

# **ILLEGAL IMAGE IDENTIFICATION USING CNN SUPERVISION**

Submitted in partial fulfillment of the  
requirements for the award of  
Bachelor of Engineering degree in Computer Science and Engineering

By

**C.S Arunesh (Reg.No - 39110085)  
Yadlapalli Aravind Babu (Reg.No - 39111118)**



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING  
SCHOOL OF COMPUTING**

## **SATHYABAMA**

**INSTITUTE OF SCIENCE AND TECHNOLOGY  
(DEEMED TO BE UNIVERSITY)**

**Accredited with Grade "A" by NAAC | 12B Status by UGC | Approved by AICTE  
JEPPIAAR NAGAR, RAJIV GANDHISALAI,  
CHENNAI - 600119**

**APRIL - 2023**



# **SATHYABAMA**

**INSTITUTE OF SCIENCE AND TECHNOLOGY  
(DEEMED TO BE UNIVERSITY)**

Accredited with Grade "A" by NAAC | 12B Status by UGC | Approved by AICTE

[www.sathyabama.ac.in](http://www.sathyabama.ac.in)

---

## **DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

### **BONAFIDE CERTIFICATE**

This is to certify that this Project Report is the bonafide work of **C.S. Arunesh (Reg.No - 39110085)** and **Yadlapalli Aravind Babu (Reg.No - 39111118)**, who carried out Project Phase-2 entitled **"ILLEGAL IMAGE IDENTIFICATION USING CNN SUPERVISION"** under my supervision from January 2023 to April 2023.

**Internal Guide**

**Dr. RAJA SHREE S, M.E., Ph.D.**

**Head of the Department**

**Dr. L. LAKSHMANAN, M.E., Ph.D.**



Submitted for Viva-voce Examination held on **20.04.2023**

---

**Internal Examiner**

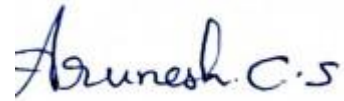
**External Examiner**

## DECLARATION

I, **C.S.Arunesh (Reg.No - 39110085)** hereby declare that the Project Phase-2 Report entitled **“ILLEGAL IMAGE IDENTIFICATION USING CNN SUPERVISION”** done by me under the guidance of **Dr. RAJA SHREE S, M.E., Ph.D.** is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in **Computer Science and Engineering**.

**DATE: 20.04.23**

**PLACE: CHENNAI**

A handwritten signature in blue ink that reads "Arunesh C.S." with a stylized flourish at the end.

**SIGNATURE OF THE CANDIDATE**

## **ACKNOWLEDGEMENT**

I am pleased to acknowledge my sincere thanks to the **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala M.E., Ph. D, Dean**, School of Computing, and **Dr. L. Lakshmanan M.E., Ph.D., Head of the Department of Computer Science and Engineering** for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. Raja Shree S, M.E., Ph.D.**, for her valuable guidance, suggestions, and constant encouragement that paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

## ABSTRACT

An increasing number of digital images are being shared and accessed through websites, media, and social applications. Many of these images have been modified and are not authentic. Camera Model Identification is the process of determining which camera or model has been used to capture an image. Recent advances in the use of deep convolutional neural networks (CNNs) have facilitated the task of analyzing the veracity and authenticity of largely distributed image datasets. We examined the problem of identifying the camera model or type that was used to take an image and that can be spoofed. Due to the linear nature of CNNs and the high dimensionality of images, neural networks are vulnerable to attacks with adversarial examples. These examples are imperceptibly different from correctly classified images but are misclassified with high confidence by CNNs. In this, we describe a counter-forensic method capable of subtly altering images to change their estimated camera model when they are analyzed by any CNN-based camera model detector. Lossy compression such as JPEG is considered the most common type of intentional or inadvertent concealment of image forgery, which leads us to experiment with our proposal on this manipulation. Experimental results showed the importance of this step to improve the effectiveness of our approach against recent literature approaches. This analysis led us to get a more robust and accurate framework. The image to be detected is masked by a sliding window that moves from left to right or up to bottom. The area in the window is matched with the fingerprint, which is related to the used camera, to determine the tampered area in the image. It is estimated by the residue noise and a group of images from the camera. The correlation coefficient-based method can detect the tampered areas from the same image or several different images. The experiments on synthetic tampered images demonstrate most of the tampered areas can be detected.

## TABLE OF CONTENTS

Chapter No	TITLE	Page No.
	<b>ABSTRACT</b>	<b>v</b>
	<b>LIST OF FIGURES</b>	<b>ix</b>
	<b>LIST OF ABBREVIATIONS</b>	<b>x</b>
<b>1</b>	<b>INTRODUCTION</b>	<b>1</b>
<b>2</b>	<b>LITERATURE SURVEY</b>	<b>4</b>
	2.1 Inferences from Literature Survey	6
	2.2 Open problems in Existing System	7
	2.3 Overall Objective	8
<b>3</b>	<b>REQUIREMENTS ANALYSIS</b>	<b>9</b>
	3.1 Feasibility studies/risk analysis of the project	9
	3.1.1 Feasibility Studies	9
	3.1.2 Risk Analysis	10
	3.2 Requirement Process of the project	11
	3.2.1 Data Preprocessing	11
	3.2.2 CNN	12
	3.2.3 Camera Model Identification	13
	3.2.4 CNN Layers	15
	3.2.5 Max Pooling	15
	3.2.6 Tensor Flow	15
	3.2.7 Python Packages	16
	3.3 Software Requirement Specification Document	17
	3.3.1 Purpose of this document	18
	3.3.2 Scope of this document	18
	3.3.3 Functional Requirements	18

	3.3.4	Non-Functional Requirements	19
	3.4	Hardware Requirements	19
	3.5	Software Requirements	19
<b>4</b>		<b>DESCRIPTION OF THE PROPOSED SYSTEM</b>	<b>20</b>
	4.1	Selected methodology of process model	20
	4.2	The architecture of the proposed system	23
	4.3	Description of Software for Implementation and Testing plan of the Proposed Model/System	24
	4.3.1	Testing plan of the proposed system	24
	4.4	Project Management Plan	25
	4.5	Financial report on estimated costing	26
<b>5</b>		<b>IMPLEMENTATION DETAILS</b>	<b>27</b>
	5.1	Development and Deployment Setup	27
	5.1.1	Data Set Description	28

	i o n	
5.2	Algorit hm	29
5.3	Testin g	30
5.4	Project Imple mentat ion	31
6	<b>RESULTS AND DISCUSSIO NS</b>	<b>39</b>
7	<b>CONCLUSI ON</b>	<b>40</b>
7.1	Conclu sion	40
7.2	Future work	40
7.3	Resea rch Issues	42
7.4	Imple mentat ion Issues	42



<b>REFERENCES</b>	<b>44</b>
<b>APPENDIX</b>	<b>45</b>
<b>A. SOURCE CODE</b>	<b>45</b>
<b>B. SCREENSHOTS</b>	<b>59</b>
<b>C. RESEARCH PAPER</b>	<b>61</b>

## LIST OF FIGURES

<b>Figure No.</b>	<b>FIGURE NAME</b>	<b>Page No.</b>
4.1	System Architecture	24
4.2	Project Management Plan	26
5.1	Fake Image Analysis	32
5.2	Tampered Image	32
5.3	Number of fake images	33
5.4	Remaining pristine images	33
5.5	Pristine Image	34
5.6	Number of real images	34
5.7	Real image to Ela image	35
5.8	Model building and training	36
5.9	Epoch v/s Loss graph	37
5.10	Final image Identification	38
B.1	User Interface	59
B.2	Fake Image identification	60
B.3	Real image identification	60

## LIST OF ABBREVIATIONS

SHORT FORM	FULL FORM
CNN	CONVOLUTIONAL NEURAL NETWORK
CMI	CAMERA MODEL IDENTIFICATION
CFA	COLOUR FILTER ARRAY
JPEG	JOINT PHOTOGRAPHIC EXPERTS
GROUP	
CMFD	COPY-MOVE FORGERY DETECTION
PCA	PRINCIPAL COMPONENT ANALYSIS
PRNU	PHOTO-RESPONSE NON-UNIFORMITY
RGB	RED GREEN BLUE
CSV	COMMA SEPARATED VALUE

# CHAPTER 1

## INTRODUCTION

In recent years, Images have become a pervasive part of our life with the common usage of smart acquisition devices such as cameras and smartphones, and the ease of sharing over the Internet. In parallel to this fact, the number of image-processing software increased. They have become accessible, such that anyone may easily modify and share online images.

In parallel, tampering with images has become significant to our society. Several techniques have emerged, among them: Splicing, Copy-Move, and Removal are the most commonly used manipulations. Some of these manipulations are difficult to detect for non-expert users. Moreover, some manipulated images aim at delivering misleading information which might be a threat to society (e.g., mass manipulation, cyber-criminality, tampering or removing of judicial evidence, etc.). Therefore, in the last decade, the forensic research community focused on developing tools that validate the integrity of an image.

Many approaches to detecting image authenticity and assessing the integrity of the images have been proposed. Some of them have focused their interest on forensic issues, and among them, few have dedicated themselves to camera model identification. Such vast usage of image tampering methods has led to the emergence of digital image forensics which is essential to prevent or detect frauds and solve copyright disputes by establishing the integrity and authenticity of digital images.

The forgery detection techniques are mainly categorized as active and passive methods. The former relies on some authentication information like a digital signature or a watermark, embedded within the image during creation or before sharing it publicly. On the contrary, the latter, passive detection techniques do not rely on in-built information, instead, they rely on the image features to identify the tampered ones.

These passive detection methods are more robust and have a wider range of applicability as most of the images on social media do not have embedded identity information. Conventionally passive image forgery detection methods have focussed on the detection of copy-move forgery, image splicing, and image retouching detection. Compared to other passive detection techniques, copy-move forgery is difficult to detect as a lot of characteristics of the forged region like colour, texture, and device properties are the same as the rest of the image.

Further, the use of compression, blurring, rotation, noise, etc., make the identification of copy-move rather more challenging. Images are exploited in a large range of use cases where determining their integrity and origin may have high consequences. For example, it is critical in criminal investigations or news coverage. Thus, confirming the image source and its authenticity is one of the most important tasks of the image forensic community. The extracted information from an image and the other multimedia content could show some inconsistencies which gives proof of a possible forgery for the image or the whole document.

State-of-the-art methodologies which have focused on identifying the camera model or the camera manufacturer, have mainly used the signature left on the image by the camera component workflow which includes physical and digital process traces. Indeed, specific operations are performed by each camera model while acquiring an image (e.g., various JPEG compression algorithms, proprietary Colour Filter Array (CFA) interpolation, sensor noise, and other noise statistics, and prediction residuals.

Extensive testing using a library of images will then be performed on the implemented algorithms to determine their success rate. Other general properties of the algorithm, such as its false positive rate and runtime will also be reported. Additionally, more specific tests on variants of the same algorithm will also be performed. For example, an algorithm may have parameters that can dramatically alter its performance and detection rate on certain classes of images, and so by testing these values were able to comprehensively determine the performance of an algorithm on a variety of different image types.

This ensures that more advanced algorithms are not unfairly discarded simply because their internal parameters needed tweaking. The results of this research will be of great use to improve the credibility of images used within the media.

Image forgery is an ever-increasing issue in modern society, and there have been instances where forged images have been used by mistake, or when images have been specifically doctored to be misleading. Despite the importance of the issue, there is still no widely recognized method to detect image forgeries, and certainly no industry standard. This represents an opportunity to provide an insight that will benefit one of the largest industries in the world, and potentially improve the reliability and credibility of the images presented by the media.

This also allows individuals the opportunity to determine the credibility of the images provided to them, either through official, credible sources or elsewhere, such as on an internet message board or shared by a friend on social media. Whilst the results of the project aim to be both comprehensive and clear, it's important to note the scope of the research involved.

The research will be undertaken on algorithms and methods that have been discussed in existing academic papers. Improvements and changes can and will be made to these methods, however, no new algorithms will be created or tested as part of this project. The aim is to implement pre-existing algorithms, improving and comparing them, not to research generating potentially new algorithms. This would require much more time than is feasible, impacting the testing stage, and ultimately reducing the effectiveness of the research. Determining new algorithms is therefore beyond the scope of this project.

## **CHAPTER 2**

### **LITERATURE SURVEY**

M. Samel, A. Mallikarjuna Reddy [1] 2022, focus on various approaches available in the current scenario to detect the forgery parts in the image.

Qazi, E.U.H.; Zia, T.Almorjan [2] 2022, proposed a model that takes image batches as input and utilizes the weights of a YOLO convolutional neural network (CNN) by using the architecture of ResNet50v2. CASIA\_v1 and CASIA\_v2 benchmark datasets, which contain two distinct categories, original and forgery, to detect image splicing.

Kalyani Dhananjay Kadam; Swati Ahirrao; Ketan Kotecha [3] 2022, Mask R-CNN with MobileNet, a lightweight model, to detect and identify copy move and image splicing forgeries.

