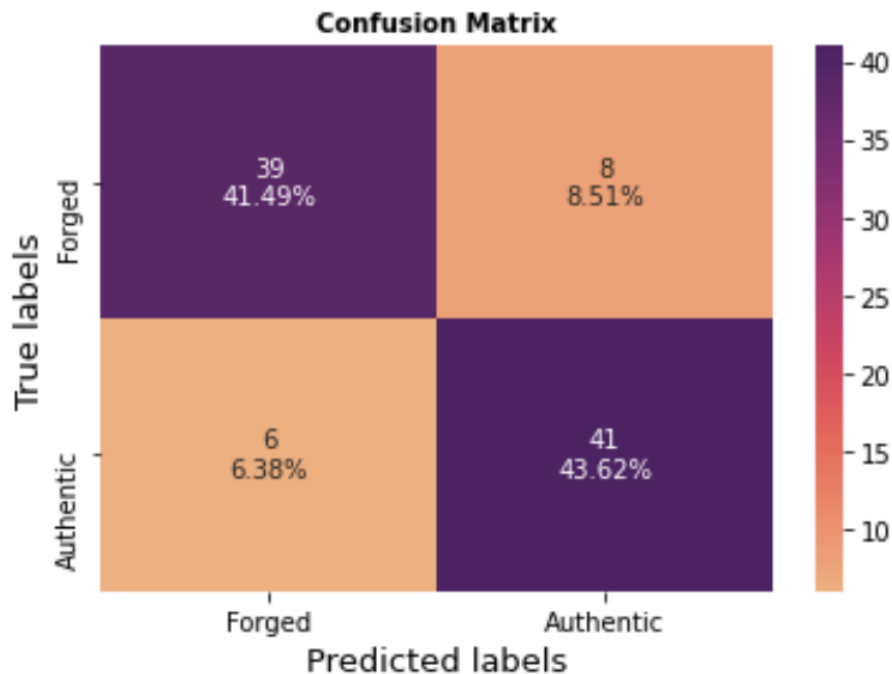


Fig 7.1 Confusion Matrix

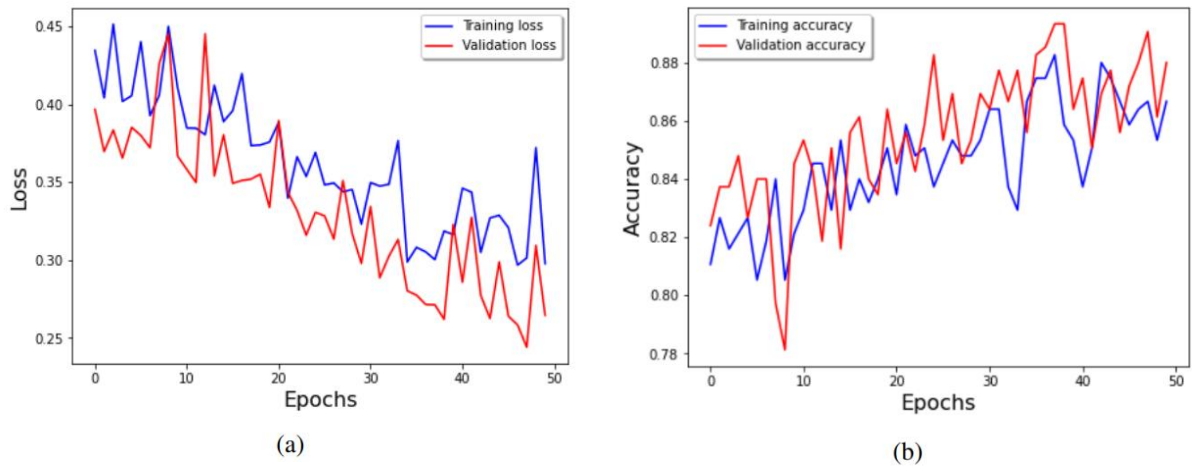


Fig 7.2 (a) The evolution of training loss (left) and (b) accuracy (right) versus number of epoch

