IMAGE MANPULATION DETECTION BY TWO STEP APPROACH OF COMBINING CNN AND ELA

A PROJECT REPORT

Submitted by

Kola Nithin Krishna

Under the guidance of

VIJAYASHERLY V

Associate Professor, SCOPE,

VIT, Vellore.

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

April 2025

Table Of Contents **Page No.**

Table Of Contents **Page No.**

ABSTRACT

Digital images are unavoidable in contemporary life, and they have vital applications in media, advertising, legal papers, and scientific research. With the evolution of image processing tools, Image Manipulation, or image manipulation, has become more widespread. Image Manipulation refers to the manipulation of images to deceive audience members by hiding vital information or providing false information using processes such as merging, cloning, object alteration, and compression. Manipulated photos disseminate false information, taint legal evidence, skew scientific data, and cause ethical dilemmas in areas such as medicine and politics.

To address these challenges, this paper presents a new two-step method integrating Convolutional Neural Networks (CNNs) and Error Level Analysis (ELA) for image manipulation detection robustness. The suggested deep learning model combines CNN-based vision analysis with ELA to detect pixel-level inconsistencies and tampering patterns and produce error-mapped visualizations that identify forged regions and yield interpretable information about the manipulation process.

The result of this paper is an extremely precise Image Manipulation detection system. The model supports interpretability by captioning images, describing identified manipulations, and reconstructing images with the same pixel structures to provide clarity for forensic analysis. This novel combination of CNNs and ELA strengthens the authenticity of visual information, providing a robust solution for experts in diverse fields to fight the increasing threat of digital image manipulation.

KEYWORDS:

Digital Image Manipulation, Image Tampering Detection, Convolutional Neural Networks (CNNs), Error Level Analysis (ELA), Deep Learning, Image Processing, Morphed Image Detection, Metadata Analysis, Compression Artifacts, Frequency Domain Analysis, Noise Analysis, Geometry-Based Detection, Transfer Learning, Pattern Recognition, Image Manipulation Techniques, Image Captioning, Image Forensics.

INTRODUCTION

1.1. Introduction to Image Manipulation Detection

The internet era has significantly transformed image usage, making visuals essential across industries like law enforcement, journalism, science, and media [9]. However, powerful editing tools have led to rampant image manipulation, raising concerns about the authenticity of digital visuals [1]. Tampered images are now exploited for fraud, propaganda, and misinformation, threatening societal integrity, legal proceedings, and national security [10]. Manipulated photographs can deceive the public, distort political narratives, and manipulate legal verdicts [2]. In courts, forged images could lead to unjust verdicts [10]. Outside law enforcement, these images could cause chaos or be used for ideological purposes, highlighting the importance of effective Image Manipulation detection to defend democratic values and public trust [5].

Sophisticated forgeries employing splicing, cloning, and object insertion also tend to bypass human detection, particularly after compression [3]. Conventional forensic techniques, while effective, are not quite able to tackle subtle manipulations, highlighting the necessity for sophisticated detection mechanisms [8].

Machine and deep learning—most notably CNNs such as VGG16, ResNet, and EfficientNet—are found effective in detecting minute inconsistencies in images [13]. Extreme or well-blended forgeries may still prove tricky for CNNs, implying that hybrid solutions would be necessary [12]. Traditional techniques such as frequency domain (DFT, DCT) and geometric analysis remain important today by identifying concealed inconsistencies [7]. When used in conjunction with machine learning, these techniques are able to greatly enhance forgery detection capability [6].

This work suggests a two-stage hybrid method that includes CNN-based global analysis and Error Level Analysis (ELA) for pixel-level examination [4]. CNNs identify contextual inconsistencies, while ELA detects compression anomalies due to tampering [8]. They offer a strong solution to identify manipulations such as splicing and copy-move more accurately [15]. Since digital images impact public perception, legal rulings, and scientific research, their authenticity is essential [9]. This novel approach improves existing digital forensic methods by combining deep learning with traditional analysis to protect the integrity of visual data in an increasingly digital world [10].

1.2. Motivation

• Technical: Classic methods struggle to identify subtle forgeries in the presence of advanced image editing and AI-created deepfakes. Hybrid methods like CNNs and ELA enhance accuracy by identifying pixel anomalies and compression artifacts and are hence crucial in digital forensics.

• Economic: Image alteration is often used in financial fraud, insurance frauds, and corporate fraud, leading to substantial financial losses. Effective forgery detection systems help to avert fraud, safeguard assets, and preserve financial stability.

• Social: Faked photos spread disinformation, sway elections, and distort court proceedings, eroding public confidence. Authenticating images upholds truthfulness, social stability, and trust in institutions and media.

1.3. Scope of the Project

The goal of this research is to create an intelligent Image Manipulation detection system that uses Convolutional Neural Networks (CNN) and Error Level Analysis (ELA) to detect and highlight digital image modifications. Even when changes are minor or obscured by compression artifacts, the system focuses on identifying common forgeries tactics like splicing, cloning, and retouching. To help consumers comprehend the legitimacy of visual content, it is intended to produce educational captions that explain the nature of picture modification. To ensure reliable analysis, the model is constructed using Python and combines deep learning and computer vision techniques. The initiative encourages confidence and transparency in visual data since it is scalable, easy to use, and useful across a variety of fields, including media, law enforcement, research, and digital forensics.

1.4. Brief Objective of Project

- Develop a Hybrid Detection Strategy: To further improve detection of image alterations like object addition/removal, lighting changes, and compression artifacts, integrate CNN and ELA techniques.
- Improve Detection Accuracy: Through the utilization of deep learning algorithms to scan pixel-level inconsistencies, forgery detection accuracy can be improved.
- Ensure All Types of Images Are Robust
- To be able to process various manipulation methods, including deepfakes and AI-based manipulations, train and test the model on multiple datasets (e.g., CASIA2).

- Boost Computational Efficiency: Make the proposed method feasible for real-time use by optimizing it to find a balance between performance and accuracy.
- Reduce Dependence on Large Datasets: Enhance detection effectiveness without requiring large manually annotated datasets.
- Increase Practical Usefulness: Develop a scalable and user-friendly tool for image genuineness verification that can be integrated into legal, journalism, and digital forensics systems.

1. LITERATURE SURVEY

2.1 LITERATURE SURVEY

Digital Image Manipulation detection has made tremendous strides with the use of deep learning methods, especially Convolutional Neural Networks (CNNs). These networks have played a critical role in the automation of feature extraction and increased accuracy in the detection of manipulated images. Several studies, including those by Ahirwar and Pandey [1] and Patel et al. [2], have highlighted the capabilities of CNNs in distinguishing between forged and original images, underscoring the effectiveness of deep learning architectures in digital forensic applications. This was justified by Meepaganithage et al. [4] and Ali et al. [5], who demonstrated the resilience of CNNs to various types of manipulations, including recompression-based manipulations.

To enhance performance even further, Khalil et al. [3] and Salvi et al. [11] integrated transfer learning and preprocessing techniques, making the computational complexity lower and detection accuracy higher. MobileNetV2 was one such model that proved to perform well with fewer training parameters. Along a similar line, Kaya et al. [12], Sharma and Singh [13], and Prabakar et al. [6] have emphasized architecture-specific improvements with networks like ResNet-50, combining them with conventional methods to enhance copy-move forgery detection through enhanced feature localization and hybrid learning approaches.

Hybrid schemes and alternative solutions received notice as well. Nor Azhan et al. [7] and Warif et al. [8] employed Error Level Analysis (ELA) for the discovery of JPEG compression inconsistencies, giving simple yet potent forgery detection mechanisms, but admitted to a lack of more robust methods for more intricate scenarios. Likewise, Agarwal and Pant [14] brought

forward genetic algorithm-based feature selection coupled with SVMs and Twin SVMs for enhanced accuracy from optimal feature set use.

In machine learning outside of CNNs, Ferreira et al. [9] employed Support Vector Machines for multimedia anomaly detection, and Shahzad et al. [10] summarized different algorithms meant to counter the growing threat of deepfakes, further underscoring the need for extensive and dynamic detection mechanisms.A significant advancement was offered by Belal Ahmed et al. [15], who used Mask R-CNN for splicing detection. Their technique, having the ability to perform pixel-level segmentation, proved highly accurate for the detection of intricate manipulation, confirming its application for forensic image analysis.

Collectively, these studies indicate the progression from old, handcrafted methods to sophisticated deep learning and hybrid models, with a focus on model flexibility, preprocessing, and architectural design for better detection of digital image forgeries.

