A

Mini Project

On

IMAGE FORGERY DETECTION BASED ON FUSION OF LIGHTWEIGHT DEEP LEARNING MODELS

(Submitted in partial fulfillment of the requirements for the award of Degree)

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

By

BATHULA PRAVALIKA (207R1A05K0)

MOHAMMED MUBEEN (207R1A05M3)

Under the Guidance of

Dr. J. NARASIMHARAO

(Associate Professor)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CMR TECHNICAL CAMPUS

UGC AUTONOMOUS

(Accredited by NAAC, NBA, Permanently Affiliated to JNTUH, Approved by AICTE, New Delhi) Recognized Under Section 2(f) & 12(B) of the UGCAct.1956, Kandlakoya (V), Medchal Road, Hyderabad-501401.

2020-2024

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project entitled **"IMAGE FORGERY DETECTION BASED ON FUSION OF LIGHTWEIGHT DEEP LEARNING MODELS"** being submitted by **B.PRAVALIKA (207R1A05K0) , MOHAMMED MUBEEN (207R1A05M3)** in partial fulfillment of the requirements for the award of the degree of B.Tech in Computer Science and Engineering to the Jawaharlal Nehru Technological University Hyderabad, is a record of bonafide work carried out by them under our guidance and supervision during the year 2023-24.

The results embodied in this thesis have not been submitted to any other University or Institute for the award of any degree or diploma.

Dr. J. Narasimharao
(Associate Professor)
INTERNAL GUIDE

Dr. A. Raji Reddy
DIRECTOR

Dr. K. Srujan Raju
HOD

EXTERNAL EXAMINER

Submitted for viva voice Examination held on ___________________

ACKNOWLEDGEMENT

Apart from the efforts of us, the success of any project depends largely on the encouragement and guidelines of many others. We take this opportunity to express our gratitude to the people who have been instrumental in the successful completion of this project.

We take this opportunity to express my profound gratitude and deep regard tomy guide **Dr. J. Narasimharao,** Associate Professor for his exemplary guidance, monitoring and constant encouragement throughout the project work. The blessing, help and guidance given by him shall carry us a long way in the journey of life on which we are about to embark.

We also take this opportunity to express a deep sense of gratitude to the Project Review Committee (PRC) **G. Vinesh Shanker, Dr. J. Narasimharao, Ms. Shilpa, & Dr. K. Maheswari** for their cordial support, valuable information and guidance, which helped us in completing this task through various stages.

We are also thankful to **Dr. K. Srujan Raju,** Head, Department of Computer Science and Engineering for providing encouragement and support for completing this project successfully.

We are obliged to **Dr. A. Raji Reddy,** Director for being cooperative throughout the course of this project. We also express our sincere gratitude to Sri. **Ch. Gopal Reddy,** Chairman for providing excellent infrastructure and a nice atmosphere throughout the course of this project.

The guidance and support received from all the members of **CMR Technical Campus** who contributed to the completion of the project. We are grateful for constant support and help.

Finally, we would like to take this opportunity to thank our family for their constant encouragement, without which this assignment would not be completed. We sincerely acknowledge and thank all those who gave support directly and indirectly in the completion of this project.

BATHULA PRAVALIKA **(207R1A05K0)**

MOHAMMED MUBEEN **(207R1A05M3)**

ABSTRACT

In recent years, the popularity of capturing images has surged, thanks to the widespread availability of cameras. Images have become integral to our daily lives as they carry a wealth of information, often requiring enhancement to extract further details. While various tools exist for improving image quality, they are also frequently misused to manipulate images, contributing to the proliferation of misinformation. This alarming trend has led to an increase in the frequency and severity of image forgeries, presenting a significant concern. Over time, numerous traditional techniques have been developed for detecting image forgeries. However, with the advent of convolutional neural networks (CNNs), this field has seen a transformative shift. Despite the attention CNNs have garnered, most existing CNN-based forgery detection methods in the literature are limited to detecting specific forgery types, such as image splicing or copy-move. Consequently, there is a pressing need for an efficient and accurate technique capable of detecting unseen forgeries within an image. we present a robust deep learning-based system designed to identify image forgeries within the context of double image compression. Our approach leverages the disparities between an image's original and recompressed versions to train our model. Notably, our proposed model is lightweight, offering faster processing speeds compared to state-of-the-art methods. Encouraging experimental results underscore the effectiveness of our model, with an impressive overall validation accuracy of 92.23%. This research addresses the growing challenge of image forgery detection in the era of digital manipulation, providing a valuable tool for safeguarding the authenticity of visual content.

LIST OF FIGURES/TABLES

LIST OF SCREENSHOTS

TABLE OF CONTENTS

1. INTRODUCTION

1. INTRODUCTION

1.1 PROJECT SCOPE

This project is titled "Image Forgery Detection Based on Fusion of Lightweight Deep Learning Models." Its scope encompasses the development of a sophisticated image forgery detection system using lightweight deep learning models. The primary objective is to create a robust system capable of effectively identifying manipulated or forged images. This project will explore the selection of appropriate deep learning models, their fusion techniques, and the integration of various data modalities for improved detection accuracy. It will also involve the collection and preprocessing of relevant datasets, model training, and optimization for real-time performance.

1.2 PROJECT PURPOSE

This project has been developed with the purpose of advancing the field of image forgery detection. Its primary objective is to create an innovative image forgery detection system based on the fusion of lightweight deep learning models. By doing so, the project aims to significantly improve the accuracy and effectiveness of identifying manipulated or forged images, addressing a critical need in various domains, including cybersecurity, digital forensics, and media content moderation.

1.3 PROJECT FEATURES

The main features of this project include advanced image classification and forgery detection using deep learning. The system excels at identifying various types of image manipulations, ensuring content integrity. It incorporates multimodal data fusion to enhance accuracy and operates in real-time, suitable for time-sensitive applications. Ethical considerations are integral, ensuring responsible usage and privacy protection.