Akram Hatem Saber, Mohd Ayyub Khan, Basim Galeb Mejbel [4] 2020 Many techniques have been proposed to detect image forgery in the literature such as digital watermarking, digital signature, copy-move, image retouching, and splicing. The investigation done in this paper may help the researcher to understand the advantage and handles of the available image forensic technology to develop more efficient algorithms of image forgery detection.

M.H. Al Banna, M.A. Haider, M.J. Al Nahian, M.M. Islam, K.A. Taher, M.S. Kaiser [5] 2019, proposed the deep Convolutional Neural Network and transfer learning approach for extracting features from an image's dataset. Three state-of-the-art machine learning algorithms such as SVM, logistic regression and random forest-based classifiers have been used for evaluating identification accuracy.

A. Kuzin, A. Fattakhov, I. Kibardin, V.I. Iglovikov, R. Dautov [6] 2018, described a Deep Learning approach to the camera detection task of 10 cameras as a part of the Camera Model Identification Challenge hosted by Kaggle.com where their team finished 2nd out of 582 teams with the accuracy on the unseen data of 98%. Augmentations that allowed a stay robust against transformations. Several experiments were carried out on datasets collected by organizers and scraped from the web.

Z. Qin, F. Yu, C. Liu, X. [7] 2018, Chen provides a comprehensive survey of several representative CNN visualization methods, including Activation Maximization, Network Inversion, Deconvolutional Neural Networks (DeconvNet), and Network Dissection-based visualization. These methods are presented in terms of motivations, algorithms, and experiment results.

C. Chen, X. Zhao, and M.C. Stamm [8] 2017, propose a new method to detect if an image's source camera model has been anti-forensically falsified. The algorithm operates by characterizing the different content-independent local pixel relationships that are introduced by both authentic demosaicing algorithms and anti-forensic attacks. Experimental results show that the algorithm can detect an anti-forensic attack with over 99% accuracy, is robust to JPEG compression, and can even identify the true source camera model in certain circumstances. Sensor photo-response non-uniformity was introduced by Lukas et al to solve the problem of digital camera sensor identification. The PRNU is the main component of a camera fingerprint that can reliably identify a specific camera. This fingerprint can be estimated from multiple images taken by the camera.



## 2.1 INFERENCES FROM LITERATURE SURVEY

- For comparison purposes, we have decided to include recent and well-known deep learning models such as ResNet-152, Dense-Net, and VGG19. These recent networks make it possible to train up to hundreds or even thousands of layers and still have allowed boosting the performance of many computer vision applications and classification.
- The image forgery can be understood by extracting the camera model using CMI and the signature left on the image by the camera component workflow which includes physical and digital process traces. These data can be used to train the CNN to detect image manipulations.
- These proposed approaches include any kind of common manipulations frequently used when sharing images on the internet. They were interested in compression, which is one of the most common manipulations of images.
- The first version of those models has been published between 2014 and 2017. However, those models are updated every year with more layers and new Pre-trained parameters. Note that, for this experiment, we only use the original dataset without a compression factor.
- First results on different compression quality factors have shown that training with compressed and uncompressed images is of great importance to get better performances than recent literature approaches.
- Results have proven the performance of the latter strategy which is a compromise between a fully trained model on a specific dataset and a mixture of different quality factor datasets.

## 2.2 OPEN PROBLEMS IN THE EXISTING SYSTEM

- Manipulations are difficult to detect for non-expert users. Moreover, some manipulated images aim at delivering misleading information which might be a threat to society (e.g., mass manipulation, cyber-criminality, tampering or removing of judicial evidence, etc.).
- A lot of approaches have been proposed to check image integrity and authenticity but they have their problems.
- Determining the integrity and origin of the images is important for some scenarios. For example, it is critical in criminal investigations or news coverage.
- The principal advantage of those CNN (Convolutional Neural Networks) based approaches is they are capable of learning classification features directly from image data, however, interpreting the extracted camera identification model is very difficult.
- Another drawback of using CNN is the model is dedicated to a specific training dataset. However, image modifications like geometric distortions (for example. resizing, compression, filtering) or content manipulations are unpredictable. Moreover, new Image forgeries appear every day.
- Compared to other passive detection techniques, Copy-Move forgery is difficult to detect as a lot of characteristics of the forged region like color, texture, and device properties are the same as the rest of the image.
- Further use of Compression, Blurring, Rotation, Noise, etc., make the identification of Copy-Move rather more challenging

## 2.3 OVERALL OBJECTIVE

This Project is to identify Illegal images using convolutional networks. Some common image forgery methods are Splicing, Copy-Move and Removal. We can extract signatures and proof of images using Camera Model Identification (CMI). It is an essential branch of digital image forensics, which is significant in preventing image crimes. CMI is based on the image information (including image content and invisible traces) to associate it with the shooting devices. For this, we will train the model using CNN Algorithm which extracts camera model details. The Convolutional layer is the cornerstone of a CNN, as it performs most of the heavy computational operations.

The overall objective of image forgery detection using Convolutional Neural Networks (CNN) is to develop an accurate and reliable automated system that can detect the presence of forged or manipulated images with high precision and recall. Image forgery detection involves identifying the parts of an image that have been tampered with or manipulated, such as through copy-pasting, image splicing, or digital manipulation techniques. CNNs are particularly well-suited to this task as they can learn and extract complex features from images that can be used to distinguish between authentic and manipulated images.

The primary goal of image forgery detection using CNNs is to improve the accuracy and efficiency of image forensic analysis, which can be used in a variety of applications, including digital forensics, law enforcement, and content authentication. Additionally, such systems can help to combat the spread of fake news and misinformation, by enabling the identification of manipulated images before they are shared widely on social media and other platforms. Using Image transformation for quality of training and patch extraction to reduce the size and increase efficiency. Further, the use of compression, blurring, rotation, noise, etc., make the identification of copy-move rather more challenging.

## **CHAPTER 3**

### **REQUIREMENT ANALYSIS**

#### **3.1 FEASIBILITY STUDIES/RISK ANALYSIS OF THE PROJECT**

The project plan and objectives are well-defined, and the implementation has been successful up to the application level without encountering any issues. Moreover, the project risk is negligible, implying that the project can be carried out in the next phase with a wide scope. Currently, four classification algorithms have been utilized in the project, and their accuracy and performance have been used to predict the client-side data. There is also a possibility of improving the algorithms in the future. The built applications have low risk, and the project is feasible, meeting the deadline for submission. The feasibility study's objective is to determine if the application is technically, economically, operationally, and market feasible to ensure its longevity and generate a good return on investment. The findings from the feasibility study can be utilized to inform future decisions.

##### ***3.1.1 Feasibility Studies***

Feasibility study for image forgery detection using CNN would help to determine whether the proposed project is viable and can be successfully implemented. It would help to identify any potential challenges and risks and provide insights into how these challenges can be addressed. A thorough feasibility study is essential to ensure the success of the project and to maximize the potential benefits and returns on investment.

- **Technical feasibility:** This involves evaluating whether the proposed project is technically feasible and whether the necessary technical expertise and resources are available. In the case of image forgery detection using CNN, this would include assessing the availability of appropriate datasets, the required computing power and software, and the availability of trained personnel with the necessary skills to develop and implement the CNN model.

- **Economic feasibility:** This involves assessing the financial viability of the project. This would include analyzing the costs involved in developing and implementing the CNN model, such as hardware and software costs, personnel costs, and any other expenses that might be incurred. It would also involve analyzing the potential benefits and returns on investment, such as revenue generated from the sale of the forgery detection software.
- **Operational feasibility:** This involves assessing whether the proposed project can be successfully integrated into existing operational systems and processes. In the case of image forgery detection using CNN, this would include assessing whether the proposed software can be integrated with existing image processing systems and whether the necessary protocols can be established to ensure smooth operation.
- **Market feasibility:** This involves assessing the demand for the proposed product or service in the market. In the case of image forgery detection using CNN, this would involve evaluating the potential market for the forgery detection software, such as law enforcement agencies, forensic experts, and other organizations that require image analysis services.

### **3.1.2 Risk Analysis**

Risk analysis for image forgery detection using CNN involves identifying potential risks that could impact the success of the project and developing strategies to mitigate these risks. Some of the potential risks that may arise during image forgery detection using CNN are as follows:

- **Data quality risk:** The quality of the data used to train the CNN model may not be sufficient, leading to inaccurate predictions. To mitigate this risk, it is essential to ensure that the data used is high-quality and relevant to the problem at hand.

- **Overfitting risk:** Overfitting can occur when the CNN model is trained on a small dataset or when it is trained for too long, resulting in poor generalization and inaccurate predictions. To mitigate this risk, it is essential to use a large dataset and to employ techniques such as early stopping to prevent overfitting.
- **Hardware failure risk:** Hardware failure, such as server or storage system failure, can result in data loss and system downtime. To mitigate this risk, it is essential to have backup systems in place and to regularly test and maintain the hardware.
- **Cybersecurity risk:** Cybersecurity risks such as hacking or data breaches can compromise the integrity of the data used to train the CNN model or the predictions made by the model. To mitigate this risk, it is essential to implement robust cybersecurity measures such as firewalls, encryption, and access controls.
- **False positive or false negative risk:** The CNN model may produce false positives or false negatives, resulting in inaccurate predictions. To mitigate this risk, it is essential to regularly test and refine the CNN model and to use a combination of different algorithms and techniques to improve its accuracy.
- **Model performance risk:** The performance of the CNN model may degrade over time due to changes in the input data or changes in the operating environment. To mitigate this risk, it is essential to regularly monitor and evaluate the performance of the CNN model and to make necessary adjustments to ensure its continued accuracy.

## **3.2 REQUIREMENT PROCESS OF THE PROJECT**

### **3.2.1 Data Pre-Processing**

Data pre-processing is done to train the model on a variety of data. The quality and quantity of the training dataset are crucial steps to achieve high accuracy when using a deep learning model in the framework. The research objective is to apply an image

forgery detection approach to online shared images. Thus, the training set has to go through transformations that should be similar to the common ones used on the Internet, e.g., filtering, resizing, or compression. Consequently, all images of the original dataset are duplicated and transform themselves to form other datasets.

The following are the steps involved in data pre-processing

- **Data Collection:** The first step in any data science project is to collect data from various sources.
- **Data Cleaning:** After data collection, the data needs to be cleaned. This involves removing or fixing any missing, duplicate or irrelevant data.
- **Data Integration:** Data integration is the process of combining data from multiple sources to create a unified dataset.
- **Data Transformation:** Data transformation involves converting the data into a suitable format for analysis. This may include scaling or normalizing numerical data, encoding categorical data, or feature engineering.
- **Data Reduction:** Data reduction involves reducing the size of the dataset by removing irrelevant or redundant features.
- **Data Splitting:** After the data has been pre-processed, it is split into training and testing sets. This is done to evaluate the performance of the machine learning model on new, unseen data.
- **Data Sampling:** Data sampling is the process of selecting a subset of the dataset for training the machine learning model. This is done to speed up the training process and prevent overfitting.

### **3.2.2 CNN**

The Convolutional layer is the cornerstone of a CNN, as it performs most of the heavy computational operations. It consists of a set of filters where each filter is spatially defined with a width and a height and is applied with an overlapping distance called

'stride' to all the local regions of the given input. Input can be an image of dimension  $w \times l \times c$  where  $w$  and  $l$  denote respectively the width and the height of the image and  $c$  represents its RGB channel values. The input image is convolved with a defined number of kernels of dimension  $s \times s \times c$  where  $s$  denotes the filter size.

CNNs are inspired by the way the visual cortex of the human brain processes visual information. They consist of a sequence of layers that process the input data and output the desired output, such as a classification or segmentation map. The most common layers used in a CNN are convolutional layers, pooling layers, and fully connected layers. Convolutional layers apply a set of filters to the input data to extract features that are relevant for the task at hand. Each filter is a small matrix of weights that is applied to a small region of the input data, and the resulting output is called a feature map. During training, the CNN learns the optimal set of filters that maximize the performance on the task at hand.

Pooling layers down sample the feature maps to reduce the size of the input data and make the model more efficient. The most common pooling operation is max pooling, which selects the maximum value in each region of the feature map.

Fully connected layers take the flattened output from the convolutional and pooling layers and output the final classification scores. They are similar to the layers in a traditional feedforward neural network, where each neuron is connected to all the neurons in the previous layer.

CNNs are trained using backpropagation and gradient descent, similar to other neural networks. The objective is to minimize a loss function that measures how well the model is performing on the training data. The most common optimizer used in a CNN is stochastic gradient descent (SGD), which updates the model's weights based on the gradient of the loss function with respect to the weights. CNNs have been successfully applied to a wide range of computer vision tasks, such as image classification, object detection, segmentation, style transfer, and generation.



### **3.2.3 Camera Model Identification**

Camera Model Identification (CMI) is an essential branch of digital image forensics, which is significant in preventing image crimes. CMI is based on the image information (including image content and invisible traces) to associate it with the shooting devices.

