FAKE IMAGE DETECTION

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

Master of Computer Applications

by

Sandeep Kumar

(Regd. No. : - 2020PGCACA75)

Under the supervision of

Dr. Mayukh Sarkar

(Internal Supervisor)

NIT Jamshedpur

Department of Computer Science and Engineering

National Institute of Technology, Jamshedpur

Jamshedpur, India

May, 2023

DECLARATION

I hereby declare that dissertation of the work titled, **"Fake Image Detection"**, submitted towards requirements of project work for partial fulfilment of **Master of Computer Applications (MCA)** is an original work of mine and the report has not formed the basis for the award of any other degree, associate ship, fellowship or similar titles. I also declare that wherever I have used materials such as data, theoretical analysis, and text from other sources, I have given due credit to them by citing source of the work in the thesis.

Sandeep Kumar
2020PGCACA75
MCA6thSemester
Dept. of Computer Science & Engineering
NIT Jamshedpur

Place :

Date :

राष्ट्रीय प्रौद्योगिकी संस्थान जमशेदपुर
NATIONAL INSTITUTE OF TECHNOLOGY, JAMSHEDPUR
(An Institution of National Importance under MHRD, Govt. of India, New Delhi)

Department of Computer Science and Engineering

Ref. NO. : …… Date: ……...

CERTIFICATE

This is to certify that the project work titled **"Fake Image Detection"** is a bonafide work carried out by **Sandeep Kumar** bearing institute registration no. **2020PGCACA75** a student of 6th semester, Master of Computer Applications (MCA), under the Department of Computer Science and Engineering, National Institute of Technology, Jamshedpur. This work has been carried out from **January** to **May.** This project report is submitted in partial fulfilment of the requirement for the award of degree of Master of Computer Applications (MCA), and has been carried out under my supervision.

I wish them all the best in his career and his future endeavour.

Dr Mayukh Sarkar
(Internal Supervisor)
Dept. of Computer Science and Engineering
NIT Jamshedpur

ACKNOWLEDGEMENT

It gives me immense pleasure to express my deep sense of gratitude to my supervisor **Dr Mayukh Sarkar** for his valuable guidance, motivation, constant inspiration and above all for their ever-cooperating attitude that enable me in bringing up this thesis in the present form.

My heartfelt gratitude also goes to **Dr D. A Khan**, Head of Department of Computer Science and Engineering. For providing me the opportunity to avail the excellent facilities and infrastructure. I am equally thankful to all other faculty members and non-teaching staffs of Computer Science and Engineering Department for their guidance and support.

I am also thankful to all my family members whose love, affection, blessings and patience encouraged me to carry out this thesis successfully. I also extend my gratitude to all my friends for their cooperation. I thank Almighty God, my lord for giving me the will power and strength to make it happen.

Lastly, I thank myself for putting for sheer hard work, dedication and perseverance.

Sandeep Kumar

2020PGCACA75

MCA6thSemester

Dept. of Computer Science & Engineering

NIT Jamshedpur

Place :

Date :

PREFACE

In the age of digital media and the widespread use of social media platforms, the sharing of images has become an integral part of our daily lives. While this has opened up new possibilities for communication and expression, it has also given rise to a new challenge: the proliferation of fake images.

Fake images are manipulated or fabricated visuals that are designed to deceive viewers by presenting a false reality. They can be created for a variety of reasons, such as spreading misinformation, propaganda, or even as a form of satire. Detecting and combating fake images is essential for maintaining the integrity of our communication channels and the trust we place in the media.

This book aims to provide a comprehensive overview of the current state of fake image detection, including the techniques used to create fake images and the methods used to detect them. It also explores the ethical considerations surrounding fake images and their impact on society.

We hope that this book will serve as a valuable resource for anyone interested in learning more about fake image detection and contributing to the fight against misinformation in the digital age.

ABSTRACT

In the present generation, social media is a big advantage for an individual to grow. On the other hand, we can't neglect the fact that it is a huge platform for negativity too. With the rapid progress of recent years, techniques that generate and manipulate multimedia content can now provide a very advanced level of realism. The boundary between real and synthetic media has become very thin. On the one hand, this opens the door to a series of exciting applications in different fields such as creative arts, advertising, film production, video games. On the other hand, it poses enormous security threats. Software packages freely available on the web allow any individual, without special skills, to create very realistic fake images and videos. These techniques can be used to manipulate public opinion regarding anything and create chaos. In this paper, we would like to overview few major facts and figures regarding exceeding image forgery techniques that exists and propose a better way on how to detect these forgeries and fakes. **Keywords:** GAN, ELA, Deep Learning, Convolutional Neural Networks, Fake Colorized Image Detection.

TABLE OF CONTENTS

LIST OF FIGURES

FAKE IMAGE DETECTION

Sandeep Kumar

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION TO THE PROJECT

Fake image detection is the process of identifying and flagging images that have been altered or manipulated in a way that deceives viewers or misrepresents the original content. With the widespread availability of digital image editing tools, it has become increasingly easy for individuals to manipulate images for various purposes, including creating fake news, propaganda, and disinformation campaigns. Fake image detection techniques use a variety of methods to analyse images for signs of manipulation, such as changes in lighting, shadows, and pixilation. The development of these techniques is critical for ensuring the integrity of visual media and maintaining public trust in the accuracy of images used in news, advertising, and other forms of media.

1.2 OBJECTIVES

The main objective is to provide the case in managing the config and details are following: -

Module 1: - User Management System

- Establish a user management system to securely store and manage user data, including personal information, login credentials, and permissions.
- Define user roles and permissions for accessing and managing the ELA and CNN models, including roles for data scientists, model developers, and system administrators.

- Implement a user-friendly interface for users to access and manage their own data, including the ability to update their profile information, reset passwords, and view their permissions.
- Enable administrators to efficiently manage user accounts, including creating new accounts, disabling or deleting accounts, and assigning or revoking permissions.
- Ensure the security and privacy of user data, including implementing strong encryption, access controls, and data backups.
- Implement single sign-on (SSO) or multi-factor authentication (MFA) to ensure the security of user accounts.
- Ensure compliance with relevant regulations and standards, such as GDPR, HIPAA, or PCI-DSS.
- Provide comprehensive reporting and auditing capabilities, including tracking user activity, permissions changes, and security incidents.
- Foster a culture of collaboration and communication among the data science, development, and operations teams to ensure that users have access to the resources they need to effectively use the ELA and CNN models.
- Continuously monitor and improve the user management system, including implementing regular software updates and security patches, and incorporating user feedback to improve the user experience.

Module 2: - Config Management

- Establish clear guidelines and policies for managing the configuration of the ELA and CNN models used for fake image detection, including version control, testing procedures, and deployment strategies.
- Define standard configurations for the hardware and software environments used to train and run the models, including operating systems, libraries, and frameworks.
- Implement automated tools for managing the configuration of the ELA and CNN models, including version control systems, continuous integration and deployment (CI/CD) pipelines, and monitoring and alerting systems.
- Establish procedures for testing and validating the ELA and CNN models, including data validation, accuracy testing, and performance testing.