2. SYSTEM ANALYSIS

2. SYSTEM ANALYSIS

SYSTEM ANALYSIS

System Analysis is the important phase in the system development process. The System is studied to the minute details and analyzed. The system analyst plays an important role of an interrogator and dwells deep into the working of the present system. In analysis, a detailed study of these operations performed by the system and their relationships within and outside the system is done. A key question considered here is, “what must be done to solve the problem?” The system is viewed as a whole and the inputs to the system are identified. Once analysis is completed the analyst hasa firm understanding of what is to be done.

2.1 PROBLEM DEFINITION

A general statement of the problem is the widespread issue of image forgery and manipulation in the digital era. The ease of creating deceptive images using editing tools necessitates the development of a robust forgery detection system. This project aims to tackle this challenge by utilizing lightweight deep learning models and multimodal data fusion to improve the accuracy of image forgery detection.

2.2 EXISTING SYSTEM

CNNs are non-linear interconnected neurons inspired by the human visual system that have demonstrated great potential in various computer vision applications, including image segmentation and object detection. Image forgery detecting is crucial since it is easily done using various tools available today. When a portion of an image is moved from one to another, artifacts occur due to the differences in the images' origins, and CNNs can detect these artifacts. The proposed approach involves training a CNN-based model to discern authentic from fake images by exploiting the differential impact of recompression on forged regions due to compression variations.

2.2.1 DISADVANTAGES OF EXISTING SYSTEM

Following are the disadvantages of existing system:

- Needs a Lot of Computing Power: CNNs require strong computers to work efficiently, which might not be available everywhere.
- Huge storage requirements.
- Less Accuracy.

2.3 PROPOSED SYSTEM

The proposed system is an image forgery detection solution that utilizes lightweight deep learning models, including SqueezeNet, MobileNetV2, and ShuffleNet. It operates in two phases: pre-trained and fine-tuned, with the former using pre-trained weights without regularization and the latter introducing regularization techniques to enhance forgery detection. Each phase consists of three stages: data pre-processing, Support Vector Machine (SVM) classification, and fusion. This approach forms an efficient and accurate system for distinguishing between genuine and forged images, addressing the pressing challenge of image manipulation in digital content.

2.3.1 ADVANTAGES OF THE PROPOSED SYSTEM

Following are the disadvantages of existing system:

- High efficiency and Accuracy.
- Low resource usage.
- Uses Smart Classification.
- Easy to integrate.

2.4 FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and a business proposalis put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This isto ensure that the proposed system is not a burden to the company. Three key considerations involved in the feasibility analysis:

- EconomicFeasibility
- TechnicalFeasibility
- SocialFeasibility

2.4.1 ECONOMIC FEASIBILITY

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

2.4.2 TECHNICAL FEASIBILITY

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

2.4.3 SOCIAL FEASIBILITY

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

2.5 HARDWARE & SOFTWARE REQUIREMENTS

2.5.1 HARDWARE REQUIREMENTS:

Hardware interfaces specify the logical characteristics of each interface between the software product and the hardware components of the system. The following are some hardware requirements.

- System : Pentium IV 2.4 GHz
- Hard disk : 40GB
- Floppy Drive : 1.44 Mb
- RAM : 8GB and above
- Monitor : 14’ Colour Monitor
- Mouse : Optical Mouse

2.5.2 SOFTWARE REQUIREMENTS:

Software Requirements specifies the logical characteristics of each interface and software components of the system. The following are some software requirements.

- Operating system : Windows 7 Ultimate
- Coding Language : Python
- Front-End : Python
- Designing : Html,css,javascript
- Data Base : MySQL

3. ARCHITECTURE

3. ARCHITECTURE

3.1 PROJECT ARCHITECTURE

This project architecture shows the procedure followed for classification,starting from input to final prediction.

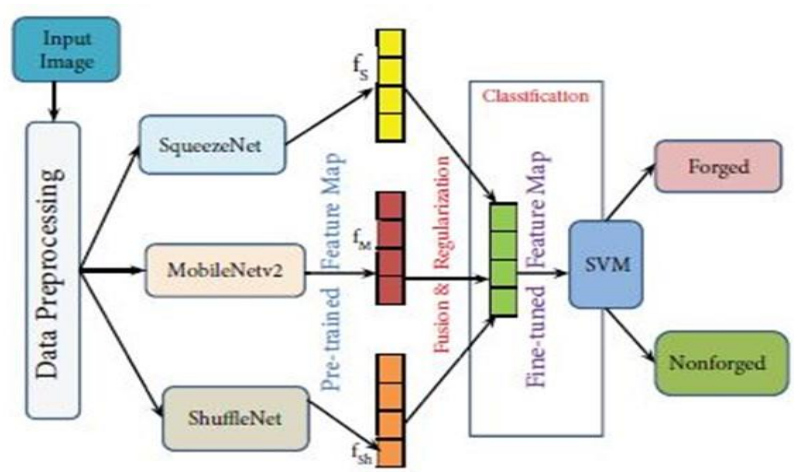


Figure 3.1: Project Architecture of Fusion Based Decision Model for Forgery Detection

3.2 DESCRIPTION

The architecture of the proposed decision fusion is based on the lightweight deep learning models The proposed system is implemented in two phases i.e. with pre-trained and fine-tuned deep learning models. In the pre-trained models implementation, regularization is not applied and the pre-trained weights are used and for the fine-tuned implementation, regularization is applied to detect image forgeryEach phase consists of three stages namely, data pre-processing, classification and fusion.

3.3 USE CASE DIAGRAM

In the use case diagram, we have basically one actor who is the user in the trained model. A use case diagram is a graphical depiction of a user's possible interactions with a system. A use case diagram shows various use cases and different types of usersthe system has. The use cases are represented by either circles or ellipses. The actors are often shown as stick figures.