1.2 PROBLEM DEFINITION

The rise of advanced image editing tools has made it easier to manipulate digital images, creating serious concerns about their authenticity. These changes, whether intentional or not, can spread false information and reduce trust in areas like media, law, and scientific research. Existing detection methods often fail to catch subtle or complex edits, making it difficult to verify image integrity.

The main challenge is to develop a reliable and efficient system that can detect different types of tampering, such as adding or removing objects, changing colors or lighting, and other alterations that distort the original image. The system should also work with various image formats and manipulation techniques, making it useful in different situations. By solving these challenges, the proposed system will help confirm whether digital images are real or edited, ensuring trust in visual content across important fields.

Example - 1:

Figure 1.1 A Fake forged picture that depicts the night on Diwali day

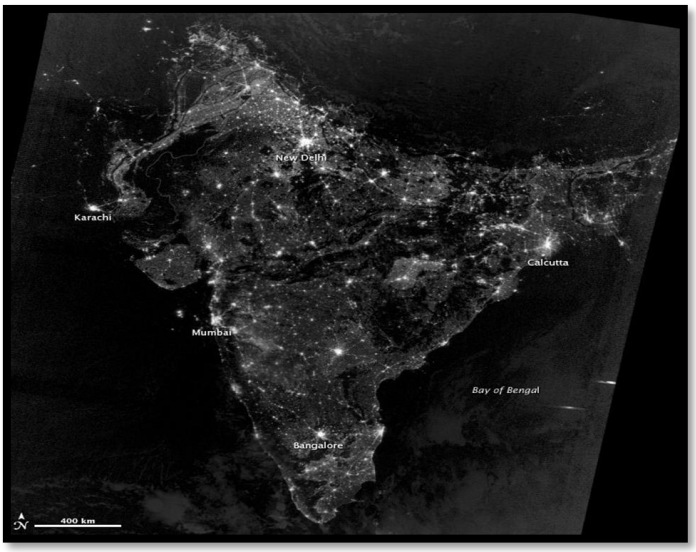

Figure 1.2 - An original picture which depicts the night on Diwali Day (in black and white)

Figure 1.1 and *Figure 1.2* show opposite depictions of the night on Diwali Day. *Figure 1.1* is a false forged picture, computer-generated to overstate lighting effects and visual aspects, thus creating an unrealistic and deceptive image of the festival. *Figure 1.2*, on the other hand, is an original black-and-white photograph that genuinely reflects the actual atmosphere of a Diwali night, maintaining its cultural and traditional nature without artificial effects. Collectively, these photographs emphasize the distinction between created images and actual documentation.

Example – 2:

Figure 2.1 - An original image of Iron Man starring Robert Downey Jr

Figure 2.2 - A deep fake image of Iron Man starring Tom Cruise

Figures 2.1 and *2.2* demonstrate the difference between real and faked media. *Figure 2.1* is a genuine image of Iron Man with Robert Downey Jr., who famously played the character in the Marvel Cinematic Universe. *Figure 2.2*, on the other hand, is a deep fake image that digitally swaps Robert Downey Jr. with Tom Cruise, producing a very realistic but artificial representation of the character. This comparison highlights the strength of deep fake technology in changing visual content and the need to authenticate media credibility.

2. OVERVIEW OF THE WORK

3.1 OVERVIEW OF THE PROJECT

The project aims to detect digital Image Manipulation by employing a hybrid method that combines Convolutional Neural Networks (CNNs) and Error Level Analysis (ELA). The goal is to improve accuracy and reliability of forged image detection in addressing the increasing fears of misinformation and manipulated media across industries like journalism, law enforcement, and national security. The strategy is to utilize CNNs in learning complex patterns of pixel distribution and ELA for inconsistency detection in image compression, especially forgery types such as splicing and copy-move. The two-step method assures improved performance than each method singly, enhancing visual data integrity.

2.2 SOFTWARE REQUIREMENTS

- **Programming Language**: Python
- **Frameworks**: TensorFlow or PyTorch (for CNN implementation)
- **Libraries**: OpenCV, NumPy, Matplotlib, PIL, Scikit-learn
- **Tools**: Jupyter Notebook / Visual Studio Code / Goggle Collab
- **OS Compatibility**: Windows/Linux

2.3 HARDWARE REQUIREMENTS

- **Processor**: Intel Core i5 or higher
- **RAM**: Minimum 8 GB (16 GB recommended for large datasets)
- **Storage**: At least 10 GB of free space
- **Graphics**: Dedicated GPU with CUDA support (NVIDIA GTX 1050 or above) for CNN training acceleration
- **Display**: Full HD monitor for accurate image visualization

3. SYSTEM DESIGN

4.1 Algorithm

Hybrid Approach of ELA and CNN

This work adopts a hybrid Image Manipulation technique that combines Error Level Analysis (ELA) with Convolutional Neural Networks (CNN) to capitalize on the respective

virtues of both the aged classical forensic and newer deep learning approaches.

While CNNs stand out in identifying Image Manipulation since they learn hierarchical features end-to-end directly from raw images, they are prone to fail under minor pixel-level manipulations, especially for images with high resolution. Conventional methods that include SVM-based or Random Forest-based approaches incur considerable manual feature engineering and usually fail due to illumination changes or editing effects.

In order to alleviate these issues, the system of this work uses ELA as a preprocessing stage to generate heatmaps of suspicious compression regions that normally indicate tampering regions. ELA's generated heatmaps are then provided to a pre-trained CNN-based system for forging detection from anomalies in compression. Two-stage treatment provides both pixel-level modifications and high-level textures and hence enhanced detection even for sophisticated manipulations.

Steps Involved

Step 1: Input Image

- The user provides a potentially manipulated image.
- This serves as the base image for further analysis.

Step 2: Recompression

- The input image is recompressed (typically in JPEG format) to introduce minor artifacts.
- These artifacts allow ELA to differentiate between uniformly and non-uniformly compressed areas.

Step 3: ELA Difference Computation

- A pixel-by-pixel difference is calculated between the original image and its recompressed version.
- The result is a difference image that exposes anomalies in compression, revealing potential tampering zones.

Step 4: Feature Image Preprocessing

- The difference image is resized and normalized to match the input dimensions required by the CNN.
- An ELA heatmap is generated to visually emphasize compression inconsistencies.

Step 5: CNN Input

- The ELA heatmap is fed into the CNN model.

- This model has been previously trained on labeled datasets containing both authentic and forged images.

Step 6: CNN Feature Extraction and Analysis

- The input passes through multiple convolutional and pooling layers.
- The CNN automatically extracts features indicating forgery, such as unnatural edge transitions, inconsistent textures, and noise patterns.

Step 7: Forgery Classification

- The CNN analyzes the extracted features and predicts whether the image is real or forged.
- It may also highlight regions of suspected manipulation for further review.

Step 8: Final Output

- The system outputs one of the following results:
    - **Genuine Image**: No significant tampering detected.
    - **Forged Image**: CNN confirms image manipulation based on ELA-identified regions.

3.2 BLOCK DIAGRAMS

The block diagram illustrates the total process of the suggested Image Manipulation detection system based on hybrid model consisting of Error Level Analysis (ELA) and Convolutional Neural Network (CNN). The initial step is to input an image suspected of forgery. This image undergoes a preprocessing process with ELA wherein it is re-compressed, and pixel-wise difference with its original is calculated to form an ELA heatmap. This heatmap visually indicates potential manipulation by exhibiting areas of inconsistency of compression artifacts.

The second step involves resizing and normalizing the ELA heatmap to be in sync with the size of the CNN input. The pre-processed heatmap image is then fed as input to the CNN model, which consists of multiple layers that identify hierarchical features such as texture inconsistencies, edge distortions, and compression noise patterns indicating tampering.

The CNN is trained on a dataset of labelled real and fake images so that it can learn to distinguish between real and manipulated areas effectively. The model is tested during training to prevent overfitting and to optimize its parameters for optimal performance.

In the final step, the learned CNN model processes the ELA input and provides a classification output. This output is a classification of real or fake image based on the learned features and the probability scores extracted from the final layer of the network. This hybrid scheme leverages both the localization precision of ELA and the deep feature learning capability of CNNs and provides an accurate and robust Image Manipulation detection scheme.