<i>DATASET</i>	<i>Accuracy</i>	<i>VGG16 (%)</i>	<i>INCEPTION (%)</i>	<i>VGG19 (%)</i>	<i>PROPOSED (%)</i>
CASIA	Training	98.95	89	97.89	78
	Validation	100	84	100	80
	Testing	71.71	69.69	77.8	60.60
FIDAC	Training	71.43	70	70.21	79
	Validation	70.84	68	72	84
	Testing	66.66	73.73	50	70.70
CASIA-FIDAC	Training	87.1	71	79.79	87
	Validation	72	64	80	80
	Testing	79.79	78.78	84.84	81.81

Table 7.2 Analysis of various Image Forgery Models (for 15 epochs)

<i>DATASET</i>	<i>Accuracy</i>	<i>VGG16 (%)</i>	<i>INCEPTION (%)</i>	<i>VGG19 (%)</i>	<i>PROPOSED (%)</i>
CASIA	Training	94.74	81	92.63	88
	Validation	100	84	100	92
	Testing	72.72	70.71	76.76	71.71
FIDAC	Training	77.66	63	76.6	74
	Validation	84	72	92	64
	Testing	71.71	68.68	59.59	64.64
CASIA-FIDAC	Training	86.17	72	80.85	85
	Validation	80	76	88	80
	Testing	80.8	69.69	83.83	84.85

Table 7.3 Analysis of various Image Forgery Models (for 50 epochs)

Coming to the computation time of each model, we have recorded the average time required per epoch in Table 7.3, for the steps per epoch kept as 20. We can see that the time taken by the proposed model is the second least. Hence our model has also performed well with respect to the overall computation time.

<i>MEASURE</i>	<i>VALUE</i>
Sensitivity	0.8269
Specificity	0.8723
Precision	0.8776
Negative Predictive Value	0.8200
False Positive Rate	0.1277
False Discovery Rate	0.1224
False Negative Rate	0.1731
Accuracy	0.8485
F1 Score	0.8515

Table 7.4 Performance measures of the proposed architecture

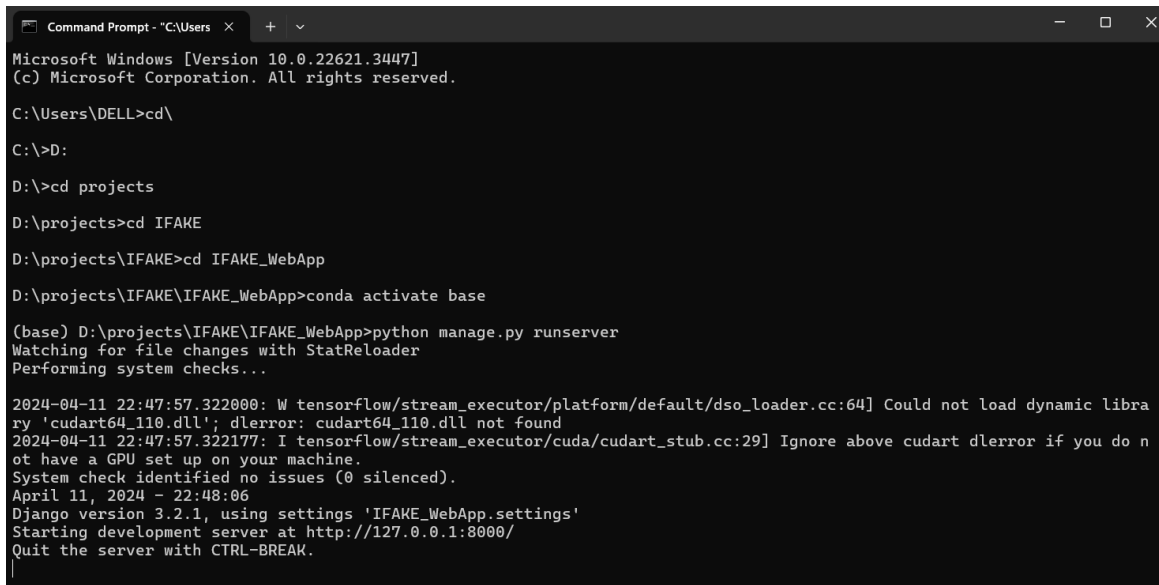
	<i>VGG16</i>	<i>INCEPTION</i>	<i>VGG19</i>	<i>PROPOSED</i>
Time/Epoch(sec)	88	35	117.5	37