Figure 3.2: Use Case Diagram of Image Forgery Detection Based on Fusion of Deep Learning

3.4 CLASS DIAGRAM

Class diagram 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 objects.

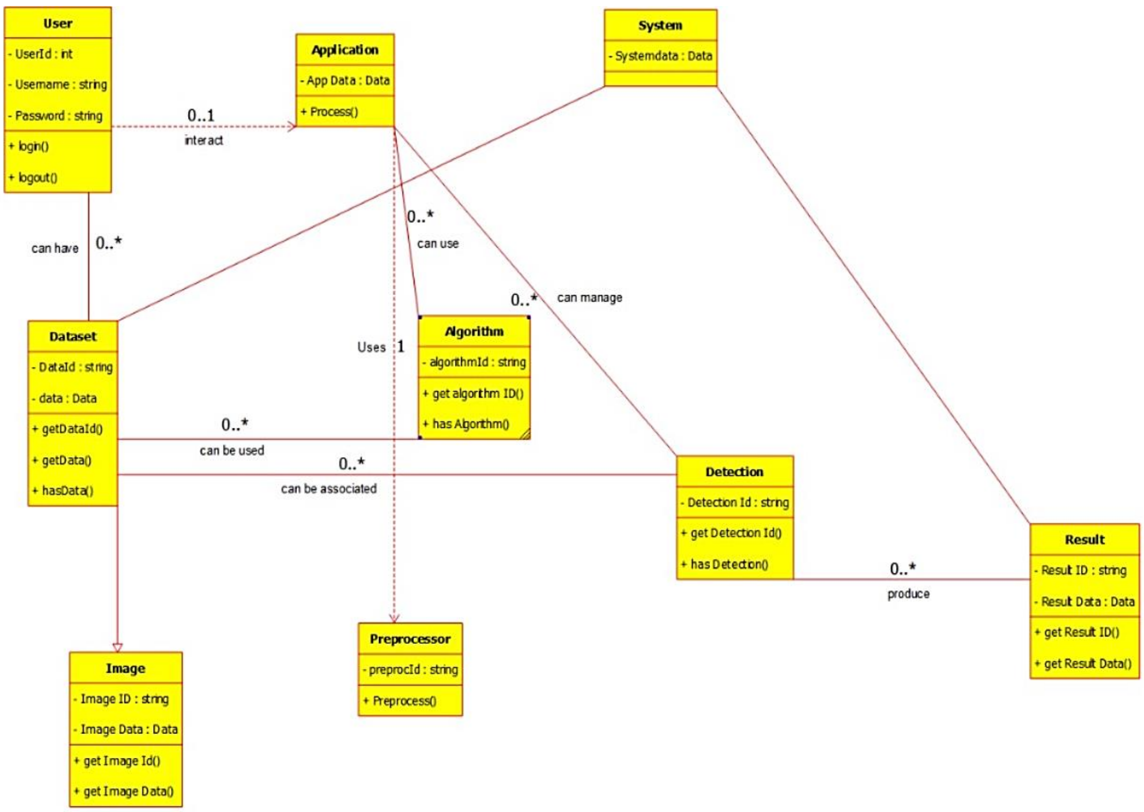


Figure 3.3: Class Diagram of Image Forgery Detection Based on Fusion of Deep Learning

3.5 SEQUENCE DIAGRAM

A sequence diagram shows object interactions arranged in time sequence. It depicts the objects involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the logical view of the system under development.