- Define roles and responsibilities for managing the ELA and CNN models, including designating a model owner, model developer, and data scientist responsible for managing the model's configuration, testing, and deployment.
- Ensure that the ELA and CNN models are compliant with relevant regulations and standards, such as GDPR, HIPAA, or PCI-DSS.
- Implement a process for monitoring and updating the ELA and CNN models, including monitoring for changes in the data distribution and updating the models as needed.
- Implement a process for monitoring and responding to model performance issues, including monitoring for accuracy degradation and implementing remediation strategies as needed.
- Foster a culture of collaboration and communication among the data science, development, and operations teams to ensure that the ELA and CNN models are integrated with other IT functions, such as data collection and pre-processing, software development, and model deployment.
- Continuously evaluate and improve the configuration management process for the ELA and CNN models, incorporating feedback from stakeholders and identifying opportunities for automation and optimization.

1.3 LITERATURE SURVEY

A literature survey on fake image detection using ELA and CNN involves reviewing published research papers, articles, and other relevant literature to identify existing approaches, methodologies, and challenges in this field. Here are some key findings from the literature survey:

- ELA is a widely used method for detecting image manipulation by analysing changes in pixel values. It has been shown to be effective in identifying regions of an image that have been edited or tampered with.
- CNNs have shown great promise in detecting fake images, especially deep learning architectures such as ResNet, VGG, and Inception. These models can learn complex features and patterns from images and have achieved high accuracy in various image classification tasks.

- Several studies have explored the use of ELA and CNN in combination for fake image detection. For example, Xiang et al. (2020) proposed a method that uses ELA to pre-process images and then applies a CNN for classification. Their results showed high accuracy in detecting fake images.
- Other studies have focused on using transfer learning to improve the performance of fake image detection models. For example, Zhang et al. (2020) used transfer learning to fine-tune a pre-trained CNN model for fake image detection and achieved better results than using a CNN model trained from scratch.
- Some challenges in fake image detection using ELA and CNN include the need for large amounts of labelled data for training, the potential for adversarial attacks to bypass the models, and the difficulty of detecting subtle and sophisticated manipulations.
- The development of benchmark datasets, such as the Deep Fake Detection Challenge dataset, has facilitated the evaluation and comparison of different fake image detection models.

Overall, the literature survey suggests that ELA and CNNs are effective methods for detecting fake images, and the combination of these methods shows promise for achieving high accuracy in this task. However, there is still much research to be done to address the challenges and limitations of these approaches.

CHAPTER 2

ANALYSIS

2.1 EXISTING SYSTEM

There are several existing systems for detecting fake images. Here are a few examples:

a) **Reverse Image Search:** This technique involves searching for the original source of an image to check if it has been manipulated or altered. By comparing the original and manipulated images, it is possible to detect the changes made to the image.

b) **Digital Forensics:** This technique involves analysing the metadata of an image, such as the date, time, and location of the image, to check for any inconsistencies. Digital forensics can also be used to detect traces of manipulation in the image.

c) **Machine Learning:** This technique involves training machine learning algorithms to detect fake images. The algorithms are trained on a dataset of real and fake images, and they learn to identify patterns that distinguish between the two. One example of a machine learning-based system is the DeepFake Detection Challenge, which was created to develop algorithms to detect deepfakes.

d) **Blockchain Technology:** This technique involves using blockchain technology to track the authenticity of an image. The blockchain can be used to store information about the original image, such as its metadata, and any changes made to the image can be recorded on the blockchain, making it possible to track the image's authenticity.

e) **Human Analysis:** This technique involves having humans analyse the image to check for any signs of manipulation. Human analysis can be time-consuming and subjective, but it can be useful in cases where other methods of detection are not effective.

Overall, there is no single technique that can detect all fake images, and a combination of techniques may be needed to effectively detect fake images.

2.2 DRAWBACK OF EXISTING SYSTEM

The use of Error Level Analysis (ELA) and Convolutional Neural Networks (CNN) for detecting fake images has some drawbacks, including:

a) Limited effectiveness against advanced image manipulation techniques: While ELA and CNN are effective at detecting certain types of image manipulation, they may not be able to detect more advanced techniques such as deepfakes, which use machine learning to create highly realistic images and videos.
b) High false positive rates: ELA and CNN can sometimes mistakenly flag authentic images as fake, leading to high false positive rates. This can be particularly problematic in situations where the consequences of a false positive could be severe, such as in legal cases.
c) Need for large amounts of training data: CNNs require large amounts of training data to effectively distinguish between real and fake images. This can be a challenge in situations where there is a limited amount of available training data or where the distribution of fake images is vastly different from that of real images.
d) Time-consuming process: The process of training a CNN to detect fake images can be time-consuming and resource-intensive, requiring significant computational power and expertise.

Overall, while ELA and CNN can be effective tools for detecting certain types of fake images, they are not fool proof and may not be effective against more advanced techniques. It's important to consider the limitations and potential drawbacks of these approaches before relying on them as the sole means of detecting fake images.

2.3 PROPOSED SYSTEM

Fake image detection builds upon the existing system and aims to provide an even better and more stable method for detecting the fake images.

2.4 PROPOSED TECHNICAL REQUIREMENT

1. Software Requirements

Operating System	Windows 10
Programming Language	Python
Technology	Machine Learning
Tools	Jupyter Notebook

2. Web Browsers

Browsers	**Requirements**
Chrome	Version 66 and above
Firefox	Version 66 and above
Safari	Version 10.1 and above
Edge	Version 41 and above

2.5 PROJECT PLANNING AND PROJECT SCHEDULING

2.5.1 Project Planning

After the project has been defined and the project team has been appointed, the project moves on to the second phase in the Project Management Life Cycle (PMLC): the detailed project planning phase. Project planning is at the heart of the project life cycle that tells everyone

to get involved that know where it is going and how it is going. In the planning phase, the project plans are documented, the project deliverables and requirements are defined, and the project schedule is created. It involved the creation of a set of plans to guide the team through the implementation and closure phases of the project. The plans are created during this phase are to manage time, cost, quality, changes, risk, and related issues. This helped to control staff and external suppliers to ensure the delivery of the project on time, within budget, and within schedule. Based on that the project is planned so that on the estimated date the project is accomplished and proper documentation is completed.

2.5.2 PROJECT CONSTRAINTS

- Scope-Project outcome as defined in the contract
- Timeline- Important milestones and client-imposed completion dates
- Budget -Funding limits are based upon the time that is required and imposed by the sponsor i.e., the Client Partner.
- Quality- Agreed quality metrics with Customers or conformance with internal quality standards
- Resources- Availability of skilled human resources or materials
- Risks- Uncertainties associated with the project

For example, a constraint on the timeline may force the project to speed up and skip certain tasks. This may impact the time frame for quality checking and force the project team to compromise on the quality of the project outcome if they have to stick to the delivery date. It is not necessary that the project constraints should always be imposed by customers/sponsors. Some of the constraints can be attributed to the external environment.

2.5.3 PROJECT APPROACH

Knowing the overall approach of a project is an important part of understanding how involvement is done and how others are made to be involved, such as the project team and business stakeholders. How to choose the right approach for a project is a large topic in itself. The methodology chosen depends on many things, including the structure and location of the project team, the technologies being used on the project, and the degree to which collaboration is a part of the company's culture. Her focus is on done on two of the most common types of approaches. The important thing to note is that most approaches involve the same steps:

- Plan the overall strategy, approach, and team structure
- Define the project requirements.
- Design interaction and visual concepts and evolve them into detailed specifications.
- Develop, test, and refine the solution.
- Deploy the solution via messaging, training, and a planned launch.
- Extend the project by making recommendations for improvements.

WATERFALL APPROACH

A waterfall approach involves treating the steps of a project as separate, distinct phases, where approval of one phase is needed before the next phase begins. For example, the Design phase does not begin in earnest until requirements have been approved by business stakeholders, who sign off on one or more requirements documents at the end of the Define phase.