Table 7.5 Avg time taken by each epoch of the model

CHAPTER 8

SCREENSHOTS

To run the project, open the command prompt and navigate to the project directory, activate the virtual environment for Django and run the server. It can be achieved by using the commands shown in the below figure 8.1.



```
Command Prompt - "C:\Users\DELL"
Microsoft Windows [Version 10.0.22621.3447]
(c) Microsoft Corporation. All rights reserved.

C:\Users\DELL>cd\

C:\>D:

D:\>cd projects

D:\projects>cd IFAKE

D:\projects\IFAKE>cd IFAKE_WebApp

D:\projects\IFAKE\IFAKE_WebApp>conda activate base

(base) D:\projects\IFAKE\IFAKE_WebApp>python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

2024-04-11 22:47:57.322000: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'cudart64_110.dll'; dlderror: cudart64_110.dll not found
2024-04-11 22:47:57.322177: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart dlerror if you do not have a GPU set up on your machine.
System check identified no issues (0 silenced).
April 11, 2024 - 22:48:06
Django version 3.2.1, using settings 'IFAKE_WebApp.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

Fig 8.1 Starting the application

After the server has started ctrl + click (or) copy and paste the link of the local server into web browser. Then it will take you to the home page of the website. It looks like figure 8.2.

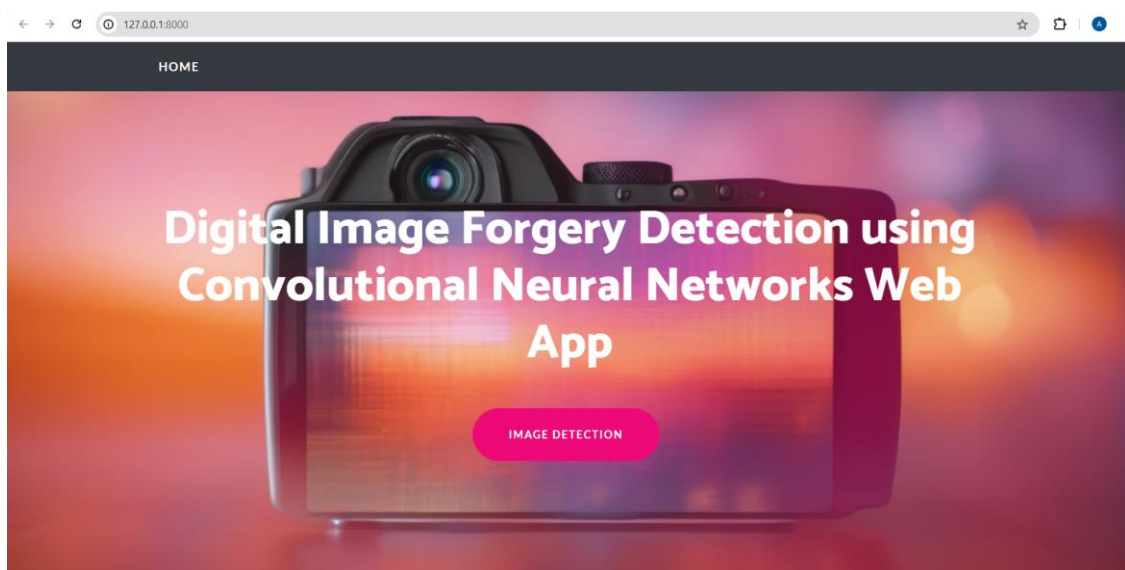