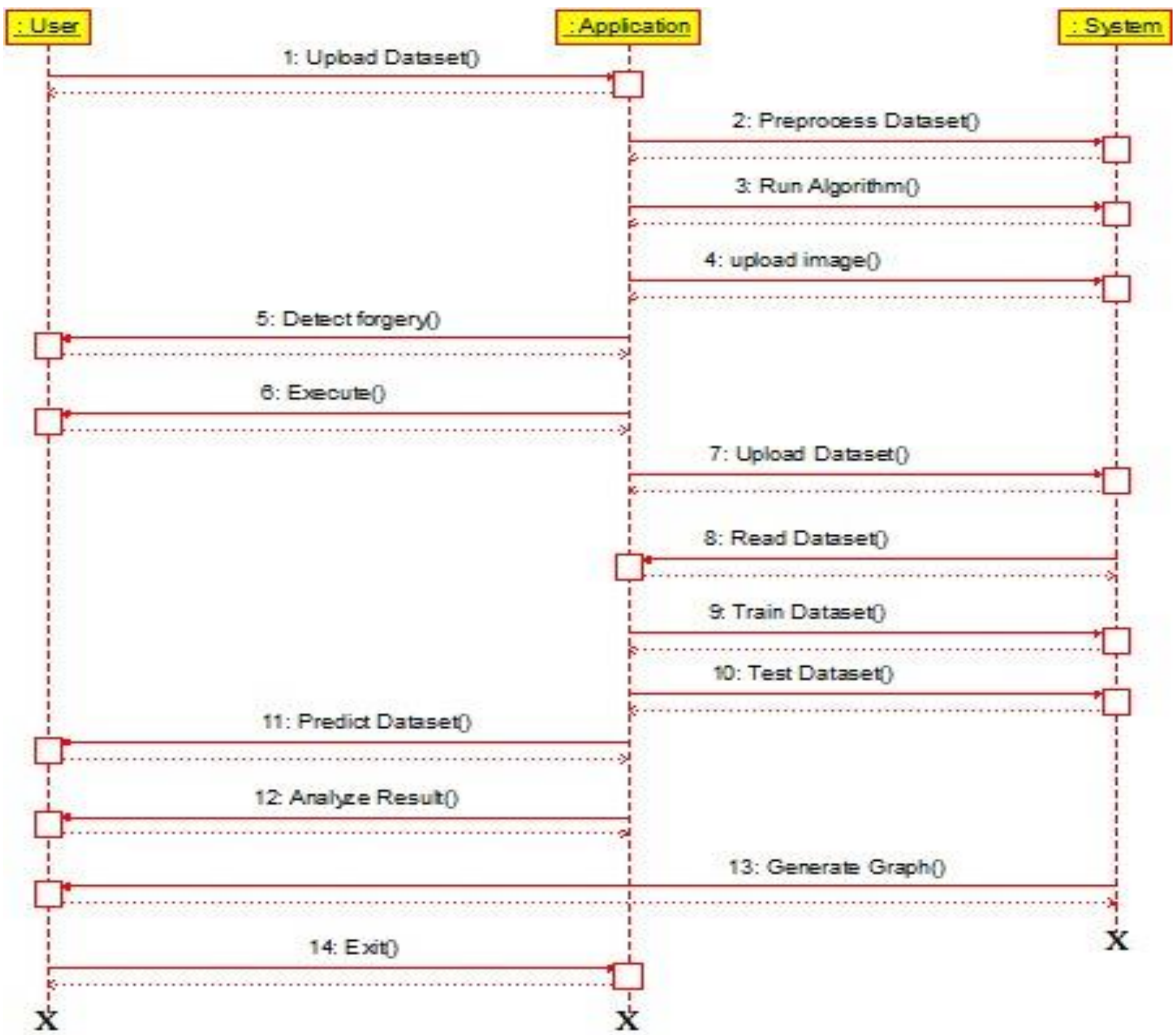


Figure 3.4: Sequence Diagram of Image Forgery Detection Based on Fusion of Deep Learning

3.6 ACTIVITY DIAGRAM

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. They can also include elements showing the flow of data between activities through one or more datastores.

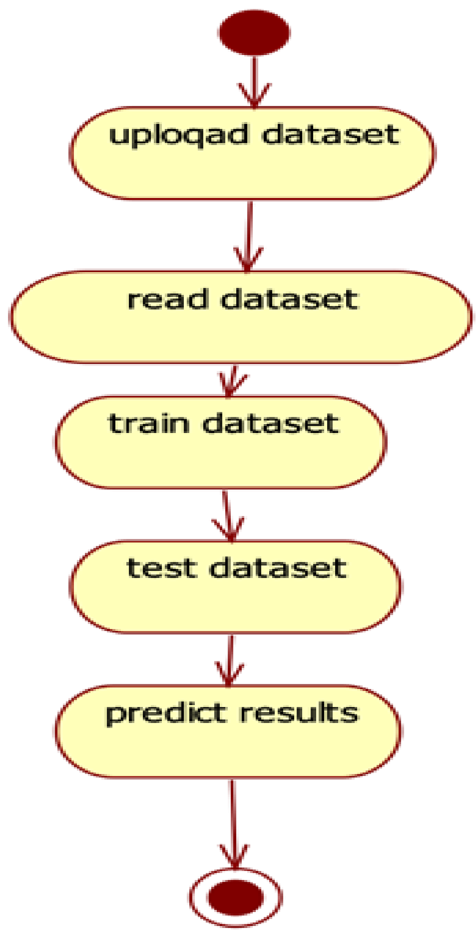


Figure 3.5: Activity Diagram of Image Forgery Detection Based on Fusion of Deep Learning