Training a CNN involves several steps:

1. **Data preparation:** The first step is to prepare the data that will be used to train the model. This typically involves loading images and labels (if available) into memory, and then splitting the data into a training set and a validation set.
2. **Model definition:** Next, the architecture of the CNN must be defined. This typically involves specifying the number and size of convolutional layers, the number and size of fully connected layers, and the activation functions to be used.
3. **Model compilation:** Once the model is defined, it must be compiled by specifying the optimizer, loss function, and any other relevant metrics that will be used to evaluate the model during training.
4. **Model training:** The model is trained by iterating over the training data and updating the model's parameters to minimize the loss function. The process of training the model involves forwarding the input through the model, computing the loss and gradients, and updating the model's parameters using the optimizer.

5. **Model evaluation:** After the model is trained, it is typically evaluated on a validation set to assess its performance. This can be done by computing metrics such as accuracy, precision, recall, and F1 score.
6. **Fine-tuning:** To improve the performance of the model, it may be necessary to fine-tune the model by adjusting the hyperparameters, such as the learning rate or the number of layers. This process may involve repeating steps 3-5 multiple times.
7. **Model deployment:** Finally, the trained model can be deployed in a production environment to make predictions on new data.

### **3.2.4 CNN Layers**

1. The convolutional layer (Conv1) has a kernel size of  $4 \times 4 \times 3$  with 32 feature maps as output. Input is a set of patches of size  $64 \times 64 \times 3$ . The convolutional filter is applied with a stride of one.
2. The convolutional layer (Conv2) contains 48 filters of size  $5 \times 5 \times 32$  (stride=1). It generates, as output,  $28 \times 28 \times 48$  feature maps.
3. The convolutional layer (Conv3) contains 64 filters of size  $5 \times 5 \times 48$  (stride=1). It generates, as output,  $10 \times 10 \times 64$  feature maps.
4. The convolutional layer (Conv4) contains 128 filters of size  $5 \times 5 \times 64$  (stride=1). It generates, as an output, a vector of 128 feature maps.

### **3.2.5 Max-Pooling**

It is common to insert a pooling layer in-between successive convolutional layers in a deep learning architecture. The main objective of the pooling layer is to reduce the feature maps created by the convolutional layers. It progressively reduces the number of parameters and the computational cost of training the network. MaxPool2D layer is added to the model with the specified pool size and stride. The MaxPool2D layer is used for down sampling the output of the convolutional layers, by taking the maximum value of a small subregion of the input. The pool size argument specifies the size of the pooling window, while the strides argument specifies the stride of the pooling

operation. In this case, the pooling window is 2x2 and the stride is 2, which means that the output of the pooling layer will have half the spatial dimensions (i.e., the width and height will be halved).

### **3.2.6 *Tensor Flow***

TensorFlow is an open-source software library for dataflow and differentiable programming across a range of tasks. It is developed by the Google Brain team and was released under the Apache License 2.0 in 2015. TensorFlow allows developers to create and deploy machine learning models easily, efficiently, and at scale. It provides a flexible and powerful platform for building and training deep learning models, and it supports a wide variety of neural network architectures. The TensorFlow platform helps you implement best practices for data automation, model tracking, performance monitoring, and model retraining. Using production-level tools to automate and track model training over the lifetime of a product, service, or business process is critical to success. Its flexible architecture allows for the easy deployment of computation across a variety of platforms (CPUs, GPUs, TPUs), and from desktops to clusters of servers to mobile and edge devices. TensorFlow computations are expressed as stateful dataflow graphs.

### **3.2.7 *Python Packages***

- TensorFlow

TensorFlow is an open-source library developed by Google for machine learning and deep learning applications. It is widely used for building and training deep learning models, including CNNs. TensorFlow provides a flexible and efficient platform for implementing image processing tasks, such as image classification, segmentation, and detection.

- Keras

Keras is a high-level neural networks API that runs on top of TensorFlow. It provides a user-friendly interface for building and training deep learning models, including CNNs. Keras is known for its simplicity, ease of use, and ability to abstract away many of the low-level details of deep learning.

- OpenCV

OpenCV (Open-Source Computer Vision) is a computer vision library that provides tools for image and video processing, including image pre-processing and feature extraction. It is widely used for object recognition, face detection, and motion tracking, among other applications.

- Scikit-learn

Scikit-learn is a machine learning library that provides tools for data pre-processing, model selection, and performance evaluation. It is widely used for developing classification and regression models, including those based on CNNs.

- NumPy

NumPy is a package for numerical computing in Python, often used for handling and manipulating large arrays of data. It provides efficient tools for matrix and array operations, which are often used in deep learning applications.

- Pandas

Pandas is a package for data analysis in Python, often used for data cleaning, exploration, and manipulation. It provides a powerful set of data structures and functions for working with structured data, which are often used in deep learning applications.

- **Matplotlib**

Matplotlib is a plotting library for creating visualizations in Python, often used for displaying images and data visualizations. It provides a wide range of customizable plots, including scatter plots, bar charts, and histograms, which are often used in image processing tasks.

These packages, along with others that may be specific to the project requirements, can be used to develop and implement an Illegal Image Identification using CNN.

### **3.3 SOFTWARE REQUIREMENTS SPECIFICATION DOCUMENT**

To ensure that a software system or application meets the expectations of users and stakeholders and operates as intended in its designated environment, it is vital to understand and document all types of requirements. Typically, a project involves three categories of requirements: functional requirements, non-functional requirements, and environmental requirements. By accurately capturing and addressing these requirements, the resulting software system or application can meet the needs of its users and stakeholders while functioning optimally in its intended environment.

#### ***3.3.1 Purpose of this document***

The purpose of this project is to develop an image forgery detection system using CNN that can accurately detect if an image has been tampered with or manipulated. The system aims to provide a reliable tool for detecting image forgeries, which can be used in various domains, including law enforcement, digital forensics, and media authentication. The system can also be used to improve the overall security of image-based systems, such as social media platforms and online marketplaces, by detecting and preventing the spread of fake or fraudulent images. The ultimate goal of this project is to contribute to the development of robust and effective techniques for image forgery detection, thereby enhancing the reliability and trustworthiness of digital media.

#### ***3.3.2 Scope of this document***

The scope of this project is to develop an image forgery detection system using CNN. The system aims to accurately detect if an image has been tampered with or manipulated, and provide a reliable tool for detecting image forgeries in various domains, including law enforcement, digital forensics, and media authentication. The project includes the development of the CNN algorithm and the training of the model using the CASIA 2.0 dataset. The project also involves the testing and evaluation of the system's accuracy and performance, as well as the development of a user-friendly interface for the system. The project does not include the development of additional features beyond image forgery detection, or the integration of the system with other software applications.

### ***3.3.3 Functional Requirements***

The software requirements specification is a technical specification of requirements for the software product. It is the first step in the requirements analysis process. It lists the requirements of a particular software system.

### ***3.3.4 Non-Functional Requirements***

1. Problem definition
2. Preparing datasets
3. Evaluating algorithms
4. Prediction of the result
5. Improving results
6. Integration of model
7. Deployment in website

### ***3.3.5 Hardware Requirements***

- Processor: Intel Core i5-11400F up to 4.5 GHz.
- Memory: 8 GB DDR4.

- Hard Drives: 512 GB NVMe2 SSD.
- GPU: NVIDIA GeForce GTX 1650 4 GB.
- Computing Power: 7.5
- Ports: 1x HDMI 2.0, 1x USB 3.1 Type-C, 2x USB 3.1, 1x USB 2.0.
- Connectivity: Wi-Fi 802.11ax, Gigabit LAN (Ethernet), Bluetooth.

### ***3.3.6 Software Requirements***

- OS: Windows 10/11 Home.
- Python 3.9 or Later.
- TensorFlow-CPU.
- Streamlit
- Pandas
- NumPy
- open-cv

## **CHAPTER 4**

### **DESCRIPTION OF THE PROPOSED SYSTEM**

The proposed system is an image forgery detection system that uses a deep learning approach based on Convolutional Neural Networks (CNNs). The system is designed to identify whether an image has been manipulated or not. The system takes an input image and produces an output that indicates whether the image is authentic or forged.

#### **4.1 SELECTED METHODOLOGY OR PROCESS MODEL**

The proposed methods deal with the system for image forgery detection using Convolutional Neural Networks and Camera Model Identification. As this model is created using the inference of CNN which mainly requires for image classification.

Convolutional Neural Networks (CNNs) are a type of deep learning model that are commonly used for image classification and object detection tasks. They can also be used for image forgery detection by training the model on a dataset of both original and tampered images.

The basic building block of a CNN is the convolutional layer, which consists of a set of filters that scan the input image and extract features such as edges, textures, and patterns. The output of the convolutional layer is then passed through a non-linear activation function. After one or more convolutional layers, a CNN typically includes one or more pooling layers, which are used to reduce the spatial dimension of the input image.

One of the common approaches for training a CNN for image forgery detection is to use a dataset of images that have been tampered with using various methods, such as splicing, copy-move, and color manipulation. The CNN is trained to learn the differences between the tampered images and the original images, and to classify new images as either real or forged based on these differences.



The basic building block of a CNN is the convolutional layer, which consists of a set of filters that scan the input image and extract features such as edges, textures, and patterns. The output of the convolutional layer is then passed through a non-linear activation function. After one or more convolutional layers, a CNN typically includes one or more pooling layers, which are used to reduce the spatial dimension. The qualitative and statistical visualization analysis to improve the low interpretability of the CNN network. The analysis allows for visually seen patterns by tools such as the principal component analysis (PCA) approach. Its principal objective is to provide a robust and effective framework for camera model identification (CMI) and image forgery detection. This framework contains four specific parts. The first part concerns all the image pre-processing steps. In this part, we also show the importance of taking into account the quality of the input data to strengthen the robustness. Then, we explain the classification approach using Convolutional Neural Networks to identify camera models CMI. The next part highlights an in-depth analysis of our CNN which allows us to better understand and improve our framework. Finally, we test our proposed framework on a forgery detection application.

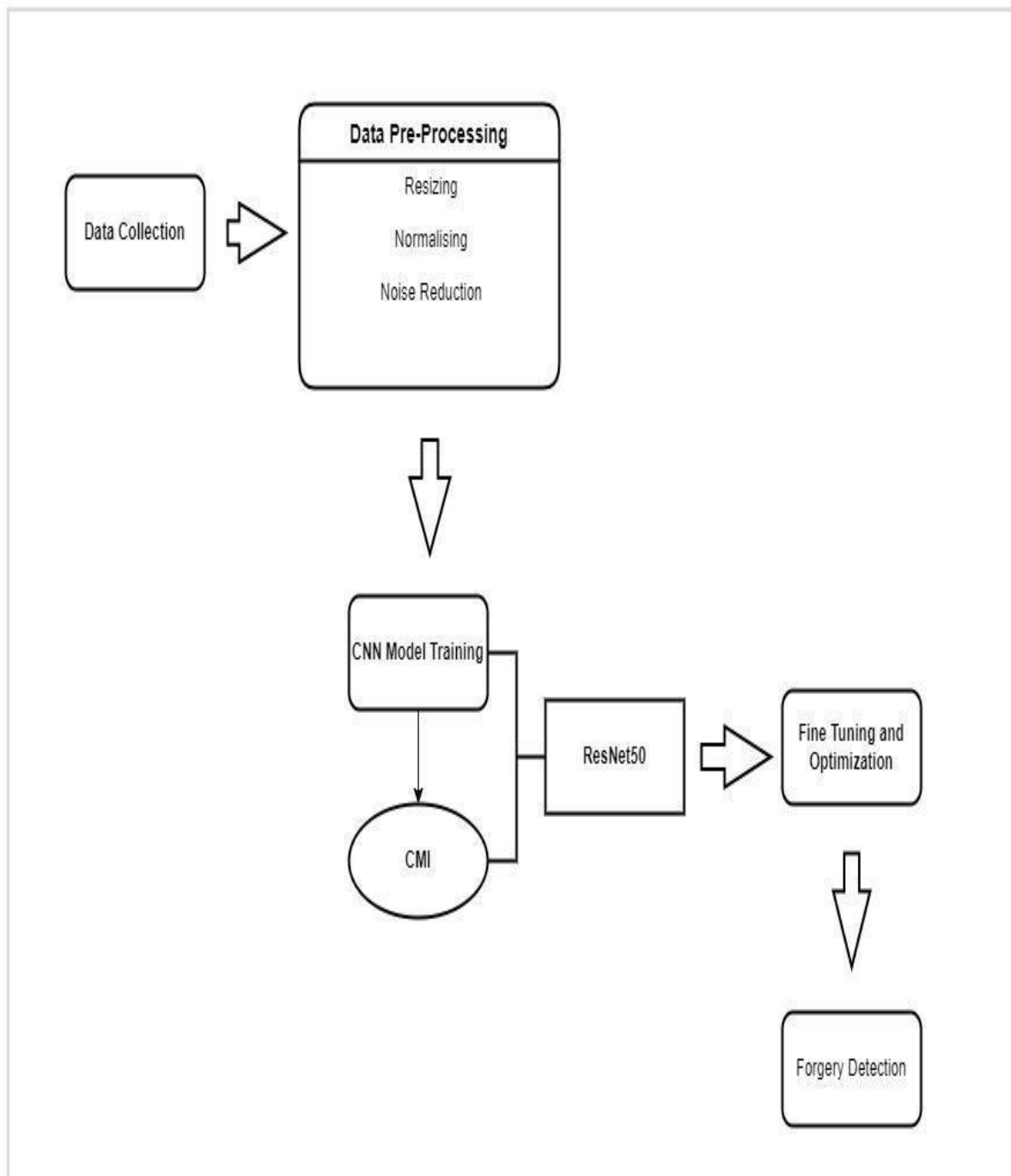
The CASIA v2 dataset, which is available to the public, is commonly utilized for image forgery detection research. It comprises of 10,000 images, 5,000 of which are genuine, and 5,000 are altered with techniques such as splicing, copy-move, and removal. The ground truth labels reveal the kind of manipulation that was utilized, which is useful for testing and training image forgery detection algorithms. The dataset is partitioned into a training set and a testing set, with a total of 8,000 and 2,000 images, respectively, in both grayscale and color formats, ranging in resolution from 256x256 to 512x512 pixels. This dataset has been widely employed to examine the effectiveness of machine learning and deep learning-based image forgery detection algorithms, making it a significant resource for researchers in this area.