Fig 8.2 Home page of the webpage

Click on the “IMAGE DETECTION” button, which will navigate to the web page for image forgery detection. And you will get a screen as shown in figure 8.3.

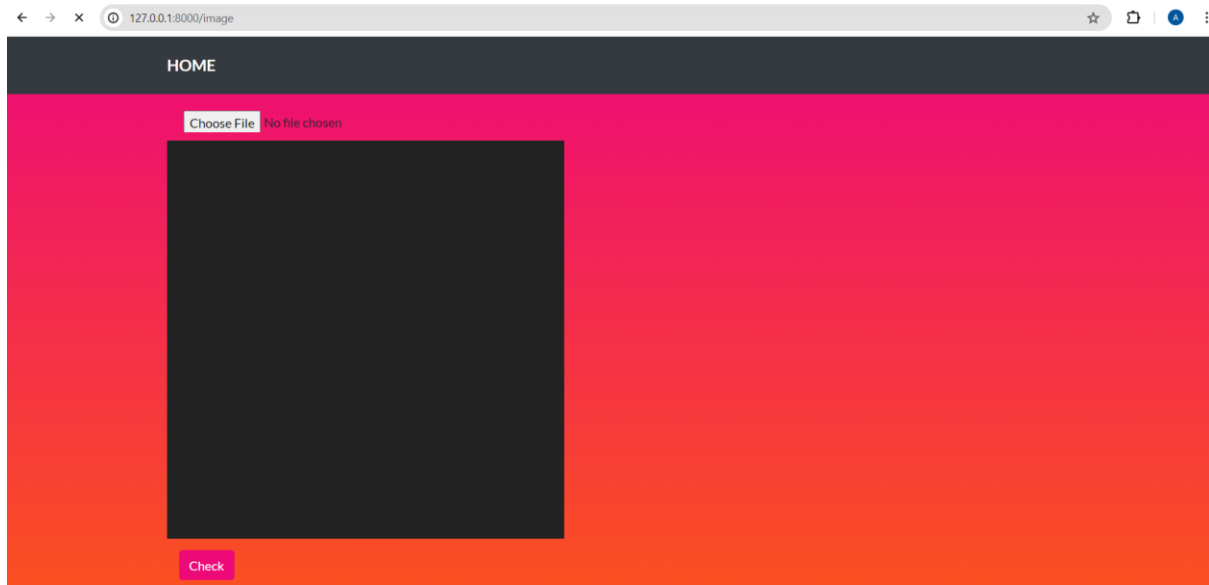


Fig 8.3 Webpage for image forgery detection

Now click on the button “Choose File” and choose the image you want to predict whether if it’s an authentic image or a forged image. The screen looks like the figure 8.4.

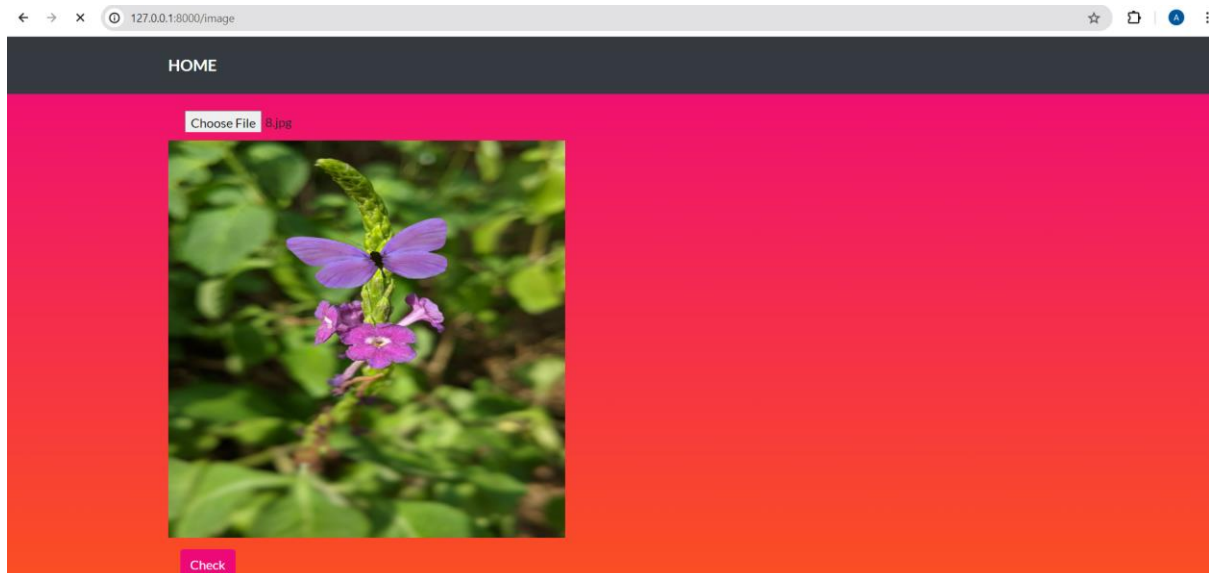


Fig 8.4 Input an image

Now click on the “Check” button bellow the image to predict whether it is an authentic image or a forged image. When we click the button, it outputs the result whether if it is a forged or an authentic one along with its confidence level. Along with the result we can also see the localized part of the forged image by clicking on the “ELA”, which means Error Level Analysis. And it is shown in the figure 8.5

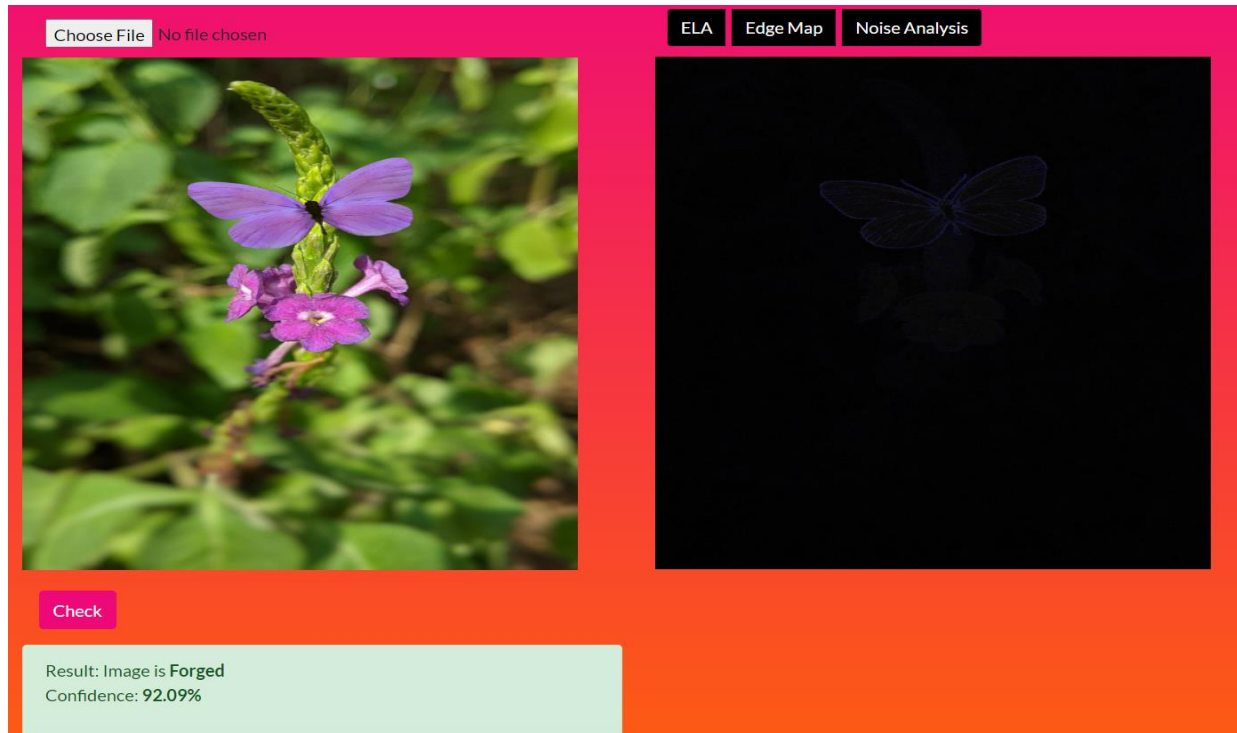


Fig 8.5 Output and ELA of predicted image

We can also view the image Edge Map and Noice Analysis by clicking on the respective buttons. For an authentic image it shows the image output as shown in the figure 8.6.

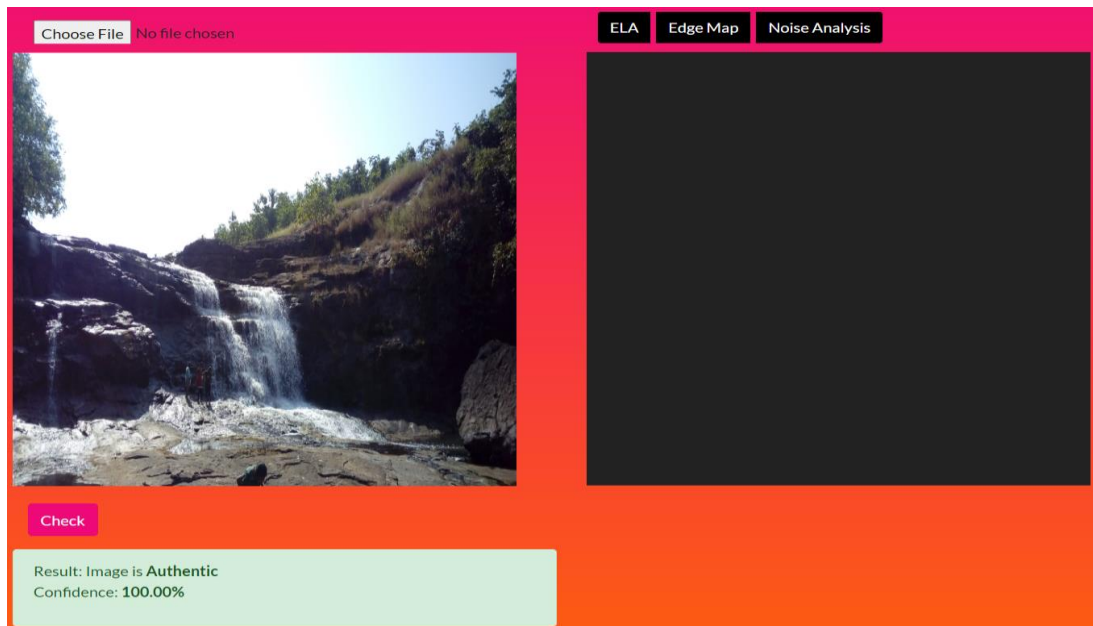


Fig 8.6 Output of an authentic image