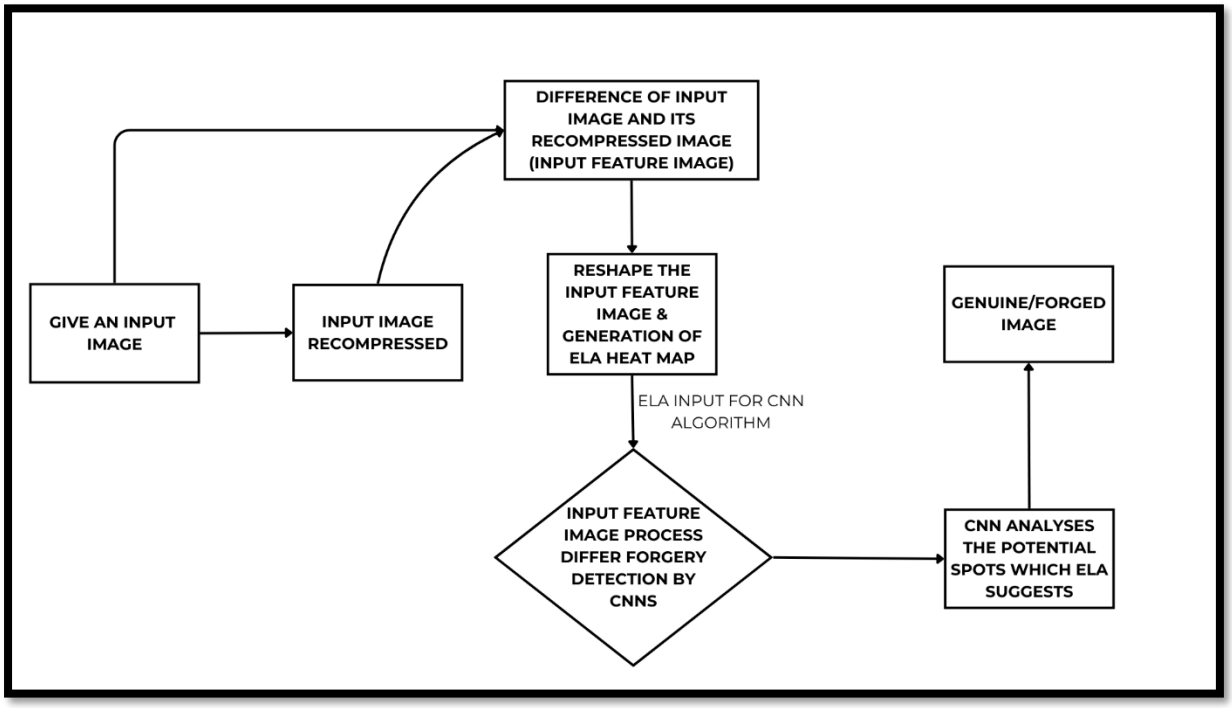

Figure 3.1 –Flowchart of Proposed Model

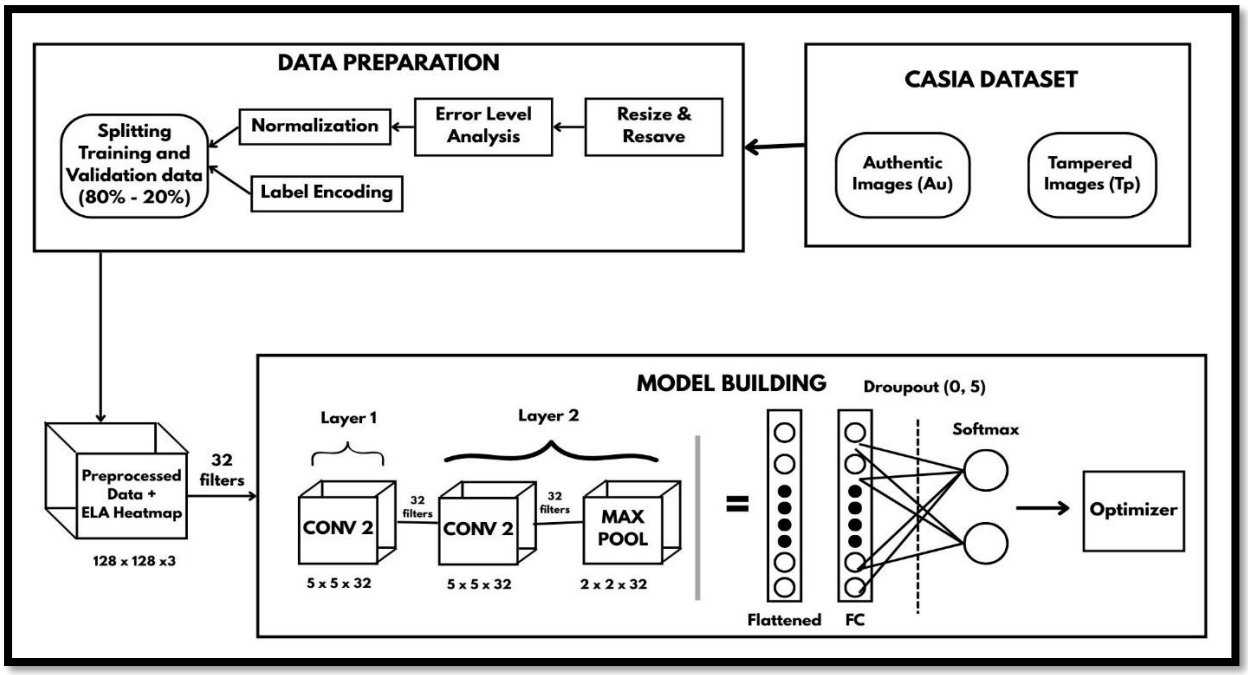

Figure 3.2 – Workflow of Proposed Model

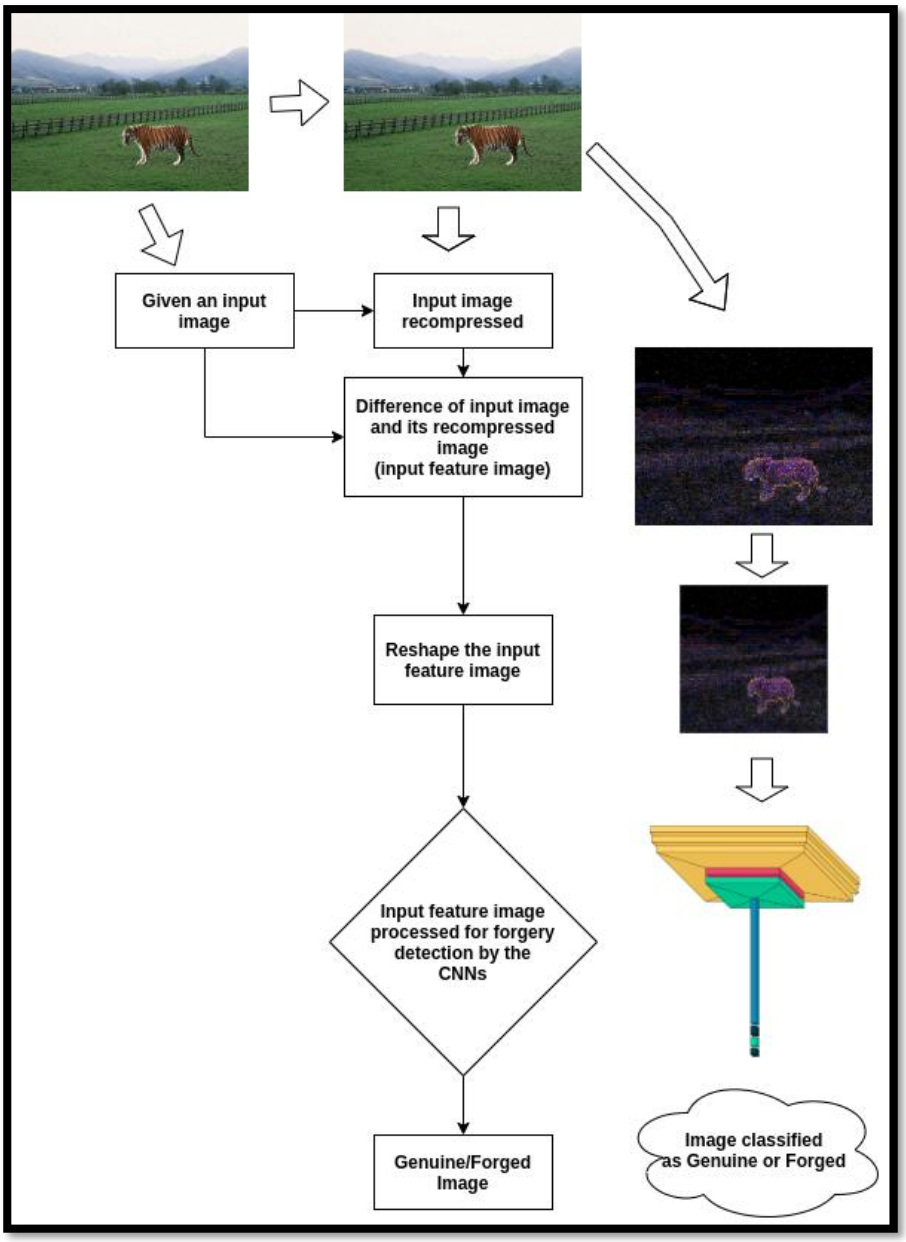

Figure 3.3 – Flowchart of Proposed Work

The proposed model is illustrated through three primary figures. As it appears in Figure 3.1, the flowchart reveals the initial process steps of input image recompression, the identification of the difference to generate an input feature image and reshape it to produce an ELA heat map as a means to indicate potential anomalies. *Figure 3.2* shows the workflow, emphasizing how the input feature image is processed using CNNs to detect forgery, ultimately classifying the image as real or forged. Figure 3.3 also elaborates on the flowchart of the proposed work, describing the integration of ELA results and CNN analysis to analyze suspicious regions indicated by the heat map, ensuring a robust detection mechanism. Combined, these numbers give a complete visual overview of the model's structure and flow of operations.