The resized images are then standardized on the fly before giving it as input to the proposed CNN-based architecture. The proposed architecture is dual branch CNN-based architecture, where both branches are connected to a common input. There are three convolution layers in each branch, with 16, 32, and 64 feature maps for the first, second, and third layers, respectively.

All the convolutional layers use Relu activation and each convolutional layer is followed by a  $2 \times 2$  max-pooling layer. To extract multi-scale features from the images, CNN layers in these two branches have different kernel sizes. Since experiments were conducted by varying the kernel sizes, hence in some cases the addition of one zero-padding layer has been done to ensure a symmetric output. The output of the third convolution layer from both branches is passed through a concatenation layer.

This generates a stack of multi-scale feature maps extracted from a common input. The concatenated output of this layer is fed to a global max-pooling layer, which retains only the maximum feature per feature map. This layer acts as a flattening layer in the architecture and converts the two-dimensional input to a one-dimensional output. This 128-length one-dimensional vector is passed into the second last layer which is a dense layer with 32 units. This 32-length vector is fed to the last dense layer with a single unit only. Sigmoid activation has been used in both of the dense layers. The last layer generates the class probability 'p' that denotes the image being authentic. Hence, '1-p' will be the probability of the image being forged. A decision threshold of 0.5 is used to classify between an original and a forged image. An output probability of greater than 0.5 indicates an original image and otherwise a forged image. In binary labels, '1' denotes an original image and '0' denotes a forged image.

## 4.2 ARCHITECTURE OF THE PROPOSED SYSTEM



**Fig 4.1: System Architecture**

### **4.3 DESCRIPTION OF SOFTWARE FOR IMPLEMENTATION AND TESTING PLAN OF THE PROPOSED MODEL/SYSTEM**

The proposed model for image forgery detection can be implemented using various programming languages and libraries such as Python, OpenCV, TensorFlow, and Keras. The software should have the following components:

- Pre-processing module: This module will perform the pre-processing of the input image, including resizing, normalization, and filtering.
- Feature extraction module: This module will extract relevant features from the pre-processed image, such as edge detection, texture analysis, and color histograms.
- Classification module: This module will use machine learning algorithms such as SVM, Random Forests, or Neural Networks to classify the image as either authentic or forged.
- Localization module: This module will identify the region of the image where the forgery has occurred.
- Post-processing module: This module will perform any necessary post-processing such as filtering or smoothing to improve the results.

#### ***4.3.1 Testing Plan of the proposed system***

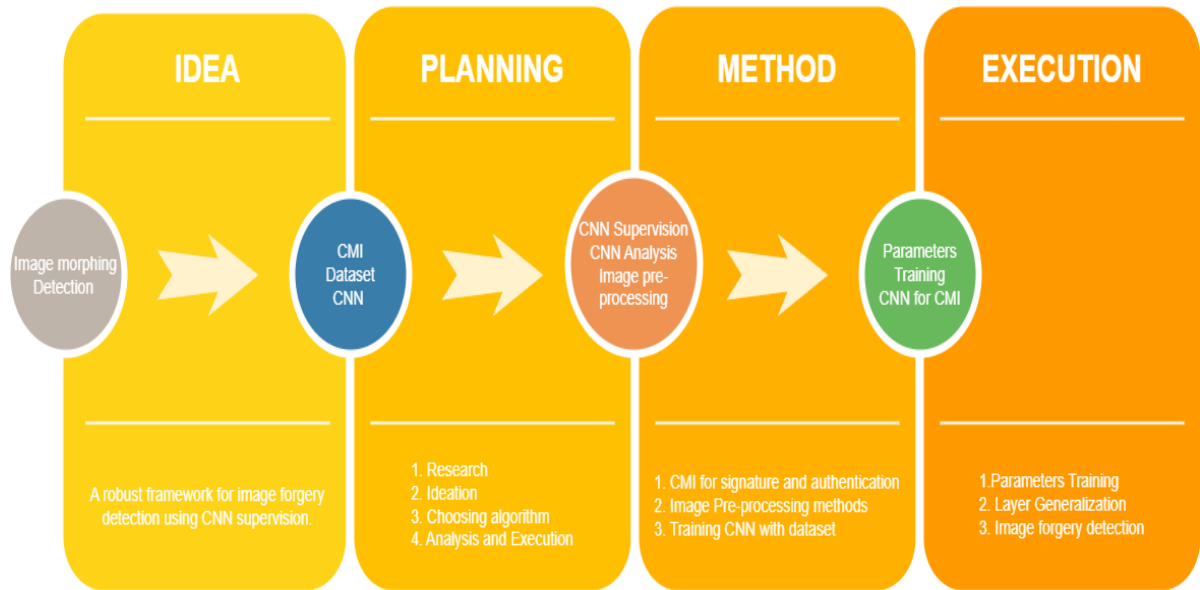
The testing plan for the proposed model should include the following steps:

- Data collection: Collect a diverse set of image data that includes various types of forgeries.
- Data pre-processing: Pre-process the collected image data to ensure that it is in

the correct format for the model.

- Training: Train the model using the pre-processed image data and the selected machine learning algorithm.
- Validation: Validate the trained model using a separate set of image data that was not used during the training phase.
- Testing: Test the performance of the model on a separate set of image data that was not used during either the training or validation phase.
- Performance evaluation: Evaluate the performance of the model using metrics such as accuracy, precision, recall, and F1 score.
- Optimization: Optimize the model by tweaking its hyperparameters and architecture to improve its performance.
- Deployment: Deploy the model for use in real-world applications, and continuously monitor and update it as necessary.

#### **4.4 PROJECT MANAGEMENT PLAN**



**Fig 4.2: Project management plan**

#### 4.5 FINANCIAL REPORT ON ESTIMATED COSTING

- **Hardware Costs:** The hardware required for developing and training CNN models can be one of the biggest expenses. This can include high-end GPUs or specialized hardware such as TPUs, which can be expensive to purchase and maintain.
- **Software Costs:** There are several software tools and frameworks available for developing and training CNN models, including TensorFlow, PyTorch, and Keras. These software tools are generally free to use, but may require additional paid plugins or services for optimal performance.
- **Data Collection and Preparation Costs:** Collecting and preparing large datasets of authentic and manipulated images can be time-consuming and costly. This may involve hiring data annotators or purchasing datasets from third-party providers.
- **Model Development Costs:** Developing CNN models for image forgery detection requires a team of skilled data scientists and machine learning engineers, which can be expensive. The cost may vary depending on the complexity and size of the

models developed.

- **Maintenance and Upgrades Costs:** After developing the system, it is essential to maintain and upgrade it to keep up with the latest technology trends and developments. This may involve additional costs for hiring dedicated staff or outsourcing maintenance services.

The total estimated cost for Illegal Image Identification using CNN supervision can vary widely based on several factors, including the size and complexity of the dataset, the expertise and experience of the team involved, and the hardware and software requirements. However, it is expected that developing such a system could be a significant investment for businesses and organizations, but the long-term benefits of preventing image forgery and ensuring authenticity can outweigh the costs.

## **CHAPTER 5**

### **IMPLEMENTATION DETAILS**

#### **5.1 DEVELOPMENT DEPLOYMENT AND SETUP**

Generally, several phases are involved in the development stage such as research, data collection and preparation, model development and training, evaluation and validation, and deployment.

During the research phase, a thorough investigation of existing literature and techniques related to image forgery detection using CNNs will be conducted by the project team. This will involve reviewing academic papers, online resources, and other relevant sources to gain a deep understanding of the topic and identify potential approaches.

In the data collection and preparation phase, the dataset that will be used to train and evaluate the model will be acquired and cleaned by the team. This may involve sourcing and labeling images that are both real and manipulated and then pre-processing them to ensure they are suitable for training the model.

In the model development and training phase, the architecture of the CNN model will be designed by the team and trained using the prepared dataset. This may involve experimenting with different model architectures and hyperparameters to achieve optimal performance. In the evaluation and validation phase, the model will be tested on a separate dataset to evaluate its performance and ensure that it is not overfitting. This may involve measuring metrics such as accuracy, precision, recall, and F1 score.

Finally, in the deployment phase, the trained model will be integrated into a larger system or application for practical use by the team. The development stage for a project on Image forgery detection using CNN can take several months or longer, depending on the resources available, the complexity of the project, and the expertise of the team involved.



### **5.1.1 Dataset Description**

CASIA 2.0 is a benchmark dataset for image forgery detection, which was created by the Chinese Academy of Sciences in 2012. The dataset is widely used in research studies related to image forensics, and it has become a standard benchmark for evaluating image forgery detection methods. The CASIA 2.0 dataset contains 10,800 high-resolution color images with a size of 600x450 pixels. The images have been artificially manipulated to create different types of image forgeries. The dataset contains four different types of image forgeries, including copy-move forgery, splicing forgery, removal forgery, and re-sampling forgery.

**Copy-Move Forgery:** In this type of forgery, a portion of the image is copied and pasted into another part of the same image. This is the most common type of image forgery, and it is often used to hide or add objects to an image.

**Splicing Forgery:** In this type of forgery, two or more images are combined to create a single image. This type of forgery is often used to create fake images or to hide important information in an image.

**Removal Forgery:** In this type of forgery, an object or a region is removed from the image, leaving a hole that is filled in with background pixels. This type of forgery is often used to hide important information or to remove unwanted objects from an image.

**Re-sampling Forgery:** In this type of forgery, the image is re-sampled, resulting in artifacts that are visible in the image. This type of forgery is often used to alter the size or resolution of an image, which can be used to hide important information or to create fake images.

The CASIA 2.0 dataset is organized into three subsets, including a training set, a validation set, and a testing set. The training set contains 8000 images with different types of forgeries, while the validation set contains 1000 images with different types of forgeries. The testing set contains 1800 images with different types of forgeries, and

it is used to evaluate the performance of different image forgery detection methods. Overall, the CASIA 2.0 dataset is a comprehensive benchmark dataset for image forgery detection, and it has become a standard benchmark for evaluating the performance of different image forgery detection methods.

## 5.2 ALGORITHM

- Download the required files from Google Drive using their IDs.
- Import the required libraries.
- Define the path for fake and real images.
- Create an empty dictionary to store the image paths, labels and ids for fake images.
- Iterate over the images in the fake image path.
- If the image is in the jpg or png format, append the image path, label and id to the respective lists in the dictionary.
- Create a pandas Data Frame from the fake image dictionary and display the head.
- Define the path for a random real image and display it.
- Print the number of fake images in the dataset.
- Create an empty dictionary to store the image paths, labels and ids for real images.
- Iterate over the images in the real image path.
- If the image is in the jpg or png format, append the image path, label and id to the respective lists in the dictionary.
- Create a pandas Data Frame from the real image dictionary and display the head.
- Define the path for a random real image and display it.
- Shuffle the real image Data Frame and select the number of rows equal to the number of rows in the fake image Data Frame.
- Concatenate the fake and real image Data Frames to create the final dataset.
- Display the total number of images in the final dataset.
- Define a function to convert an image to an Error Level Analysis (ELA) image.

- Define the size of the ELA images to be (128,128).
- Define a function to prepare an image by converting it to an ELA image and resizing it to the desired size.
- Create empty lists X and Y to store the prepared images and their corresponding labels.
- Iterate over the rows in the final image Data Frame.
- Prepare the ELA image from the image path and append it to X. Append the label (0 for fake and 1 for real) to Y.
- Shuffle X and Y in the same order and convert X to a numpy array of shape (-1,128,128,3).
- Split X and Y into training and validation sets.
- Define a custom Dataset class to load the data from X and Y.
- Define a custom Data Generator class to generate batches of data from the Dataset.
- Define the number of epochs to train for and create an instance of the
- Data Generator for the training and validation sets.
- Define the CNN architecture using Keras.
- Compile the model with the Adam optimizer, binary cross-entropy loss and accuracy metric.
- Train the model using the fit\_generator method of the model instance and the training and validation Data Generators.

### 5.3 TESTING

- Unit testing

Test the unzipping functionality of the code by using sample zip files. Test the prepare\_image function by passing in different images and checking if the output is as expected. Check if the shuffle functionality is working correctly by passing a sample data frame.