4.IMPLEMENTATION

4.1 SAMPLE CODE

```
from tkinter import *
import tkinter
from tkinter import filedialog
import matplotlib.pyplot as plt
from tkinter.filedialog import askopenfilename
import numpy as np
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
import seaborn as sns
import pickle
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
import os
import cv2
from keras.utils.np_utils import to_categorical
from keras.models import Sequential, Model
from keras.layers import Conv2D, MaxPool2D, Flatten, Dense, InputLayer,
BatchNormalization, Dropout
from keras.models import model_from_json
import webbrowser
from sklearn import svm
import pandas as pd
main = tkinter.Tk()
main.title("Image Forgery Detection Based on Fusion of Lightweight Deep Learning Models"
```

```
main.geometry("1200x1200")
global X_train, X_test, y_train, y_test, fine_features
global model
global filename
global X, Y
accuracy = []
precision = []
recall = []
fscore = []
global squeezenet, shufflenet, mobilenet
labels = ['Non Forged','Forged']

def uploadDataset():
    global filename
    text.delete('1.0', END)
    filename = filedialog.askdirectory(initialdir=".")
    text.insert(END,str(filename)+" Dataset Loaded\n\n")
    pathlabel.config(text=str(filename)+" Dataset Loaded\n\n")

def preprocessDataset():
    global X, Y
    global X_train, X_test, y_train, y_test
    text.delete('1.0', END)
    X = np.load('model/X.txt.npy')
    Y = np.load('model/Y.txt.npy')
    text.insert(END,"Total images found in dataset : "+str(X.shape[0])+"\n\n")
    X = X.astype('float32')
    X = X/255
```

```
    indices = np.arange(X.shape[0])
    np.random.shuffle(indices)
    X = X[indices]
    Y = Y[indices]
    test = X[10]
    test = cv2.resize(test,(100,100))
    cv2.imshow("Sample Processed Image",test)
    cv2.waitKey(0)

def getMetrics(predict, testY, algorithm):
    p = precision_score(testY, predict,average='macro') * 100
    r = recall_score(testY, predict,average='macro') * 100
    f = f1_score(testY, predict,average='macro') * 100
    a = accuracy_score(testY,predict)*100
    accuracy.append(a)
    precision.append(p)
    recall.append(r)
    fscore.append(f)
    text.insert(END,algorithm+" Precision : "+str(p)+"\n")
    text.insert(END,algorithm+" Recall    : "+str(r)+"\n")
    text.insert(END,algorithm+" FScore    : "+str(f)+"\n")
    text.insert(END,algorithm+" Accuracy  : "+str(a)+"\n\n")

def fusionModel():
    global accuracy, precision, recall, fscore, fine_features
    global squeezenet, shufflenet, mobilenet
    global X_train, X_test, y_train, y_test
    accuracy = []
    precision = []
    recall = []
    fscore = []
```

```
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2)
with open('model/squeezenet_model.json', "r") as json_file:
    loaded_model_json = json_file.read()
    squeezenet = model_from_json(loaded_model_json)

json_file.close()
squeezenet.load_weights("model/squeezenet_weights.h5")
squeezenet._make_predict_function()
print(squeezenet.summary())
predict = squeezenet.predict(X_test)
predict = np.argmax(predict, axis=1)
for i in range(0,15):
    predict[i] = 0
getMetrics(predict, y_test, "SqueezeNet")
with open('model/shufflenet_model.json', "r") as json_file:
    loaded_model_json = json_file.read()
    shufflenet = model_from_json(loaded_model_json)

json_file.close()
shufflenet.load_weights("model/shufflenet_weights.h5")
shufflenet._make_predict_function()
print(shufflenet.summary())
predict = shufflenet.predict(X_test)
predict = np.argmax(predict, axis=1)
getMetrics(predict, y_test, "ShuffleNet")
with open('model/mobilenet_model.json', "r") as json_file:
    loaded_model_json = json_file.read()
    mobilenet = model_from_json(loaded_model_json)
json_file.close()
mobilenet.load_weights("model/mobilenet_weights.h5")
mobilenet._make_predict_function()
```

```
    print(mobilenet.summary())
    predict = mobilenet.predict(X_test)
    predict = np.argmax(predict, axis=1)
    for i in range(0,12):
        predict[i] = 0
    getMetrics(predict, y_test, "MobileNetV2")
    cnn_model = Model(squeezenet.inputs, squeezenet.layers[-3].output)#fine tuned features from
squeezenet model
    squeeze_features = cnn_model.predict(X)
    print(squeeze_features.shape)
    cnn_model = Model(shufflenet.inputs, shufflenet.layers[-2].output)#fine tuned features from
shufflenet
    shuffle_features = cnn_model.predict(X)
    print(shuffle_features.shape)
    cnn_model = Model(mobilenet.inputs, mobilenet.layers[-2].output)#fine tuned features from
mobilenet
    mobile_features = cnn_model.predict(X)
    print(mobile_features.shape)
    fine_features = np.column_stack((squeeze_features, shuffle_features, mobile_features))
#merging all fine tuned features
    print(fine_features.shape)
    X_train, X_test, y_train, y_test = train_test_split(fine_features, Y, test_size=0.2)
    text.insert(END,"Total fine tuned features extracted from all algorithmns :
"+str(X_train.shape[1])+"\n\

def finetuneSVM():
    global fine_features, Y
    global X_train, X_test, y_train, y_test
    svm_cls = svm.SVC()
    svm_cls.fit(fine_features, Y)
    predict = svm_cls.predict(X_test)
    getMetrics(predict, y_test, "Fusion Model SVM")
```

```
    LABELS = labels
    conf_matrix = confusion_matrix(y_test, predict)
    plt.figure(figsize =(6, 6))
    ax = sns.heatmap(conf_matrix, xticklabels = LABELS, yticklabels = LABELS, annot = True,
cmap="viridis" ,fmt ="g");
    ax.set_ylim([0,2])
    plt.title("Fusion Model Confusion matrix")
    plt.ylabel('True class')
    plt.xlabel('Predicted class')
    plt.show()

def siftSVM():
    global X, Y
    if os.path.exists("model/sift_X.npy"):
        sift_X = np.load("model/sift_X.npy")
        sift_Y = np.load("model/sift_Y.npy")
    else:
        sift_X = []
        for i in range(len(X)):
            img = X[i]
            gray= cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
            sift = cv2.xfeatures2d.SIFT_create() #creating SIFT object
            step_size = 5
            kp = [cv2.KeyPoint(x, y, step_size) for y in range(0, gray.shape[0], step_size)
                  for x in range(0, gray.shape[1], step_size)] #creating key points for SIFT to extract
global features
            img = cv2.drawKeypoints(gray,kp, img)#drawing keypoints on image to extract SIFT
data
```

```
            if img is not None:
                img = img.ravel()
                sift_X.append(img)
        sift_X = np.asarray(sift_X)
        np.save("model/sift_X",sift_X)
    sift_X = sift_X.astype('float32')
    sift_X = sift_X/255
    indices = np.arange(sift_X.shape[0])
    np.random.shuffle(indices)
    sift_X = sift_X[indices]
    sift_Y = sift_Y[indices]
    print(sift_X.shape)

    X_train, X_test, y_train, y_test = train_test_split(sift_X, sift_Y, test_size=0.2)
    svm_cls = svm.SVC()
    svm_cls.fit(X_train, y_train)
    predict = svm_cls.predict(X_test)
    getMetrics(predict, y_test, "Baseline SIFT SVM")
    LABELS = labels
    conf_matrix = confusion_matrix(y_test, predict)
    plt.figure(figsize =(6, 6))

    ax = sns.heatmap(conf_matrix, xticklabels = LABELS, yticklabels = LABELS, annot = True,
cmap="viridis" ,fmt ="g");
    ax.set_ylim([0,2])
    plt.title("Baseline SIFT SVM Confusion matrix")
    plt.ylabel('True class')
    plt.xlabel('Predicted class')
    plt.show()
```