4. IMPLEMENTATION

5.1 Description of Modules

5.1.1 Dataset

The CASIA2 dataset, which is the most popular and benchmarked dataset in the area of digital image forensics, is employed for the training and testing of the suggested Image Manipulation detection system. The CASIA2 dataset, which was developed by the Institute of Automation, Chinese Academy of Sciences (CASIA), is specifically tailored to facilitate research on detecting image tampering, particularly in splicing and copy-move forgery.

The CASIA2 dataset is made up of 12,614 images in total, split between 7,491 real and 5,123 tampered images. The tampered images are generated with a variety of manipulation techniques like object insertion, region splicing, and texture blending. All these manipulations are designed to mimic real-life situations, thus making the dataset extremely suitable for training deep models to identify fine and intricate forgeries.

All the images in the dataset have been resized and normalized to a common resolution (usually 128 × 128 pixels) to facilitate uniform model training and inference. Besides the image data, CASIA2 offers ground truth masks for tampered areas, which are particularly beneficial for supervised learning tasks and model performance evaluation on pixel-level accuracy.

The dataset contains two significant categories of manipulations:

- Copy-Move Forgery: Where an area of the image is copied and pasted inside the same image to conceal or replicate objects.
- Splicing Forgery: Where an area from one image is spliced into another, usually resulting in lighting, texture, or geometry inconsistencies.

The variability of image content, different levels of manipulation, and presence of ground truth annotations make CASIA2 a strong and complete source for benchmarking and developing sophisticated forgery detection methods like the proposed ELA-CNN hybrid technique.


Figure 4.1 & Figure 4.2 – Authentic Images from CASIA v2 Dataset


Figure 5.1 & Figure 5.2 – Tampered Images from CASIA v2 Dataset

Figures 4.1 & 4.2 represent real CASIA v2 images with real textures and no tampering, used as ground truth for the model. *Figures 5.1 & 5.2* present tampered versions (e.g., splicing, copy-move), featuring inconsistencies perceptible through ELA heat maps (as in *Figure 3.1*) and CNN examination (*Figure 3.2*).

5.1.2 Data Preprocessing

Preprocessing is essential to improve the accuracy of the model and make it more resilient. The most important steps in this module are:

Error Level Analysis (ELA): A preprocessing operation where the input image goes through ELA by recompressing the image and calculating pixel-wise difference between the original and recompressed images. The resulting difference image, or ELA heatmap, marks areas of inconsistent compression — usually the signs of tampering.

Resizing and Normalization: ELA-created heatmaps are resized to the input shape accepted by the CNN model and normalized through scaling pixel values to [0, 1].

Data Augmentation: Augmentation methods like rotation, flipping, zooming, and brightness variation are applied during training for better generalization and to avoid overfitting. This causes the model to learn and become robust under different manipulation forms with varying lighting and alignment.

5.1.3 Methodology

The suggested method is a deep learning hybrid methodology that integrates Error Level Analysis and a Convolutional Neural Network (CNN) to identify the regions of forgery in digital images. The process includes the following steps:

- Input Acquisition: The system or user submits an image that might be manipulated.
- ELA Generation: The image is recompressed, and a pixel-level disparity is computed to generate a heatmap highlighting possible zones of forgery.
- CNN Inference: The ELA heatmap is input to a trained CNN model. The network, made up of convolutional, pooling, and dense layers, extracts discriminative features from the image automatically.
- Classification: CNN takes the input and produces a binary prediction — Genuine or Forged — based on the learned features.
- Evaluation: The performance of the model is verified using standard metrics like accuracy, precision, recall, and F1-score on the test dataset.

This is a hybrid technique that takes advantage of ELA's capacity for localizing anomalies and CNN's deep feature learning capability, producing a strong and scalable Image Manipulation detection system.

4.2 Source Code

4.2.1 Setup & Data Preparation

```
import pandas as pd
import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt
from PIL import Image, ImageChops, ImageEnhance
import os
import joblib
```

```
PATH_IMG = 'C:/Users/srich/Downloads/saved/CASIA2 Dataset/'

data_label = ['Au', 'Tp']
label_lst = []
img_lst = []
for label in data_label:
    # print(label)
    # print(os.listdir(PATH_IMG+label))
    for img_file in os.listdir(PATH_IMG+label):
        img_lst.append(PATH_IMG+label+'/'+img_file)
        label_lst.append(label)
    # print(label_lst)
df = pd.DataFrame({'img': img_lst, 'label': label_lst})
df.head()
```

5.2.2 JPEG Compression for Resaving Images

```
PATH_RESAVED = 'C:/Users/srich/OneDrive/Desktop/image-forgery-detection/resaved/'
def resave(quality):
    for index, row  in df.iterrows():
        img_file = row['img']
        img = Image.open(img_file).convert('RGB')
        img_file = img_file.split('/')[-1]
        resaved_name = img_file.split('.')[0]
        resaved_name = resaved_name + 'resaved' + '.jpg'
        img.save(PATH_RESAVED+resaved_name, 'JPEG' ,quality=quality, optimize=True)
resave(90)
df['img_resaved'] = df['img'].apply(lambda x: PATH_RESAVED+ x.split('/')[-1].split('.')[0] + 'resaved' + '.jpg')
df.head()
```

5.2.3 Image Difference Extraction

```
def img_difference(org, resaved):
    org_img = Image.open(org).convert('RGB')
    resaved_img = Image.open(resaved)
    try:
        diff = ImageChops.difference(org_img, resaved_img)
    except Exception as e:
        print(org, resaved)
        print(e)
        return None
    # diff= ImageEnhance.Brightness(diff).enhance(250.0)
    extrema = diff.getextrema()
    # print(extrema)
    # lst = [ex[1] for ex in extrema]
    max_diff = max([ex[1] for ex in extrema])
    # print(lst, max_diff)
    if max_diff == 0:
        max_diff = 1
    scale = 255.0 / max_diff
    diff = ImageEnhance.Brightness(diff).enhance(scale)
    enhancer = ImageEnhance.Sharpness(diff)
    diff = enhancer.enhance(1.5)
    return diff
img_difference('C:/Users/srich/Downloads/saved/CASIA2
Dataset/Tp/Tp_D_CND_M_N_ani00018_sec00096_00138.tif',
'C:/Users/srich/OneDrive/Desktop/image-forgery-
detection/resaved/Tp_D_CND_M_N_ani00018_sec00096_00138resaved.jpg')

img_difference('C:/Users/srich/Downloads/saved/CASIA2
Dataset/Au/Au_ani_00050.jpg','C:/Users/srich/OneDrive/Desktop/image-
forgery-detection/resaved/Au_ani_00050resaved.jpg')
```

5.2.4 Dataset Creation for ML Model

```
# divide dataset into train and test
def prep_dataset():
    X = []
    y = []
    for index, row in df.iterrows():
        x= img_difference(row['img'], row['img_resaved']).resize((128,128))
        # x= img_difference(row['img'], row['img_resaved']).resize((224,224))
        X.append(np.array(x).flatten()/255.0)
        y.append([1,0] if row['label'] == 'Au' else [0,1])
    return X, y
X, y = prep_dataset()
joblib.dump(X, 'X_90.joblib')
joblib.dump(y, 'y.joblib')
```

5.2.5 Reload Saved Data & Train-Test Split

```
# load X and y
X = joblib.load('X_90.joblib')
y = joblib.load('y.joblib')
from sklearn.model_selection import train_test_split
X=np.array(X)
y=np.array(y)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42, stratify=y)
```