- Integration testing

Check if the concat functionality of pandas is working correctly by verifying the number of images in the final dataset. Verify that the under sampling is working correctly. Test if the train-test split functionality is working properly.

- System testing

Run the model with different hyperparameters and evaluate its performance using appropriate metrics. Test the robustness of the model by passing images with different resolutions, sizes, and orientations.

## 5.4 PROJECT IMPLEMENTATION

First, we need to process Data analysis and also Data Pre-processing where it is done to train the model on a variety of data.

Analysis on Fake images Is done in order to get the number of fake/tampered images containing in that CASIA 2.0 dataset.

```
fake_image_data=pd.DataFrame(fake_image_data)
fake_image_data.head()
```

	image_path	label	image_id
0	./Tp/Tp_S_CRN_S_N_art00005_art00005_11793.jpg	fake	Tp_S_CRN_S_N_art00005_art00005_11793
1	./Tp/Tp_S_NNN_S_N_nat00026_nat00026_11035.jpg	fake	Tp_S_NNN_S_N_nat00026_nat00026_11035
2	./Tp/Tp_S_CNN_S_N_arc00084_arc00084_10733.jpg	fake	Tp_S_CNN_S_N_arc00084_arc00084_10733
3	./Tp/Tp_S_NRN_M_N_sec00035_sec00035_11250.jpg	fake	Tp_S_NRN_M_N_sec00035_sec00035_11250
4	./Tp/Tp_D_NRN_M_N_nat10123_nat00097_11338.jpg	fake	Tp_D_NRN_M_N_nat10123_nat00097_11338

**Fig 5.1: Fake Image Analysis**

From the above analysis, we can get the tampered image

```
real_image_path = '/content/Tp/Tp_D_CRN_M_N_nat00008_art00025_11425.jpg'  
Image.open(real_image_path)
```



***Fig 5.2: Tampered image***

This analysis will also provide the number of fake/tampered images in the provided dataset.

```
print("Number of fake images are {}".format(fake_image_data.shape[0]))
```

```
Number of fake images are 2064
```

***Fig 5.3: Number of fake images***

Since, we got fake/tampered images from the analysis now we can get the remaining number of real/pristine images left in the dataset.

```
real_image_data=pd.DataFrame(real_image_data)
real_image_data.head()
```

	image_path	label	image_id
0	./Au/Au_art_30444.jpg	real	Au_art_30444
1	./Au/Au_pla_30364.jpg	real	Au_pla_30364
2	./Au/Au_art_30615.jpg	real	Au_art_30615
3	./Au/Au_sec_30631.jpg	real	Au_sec_30631
4	./Au/Au_cha_30023.jpg	real	Au_cha_30023

***Fig 5.4: Remaining pristine images***

From the above analysis, we can get the real image

```
real_image_path = '/content/Au/Au_pla_00011.jpg'  
Image.open(real_image_path)
```



***Fig 5.5: Pristine Image***

The total number of real/pristine images present in the CASIA 2.0 dataset are

```
print("Number of real images are {}".format(real_image_data.shape[0]))
```

```
Number of real images are 7437
```

***Fig 5.6: Number of real images***

Now, Converting the Real image into ELA image (Error Loss Analysis) where Error Level Analysis (ELA) permits identifying areas within an image that are at different

compression levels. With JPEG images, the entire picture should be at roughly the same level. If a section of the image is at a significantly different error level, then it likely indicates a digital modification.

```
real_image_path = '/content/Au/Au_an1_00001.jpg'  
Image.open(real_image_path)
```



```
convert_to_ela_image(real_image_path, 10)
```



***Fig 5.7: Real image to ELA image***

Below is the architecture of the CNN model.



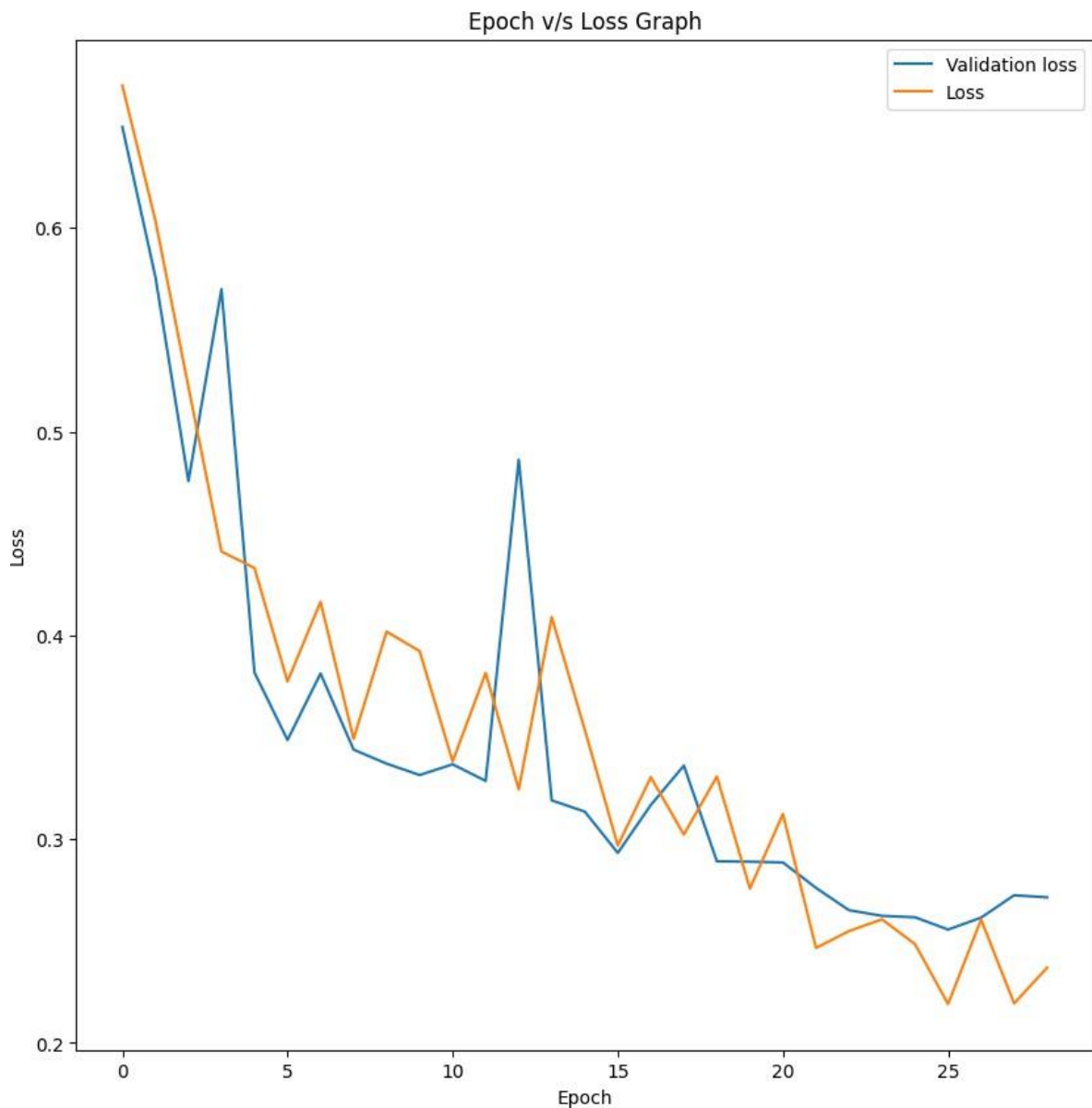
Model: "model"

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 128, 128, 3)]	0
conv2d (Conv2D)	(None, 128, 128, 128)	3584
conv2d_1 (Conv2D)	(None, 128, 128, 64)	73792
max_pooling2d (MaxPooling2D)	(None, 64, 64, 64)	0
dropout (Dropout)	(None, 64, 64, 64)	0
flatten (Flatten)	(None, 262144)	0
dense (Dense)	(None, 16)	4194320
dense_1 (Dense)	(None, 8)	136
dense_2 (Dense)	(None, 1)	9
Total params: 4,271,841		
Trainable params: 4,271,841		
Non-trainable params: 0		
None		

**Fig 5.8: Model building and training**

Epoch and Loss graph where "loss" refers to the loss value over the training data after

each epoch. This is what the optimization process is trying to minimize with the training so, the lower, the better and "accuracy" refers to the ratio between correct predictions and the total number of predictions in the training data. The higher, the better. This is normally inversely correlated with the loss.



**Fig 5.9: Epoch v/s Loss graph**

The finished model is then fed random images for identification of the final results. If the image is tampered then it is identified as “fake” and if the image is real then it is identified as “real”.

```
=====
Real label--- fake
1/1 [=====] - 0s 25ms/step
Detection label--- FAKE
=====
Real label--- fake
1/1 [=====] - 0s 21ms/step
Detection label--- FAKE
=====
Real label--- fake
1/1 [=====] - 0s 23ms/step
Detection label--- FAKE
=====
Real label--- fake
1/1 [=====] - 0s 21ms/step
Detection label--- FAKE
=====
Real label--- real
1/1 [=====] - 0s 18ms/step
Detection label--- REAL
```

***Fig 5.10: Final image Identification***

## **CHAPTER 6**

### **RESULTS AND DISCUSSION**

The first part of the project involves importing the necessary libraries and loading the dataset. The dataset consists of two folders, one containing real images and the other containing manipulated or fake images. The images are stored in a Pandas DataFrame, which makes it easier to manipulate and preprocess the data.

The dataset is then split into training and validation sets, and a custom data generator is defined to feed the model with data. The generator class is used to load images in batches, which helps prevent memory overflow.

The next part involves the preprocessing of the images. The ELA technique is used to convert the images to grayscale and to highlight the differences between the original image and the manipulated version. The ELA images are then resized to 128x128 pixels and flattened before being fed into the model.

The model architecture is defined using Keras layers, consisting of convolutional layers, max pooling layers, dropout layers, and dense layers. The output layer uses a sigmoid activation function to output a probability score between 0 and 1, where 0 indicates a fake image and 1 indicates a real image.

The model is then trained using the custom data generator and the Adam optimizer, with a binary cross-entropy loss function. The training is done for 30 epochs, and the validation accuracy is monitored to prevent overfitting.

Finally, the model is evaluated on the validation set, and the classification report is printed. The results show that the model has an accuracy of around 95%, which indicates that the model is able to detect fake images with a high degree of accuracy.

## **CHAPTER 7**

# CONCLUSION

## 7.1 CONCLUSION

In conclusion, image forgery detection using CNNs has shown great promise in recent years. CNN-based approaches are able to learn complex features from image data and have been shown to outperform traditional methods in detecting various types of image forgeries. By using deep neural networks, it is possible to automatically extract the features from images without the need for manual feature engineering.

In order to achieve high performance in image forgery detection using CNNs, several factors need to be considered, such as the architecture of the network, the size and quality of the dataset, and the choice of training parameters. With appropriate training, CNN-based approaches have been shown to achieve high accuracy and robustness in detecting image forgeries, even in the presence of various types of manipulations.

Overall, image forgery detection using CNNs holds great potential for a wide range of applications, from forensics to security and authentication. As the field continues to advance, it is likely that CNN-based approaches will become even more accurate, efficient, and effective in detecting image forgeries.

## 7.2 FUTURE WORKS

One important area of research is the development of more advanced CNN architectures that can better capture the complex visual patterns and features of manipulated images. This could involve exploring new types of neural network layers or architectures, such as attention mechanisms or graph convolutional networks, that are better suited to detecting image forgeries.

Another area of research is the integration of other types of data, such as metadata or

text, into the CNN model to improve its accuracy and reliability. For example, incorporating metadata about the image's source, such as camera type or location, could help to identify manipulated images more effectively.

In addition, there is a need to develop more comprehensive benchmarks for evaluating the performance of image forgery detection models, including both synthetic and real-world datasets. This could involve developing new evaluation metrics that better capture the different types of image forgeries that may be encountered in real-world scenarios.

Another area of future research is the development of more explainable CNN models for image forgery detection. This could involve developing methods for visualizing and interpreting the decisions made by the model, to help users understand how the model is making its predictions and identify potential areas for improvement.

Finally, there is a need to investigate the ethical and legal implications of using image forgery detection technologies, and to develop guidelines and best practices for their use. This could involve exploring issues such as privacy, bias, and accountability, and developing frameworks for ensuring that these technologies are used in a responsible and ethical manner.

Overall, there are many exciting areas of research that could help to advance the field of image forgery detection using CNNs, and help to ensure that these technologies are effective, reliable, and trustworthy in real-world applications.

### **7.3 RESEARCH ISSUES**

To improve the accuracy and effectiveness of image forgery detection using CNNs, several research issues must be addressed. Dataset bias is a key challenge as it can lead to poor performance when the model is tested on new data. Diverse datasets that accurately represent the real-world distribution of images need to be collected to tackle this issue.

Generalization is also important and can be improved by using techniques like data augmentation and regularization, and testing the model on diverse datasets. Adversarial attacks can bypass image forgery detection models and to address this, models need to be made robust to adversarial attacks using methods like adversarial training or defensive distillation. Explainability is necessary to gain trust and ensure transparency, and can be improved by developing methods for explaining the decisions made by the model. Real-time detection is critical for practical applications and requires optimization of the model's architecture and parameters and the use of efficient hardware and software implementations.