Along with the Error Level Analysis, it also gives the output of Edge Map and Noise Analysis as a result for the input image. Along with the output of predicted result it also gives the metadata of the image. It is as shown in the figure 8.7.

Metadata	
Attribute	Value
- Image width	4160 pixels
- Image height	2368 pixels
- Image orientation	Horizontal (normal)
- Bits/pixel	24
- Pixel format	YCbCr
- Creation date	42
- Compression	JPEG (Baseline)
- Date-time original	42
- Date-time digitized	42
- Comment	94% (approximate)
- Format version	JFIF 1.01
- MIME type	image/jpeg
- Endianness	Big endian

Fig 8.7 Output of metadata

CHAPTER 9

CONCLUSION AND FUTURE SCOPE

With the widespread proliferation of manipulated images and the ever-increasing sophistication of forgery techniques, the importance of this field cannot be overstated. It safeguards the integrity of images across a multitude of domains, from journalism to healthcare, ensuring that the visual information we encounter is reliable and unaltered. We proposed our CNN architecture and our Dataset FIDAC through this paper. By conducting a comparative analysis of combination of the image forgery techniques with the mentioned datasets, we have effectively proved that using a combination of two datasets i.e. CASIA and FIDAC results in better accuracy. Along with this, our proposed network is able to compete with the most famous classifiers in a much faster and efficient way. This being just an experiment, the accuracy value