AGILE APPROACH

Because change is constant, project teams are continually looking for more flexible approaches than the waterfall model. Many methodologies follow a more fluid approach, with some steps happening alongside each other; for example, versions of the website could be released on a rapid, iterative schedule using agile or rapid development. An agile approach generally has a greater focus on rapid collaboration and a reduced focus on detailed documentation and formal sign-off.

2.5.4 PROJECT COMMUNICATION

The Project required a sound communication plan, but not as the other projects having the same types of communication or the same methods for distributing the information. The communication plan documented the types of information needed from the stakeholders such as when the information should be distributed, and how the information will be delivered. The types of information which will be going to communicate typically include project status, project scope statements and updates, project baseline information, risks, action items, performance measures, project acceptance, and so on. It's important that the information needed from the stakeholders should be determined as early in the planning phase of the project management life cycle as possible so that project planning documents can be developed, who should receive copies of them, and how they should be delivered is taken into consideration.

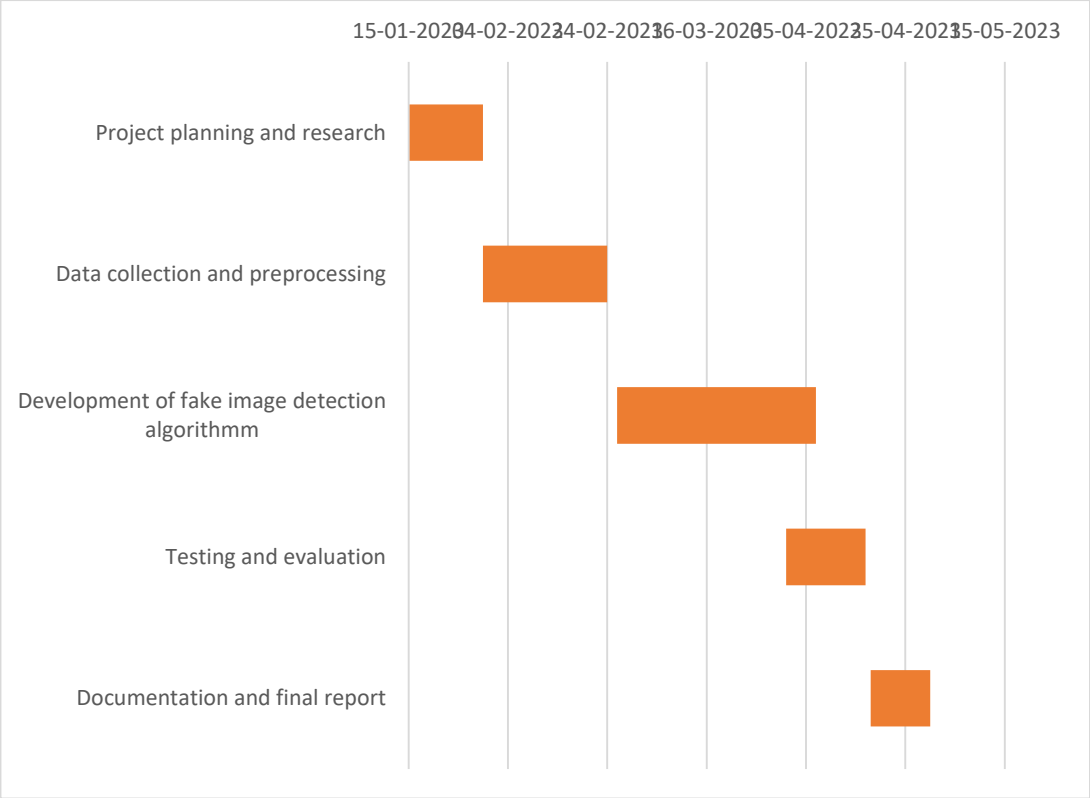
2.5.5 PROJECT SCHEDULING

- Project planning and research
- Data collection and pre-processing
- Development of fake image detection algorithm.
- Testing and evaluation.

- Documentation and final report.

2.5.6 GANTT CHART

A Gantt chart, commonly used in project management, is one of the most popular and useful ways of showing activities (tasks or events) displayed against time. On the left of the chart is a list of the activities and along the top is a suitable time scale. Each activity is represented by a bar; the position and length of the bar reflect the start date, duration, and end date of the activity. This allows you to see at a glance:



2.6 SOFTWARE REQUIREMENT SPECIFICATIONS (SRS)

2.6.1 DOCUMENT PURPOSE

The purpose of the Software Requirement Specification is to outline the requirements for the enhancement in the live business and the innovative ideas project as the sample for future inclusion in live business as well as organizational purpose. When live business is concerned it is basically

based on the tools and the framework that the organization is been working on cloud-based applications for Planning, Consolidation, Reporting, and Analytics.

2.6.2 PRODUCT

SCOPE:

The following key requirements covered under the project are as follows:

1. **Demo Project**

- Providing a new feature.
- Removing the time consumption.
- Removing the Technical guy.

2. **Innovation Idea**

- Using new algorithm like ELA and CNN

2.6.3 PRODUCT PERSPECTIVE

The Key purpose of this project is to remove the technical person and to make the workflow smoother. Any user can work as per assigned roles.

2.6.4 PRODUCT FUNCTIONALITY

1. Functional Requirement

- **Image analysis:** The system should be able to analyse the image to determine whether it is fake or not. This can include examining the metadata of the image, such as its source and date of creation, as well as analysing the image itself to detect any signs of manipulation.
- **Machine learning:** The system should use machine learning algorithms to learn from a large dataset of real and fake images, in order to improve its accuracy over time. This can involve techniques such as deep learning, convolutional neural networks, and other forms of artificial intelligence.

- **Detection accuracy:** The system should be able to accurately detect fake images with a high level of precision and recall, and should be able to identify the specific type of manipulation that has been applied to the image.
- **Real-time detection:** The system should be able to detect fake images in real-time, allowing for rapid identification and response to potential threats.
- **Scalability:** The system should be able to handle large volumes of images, as well as be easily scalable to accommodate future growth in image data.
- **User interface:** The system should provide a user-friendly interface that allows users to easily upload and analyse images, as well as view reports and insights generated by the system.
- **Integration:** The system should be easily integrated with other software applications and systems, such as social media platforms or content management systems, in order to provide comprehensive image analysis and detection capabilities.

2. Non-Functional Requirements

- **Accuracy:** The system should have a high accuracy rate in detecting fake images. The system should be able to detect the fake images with a high level of accuracy, preferably with a low false-positive rate.
- **Speed:** The system should be able to process images quickly and efficiently, especially when dealing with large volumes of images. The speed of image processing should be sufficient to detect fake images in real-time applications.
- **Scalability:** The system should be able to scale up or down as per the requirement. It should be able to handle the increasing amount of images with the same accuracy rate.

- **Robustness:** The system should be able to detect fake images irrespective of the image's quality, size, or format.
- **Security:** The system should be able to maintain data privacy and security while processing images.

2.6.5 USER & CHARACTERISTICS

- **Developers-** From this document, developers would be able to know what all changes or enhancements are done. If any further enhancement is to be done it would give quite an idea about it.
- **Policy Makers-** Can use it for analysis purposes so that in the future to the development of new systems or enhancement work can be carried out later.

2.6.6 TOOLS & TECHNOLOGY

The tech stack for the application includes:

Programming languages: Python programming languages for developing image processing algorithms.