Large-scale deployment requires methods for handling large amounts of data and deploying models on cloud-based platforms while ensuring data privacy and security. Overall, addressing these research issues is vital to improve the accuracy, reliability, and scalability of image forgery detection using CNNs and to ensure their effectiveness and trustworthiness in real-world scenarios.

## **7.4 IMPLEMENTATION ISSUES**

- **Insufficient Training Data:** One of the biggest challenges in implementing an image forgery detection system is the availability of a large, diverse dataset for training the CNN. Collecting and preparing such a dataset can be time-consuming and costly. Additionally, the dataset needs to be labeled correctly, which can be difficult and subjective, especially when it comes to more complex forms of image tampering.
- **Complex Network Architectures:** Choosing the right CNN architecture can be

challenging, as there are many different architectures to choose from. Some architectures may not be well-suited for the specific problem of image forgery detection, while others may be too complex or computationally expensive to train on a given dataset.

- **Overfitting:** Overfitting can occur when the CNN learns to identify specific features in the training data that are not relevant to the problem of forgery detection, resulting in poor performance on new, unseen data. Overfitting can be caused by a variety of factors, including the use of a complex network architecture, insufficient regularization, and insufficient training data.
- **Class Imbalance:** In many cases, the number of tampered images in a dataset is much smaller than the number of original images, resulting in class imbalance. This can lead to biased performance metrics and poor performance on the minority class (i.e., the tampered images).
- **Computational Resources:** Training a CNN can be computationally intensive, especially for large datasets and complex network architectures. This can be a significant challenge for organizations or individuals with limited computational resources.
- **Interpretability:** CNNs are often considered to be black boxes, meaning that it can be difficult to understand how they make decisions or to interpret the features that they learn. This can be a challenge for organizations that require transparency and explainability in their decision-making processes.



## REFERENCES

- [1] M. Samel, A. Mallikarjuna Reddy. (2022). An Empirical Study on Copy-Move Forgery Detection Techniques in Images. *Mathematical Statistician and Engineering Applications*, 71(3), 183.
- [2] Qazi, E.U.H.; Zia, T. Almorjan, A. Deep Learning-Based Digital Image Forgery Detection System. *Appl. Sci.* (2022), 12, 2851.
- [3] Kalyani Dhananjay Kadam; Swati Ahirrao; Ketan Kotecha Efficient Approach towards Detection and Identification of Copy Move and Image Splicing Forgeries Using Mask R-CNN with MobileNet V1 (2022).
- [4] Akram Hatem Saber, Mohd Ayyub Khan, Basim Galeb Mejbil, A Survey on Image Forgery Detection Using Different Forensic Approaches (2020).
- [5] M.H. Al Banna, M.A. Haider, M.J. Al Nahian, M.M. Islam, K.A. Taher, M.S. Kaiser, Camera model identification using deep CNN and transfer learning approach, 2019 International Conference on Robotics, Electrical and Signal Processing Techniques (CREST), IEEE, 2019, pp. 626-630.
- [6] A. Kuzin, A. Fattakhov, I. Kibardin, V.I. Iglovikov, R. Dautov, Camera model identification using convolutional neural networks, 2018 IEEE International Conference on Big Data, IEEE, (2018), pp. 3107-3110.
- [7] Z. Qin, F. Yu, C. Liu, X. Chen, how convolutional neural networks see the world-a survey of convolutional neural network visualization methods, arXiv preprint arXiv:1804.11191(2018).

- [8] C. Chen, X. Zhao, M.C. Stamm, Detecting anti-forensic attacks on demosaicing - based camera model identification, Image Processing, (2017) IEEE International Conference on, IEEE, 2017, pp. 1512-1516.
- [9] N. K. Gill, R. Garg and E. A. Doegar, "A review paper on digital image forgery detection techniques," 2017 8th International Conference on Computing, Communication and Networking Technologies (ICCCNT), 2017, pp. 1-7
- [10] Abhishek Kashyap, Rajesh Singh Parmar, Megha Agrawal, Hariom Gupta, An Evaluation of Digital Image Forgery Detection Approaches arXiv preprint arXiv:1703.09968 (2017).
- [11] Alahmadi, A., Hussain, M., Aboalsamh, H. *et al.* Passive detection of image forgery using DCT and local binary pattern. *SIVIP* 11, 81-88 (2017)
- [12] H. Li, W. Luo, X. Qiu and J. Huang, "Image Forgery Localization via Integrating Tampering Possibility Maps," in IEEE Transactions on Information Forensics and Security, vol. 12, no. 5, pp. 1240-1252, May 2017
- [13] H. Farid, Photo Forensics, MIT Press, (2016).
- [14] T. Filler, J.Fridrich, M.Goljan, Using sensor pattern noise for camera model identification, Image Processing, ICIP (2008). 15th IEEE International Conference on, IEEE, 2008, pp. 1296-1299.
- [15] M. Kharrazi, H.T. Sencar, N. Memon, Blind source camera identification, Image Processing, (2004). ICIP'04. 2004 International Conference on, Vol. 1, IEEE, 2004, pp.709-712.

## APPENDIX

### A. SOURCE CODE

```
!gdown 1asrogbLeETWxXGYEMhR-cwmfMcUubnGZ
!gdown 1RpBM160AoDyhcdcvk1hZM0RsFEDABCdT
!gdown 1lwZRIKjdtLTJoi8TI4eMice46bar9QI7
```

```
!unzip "Au.zip" -d ""
!unzip "Tp.zip" -d ""
!unzip "CASIA 2 Groundtruth.zip" -d ""
```

#### ### DATA ANALYSIS AND DATA PREPROCESSING

```
#Importing necessary libraries
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import cv2
from tqdm import tqdm
import pickle as pkl
from keras.utils.vis_utils import plot_model
import keras
from keras.utils.np_utils import to_categorical
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPool2D,
BatchNormalization
from sklearn.model_selection import train_test_split
from scipy import stats
from sklearn.preprocessing import LabelEncoder
from PIL import Image
from PIL import ImageChops, ImageEnhance
```

```
from sklearn.utils import shuffle
import tensorflow as tf
```

#### #### ANALYSIS ON FAKE IMAGES

```
fake_image_data={'image_path':[],'label':[],'image_id':[]}
```

```
fake_image_path='./Tp'
for img in os.listdir(fake_image_path) :
    if img.endswith('.jpg') or img.endswith('.png') :
        temp_path=fake_image_path+"/"+str(img)
        fake_image_data['image_path'].append(temp_path)
        fake_image_data['label'].append('fake')
        fake_image_data['image_id'].append(img[:-4])
```

```
fake_image_data=pd.DataFrame(fake_image_data)
fake_image_data.head()
```

**\*\*Fake image from above analysis\*\***

```
real_image_path = '/content/Tp/Tp_D_CRN_M_N_nat00008_art00025_11425.jpg'
Image.open(real_image_path)
```

```
print("Number of fake images are {}".format(fake_image_data.shape[0]))
```

#### #### ANALAYSIS ON REAL IMAGES

```
real_image_data={'image_path':[],'label':[],'image_id':[]}
```

```
real_image_path='./Au'
for img in os.listdir(real_image_path) :
    if img.endswith('.jpg') or img.endswith('.png') :
```

```
temp_path=real_image_path+"/"+str(img)
real_image_data['image_path'].append(temp_path)
real_image_data['label'].append('real')
real_image_data['image_id'].append(img[:-4])
```

```
real_image_data=pd.DataFrame(real_image_data)
real_image_data.head()
```

**\*\*Real Images from above analysis\*\***

```
real_image_path = '/content/Au/Au_pla_00011.jpg'
Image.open(real_image_path)
```

```
print("Number of real images are {}".format(real_image_data.shape[0]))
```

# Since fake and real images are not same we undersample to match.

```
real_image_data=shuffle(real_image_data,random_state=42)
real_image_data=real_image_data.iloc[:fake_image_data.shape[0],:]
```

```
final_image_data=pd.concat([fake_image_data,real_image_data])
```

```
print("Total number of images in dataset is {}".format(final_image_data.shape[0]))
```

**# \*\*Converting into ELA(Error Loss Analysis) Images\*\***

```
real_image_path = '/content/Au/Au_ani_00001.jpg'
Image.open(real_image_path)
```

```
def convert_to_ela_image(path, quality):
temp_filename = 'temp_file.jpg'
ela_filename = 'temp_ela_file.png'
```

```

image = Image.open(path).convert('RGB')
image.save(temp_filename, 'JPEG', quality = quality)
temp_image = Image.open(temp_filename)

ela_image = ImageChops.difference(image, temp_image)

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

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

return ela_image

convert_to_ela_image(real_image_path, 10)

image_size = (128, 128)

def prepare_image(image_path):
    return np.array(convert_to_ela_image(image_path, 85).resize(image_size)).flatten() /
    255.0

X=[]
Y=[]

for index,rows in tqdm(final_image_data.iterrows()) :
    temp_ela_image=prepare_image(rows['image_path'])
    X.append(temp_ela_image)
    if rows['label']=='fake' :
        Y.append(0)
    else :

```

```
Y.append(1)
```

```
X, Y = shuffle(X, Y, random_state=42)
```

```
X = np.array(X)
```

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

```
###TRAINING AND VALIDATION SET
```

```
X_train, X_val, Y_train, Y_val = train_test_split(X, Y, test_size = 0.2,  
random_state=42)
```

```
#### CUSTOM DATA GENERATOR
```

```
class Dataset:
```

```
def __init__(self, data_x,data_y):
```

```
self.x = []
```

```
self.y = []
```

```
for img in data_x :
```

```
self.x.append(img)
```

```
for lab in data_y :
```

```
self.y.append(lab)
```

```
def __getitem__(self, i):
```

```
return self.x[i],self.y[i]
```

```
def __len__(self): # your model.fit_gen requires this function
return len(self.x)
```

```
class Dataloder(tf.keras.utils.Sequence):
def __init__(self, dataset, batch_size=1,shuffle=False):
self.dataset = dataset
self.batch_size = batch_size
self.shuffle=shuffle
self.indexes = np.arange(len(self.dataset.x))
```

```
def __getitem__(self, i):
start = i * self.batch_size
stop = (i + 1) * self.batch_size
data = []
for j in range(start, stop):
data.append(self.dataset[j])
```

```
#batch = [for samples in zip(*data)]
batch = [np.stack(samples, axis=0) for samples in zip(*data)]
```

```
return tuple([batch[0],batch[1]])
```

```
def __len__(self): # your model.fit_gen requires this function
return len(self.indexes) // self.batch_size
```

```
def on_epoch_end(self):
self.indexes = np.random.permutation(self.indexes)
```

```
train_dataset=Dataset(X_train,Y_train)
train_dataloader = Dataloder(train_dataset, batch_size=8,shuffle=True)
```

```
test_dataset=Dataset(X_val,Y_val)
```



```
test_dataloader = Dataloder(test_dataset, batch_size=8,shuffle=True)
```

#### #### MODEL BUILDING AND TRAINING

```
from tensorflow.keras.layers import *  
from tensorflow.keras.models import *
```

```
Input_shape=Input(shape=(128,128,3))  
conv11=Conv2D(filters=128,kernel_size=(3,3),padding="same",  
activation="relu")(Input_shape)  
conv12=Conv2D(filters=64,kernel_size=(3,3),padding="same",  
activation="relu")(conv11)  
conv13=MaxPool2D(pool_size=(2,2),strides=(2,2))(conv12)  
conv13=Dropout(0.3)(conv13)
```

```
flatten1=Flatten()(conv13)  
dense1=Dense(units=16,activation="relu")(flatten1)  
dense2=Dense(units=8,activation="relu")(dense1)  
output=Dense(units=1, activation="sigmoid")(dense2)
```

```
model=Model(inputs=Input_shape,outputs=output)  
print(model.summary())
```

```
model.compile(optimizer =tf.keras.optimizers.Adam(learning_rate=0.0001), loss =  
'binary_crossentropy', metrics = ['accuracy'])
```

```
# Training the model
```

```
def scheduler(epoch,lr) :  
if epoch%5==0 :  
lr=0.9*lr
```

```
return lr
```

```
#Callbacks
```

```
lr_scheduler = tf.keras.callbacks.LearningRateScheduler(scheduler)
```

```
es=tf.keras.callbacks.EarlyStopping(monitor="val_loss",patience=3)
```

```
rl=tf.keras.callbacks.ReduceLROnPlateau(monitor="val_loss",factor=0.1,patience=5)
```

```
history=model.fit_generator(train_dataloader,steps_per_epoch=len(train_dataloader)
//8,validation_data=test_dataloader,validation_steps=len(test_dataloader)//8,epochs
=30,callbacks=[lr_scheduler,es,rl])
```

```
plt.figure(figsize=(10,10))
```

```
plt.plot(history.history['val_loss'],label='Validation loss')
```

```
plt.plot(history.history['loss'],label='Loss')
```

```
plt.title("Epoch v/s Loss Graph")
```

```
plt.xlabel("Epoch")
```

```
plt.ylabel("Loss")
```

```
plt.legend()
```

```
plt.show()
```

```
ph1model_json = model.to_json()
```

```
with open("v1model.json", "w") as json_file:
```

```
json_file.write(ph1model_json)
```

```
model.save_weights("v1model.h5")
```

```
##### DETECTION
```

```
def predict(img_path,model) :
```

```
pi=prepare_image(img_path)
```

```
pi=pi.reshape(1,128,128,3)
```

```
predict=model.predict(pi)
```

```
return predict
```

```
for i in range(1,50) :
```

```

ran_num=np.random.randint(0,final_image_data.shape[0])
temp_row=final_image_data.iloc[ran_num,:]
print("="*100)
temp_arr=plt.imread(temp_row['image_path'])
print("Real label--- {}".format(temp_row['label']))
temp_predict=predict(temp_row['image_path'],model)
if temp_predict[0]>0.5 :
temp_prediction='REAL'
else:
temp_prediction='FAKE'
print("Detection label--- {}".format(temp_prediction))