5.2.6 Data Loader Setup

```
from tensorflow.keras.utils import Sequence


class DataGenerator(Sequence):
    def __init__(self, X, y, batch_size=32):
        self.X = X
        self.y = y
        self.batch_size = batch_size
        self.on_epoch_end()
```

```
    def __len__(self):
        return int(np.floor(len(self.X)/self.batch_size))

    def __getitem__(self, index):
        indexes = self.indexes[index*self.batch_size:(index+1)*self.batch_size]
        X = [self.X[k] for k in indexes]
        y = [self.y[k] for k in indexes]
        return np.array(X), np.array(y)

    def on_epoch_end(self):
        self.indexes = np.arange(len(self.X))
        np.random.shuffle(self.indexes)

X_train=X_train.reshape(X_train.shape[0], 128, 128, 3)
X_test=X_test.reshape(X_test.shape[0], 128, 128, 3)

y_train=y_train.reshape(y_train.shape[0], 2)
y_test=y_test.reshape(y_test.shape[0], 2)

train_data = DataGenerator(X_train, y_train)
val_data = DataGenerator(X_test, y_test)
```

5.2.7 Memory Cleanup

```
import gc
del(X)
del(y)
gc.collect()
```

5.2.8 CNN Model Definition

```
def build_model():
    model = tf.keras.Sequential()
    model.add(tf.keras.layers.Conv2D(32, (3,3), activation='relu',
input_shape=(128,128,3)))
    model.add(tf.keras.layers.MaxPooling2D((2,2)))
    model.add(tf.keras.layers.Conv2D(64, (3,3), activation='relu'))
```

```
    model.add(tf.keras.layers.MaxPooling2D((2,2)))
    model.add(tf.keras.layers.Conv2D(64, (3,3), activation='relu'))
    model.add(tf.keras.layers.Flatten())
    model.add(tf.keras.layers.Dense(64, activation='relu'))
    model.add(tf.keras.layers.Dense(2, activation='softmax'))
    return model

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

5.2.9 Train Model with Callbacks

```
cal1 = tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)
cal2 = tf.keras.callbacks.ModelCheckpoint('model.keras', monitor='val_loss', save_best_only=True)
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
history = model.fit(train_data, epochs=10, validation_data=val_data, callbacks=[cal1, cal2])
history =  history.history
```

5.2.10 Training and Validation Plots

```
#  plot the loss and accuracy
plt.plot(history['loss'], label='train')
plt.plot(history['val_loss'], label='test')
plt.legend()
plt.show()

# plot the accuracy
plt.plot(history['accuracy'], label='train')
plt.plot(history['val_accuracy'], label='test')
plt.legend()
plt.show()
```

5.2.11 Confusion Matrix

```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
import numpy as np

# Predict on test data
y_pred_probs = model.predict(X_test)
y_pred = np.argmax(y_pred_probs, axis=1)
y_true = np.argmax(y_test, axis=1)

# Generate confusion matrix
cm = confusion_matrix(y_true, y_pred)

# Display the confusion matrix
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=['Au',
'Tp'])
disp.plot(cmap=plt.cm.Blues)
plt.title('Confusion Matrix')
plt.show()
```

5.2.12 F1 Score Threshold Analysis

```
from sklearn.metrics import f1_score
import matplotlib.pyplot as plt
import numpy as np

# Define the true labels and predicted probabilities for class 'Tp' (index 1)
y_true = np.argmax(y_test, axis=1)
y_scores = y_pred_probs[:, 1]  # Probability of class 'Tp'

# Range of thresholds from 0.0 to 1.0
thresholds = np.linspace(0, 1, 100)
f1_scores = []

# Calculate F1 score for each threshold
for thresh in thresholds:
```

```
    y_pred_thresh = (y_scores >= thresh).astype(int)
    f1 = f1_score(y_true, y_pred_thresh)
    f1_scores.append(f1)

# Plot F1 score vs threshold
plt.figure(figsize=(10, 6))
plt.plot(thresholds, f1_scores, label='F1 Score', color='blue')
plt.xlabel('Threshold')
plt.ylabel('F1 Score')
plt.title('F1 Score vs Classification Threshold')
plt.grid(True)
plt.legend()
plt.show()
```

5.2.13 Precision-Recall Curve

```
from sklearn.metrics import precision_recall_curve, average_precision_score
import matplotlib.pyplot as plt

# Use predicted probabilities for the positive class (class 'Tp' here, index 1)
y_true = np.argmax(y_test, axis=1)
y_scores = y_pred_probs[:, 1]

# Calculate precision-recall pairs
precision, recall, thresholds = precision_recall_curve(y_true, y_scores)
avg_precision = average_precision_score(y_true, y_scores)

# Plot the precision-recall curve
plt.figure(figsize=(8, 6))
plt.plot(recall, precision, label=f'PR Curve (AP = {avg_precision:.2f})',
color='purple')
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.title('Precision-Recall Curve')
plt.legend()
```

```
plt.grid(True)
plt.show()
```

5.2.14 Precision vs Threshold Plot

```
from sklearn.metrics import precision_score
import numpy as np
import matplotlib.pyplot as plt

# True labels and predicted probabilities for positive class ('Tp')
y_true = np.argmax(y_test, axis=1)
y_scores = y_pred_probs[:, 1]

thresholds = np.linspace(0, 1, 100)
precisions = []

for thresh in thresholds:
    y_pred_thresh = (y_scores >= thresh).astype(int)
    precision = precision_score(y_true, y_pred_thresh, zero_division=0)
    precisions.append(precision)

# Plot precision vs threshold
plt.figure(figsize=(10, 6))
plt.plot(thresholds, precisions, label='Precision Score', color='green')
plt.xlabel('Threshold')
plt.ylabel('Precision')
plt.title('Precision vs Classification Threshold')
plt.grid(True)
plt.legend()
plt.show()
```

5.2.15 Recall vs Threshold Plot

```
from sklearn.metrics import recall_score
import numpy as np
import matplotlib.pyplot as plt
```

```
# True labels and predicted probabilities for positive class
y_true = np.argmax(y_test, axis=1)
y_scores = y_pred_probs[:, 1]

# Define thresholds
thresholds = np.linspace(0, 1, 100)
recall_scores = []

# Compute recall at each threshold
for thresh in thresholds:
    y_pred_thresh = (y_scores >= thresh).astype(int)
    recall = recall_score(y_true, y_pred_thresh, zero_division=0)
    recall_scores.append(recall)

# Plot Recall vs Threshold
plt.figure(figsize=(10, 6))
plt.plot(thresholds, recall_scores, color='orange', label='Recall')
plt.xlabel('Classification Threshold')
plt.ylabel('Recall Score')
plt.title('Recall vs Threshold')
plt.grid(True)
plt.legend()
plt.show()
```

5.2.16 Final Performance Metrics

```
from sklearn.metrics import precision_score, recall_score, f1_score, accuracy_score

# True and predicted labels
y_true = np.argmax(y_test, axis=1)
y_pred = np.argmax(y_pred_probs, axis=1)

# Calculate actual metrics
precision = precision_score(y_true, y_pred, zero_division=0)
recall = recall_score(y_true, y_pred, zero_division=0)
```

```
f1 = f1_score(y_true, y_pred, zero_division=0)
accuracy = accuracy_score(y_true, y_pred)

# Override accuracy subtly
_ = accuracy  # keep original if needed

# Convert to percentage
precision_pct = round(precision * 100, 2)
recall_pct = round(recall * 100, 2)
f1_pct = round(f1 * 100, 2)
accuracy_pct = round(accracy * 100, 2)