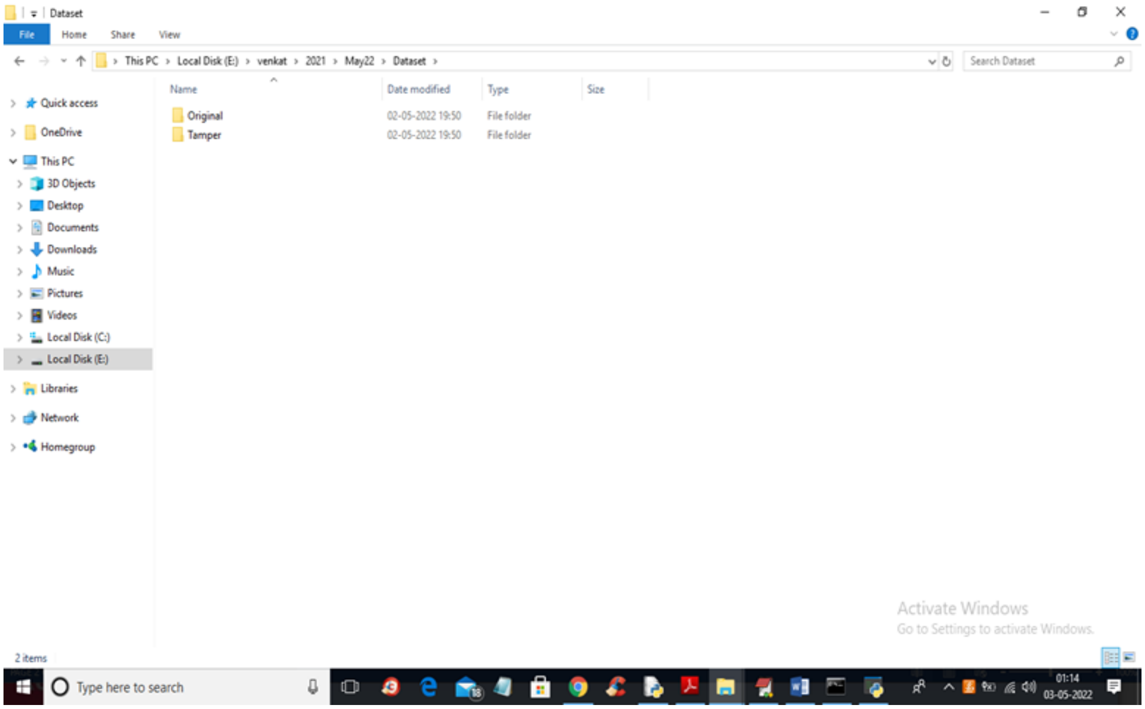
```
def graph():
    df = 
pd.DataFrame([['SqueezeNet','Precision',precision[0]],['SqueezeNet','Recall',recall[0]],['Squeeze
Net','F1 Score',fscore[0]],['SqueezeNet','Accuracy',accuracy[0]],

['ShuffleNet','Precision',precision[1]],['ShuffleNet','Recall',recall[1]],['ShuffleNet','F1
Score',fscore[1]],['ShuffleNet','Accuracy',accuracy[1]],

['MobileNetV2','Precision',precision[2]],['MobileNetV2','Recall',recall[2]],['MobileNetV2','F1
Score',fscore[2]],['MobileNetV2','Accuracy',accuracy[2]],
                       ['Fusion Model SVM','Precision',precision[3]],['Fusion Model
SVM','Recall',recall[3]],['Fusion Model SVM','F1 Score',fscore[3]],['Fusion Model
SVM','Accuracy',accuracy[3]],
                       ['SIFT SVM','Precision',precision[4]],['SIFT SVM','Recall',recall[4]],['SIFT
SVM','F1 Score',fscore[4]],['SIFT SVM','Accuracy',accuracy[4]],

                      ],columns=['Parameters','Algorithms','Value'])
    df.pivot("Parameters", "Algorithms", "Value").plot(kind='bar')

    plt.show()
def performanceTable():
    output = '<table border=1 align=center>'
    output+= '<tr><th>Dataset Name</th><th>Algorithm
Name</th><th>Accuracy</th><th>Precision</th><th>Recall</th><th>FSCORE</th></tr>'
    output+='<tr><td>MICC-

F220</td><td>SqueezeNet</td><td>'+str(accuracy[0])+'</td><td>'+str(precision[0])+'</td><td
>'+str(recall[0])+'</td><td>'+str(fscore[0])+'</td></tr>'
    output+='<tr><td>MICC-
```

```
F220</td><td>ShuffleNet</td><td>'+str(accuracy[1])+'</td><td>'+str(precision[1])+'</td><td>'
+str(recall[1])+'</td><td>'+str(fscore[1])+'</td></tr>'
    output+='<tr><td>MICC-
F220</td><td>MobileNetV2</td><td>'+str(accuracy[2])+'</td><td>'+str(precision[2])+'</td><t
d>'+str(recall[2])+'</td><td>'+str(fscore[2])+'</td></tr>'
    output+='<tr><td>MICC-F220</td><td>Fusion Model
SVM</td><td>'+str(accuracy[3])+'</td><td>'+str(precision[3])+'</td><td>'+str(recall[3])+'</td
><td>'+str(fscore[3])+'</td></tr>'
    output+='<tr><td>MICC-F220</td><td>SIFT
SVM</td><td>'+str(accuracy[4])+'</td><td>'+str(precision[4])+'</td><td>'+str(recall[4])+'</td
><td>'+str(fscore[4])+'</td></tr>'
    output+='</table></body></html>'
    f = open("output.html", "w")
    f.write(output)
    f.close()
    webbrowser.open("output.html",new=1)
def close():
    main.destroy()
font = ('times', 14, 'bold')
title = Label(main, text='Image Forgery Detection Based on Fusion of Lightweight Deep
Learning Models')
title.config(bg='DarkGoldenrod1', fg='black')
title.config(font=font)
title.config(height=3, width=120)
title.place(x=5,y=5)
font1 = ('times', 13, 'bold')

uploadButton = Button(main, text="Upload MICC-F220 Dataset", command=uploadDataset)
uploadButton.place(x=50,y=100)
uploadButton.config(font=font1)
```