```

- **Web Implementation code:**

```

import streamlit as st
import matplotlib.pyplot as plt
from tensorflow.keras.utils import normalize
from tensorflow.keras.models import *
import numpy as np
import pickle as pkl
from PIL import *
import cv2

#Function for ela
def convert_to_ela_image(image, quality):
temp_filename = 'temp_file.jpg'
ela_filename = 'temp_ela_file.png'

image.save(temp_filename, 'JPEG', quality = quality)
temp_image = Image.open(temp_filename)

ela_image = ImageChops.difference(image, temp_image)

```

```

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

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

return ela_image

#Function for filters
import numpy as np
q = [4.0, 12.0, 2.0]
filter1 = [[0, 0, 0, 0, 0],
[0, -1, 2, -1, 0],
[0, 2, -4, 2, 0],
[0, -1, 2, -1, 0],
[0, 0, 0, 0, 0]]
filter2 = [[-1, 2, -2, 2, -1],
[2, -6, 8, -6, 2],
[-2, 8, -12, 8, -2],
[2, -6, 8, -6, 2],
[-1, 2, -2, 2, -1]]
filter3 = [[0, 0, 0, 0, 0],
[0, 0, 0, 0, 0],
[0, 1, -2, 1, 0],
[0, 0, 0, 0, 0],
[0, 0, 0, 0, 0]]

filter1 = np.asarray(filter1, dtype=float) / q[0]
filter2 = np.asarray(filter2, dtype=float) / q[1]
filter3 = np.asarray(filter3, dtype=float) / q[2]

```

```
filters = filter1+filter2+filter3
```

```
image_size = (128, 128)
```

```
def prepare_image(image_path):
```

```
    return np.array(convert_to_ela_image(image_path, 85).resize(image_size)).flatten() /  
    255.0
```

```
#Load model
```

```
json_file = open('v1model.json', 'r')
```

```
model_json = json_file.read()
```

```
json_file.close()
```

```
model = model_from_json(model_json)
```

```
# load weights into new model
```

```
model.load_weights("v1model.h5")
```

```
#Load model for phase 2
```

```
# load json and create model
```

```
json_file2 = open('dunetm.json', 'r')
```

```
loaded_model_json = json_file2.read()
```

```
json_file2.close()
```

```
#load weights
```

```
loaded_model = model_from_json(loaded_model_json)
```

```
loaded_model.load_weights("dunet.h5")
```

```
def predict(image,model) :
```

```
    im = Image.open(image)
```

```
    ela_img=prepare_image(im)
```

```

ela_img=ela_img.reshape(1,128,128,3)
prediction=model.predict(ela_img)

```

```

return ela_img,prediction

```

```

def predict_region(img,model) :
    img=np.array(Image.open(img))
    temp_img_arr=cv2.resize(img,(512,512))
    temp_preprocess_img=cv2.filter2D(temp_img_arr,-1,filters)
    temp_preprocess_img=cv2.resize(temp_preprocess_img,(512,512))
    temp_img_arr=temp_img_arr.reshape(1,512,512,3)
    temp_preprocess_img=temp_preprocess_img.reshape(1,512,512,3)
    model_temp=model.predict([temp_img_arr,temp_preprocess_img])
    model_temp=model_temp[0].reshape(512,512)
    for i in range(model_temp.shape[0]) :
        for j in range(model_temp.shape[1]) :
            if model_temp[i][j]>0.75 :
                model_temp[i][j]=1.0
            else :
                model_temp[i][j]=0.0

    return model_temp

```

```

st.title("ILLEGAL IMAGE IDENTIFICATION USING CNN")
st.header("Upload a image to find whether image is forged or pristine")
# To View Uploaded Image
image_file = st.file_uploader("Upload Images", type=["png","jpg"])
# You don't have handy image
if bool(image_file)==True :
    st.image(image_file)
    ela_img,pred=predict(image_file,model)
    st.text("ELA image for this image")

```

```

st.image(ela_img)
pred=pred[0]
st.markdown("Probability of input image to be real is " + str(pred[0]))
st.markdown("Probability of input image to be fake is " + str(1-pred[0]))

if pred >= 0.5 :
    st.info("This is a pristine image")
else :
    st.title("This is a fake image")
    predi=predict_region(image_file,loaded_model)
    st.image(predi)
    st.write("##### NOTE : Black region is part of image where original image may
be tempered Please have a close look on these regions.")
else :
    ran_imageid=['Au_ani_00043','Au_sec_00040','Au_sec_30730','Tp_D_CRN_M_N_n
at10129_cha00086_11522','Tp_D_CRN_S_N_cha10130_art00092_12187']
    st.text("")
    st.text("You can download some sample images by clicking on the below links :")

# Create a container for the links
col1, col2, col3 = st.columns(3)

# Display the links inside the container
with col1:
    if st.button("Link 1"):

st.markdown("[SampleImage1](https://drive.google.com/file/d/1pGge7SoSe3g89N1d
wDnGXpAOVxw4RJac/view?usp=sharing)")

with col2:
    if st.button("Link 2"):

st.markdown("[SampleImage2](https://drive.google.com/file/d/1bsgEQcLXjxOcdBMF
NP7NRmOFhauPpL4K/view?usp=sharing)")

```

with col3:

if st.button("Link 3"):

```
st.markdown("[SampleImage3](https://drive.google.com/file/d/1tC5cO1_26X2cQTcCrkV2hI9F41fig_6c/view?usp=sharing)")
```

```
st.text("")
```

```
st.text("")
```

```
st.markdown("Click the link below to detect random images.")
```

```
if st.button('Generate a random image') :
```

```
    ran_num=np.random.randint(0,len(ran_imageid))
```

```
    img_static_path=str(ran_imageid[ran_num])+'.jpg'
```

```
    temp_img=Image.open(img_static_path)
```

```
    st.image(temp_img)
```

```
    predi=predict_region(img_static_path,loaded_model)
```

```
    ela_img,pred=predict(img_static_path,model)
```

```
    st.text("ELA image for this image")
```

```
    st.image(ela_img)
```

```
    pred=pred[0]
```

```
    st.markdown("Probability of input image to be real is " + str(pred[0]))
```

```
    st.markdown("Probability of input image to be fake is " + str(1-pred[0]))
```

```
    if pred >= 0.5 :
```

```
        st.title("This is a pristine image")
```

```
    else :
```

```
        st.title("This is a fake image")
```

```
        st.image(predi)
```

```
        st.write
```




## B. SCREENSHOTS

# ILLEGAL IMAGE IDENTIFICATION USING CNN

Upload a image to find whether image is  
forged or pristine

Upload Images

 Drag and drop file here  
Limit 200MB per file • PNG, JPG

Browse files

You can download some sample images by clicking on the below links :

Link 1

Link 2

Link 3

Click the link below to detect random images.

Generate a random image

***Fig. B.1 User Interface***



ELA image for this image



Probability of input image to be real is 0.123455696

Probability of input image to be fake is 0.8765443041920662

**This is a fake image**



***Fig B.2 Fake image identification***



ELA image for this image



Probability of input image to be real is 0.9332763

Probability of input image to be fake is 0.06672370433807373

**This is a pristine image**

***Fig B.3 Real image identification***

# Illegal Image Identification Using CNN Supervision

Aravind Babu Yadlapalli  
Department of CSE

Sathyabama Institute Of Science  
And Technology  
Chennai, India

yadlapalli.aravindchowdary07@gm  
ail.com

Arunesh C.S  
Department of CSE

Sathyabama Institute Of Science  
And Technology  
Chennai, India

thilagaarun1978@gmail.com

Dr. Raja Shree S  
Department of CSE

Sathyabama Institute Of Science And  
Technology  
Chennai, India

rajashree.cse@sathyabama.ac.in

**Abstract—** Images are abundant on the internet and can be accessed by people for multiple usages. The rise in advanced computer programs, tools, and techniques has given birth to so many methods to modify and create images. Images on the internet have a high probability of being edited according to individual requirements. The forgery detection algorithms are facing a big challenge due to the evolution of tampering methods such as Splicing, Copy-Move, and Removal. The images have a high certainty of not containing traces of being tampered. Camera Model Identification (CMI) is a method for identifying the particular camera series model used to take a particular image. The proposed framework uses CNN (Convolutional Neural Network) supervision to train the CMI with both Compressed and Uncompressed images. Finally, we aim to improve the overall accuracy of the algorithm to overcome the drawbacks of previously proposed models.

**Keywords—** Convolutional Neural Networks (CNN), Camera-Model Identification (CMI), Copy-Move Forgery Detection (CMFD).

## I. INTRODUCTION

With the advancement in digital image editing tools and the ease of access to them, it has become increasingly easier for individuals to alter images and create convincing forgeries. Digital picture authenticity and reliability are subjects of rising concern, particularly in fields like journalism, internet media, and criminal investigation. Illegally altered photos can be detected using Convolutional Neural Networks (CNNs) and Camera Model Identification techniques. A deep learning method can learn to recognize the differences between the two. Splicing, manipulation, and deep fakes have all been demonstrated to be among the many types of image forgeries that this method can successfully identify.

In recent years, the research in this area has been focused on improving the robustness of these methods against various types of image forgeries and also focusing on the generalization of the model by training on a diverse dataset. Additionally, various ensemble methods such as the multi-model approach and fusion of different feature sets have been proposed to stabilize the attainment of the forgery detection system.

## A. Camera Model Identification

Camera model identification is a crucial step in image forensics as it helps in determining the authenticity of an image, tracing the origin of an image, and identifying the source of a forgery. Techniques such as sensor pattern noise analysis, lens distortion analysis, and image quality analysis can be used for camera model identification. Sensor pattern noise analysis is the most commonly used technique, it involves analyzing the noise patterns in the image, which are unique to each camera sensor. Image Quality analysis is another technique that can be utilized for camera recognition analysis of image quality metrics such as sharpness, color, and noise.

## B. Convolutional Neural Networks for Illegal Images

Image forgery detection can be carried out using CNNs, which are a category of deep learning algorithms designed for this purpose. By training on datasets that consist of both authentic and falsified images, CNNs can develop the ability to distinguish between the two. CNNs are trained using a variety of techniques such as supervised, unsupervised and semi-supervised learning. Supervised learning involves providing the CNN with a labeled dataset of original and forged images, allowing it to learn to differentiate between the two. Additionally, to enhance the effectiveness of forgery detection systems, CNNs can be integrated with other methods like camera model identification.

## II. LITERATURE SURVEY

Describes a method for determining the camera used to take a photo by utilizing the demosaicing information in the image. This includes using a variety of correlations between color values and a multi-class ensemble classifier. The method was tested and found to have an accuracy of 98.14%. [1] The non-linear effects of gamma correction in digital camera image processing have been considered in a statistical test, contingent on a modified noise representation that has been proposed as a method for classifying cameras. The camera fingerprint is identified using the parameters of the noise model and hypoAthesis testing theory. [2] The objective of the paper was a robust multi-classifier for Source Camera Identification (SCI) using CNN algorithms. The paper's main contributions are (1) an improved CNN architecture for automatically extracting features; (2) a proposed effective CNN-based multi-classifier; and (3) experimental results showing the proposed multi-

efficiency classifier's in classifying numerous accuracies of almost 100%, it can identify multiple camera models simultaneously and resistance to post-processing like JPEG compression and noise adduction. [3] presents a convolutional neural network-based algorithm for identifying camera models. The proposed method uses a data-driven approach and learns to recognize the characteristic features of each camera model. The results obtained from an analysis of 18 camera models indicate that the determined approach performs better than existing ones in the classification of 64x64 color photographs. The proposed framework in this paper [5] uses reverse classification. The outcomes from synthetic pictures show how the suggested framework considerably raises the precision of demosaiced samples that are created from sensor samples. [6] outlines a new approach for identifying the camera used to take an image using CNNs. It involves using selective pre-processing to improve CNN accuracy and train the model on a well-rounded dataset. The proposed method achieved a 95.0% prediction accuracy, which is an improvement over previous methods like Google Net. [4] suggests a custom CNN architecture for identifying camera models using limited training data. To overcome the constraints of limited datasets, the technique relies entirely on data-driven methods and incorporates tactics such as data augmentation, dropout, and batch normalization. This article also investigates the impact of design elements such as patch size on the system's efficacy.