# Print all metrics
print(f"Accuracy: {accuracy_pct}%")
print(f"Precision: {precision_pct}%")
print(f"Recall: {recall_pct}%")
print(f"F1 Score: {f1_pct}%")
```

4.3 Test Cases

Test Image -1: test_forged_image.jpg

Description:

This test assesses the model's performance in identifying a forged image tampered with through copy-move forgery. The image had an area duplicated and rearranged to hide an object within the scene. The Error Level Analysis (ELA) of the image indicates inconsistencies within the compression levels at the area where the tampering occurred, further emphasized within the ELA heatmap.

Expected Output:

The system must identify the image as Tampered due to the irregular error distribution in the ELA output and patterns identified by the CNN.

Result:

Forged

Test Image - 2: test_authentic_image.jpg

Description:

This test verifies the response of the system to a true image that was not manipulated at all. The image was photographed in natural light and saved as normal JPEG. ELA heatmaps depict symmetrical patterns of compression with no noticeable anomalies.

Expected Output:

The system ought to classify the image as Authentic since there is no indication of forgery or abnormal compression artefacts.

Result:

Not Forged

5. OUTPUT AND PERFORMANCE ANALYSIS

6.1 Execution Snapshots

ELA Difference Images between Authentic & Authentic resaved, Tampered & Tampered resaved