should not be considered as a final result. The accuracy received can be further improved by increasing the size of the input dataset and increasing the number of epochs. Also considering better optimizers and augmentation techniques might also give better results.

CHAPTER 10

REFERENCES

- [1] M. Kunj and V. Tyagi, “Image forgery detection: Survey and future directions,” *Data, Engineering and Applications*, p. 163–194, 2019.
- [2] B. Bayar and M. C. Stamm, “A Deep Learning Approach to Universal Image Manipulation Detection Using a New Convolutional Layer,” *Proceedings of the 4th ACM Workshop on Information Hiding and Multimedia Security*, 2016.
- [3] I. B. Sudiatmika, F. Rahman, T. Trisno and S. Suyoto, “Image forgery detection using error level analysis and deep learning,” *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, vol. 17, no. 2, p. 653, 2018.
- [4] Qurat-ul-ain, N. Nida, A. Irtaza and N. Ilyas, “Forged face detection using ELA and Deep Learning Techniques,” *2021 International Bhurban Conference on Applied Sciences and Technologies (IBCAST)*, 2021.
- [5] A. Kuznetsov, “Digital image forgery detection using deep learning approach,” *Journal of Physics: Conference Series*, vol. 1368, p. 032028, 2019.
- [6] N. B. A. Warif, M. Y. I. Idris, A. W. A. Wahab and R. Salleh, “An evaluation of Error Level Analysis in image forensics,” *2015 5th IEEE International Conference on System Engineering and Technology (ICSET)*, pp. 23-28, 2015.