Deep learning frameworks: TensorFlow, Keras, and PyTorch are widely used deep learning frameworks for developing deep neural networks for fake image detection.

Image processing libraries: OpenCV and Pillow are popular image processing libraries that can be used for manipulating images.

Cloud platforms: Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform (GCP) offer cloud-based image processing services that can be used for developing fake image detection systems.

Data storage and processing tools: Hadoop, Apache Spark, and Apache Kafka can be used for storing and processing large volumes of image data.

Visualization tools: Tableau, Power BI, and D3.js can be used for visualizing the results of the fake image detection system.

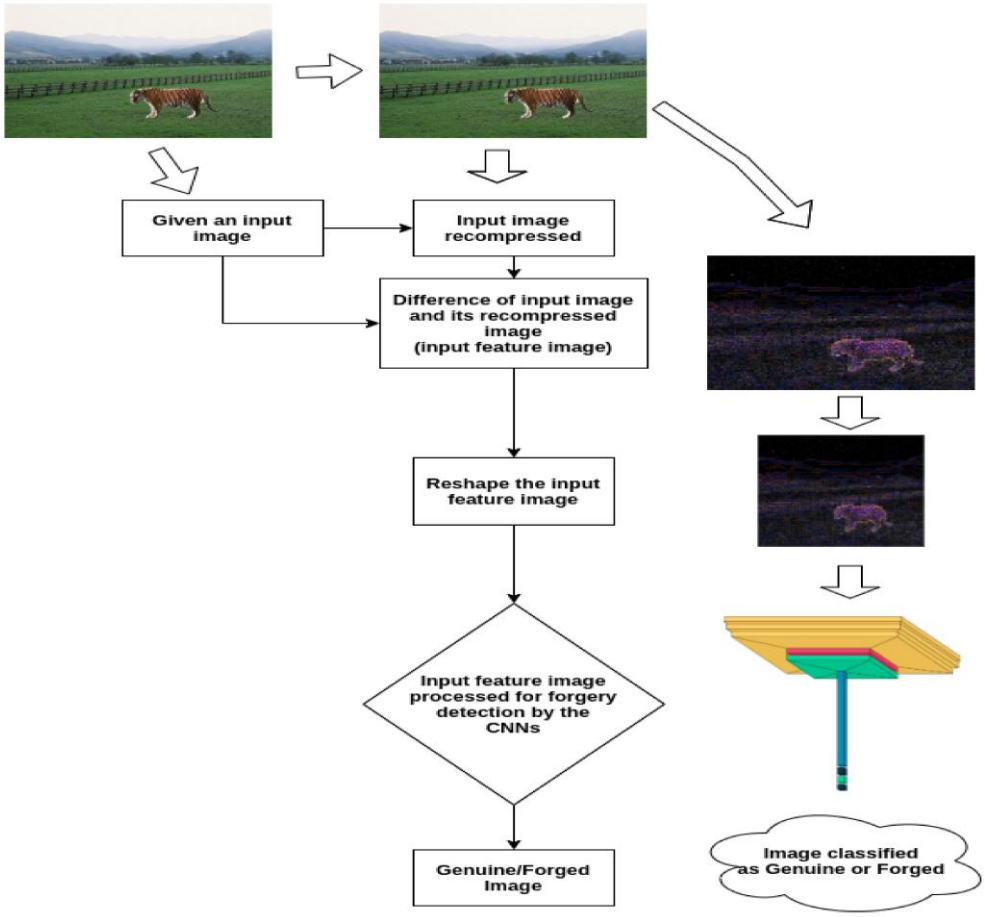
APIs: Various APIs such as Google Vision API, Microsoft Cognitive Services, and Amazon Rekognition can be used for developing fake image detection systems.

Other tools and libraries: Some other useful tools and libraries include NumPy, SciPy, Matplotlib, Pandas, Scikit-learn, and Flask.

2.7 DATA MODEL, DATA FLOW DIAGRAM & OTHER SIMILAR MODELS & DICTIONARY

2.7.1 FLOW CHARTS

A flowchart is a type of diagram that represents a workflow or process. A flowchart can also be defined as a step-by-step approach to solving a task.



2.7.2 DATA FLOW DIAGRAMS

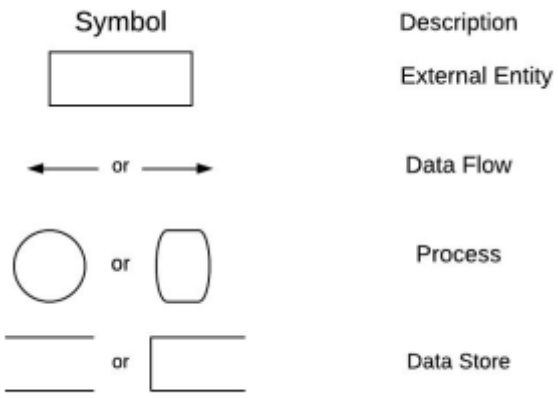
DFDs are the model of the proposed system. They clearly should show the requirements on which the new system should be built. Later during design activity this is taken as the basis for drawing the system's structure charts. The **Basic Notation** used to **create a DFD's** are as follows:

1. Data Flow: Data move in a specific direction from an origin to a destination.

2. Process: People, procedures, or devices that use or produce (Transform) Data. The physical component is not identified.

3. External Entity (Source or Destination): External sources or destination of data, which may be People, programs, organizations or other entities.

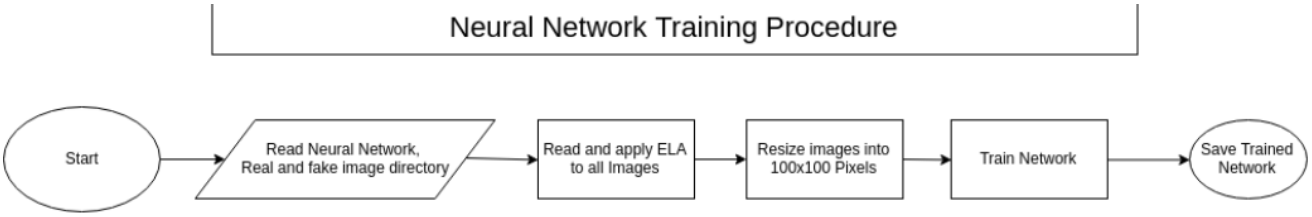
4. Data Store: A data store represents the storage of persistent data required and/or produced by the process. Here are some examples of data stores: membership forms, database tables, etc.