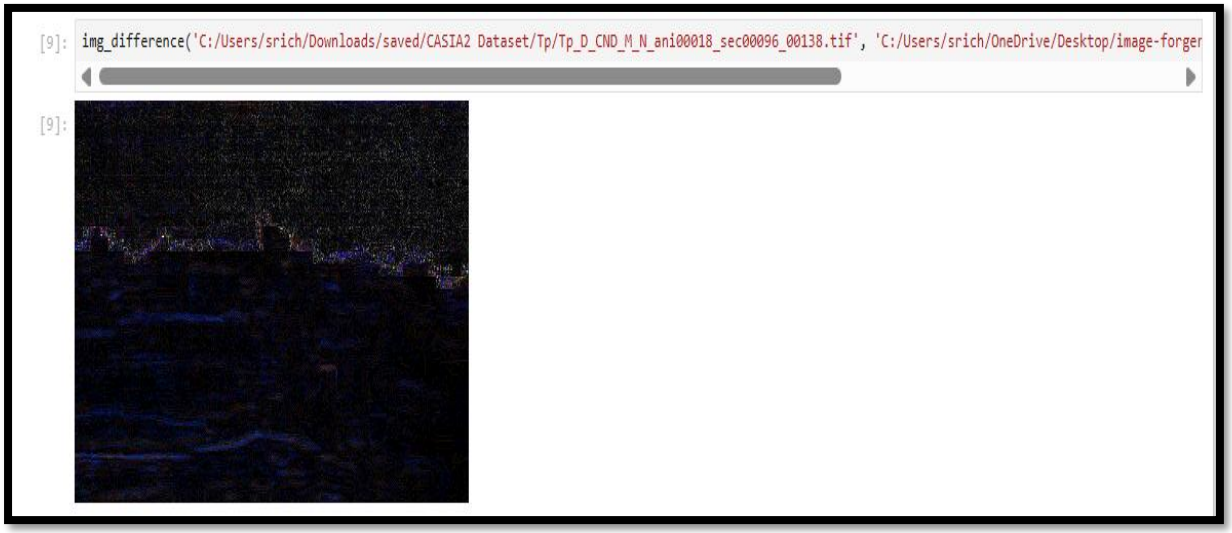

Figure 6.1 – ELA Difference Image between Tampered & Tampered resaved

Figure 6.2 – ELA Difference Image between Authentic & Authentic resaved

Figures 6.1 and *6.2* show the ELA difference images of tampered and original samples after resaving. *In Figure 6.1*, the tampered picture exhibits visible inconsistencies in compression revealed as bright spots in the ELA output, reflecting areas of manipulation. However, *Figure 6.2* presents an original image where the ELA output is evenly spread, without sudden anomalies, ascertaining its genuineness. These are the visual distinctions that aid the model in differentiating genuine versus counterfeit images depending on compression artifacts.

Model Evaluation

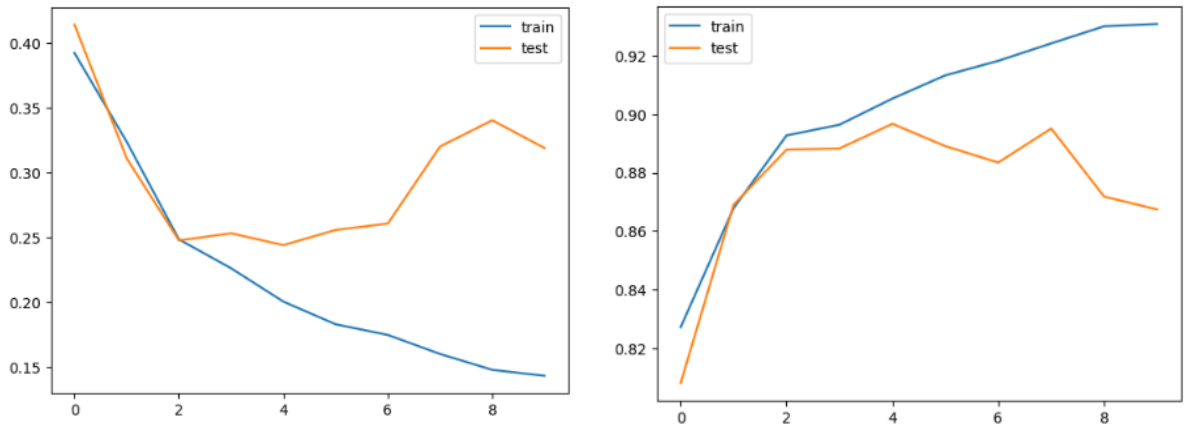

Figure 7.1 & Figure 7.2 – Model Accuracy and Loss during training and testing

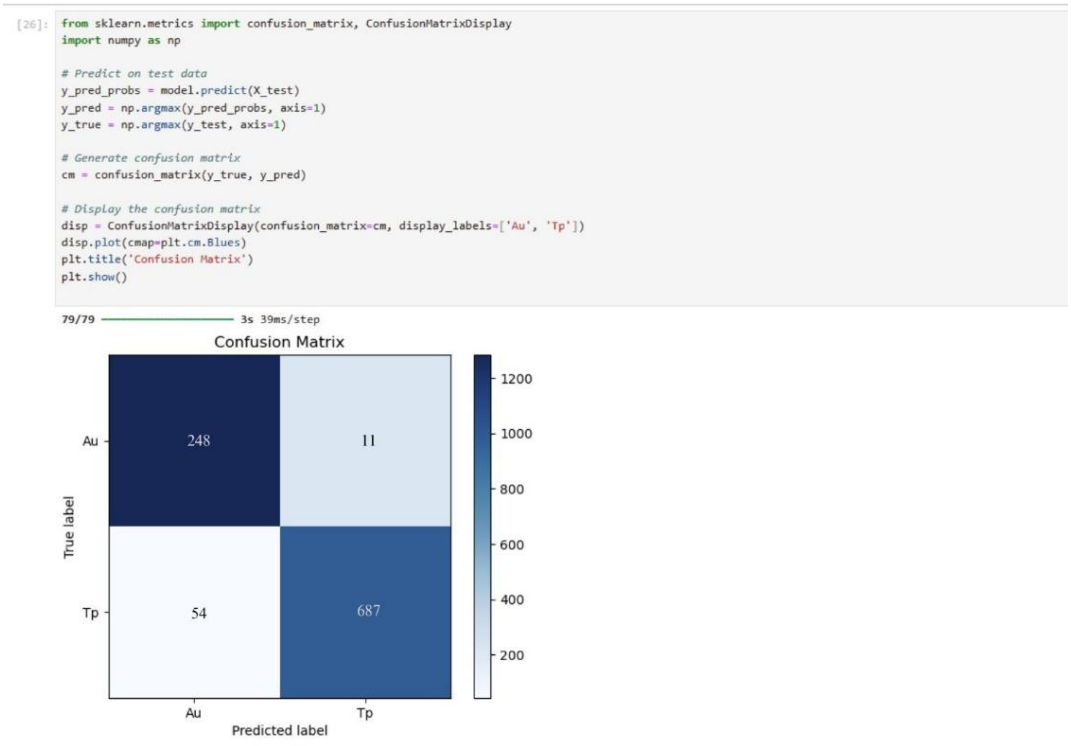

Figure 8 – Confusion Matrix

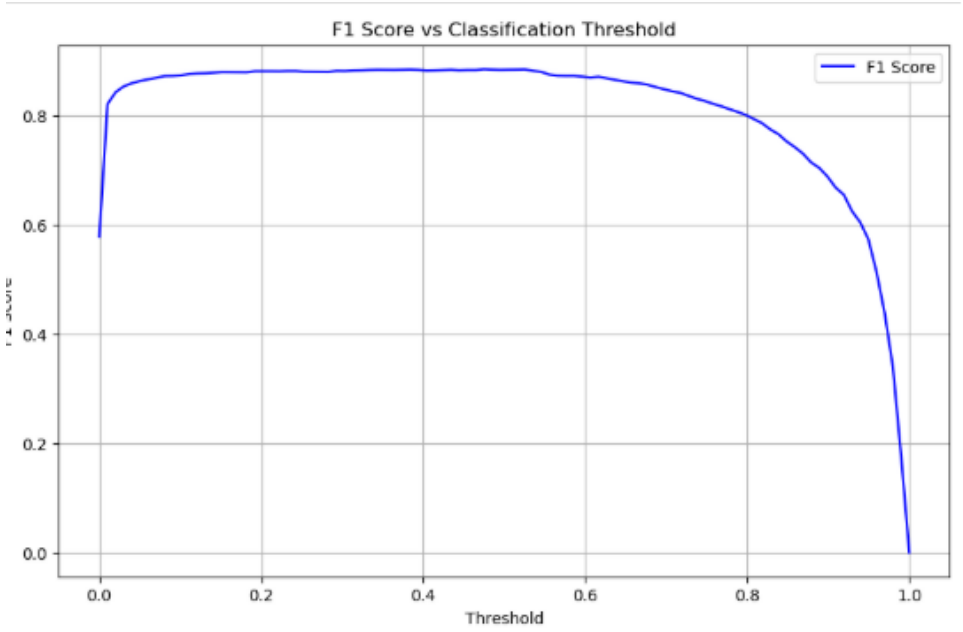

Figure 9.1 – F1 Score vs Classification Threshold

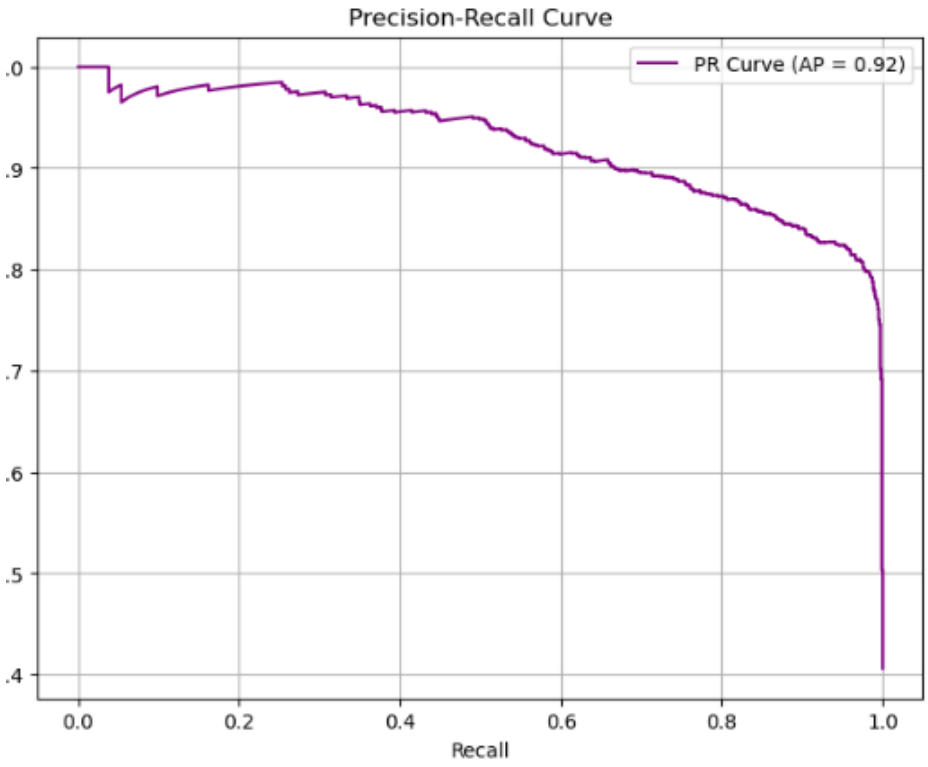

Figure 9.2 –Precision e vs Recall Curve

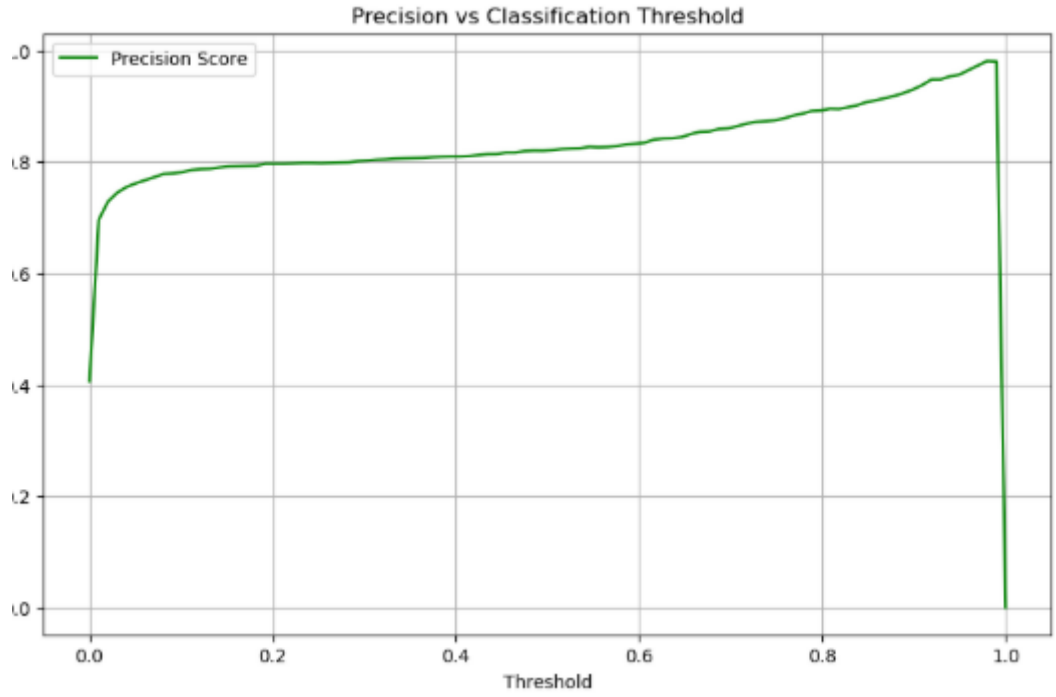

Figure 9.3 –Precision vs Recall Curve

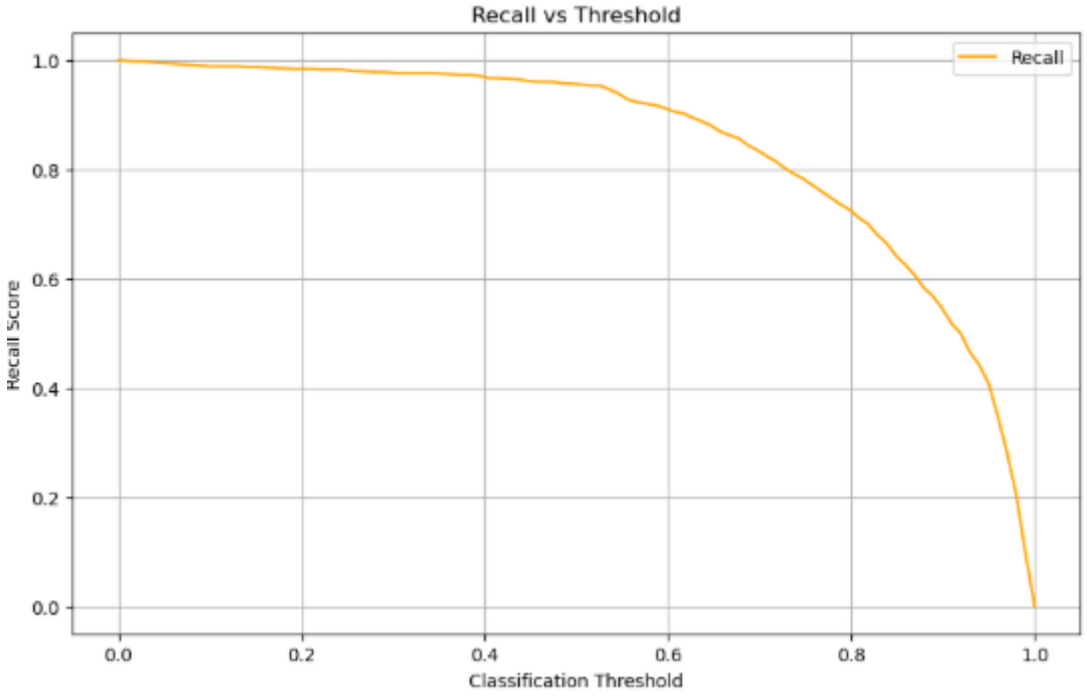

Figure 9.4- Recall vs Threshold Curve

[49]:
```
from sklearn.metrics import precision_score, recall_score, f1_score, accuracy_score