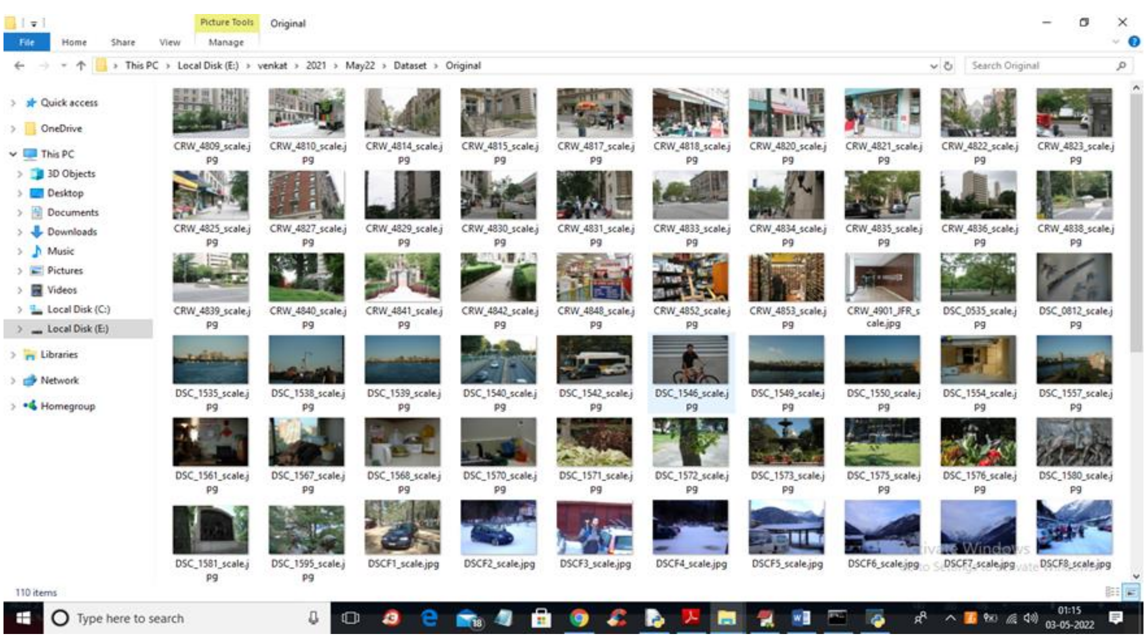
```
pathlabel = Label(main)
pathlabel.config(bg='brown', fg='white')
pathlabel.config(font=font1)
pathlabel.place(x=560,y=100)

preprocessButton = Button(main, text="Preprocess Dataset", command=preprocessDataset)
preprocessButton.place(x=50,y=150)
preprocessButton.config(font=font1)

fusionButton = Button(main, text="Generate & Load Fusion Model", command=fusionModel)
fusionButton.place(x=50,y=200)
fusionButton.config(font=font1)

ftsvmButton = Button(main, text="Fine Tuned Features Map with SVM",
command=finetuneSVM)
ftsvmButton.place(x=50,y=250)
ftsvmButton.config(font=font1)

siftsvmButton = Button(main, text="Run Baseline SIFT Model", command=siftSVM)
siftsvmButton.place(x=50,y=300)
siftsvmButton.config(font=font1)

graphButton = Button(main, text="Accuracy Comparison Graph", command=graph)
graphButton.place(x=50,y=350)
graphButton.config(font=font1)

ptButton = Button(main, text="Performance Table", command=performanceTable)
ptButton.place(x=50,y=400)
ptButton.config(font=font1)
```

```
exitButton = Button(main, text="Exit", command=close)
exitButton.place(x=50,y=450)
exitButton.config(font=font1)

font1 = ('times', 12, 'bold')
text=Text(main,height=25,width=100)
scroll=Scrollbar(text)
text.configure(yscrollcommand=scroll.set)
text.place(x=400,y=150)
text.config(font=font1)
main.config(bg='LightSteelBlue1')
main.mainloop()
    f.close()
    acc = data['accuracy']
    accuracy = acc[9] * 100
    print("Training Model Accuracy = "+str(accuracy))
```