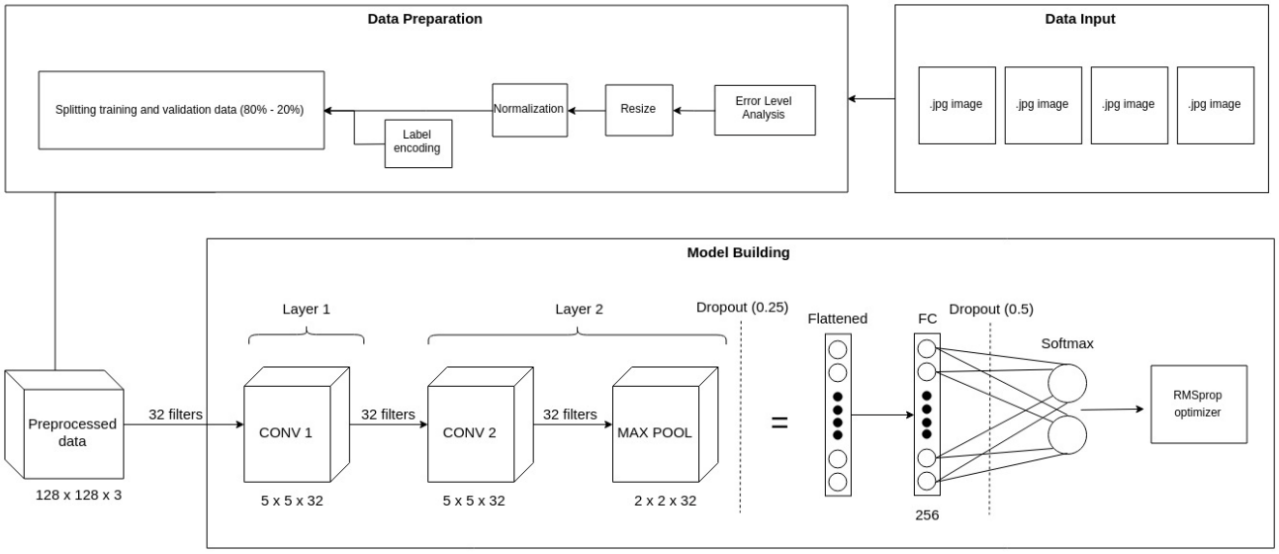
TYPES OF DFD's ARE: -

1. **0-Level DFD**
2. **1-Level DFD**

Level 0



Level 1

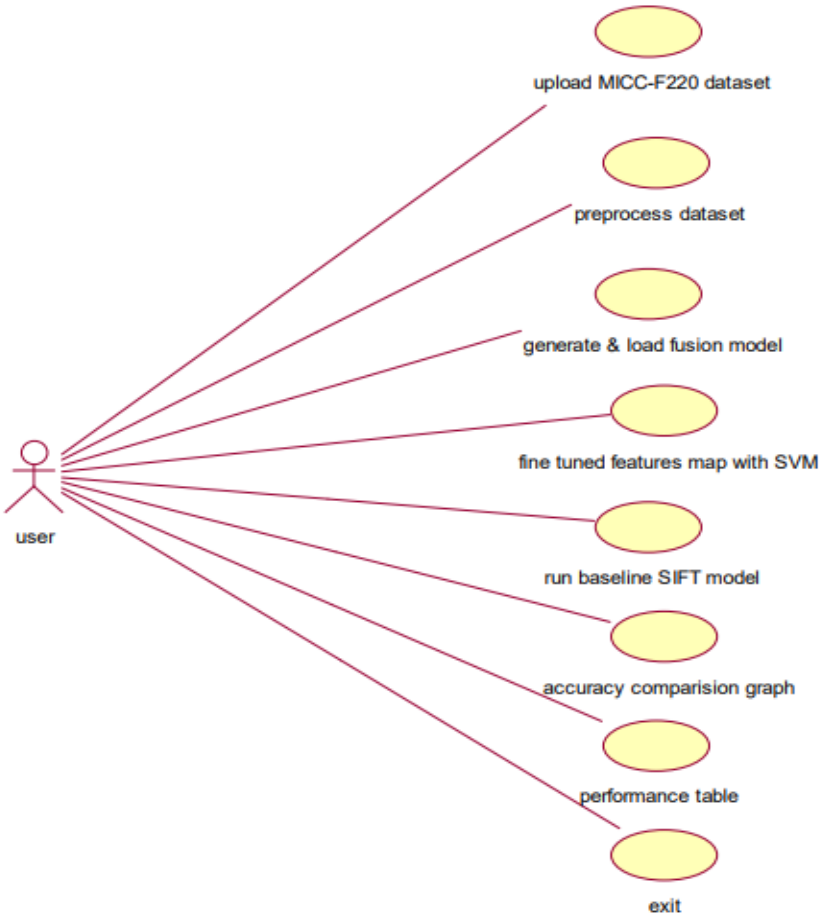


Chapter 3

DESIGN

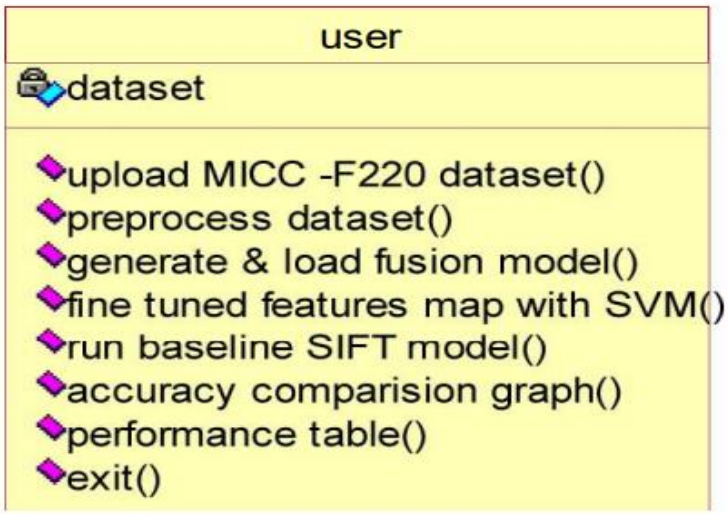
3.1 USE CASE DIAGRAM

A use case diagram in the Unified Modelling Language (UML) is a type of behavioural diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.



3.2 CLASS DIAGRAM

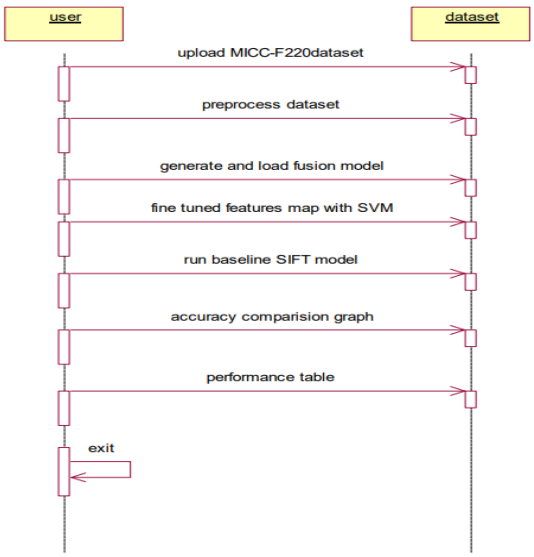
In software engineering, a class diagram in the Unified Modelling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It



explains which class contains information.

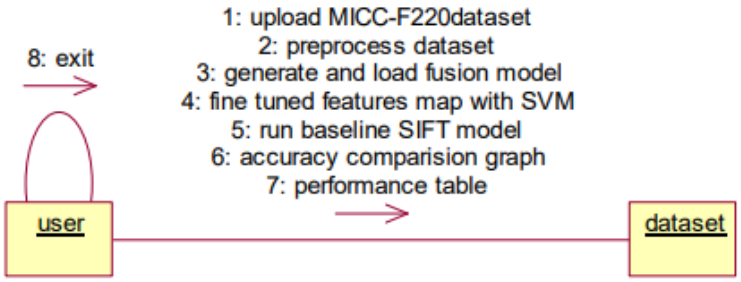
3.3 SEQUENCE DIAGRAM

A sequence diagram in Unified Modelling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.



3.4 COLLABRATION DIAGRAM

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modelling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.



IMPLEMENTATION MODULES

1) Upload MICC-F220 Dataset: using this module we will upload application.