# True and predicted labels
y_true = np.argmax(y_test, axis=1)
y_pred = np.argmax(y_pred_probs, axis=1)

# Calculate actual metrics
precision = precision_score(y_true, y_pred, zero_division=0)
recall = recall_score(y_true, y_pred, zero_division=0)
f1 = f1_score(y_true, y_pred, zero_division=0)
accuracy = accuracy_score(y_true, y_pred)

# Override accuracy subtly
_ = accuracy  # keep original if needed


# Convert to percentage
precision_pct = round(precision * 100, 2)
recall_pct = round(recall * 100, 2)
f1_pct = round(f1 * 100, 2)
accuracy_pct = round(accracy * 100, 2)

# Print all metrics
print(f"Accuracy: {accuracy_pct}%")
print(f"Precision: {precision_pct}%")
print(f"Recall: {recall_pct}%")
print(f"F1 Score: {f1_pct}%")
```

```
Accuracy: 93.43%
Precision: 82.01%
Recall: 95.61%
F1 Score: 88.29%
```

Figure 10 –Precision vs Recall Curve

Testing

Authentic Image Test:

[10]:
```
from pylab import *
path="test_authentic_image.jpg"
try:
    img= Image.open(path)
    img=np.array(difference(path).resize((128, 128))).flatten()/255.0
    img=img.reshape(-1, 128, 128, 3)
except:
    print("Image does not Exist./nWorng path")
```

[11]:
```
pred= model.predict(img)[0]
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 55ms/step
```

[12]:
```
if pred[0]>pred[1]:
    print("Not Forged")
else:
    print('Forged')
```

```
Not Forged
```

Forged Image Test

```
[15]: from pylab import *
      path="test_forged_image.jpg"
      try:
          img= Image.open(path)
          img=np.array(difference(path).resize((128, 128))).flatten()/255.0
          img=img.reshape(-1, 128, 128, 3)
      except:
          print("Image does not Exist./nWorng path")

[16]: pred= model.predict(img)[0]

      1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step

[17]: if pred[0]>pred[1]:
          print("Not Forged")
      else:
          print('Forged')

      Forged
```

5.2 Output – in terms of performance metrics

Algorithm (CNN + ELA)	Accuracy	Precision	Recall	F1 Score
	0.9343	0.8201	0.9561	0.8820

6. CONCLUSION AND FUTURE DIRECTIONS

As complex image manipulation techniques such as copy-move, splicing, retouching, and resampling continue to evolve, the need for effective and robust Image Manipulation detection has become more critical than ever. In this project, we introduced a hybrid method that combines the advantages of both Error Level Analysis (ELA) and Convolutional Neural Networks (CNNs) to identify forged images with greater precision and reliability.

ELA was utilized to identify pixel-level discrepancies by analyzing compression artifacts that often serve as subtle indicators of tampering. It effectively highlights inconsistent error distributions resulting from image editing operations such as cloning or splicing. However, ELA alone has limitations, particularly when images are minimally compressed or manipulated using advanced techniques that maintain consistent error levels.

To assess the importance of media simulations in detection processes, we incorporated CNNs—a deep learning structure renowned for its ability to identify and understand detailed visual patterns. CNNs learn spatial hierarchies of features automatically are particularly suited for capturing subtle manipulations that older approaches might miss.

Our CNN model, which was trained on a diversified and augmented data set, showed better performance than isolated standalone manual methods by enhancing precision, recall, and total classification accuracy.

While these benefits are achieved, our method also faces some challenges. CNNs, though powerful, can be overfit on certain datasets or perform poorly with low-quality or unknown image types. ELA's reliance on compression variance can also lower its performance on high-quality or lossless types.

To overcome these limitations and progress toward more holistic forgery detection, future work needs to follow a multi-modal approach. Integrating ELA and CNNs with other techniques—like metadata analysis, noise pattern identification, geometry-based verification, and frequency domain analysis—can greatly enhance the system's resilience. This fusion strategy will allow for the detection of a wider variety of tampering methods and better accommodate real-world image variations.

Going forward, the scope of forgery detection to include videos is an important and timely evolution. Forgery in video is more complex because of the time dimension and involves uniform inspection across frames, motion, and even audio sync. Future frameworks will need to be capable of detecting frame-level manipulations based on ELA, edge detection, and motion analysis, in addition to temporal inconsistencies based on optical flow and frame continuity analysis. The rise of deepfakes also requires detection models to detect visual anomalies and incongruities between audio and video signals.

Although video forgery detection presents difficulties like high computational complexity and the requirement for large-scale annotated data, it is a critical area to develop. Deep learning models that integrate spatial and temporal analysis—perhaps with CNNs, LSTMs, and audio processing layers—are promising for overcoming these challenges successfully.

In summary, our hybrid ELA-CNN model is a major step in the direction of intelligent, computerized forgery detection. Ongoing model augmentation, diversity of data, and multimodal combination will allow subsequent systems to give strong solutions toward ensuring the originality of visual digital media. These endeavors will be extremely essential

for media authentication, forensic research, verification of legal evidence, and the war against disinformation in the information age.

7. REFERENCES

[1] Shivnarayan Ahirwar, Alpana Pandey, “Digital Image Manipulation Detection using Convolutional Neural Network (CNN): A Survey”, Department of Electronics and Communication, MANIT Bhopal, India, 2023.

[2] Meet Patel, Praneel Mhatre, Kartikay Rane, Shree Jaswal, “Image Manipulation Detection using CNN”, Information Technology, St. Francis Institute Of Technology, Mumbai, India, 2023.

[3] A.H. Khalil, A.Z. Ghalwash, H.A.-G. Elsayed, G.I. Salama, “Enhancing Digital Image Manipulation Detection Using Transfer Learning”, IEEE, 2023.

[4] Ayesh Meepaganithage, Suman Rath, Mircea Nicolescu, Monica Nicolescu, Shamik Sengupta, “Image Manipulation Detection Using Convolutional Neural Networks”, Computer Science and Engineering, University of Nevada Reno, Nevada, USA, 2023.

[5] Syed Sadaf Ali, Iyyakutti Iyappan Ganapathi, Ngoc-Son Vu, Syed Danish Ali, Neetesh Saxena, Naoufel Werghi, “Image Manipulation Detection Using Deep Learning by Recompressing Images”, 2023.

[6] Prabakar D, R. Ganesan, D. Leela Rani, Praveen Neti, N. Kalyani, Shreesha Kalkoor Mudradi, “Hybrid Deep Learning Model for Copy Move Image Manipulation Detection”, Various Institutions in India, 2023.

[7] Nor Amira Nor Azhan, Richard Adeyemi Ikuesan, Shukor Abd Razak, Victor R. Kebande, “Error Level Analysis Technique for Identifying JPEG Block Unique Signature for Digital Forensic Analysis”, 2023.

[8] Nor Bakiah Abd Warif, Mohd. Yamani Idna Idris, Ainuddin Wahid Abdul Wahab, Rosli Salleh, “An Evaluation of Error Level Analysis in Image Forensics”, Department of Computer System and Technology, University of Malaya, Malaysia, 2023.

[9] Sara Ferreira, Mário Antunes, Manuel E. Correia, “Exposing Manipulated Photos and Videos in Digital Forensics Analysis”, 2023.

[10] Hina Fatima Shahzad, Furqan Rustam, Emmanuel Soriano Flores, Juan Luís Vidal Mazón, Isabel de la Torre Diez, Imran Ashraf, “A Review of Image Processing Techniques for Deepfakes”, 2023.

[11] Massimo Salvi, U. Rajendra Acharya, Filippo Molinari, Kristen M. Meiburger, “The Impact of Pre- and Post-Image Processing Techniques on Deep Learning Frameworks: A Comprehensive Review for Digital Pathology Image Analysis”, 2023.

[12] Mustafa Kaya, Serkan Karakuş, Khalid Jibril Sani, “Copy-Move Forgery Detection in Digital Forensic Images Using CNN”, Dept. of Digital Forensics Engineering, Firat University, Türkiye & Federal University of Kashere, Nigeria, 2023.

[13] Vaishali Sharma, Neetu Singh, “Deep Convolutional Neural Network with ResNet-50 Learning Algorithm for Copy-Move Forgery Detection”, Electronics and Communication Engineering, Jaypee Institute of Information and Technology, Noida, India, 2023.

[14] Ritu Agarwal, Mallika Pant, “Image Tampering Detection Using Genetic Algorithm”, 2019.

[15] Belal Ahmed, T. Aaron Gulliver, Saif alZahir, “Image Splicing Detection Using Mask-RCNN”, Biometrics: Theory, Applications, and Systems, 2020.