- [7] K. Team, “Keras Documentation: VGG16,” Keras, [Online].
Available: <https://keras.io/api/applications/vgg/>. [Accessed 06 April 2022].
- [8] K. Team, “Keras Documentation: VGG19,” Keras, [Online].
Available: <https://keras.io/api/applications/vgg/>. [Accessed 06 April 2022].
- [9] K. Team, “Keras Documentation: Inceptionv3,” Keras, [Online].
Available: <https://keras.io/api/applications/inceptionv3/>. [Accessed 06 April 2022].
- [10] S. Pawar, G. Pradhan, B. Gaurangi and S. Bhutad, “FIDAC- Forged Images Detection And Classification,” *IEEE Dataport*, 2022. [Online].
Available: <https://dx.doi.org/10.21227/4det-6512>.

[11] J. Dong, W. Wang and T. Tan, "2013 IEEE China Summit and International Conference on Signal and Information Processing," CASIA Image Tampering Detection Evaluation Database, pp. 422-426, 2013.

[12] S. Pawar, G. Pradhan, B. Gaurangi and S. Bhutad, "ViFoDAC- Video Forgery Detection And Classification," <https://dx.doi.org/10.21227/63t2-ea77> IEEE Dataport, 2022. [Online].

Available: <https://dx.doi.org/10.21227/63t2-ea77>.

[13] L. Bondi, S. Lameri, D. Guera, P. Bestagini, E. J. Delp and S. Tubaro, "Tampering detection and localization through clustering of camera-based CNN features," 2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), pp. 1855-1864, 2017.