2) Pre-process Dataset: using this module we will read all images and normalize their pixel values and then resize them to equal size

3) Generate & Load Fusion Model: using this module we will train 3 algorithms called Squeeze Net, MobileNetV2 and Shuffle Net and then extract features from it to train fusion model. All algorithms prediction accuracy will be calculated on test data

4) Fine Tuned Features Map with SVM: using this module we will extract features from all 3 algorithms to form a fusion model and then fusion data get trained with SVM and then calculate its prediction accuracy.

5) Run Baseline SIFT Model: using this module we will extract SIFT existing technique features from images and then train with SVM and get its prediction accuracy

6) Accuracy Comparison Graph: using this module we will plot accuracy graph of all algorithms

7) Performance Table: using this module we will display all algorithms performance table.

Chapter 4

CODING

4.1 Importing all the libraries

The libraries are pandas, numpy, matplotlib, seaborn, sklearn, itertools and keras

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import seaborn as sns
%matplotlib inline

np.random.seed(2)

from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
import itertools

from keras.utils.np_utils import to_categorical # convert to one-hot-encoding
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPool2D
from keras.optimizers import RMSprop
from keras.preprocessing.image import ImageDataGenerator
from keras.callbacks import ReduceLROnPlateau, EarlyStopping

sns.set(style='white', context='notebook', palette='deep')
```

4.2 Initial Preparation

```
from PIL import Image
import os
from pylab import *
import re
```

```
from PIL import Image, ImageChops, ImageEnhance
```

4.2.1 Limit Function

```
def get_imlist(path):
    return [os.path.join(path,f) for f in os.listdir(path) if f.endswith('.jpg')
or f.endswith('.png')]
```

4.2.2 ELA Conversion

```
def convert_to_ela_image(path, quality):
    filename = path
    resaved_filename = filename.split('.')[0] + '.resaved.jpg'
    ELA_filename = filename.split('.')[0] + '.ela.png'

    im = Image.open(filename).convert('RGB')
    im.save(resaved_filename, 'JPEG', quality=quality)
    resaved_im = Image.open(resaved_filename)

    ela_im = ImageChops.difference(im, resaved_im)

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

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

    return ela_im
```

4.3 Input and Conversion

4.3.1 Real Image Input and ELA Conversion

```
Image.open('kaggle/input/casia-
dataset/CASIA2/Au/Au_ani_00001.jpg')
```

```
In [5]: Image.open('kaggle/input/casia-dataset/CASIA2/Au/Au_ani_00001.jpg')
```

Out[5]:

convert_to_ela_image('kaggle/input/casia-dataset/CASIA2/Au/Au_ani_00001.jpg', 90)

```
In [6]: convert_to_ela_image('kaggle/input/casia-dataset/CASIA2/Au/Au_ani_00001.jp
```

Out[6]:

4.3.2 Fake Image Input and ELA Conversion

Image.open('kaggle/input/casia-dataset/CASIA2/Tp/Tp_D_NRN_S_N_ani10171_ani00001_12458.jpg')

```
In [7]: Image.open('kaggle/input/casia-dataset/CASIA2/Tp/Tp_D_NRN_S_N_ani10171_ani
```

Out[7]:

convert_to_ela_image('kaggle/input/casia-dataset/CASIA2/Tp/Tp_D_NRN_S_N_ani10171_ani00001_12458.jpg', 90)

```
In [8]: convert_to_ela_image('kaggle/input/casia-dataset/CASIA2/Tp/Tp_D_NRN_S_N_ar
```

Out[8]:

4.4 Data Prepration

4.4.1 Image Prepration

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

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

4.4.2 Image Processing

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

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

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

print(len(X), len(Y))
```

```
X = np.array(X)
Y = to_categorical(Y, 2)
X = X.reshape(-1, 128, 128, 3)

X_train, X_val, Y_train, Y_val = train_test_split(X, Y, test_size = 0.2,
random_state=5)
X = X.reshape(-1,1,1,1)
print(len(X_train), len(Y_train))
print(len(X_val), len(Y_val))
```

4.5 CNN Model

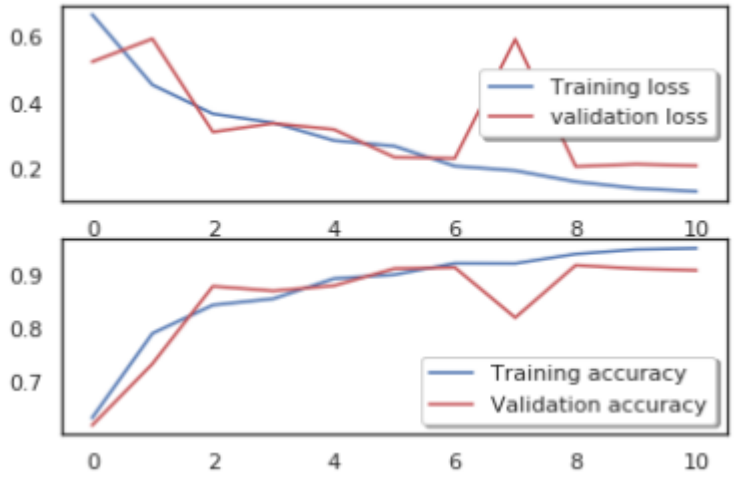
```
def build_model():
    model = Sequential()
    model.add(Conv2D(filters = 32,
                     kernel_size = (5, 5),
                     padding = 'valid',
                     activation = 'relu',
                     input_shape = (128, 128, 3)))

    model.add(MaxPool2D(pool_size = (2, 2)))

    model.add(Conv2D(filters = 32,
                     kernel_size = (5, 5),
                     padding = 'valid',
                     activation = 'relu',
                     input_shape = (128, 128, 3)))

    model.add(MaxPool2D(pool_size = (2, 2)))

    model.add(Dropout(0.25))

    model.add(Flatten())

    model.add(Dense(256, activation = 'relu'))

    model.add(Dropout(0.5))

    model.add(Dense(2, activation = 'softmax'))

    return model
```

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

```
In [16]: model = build_model()
         model.summary()

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

          max_pooling2d (MaxPooling2D  (None, 62, 62, 32)       0
          )

          conv2d_1 (Conv2D)            (None, 58, 58, 32)        25632

          max_pooling2d_1 (MaxPooling  (None, 29, 29, 32)       0
          2D)

          dropout (Dropout)            (None, 29, 29, 32)        0

          flatten (Flatten)            (None, 26912)             0

          dense (Dense)                (None, 256)               6889728

          dropout_1 (Dropout)          (None, 256)               0

          dense_1 (Dense)              (None, 2)                 514

         =================================================================
         Total params: 6,918,306
         Trainable params: 6,918,306
         Non-trainable params: 0
         _________________________________________________________________
```