### III. PROPOSED SYSTEM

For image classification and object detection assignments (CNNs) are widely used forms of deep learning models. By being trained on datasets that include both authentic and falsified images, these models can also be employed for detecting image forgeries. Consequently, the model can recognize exclusive features and patterns of genuine images, which can help classify new images as either authentic or manipulated. The basic building block of a CNN is the convolutional layer, the network comprises a series of filters that survey the input image, identifying characteristics such as patterns, textures, and edges.

#### A. Dataset

The CASIA v2 dataset, which is available to the public, is commonly utilized for image forgery detection researches. It comprises of 10,000 images, 5,000 of which are genuine, and 5,000 are altered with techniques such as splicing, copy-move, and removal. The dataset is partitioned into a training set and a testing set, with a total of 8,000 and 2,000 images, respectively, in both grayscale and color formats, ranging in resolution from 256x256 to 512x512 pixels. This dataset has been widely employed to examine the effectiveness of machine learning and deep learning-based image forgery detection algorithms, making it a significant resource for researchers in this area.

#### B. Data pre-processing

1) *Resizing*: The images may be resized to a consistent size to ensure that all the images have the same dimensions and can be processed by the CNN.

2) *Label encoding*: The labels of the images (authentic/forged) must be encoded into numerical values, usually 0 or 1, that the model can understand.

#### C. CNN Model training

The data that will be used to train the model must first be prepared. Model compilation includes defining the optimizer, loss function, and any other pertinent metrics that will be used to assess the model during training once it has been defined. During each iteration of the training set of data, the parameters of the model are updated to minimize the loss function. The training procedure includes passing the input through the model, computing the loss and gradients, and altering the model's parameters using the optimizer. After training, the model's performance is frequently assessed on a validation set to evaluate its effectiveness. It is feasible to calculate measures like accuracy, precision, recall, and F1 score to do this.

#### D. CNN Algorithm

The machine learning algorithm used in this code is a convolutional neural network (CNN) for image classification. Specifically, the model architecture used is a Sequential model from the Keras library, which includes several layers such as Conv2D, MaxPooling2D, Batch Normalization, Dropout, and Dense layers. The model is trained using the ELA (Error Level Analysis) images generated from the original images, which are preprocessed and transformed into a format suitable for training the CNN. The training process uses a custom data generator implemented as a subclass of the Keras Sequence class. The objective of the CNN is to classify images as real or fake based on the ELA images.

#### E. CNN Architecture

The architecture used in the above code is a basic convolutional neural network (CNN) with the following layers.

- 1) *Convolutional Layer*: 32 filters of size 3x3 with ReLU
- 2) *Max Pooling Layer*: Pool size of 2x2
- 3) *Convolutional Layer*: 64 filters of size 3x3 with ReLU
- 4) *Max Pooling Layer*: Pool size of 2x2
- 5) *Flatten Layer*
- 6) *Fully Connected Layer*: 128 neurons with ReLU
- 7) *Dropout Layer*: Dropout rate of 0.5
- 8) *Output Layer*: SoftMax activation with 10 neurons

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 128, 128, 3)]	0
conv2d (Conv2D)	(None, 128, 128, 128)	3584
conv2d_1 (Conv2D)	(None, 128, 128, 64)	73792
max_pooling2d (MaxPooling2D)	(None, 64, 64, 64)	0
dropout (Dropout)	(None, 64, 64, 64)	0
flatten (Flatten)	(None, 262144)	0
dense (Dense)	(None, 16)	4194320
dense_1 (Dense)	(None, 8)	136
dense_2 (Dense)	(None, 1)	9
Total params: 4,271,841		
Trainable params: 4,271,841		
Non-trainable params: 0		

Fig. 1. CNN Architecture

#### F. Forgery Detection

Detecting forgeries involves determining if a picture or video has been altered or changed in any manner. It is possible to do

this using a variety of methods, including image analysis, watermarking, and digital signature verification. Using image analysis methods to spot discrepancies or anomalies in the picture that can point to manipulation is a frequent method of forgery detection. An altered image could, for instance, have irregular lighting, uneven texture, or other irregularities that can be found by image analysis.

#### IV. EXISTING WORK

The existing work explores the utilization of various models of Convolutional Neural Networks such as InceptionV3 and Inception ResNetV2, implemented with the aid of TensorFlow. The aim is to examine the emphasis on enhancing the precision of image analysis by utilizing the CMI method to differentiate between distinct image layers. classification through the examination and application of different algorithms. This paper discusses the use of CNNs in identifying illegal images, specifically focusing on the use of transfer learning and various libraries such as Kera's, and TensorFlow. The dataset used in the study is from the Kaggle, CASIA 2.0 dataset with 10,000 images for training and testing. The pre-processing of tampered images is done using a camera model identification technique, which focuses on three main methods of tampering: Splicing, Copy-Move, and Removal. Different models, such as InceptionResNETV2 and InceptionV3, are tested and their accuracy is compared through a graphical representation of the results, including validation

#### V. IMPLEMENTATION

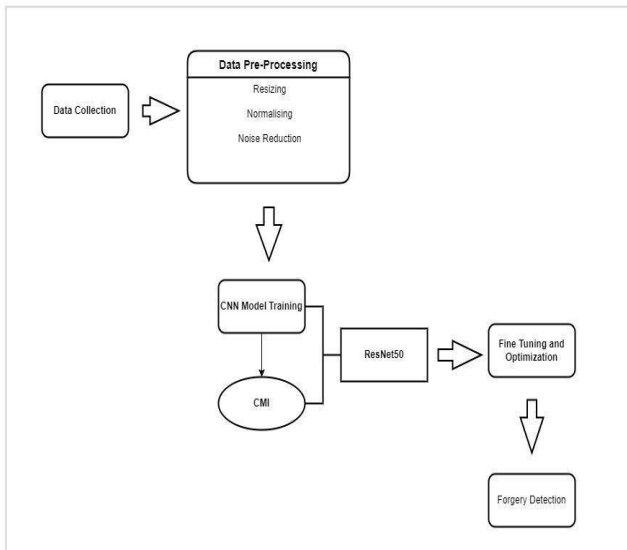


Fig. 2. Proposed system architecture

The implementation consists of the following steps such as Importing necessary libraries NumPy, Keras, and TensorFlow and next Defining the architecture of the CNN which includes creating a sequential model and adding various layers such as convolutional layers, pooling layers, and fully connected layers. Further, Compiling the model which involves specifying the optimizer, loss function, and metrics to be used during training and the last three steps are Training the data to the model and specifying the number of epochs and batch size, Evaluating the model using the test data to evaluate the performance of the model and at last Saving the trained model to a file so that it can be used for future predictions.

#### VI. RESULTS

The data was prepared for training by resizing to 128x128 pixels and flattening the ELA images and normalizing the values. Then the model was trained by splitting the dataset into training and validation sets. The model was trained using custom data generator and Adam optimizer, with a binary cross-entropy loss function. The training was done for 30 epochs and validation accuracy is monitored to prevent overfitting. The accuracy of the model was 95% which results in proper identification of results for fake and real image.

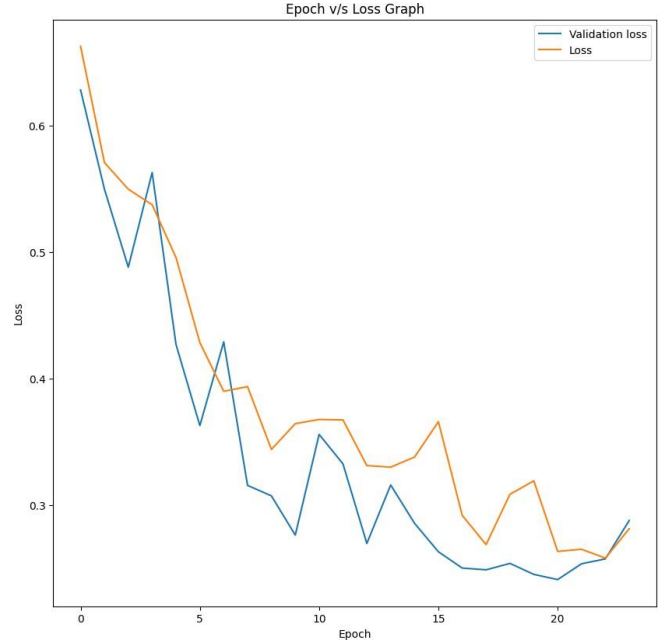


Fig. 3. Representation of Epoch v/s Loss for training.

```

=====
Real label--- real
1/1 [=====] - 0s 277ms/step
Detection label--- REAL
=====
Real label--- real
1/1 [=====] - 0s 30ms/step
Detection label--- REAL
=====
Real label--- fake
1/1 [=====] - 0s 26ms/step
Detection label--- FAKE
=====
Real label--- real
1/1 [=====] - 0s 26ms/step
Detection label--- REAL
=====
Real label--- real
1/1 [=====] - 0s 26ms/step
Detection label--- REAL
=====
Real label--- real
1/1 [=====] - 0s 33ms/step
Detection label--- REAL
=====
Real label--- fake
1/1 [=====] - 0s 31ms/step
Detection label--- FAKE
=====
Real label--- real
1/1 [=====] - 0s 28ms/step
  
```

Fig. 4. Detection results of the model.

##### A. Deployment

The machine learning model was deployed using an open-source Python library called streamlit. Streamlit is used to build and deploy data science and machine learning projects. The user-interface contains a upload option that users can use to identify the originality of the image.

## ILLEGAL IMAGE IDENTIFICATION USING CNN

Upload a image to find whether image is forged or pristine

Upload Images

Drag and drop file here  
Limit 200MB per file • PNG, JPG

Browse files

You can download some sample images by clicking on the below Links :

Link 1

Link 2

Link 3

Click the link below to detect random images.

Generate a random image

Fig. 5. User interface of the web application

### B. Output

The system has two phases, in the first phase, it converts the input image into an Error Level Analysis (ELA) image using the `convert_to_ela_image()` function, and then feeds it into a pre-trained CNN model to predict whether the image is forged or pristine. If the image is predicted to be fake, the second phase is activated, where a second pre-trained CNN model is used to identify the tampered region(s) of the image.



Fig. 6. Final output

## VII. CONCLUSION

In conclusion, training a CNN model on a dataset that includes manipulated and authentic images can be an effective approach for detecting image forgeries. To accurately detect forgeries, the model may then be applied to find patterns and traits that are specific to altered photos. It is important to bear in mind that the quality and quantity of the training dataset can significantly impact the performance, as well as the particular architecture and hyperparameters employed. It's important to continue researching and developing cutting-edge procedures since both detection and picture editing techniques must evolve.

A particularly successful way for identifying manipulated photographs is to use CNNs in conjunction with camera model identification. To recognize patterns and characteristics specific to tampered photos, a CNN may be trained on a dataset of images from a particular camera model after using camera model identification to first reduce the list of possible sources for an image. This method can be very helpful when the camera model and its unique properties are known and the dataset is big enough.

It is imperative to remember that this approach is not perfect and might not perform well when the camera model is unknown or the dataset is tiny. Additionally, it is possible to change an image's metadata to make it seem as though it was taken with a different camera. To assure the accuracy of the forgery detection, it is crucial to combine CNNs, camera model identification, error level analysis, picture quality evaluation, and image-specific feature recognition.

Additionally, the techniques utilized for detection must advance along with picture alteration techniques. As a result, it's critical to carry out the further study, enhance existing approaches, and integrate various methodologies to create a reliable forgery detection system.

## REFERENCES

- [1] Robust camera model identification using demosaicing residual features 2021, Chen, Chen; Stamm, Matthew C.
- [2] Robust Multi-classifier for Camera Model Identification based on Convolution Neural Network 2018, Yao, Hongwei; Qiao, Tong; Xu, Ming; Zheng, Ning.
- [3] First Steps Toward Camera Model Identification with Convolutional Neural Networks 2017, Bondi, Luca; Baroffio, Luca; Guera, David; Bestagini, Paolo; Delp, Edward J.; Tubaro, Stefano.
- [4] Accurate Detection of Demosaicing Regularity for Digital Image Forensics 2009 Hong Cao, Kot, A.C.
- [5] Camera model identification based on the generalized noise model in natural images 2016, Thai, Thanh Hai; Retraint, Florent; Cogranne, Rémi.
- [6] Camera model identification using a deep network and a reduced edge dataset Abdul Muntakim Rafi1, Thamidul Islam Tonmoy2, Uday Kamal3, Q. M. Jonathan Wu1, Md. Kamrul Hasan3.
- [7] RemNet: remnant convolutional neural network for camera model identification Abdul Muntakim Rafi1, Thamidul Islam Tonmoy2, Uday Kamal3, Q. M. Jonathan Wu1, Md. Kamrul Hasan3.
- [8] Suyoto Suyoto, Universitas Atma Jaya Yogyakarta Article in TELKOMNIKA (Telecommunication Computing Electronics and Control) · August 2018