4.6 Accuracy and loss curves during training-validation

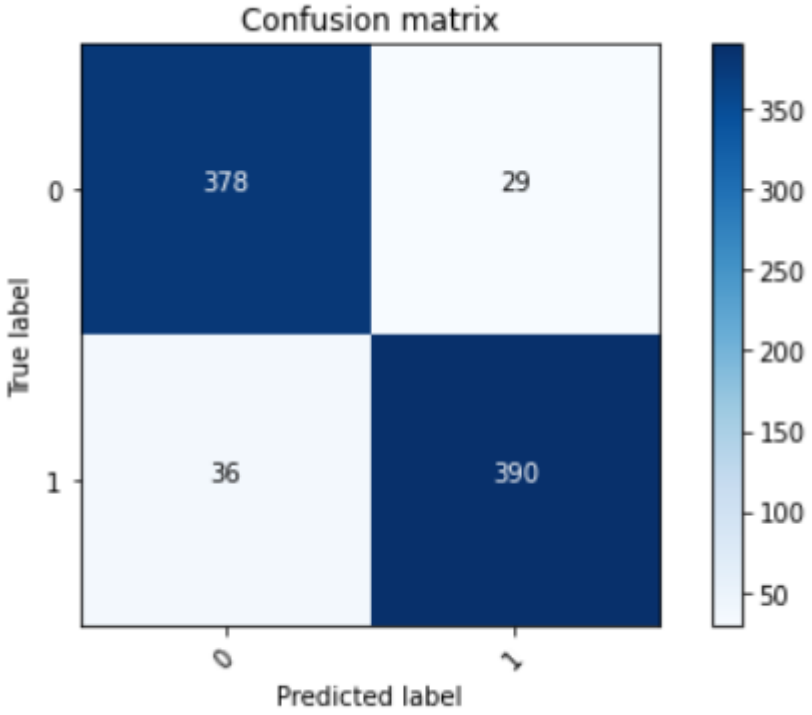
```
# Plot the loss and accuracy curves for training and validation
fig, ax = plt.subplots(2,1)
ax[0].plot(history.history['loss'], color='b', label="Training loss")
ax[0].plot(history.history['val_loss'], color='r', label="validation loss",axes =ax[0])
legend = ax[0].legend(loc='best', shadow=True)

ax[1].plot(history.history['accuracy'], color='b', label="Training accuracy")
ax[1].plot(history.history['val_accuracy'], color='r',label="Validation accuracy")
legend = ax[1].legend(loc='best', shadow=True)
```



4.7 Confusion Matrix

```
  def plot_confusion_matrix(cm, classes,
                            normalize=False,
                            title='Confusion matrix',
                            cmap=plt.cm.Blues):
"""
This function prints and plots the confusion matrix.
Normalization can be applied by setting `normalize=True`.
"""
plt.imshow(cm, interpolation='nearest', cmap=cmap)
plt.title(title)
plt.colorbar()
tick_marks = np.arange(len(classes))
plt.xticks(tick_marks, classes, rotation=45)
plt.yticks(tick_marks, classes)

if normalize:
    cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]

thresh = cm.max() / 2.
for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
    plt.text(j, i, cm[i, j],
             horizontalalignment="center",
             color="white" if cm[i, j] > thresh else "black")

plt.tight_layout()
plt.ylabel('True label')
```

```
    plt.xlabel('Predicted label')


# Predict the values from the validation dataset
Y_pred = model.predict(X_val)
# Convert predictions classes to one hot vectors
Y_pred_classes = np.argmax(Y_pred,axis = 1)
# Convert validation observations to one hot vectors
Y_true = np.argmax(Y_val,axis = 1)
# compute the confusion matrix
confusion_mtx = confusion_matrix(Y_true, Y_pred_classes)
# plot the confusion matrix
plot_confusion_matrix(confusion_mtx, classes = range(2))
```



Chapter 5

SOFTWARE ENVIRONMENT

5.1 What is Python?

Below are some facts about Python.

Python is currently the most widely used multi-purpose, high-level programming language.

Python allows programming in Object-Oriented and Procedural paradigms. Python programs generally are smaller than other programming languages like Java.

Programmers have to type relatively less and indentation requirement of the language, makes them readable all the time.

Python language is being used by almost all tech-giant companies like – Google,

Amazon, Facebook, Instagram, Dropbox, Uber… etc.

The biggest strength of Python is huge collection of standard library which can be used for the following –

- Machine Learning
- GUI Applications (like Kivy, Tkinter, PyQt etc.)
- Web frameworks like Django (used by YouTube, Instagram, Dropbox)
- Image processing (like Opencv, Pillow)
- Web scraping (like Scrapy, BeautifulSoup, Selenium)
- Test frameworks
- Multimedia

5.2 Advantage of Python

1. **Easy to Learn and Use:** Python is a high-level language, which means it's designed to be easily understood by humans. Its syntax is simple and intuitive, making it an ideal language for beginners to learn.
2. **Open Source:** Python is free to use, distribute, and modify, making it an accessible language for developers with different budgets.
3. **Versatile:** Python is a versatile language that can be used for various purposes such as web development, scientific computing, data analysis, artificial intelligence, and more.
4. **Large Standard Library:** Python comes with a large standard library that contains a vast range of modules and functions for various tasks. This makes it easy for developers to write complex programs without having to reinvent the wheel.
5. **Cross-Platform:** Python is a cross-platform language, meaning that code written in Python can run on different operating systems such as Windows, Mac, and Linux.
6. **High-Level Language:** Python is a high-level language that abstracts away low-level details such as memory management, making it easier for developers to focus on solving problems.
7. **Great Community Support:** Python has a large and active community of developers who contribute to the language by creating libraries, modules, and tools. This community provides support and resources for beginners and experts alike.
8. **Scalable:** Python is a scalable language that can handle large-scale projects with ease. It's used by big companies such as Google, Facebook, and Netflix for their large-scale applications.
9. **Readable Code:** Python emphasizes readability and clean code, making it easier for developers to understand and maintain code written by others.
10. **Easy to Integrate:** Python can be easily integrated with other languages and technologies such as C, C++, Java, and .NET. This makes it a great language for building complex systems that require integration with multiple technologies.

5.3 Advantage of Python over other languages

1. Less Coding

Almost all of the tasks done in Python requires less coding when the same task is done in other languages. Python also has an awesome standard library support, so you don't have to search for any third-party libraries to get your job done. This is the reason that many people suggest learning Python to beginners.

2. Affordable

Python is free therefore individuals, small companies or big organizations leverage the free available resources to build applications. Python is popular and widely used so it gives you better community support.

The 2019 Github annual survey showed us that Python has overtaken Java in the most popular programming language category.

3 . Python is for everyone

Python code can run on any machine whether it is Linux, Mac or Windows. Programmers need to learn different languages for different jobs but with Python, you can professionally build web apps, perform data analysis and machine learning, automate things, do web scraping and also build games and powerful visualizations. It is an all-rounder programming language.

5.4 Disadvantages of Python

So far, we've seen why Python is a great choice for your project. But if you choose it, you should be aware of its consequences as well. Let's now see the downsides of choosing Python over another language.

1. Speed Limitations

We have seen that Python code is executed line by line. But since Python is interpreted, it often results in slow execution. This, however, isn't a problem unless speed is a focal point for the project. In other words, unless high speed is a requirement, the benefits offered by
Python are enough to distract us from its speed limitations.

2. Weak in Mobile Computing and Browsers

While it serves as an excellent server-side language, Python is much rarely seen on the clientside. Besides that, it is rarely ever used to implement smartphone-based applications. One such application is called Carbonnelle.
The reason it is not so famous despite the existence of Brython is that it isn't that secure.

3. Design Restrictions

As you know, Python is dynamically-typed. This means that you don't need to declare the type of variable while writing the code. It uses duck-typing. But wait, what's that? Well, it just means that if it looks like a duck, it must be a duck. While this is easy on the programmers during coding, it can raise run-time errors.

4. Underdeveloped Database Access Layers

Compared to more widely used technologies like JDBC (Java DataBase Connectivity) and ODBC (Open DataBase Connectivity), Python's database access layers are a bit underdeveloped. Consequently, it is less often applied in huge enterprises.

5. Simple

No, we're not kidding. Python's simplicity can indeed be a problem. Take my example. I don't do Java, I'm more of a Python person. To me, its syntax is so simple that the verbosity of Java code seems unnecessary.

This was all about the Advantages and Disadvantages of Python Programming Language.

5.5 What is Machine Learning?

Before we take a look at the details of various machine learning methods, let's start by looking at what machine learning is, and what it isn't. Machine learning is often categorized as a subfield of artificial intelligence, but I find that categorization can often be misleading at first brush. The study of machine learning certainly arose from research in this context, but in the data science application of machine learning methods, it's more helpful to think of machine learning as a means of building models of data.

Fundamentally, machine learning involves building mathematical models to help understand data. "Learning" enters the fray when we give these models tunable parameters that can be adapted to observed data; in this way the program can be considered to be "learning" from the data. Once these models have been fit to previously seen data, they can be used to predict and understand aspects of newly observed data. I'll leave to the reader the more philosophical digression regarding the extent to which this type of mathematical, model-based "learning" is similar to the "learning" exhibited by the human brain. Understanding the problem setting in machine learning is essential to using these tools effectively, and so we will start with some broad categorizations of the types of approaches we'll discuss here.

Categories of Machine Learning

At the most fundamental level, machine learning can be categorized into two main types:

Supervised learning involves somehow modeling the relationship between measured features of data and some label associated with the data; once this model is determined, it can be used to apply labels to new, unknown data. This is further subdivided into classification tasks and

regression tasks: in classification, the labels are discrete categories, while in regression, the labels are continuous quantities. We will see examples of both types of supervised learning in the following section.

Unsupervised learning involves modeling the features of a dataset without reference to any label, and is often described as "letting the dataset speak for itself." These models include tasks such as clustering and dimensionality reduction. Clustering algorithms identify distinct groups of data, while dimensionality reduction algorithms search for more succinct representations of the data. We will see examples of both types of unsupervised learning in the following section.

Need for Machine Learning

Human beings, at this moment, are the most intelligent and advanced species on earth because they can think, evaluate and solve complex problems. On the other side, AI is still in its initial stage and haven't surpassed human intelligence in many aspects. Then the question is that what is the need to make machine learn? The most suitable reason for doing this is, "to make decisions, based on data, with efficiency and scale".

Lately, organizations are investing heavily in newer technologies like Artificial Intelligence,

Machine Learning and Deep Learning to get the key information from data to perform several real-world tasks and solve problems. We can call it data-driven decisions taken by machines, particularly to automate the process. These data-driven decisions can be used, instead of using programing logic, in the problems that cannot be programmed inherently. The fact is that we can't do without human intelligence, but other aspect is that we all need to solve real-world problems with efficiency at a huge scale. That is why the need for machine learning arises.

5.6 Packages used in project

Tensorflow

TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. It is used for both research and production at Google.
TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache 2.0 open-source license on November 9, 2015.

NumPy

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

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

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

Pandas

Pandas an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem.

Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data load, prepare, manipulate, model, and analyze. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

Matplotlib

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter Notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery.
For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object oriented interface or via a set of functions familiar to MATLAB users.

Scikit – learn

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use.

Chapter 6

SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

6.1 TEST REPORTS

TYPES OF TESTS

Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of component.

FUNCTIONAL TEST

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input: identified classes of valid input must be accepted.
Invalid Input: identified classes of invalid input must be rejected.
Functions: identified functions must be exercised.
Output: identified classes of application outputs must be exercised.
Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

System Testing

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is used to test areas that cannot be reached from a black box level.

Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box you cannot "see" into it. The test provides inputs and responds to outputs without considering how the software works.

Unit Testing

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

6.2 TEST STRATEGY AND APPROACH

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.
The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level –interact without error.
Test Results: All the test cases mentioned above passed successfully. No defects encountered.

Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered

Chapter 7

CONCLUSION & FUTURE WORK

7.1 Conclusion

The increased availability of cameras has made photography popular in recent years. Images play a crucial role in our lives and have evolved into an essential means of conveying information since the general public quickly understands them. There are various tools accessible to edit images; these tools are primarily intended to enhance images; however, these technologies are frequently exploited to forge the images to spread misinformation. As a result, image forgery has become a significant problem and a matter of concern. In this paper, we provide a unique image forgery detection system based on neural networks and deep learning, emphasizing the CNN architecture approach. To achieve satisfactory results, the suggested method uses a CNN architecture that incorporates variations in image compression. We use the difference between the original and recompressed images to train the model. The proposed technique can efficiently detect image splicing and copy-move types of image forgeries. The experiments results are highly encouraging, and they show that the overall validation accuracy is 95.23%, with a defined iteration limit.

7.2 Future Scope

We plan to extend our technique for image forgery localization in the future. We will also combine the suggested technique with other known image localization techniques to improve their performance in terms of accuracy and reduce their time complexity. We will enhance the proposed technique to handle spoofing [50] as well. The present technique requires image resolution to be a minimum of 128 × 128, so we will enhance the proposed technique to work well for tiny images. We will also be developing a challenging extensive image forgery database to train deep learning networks for image forgery detection.

References:

[1] Özgöbek, Özlem, J. A. Gulla, “Towards an Understanding of Fake News”, Norwegian Big Data Symposium (2017).

[2] Kshetri, Nir, J. Voas, “The Economics of ‘Fake News’”, IT Pro (November/December 2017), IEEE Computer Society.

[3] Chinese Academy of Science. “CASIA Image Tempering Detection Evaluation Database (CAISA TIDE) V2.0. Diambil dari http://forensics.idealtest.org

[4] N. Krawetz, "A pictures worth digital image analysis and forensics," Black Hat Briefings, hlm. 1-31, 2007.

[5] Rawat, Waseem, Z. Wang, “Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review”, Neural Computation 29 (2017), hlm. 2352-2449.

[6] Gunawan, Teddy Surya, Hanafiah, S. A. M., Kartiwi, M., Ismail, N., Za’bah, N. F., Nordin, A. N., “Development of Photo Forensics Algorithm by Detecting Photoshop Manipulation Using Error Level Analysis”, Indonesian Journal of Electrical Engineering and Computer Science, Vol. 7, No. 1, Juli 2017, hlm. 131-137.

[6] Photo Forensics: Detect Photoshop Manipulation with Error Level Analysis, September 2018. Diambil dari https://resources.infosecinstitute.com/error-level-analysis-detect-i mage-manipulation/#gref

[7] Edelman, “2018 Edelman Trust Barometer Global Report”, diambil dari https://cms.edelman.com/sites/default/files/2018-01/2018%20Edel man%20Trust%20Barometer%20Global%20Report.pdf

[8] Bullas, Jeff. “6 Powerful Reasons Why you should include Images in your Marketing”, diambil dari https://www.jeffbullas.com/6-powerful-reasons-why-you-should-in clude-images-in-your-marketing-infographic/

[9] V. Nair and G. E. Hinton. “Rectified linear units improve restricted boltzmann machines,” Proceedings of the 27th International Conference on Machine Learning, 21-24 Juni 2010, hlm. 807-814.

[10] Villan, M. Afsal, Kuruvilla, K., Paul, J., Elias, E. P., “Fake Image Detection Using Machine Learning”, International Journal of Computer Science and Information Technology & Security (IJCSITS), Vol. 7, No. 2, 2017.