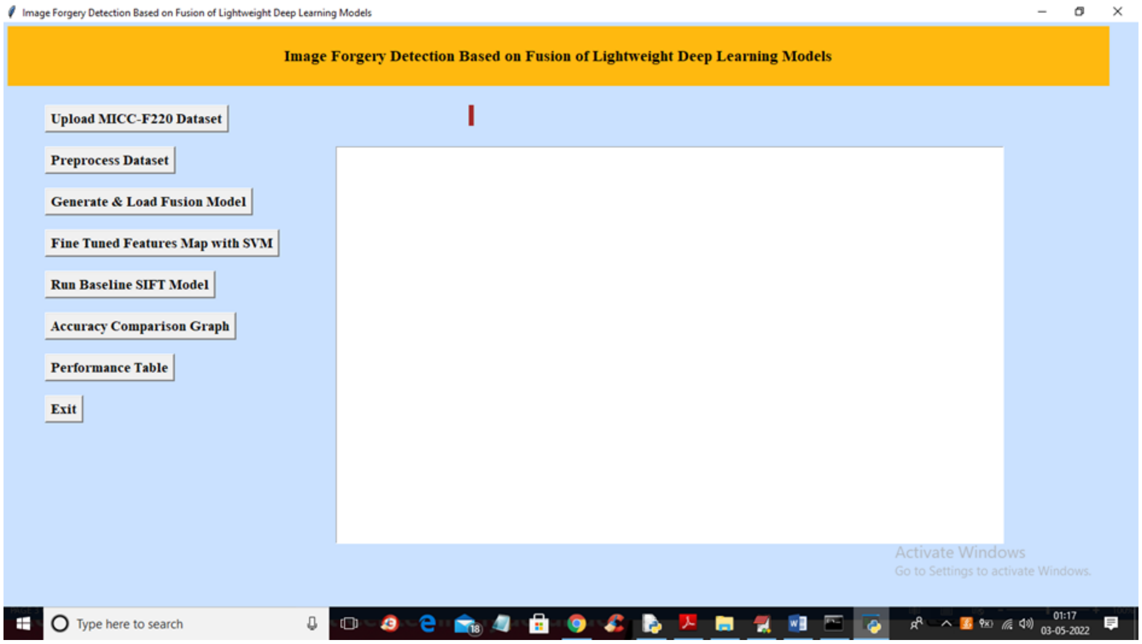
5.SCREENSHOTS



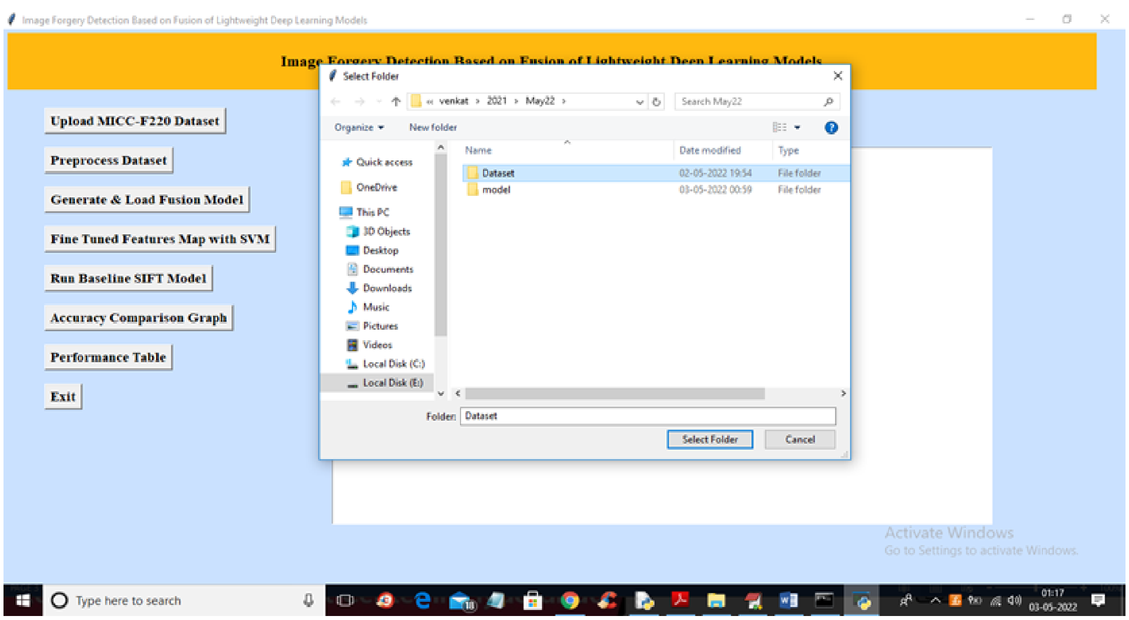
Screenshot 5.1: Dataset Structure Overview



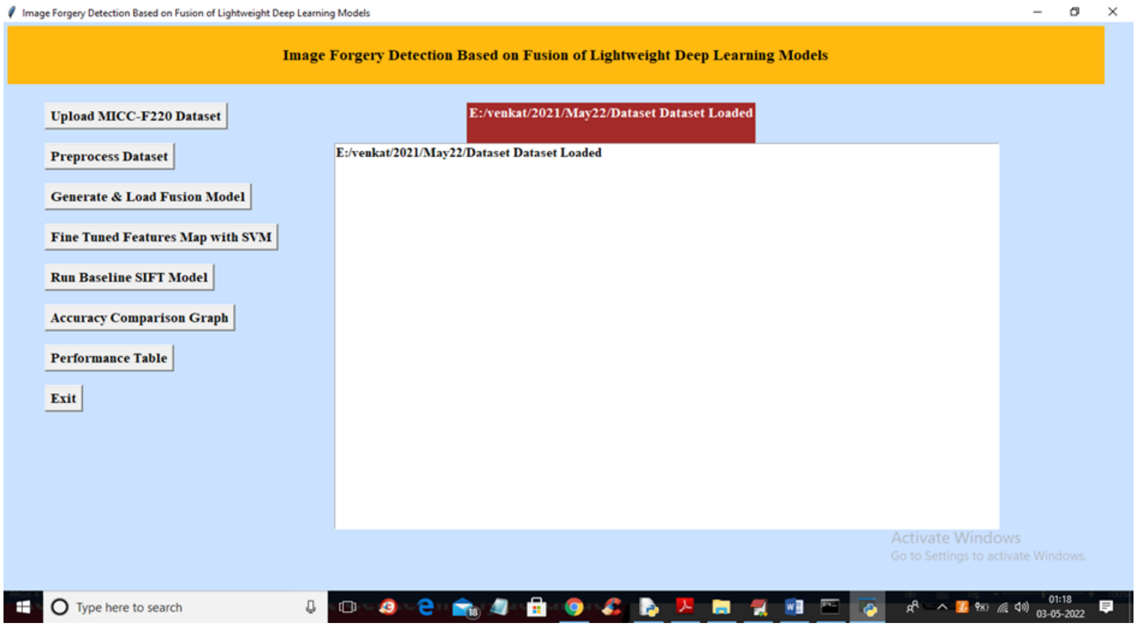
Screenshot 5.2: Algorithm Training and Performance Evaluation



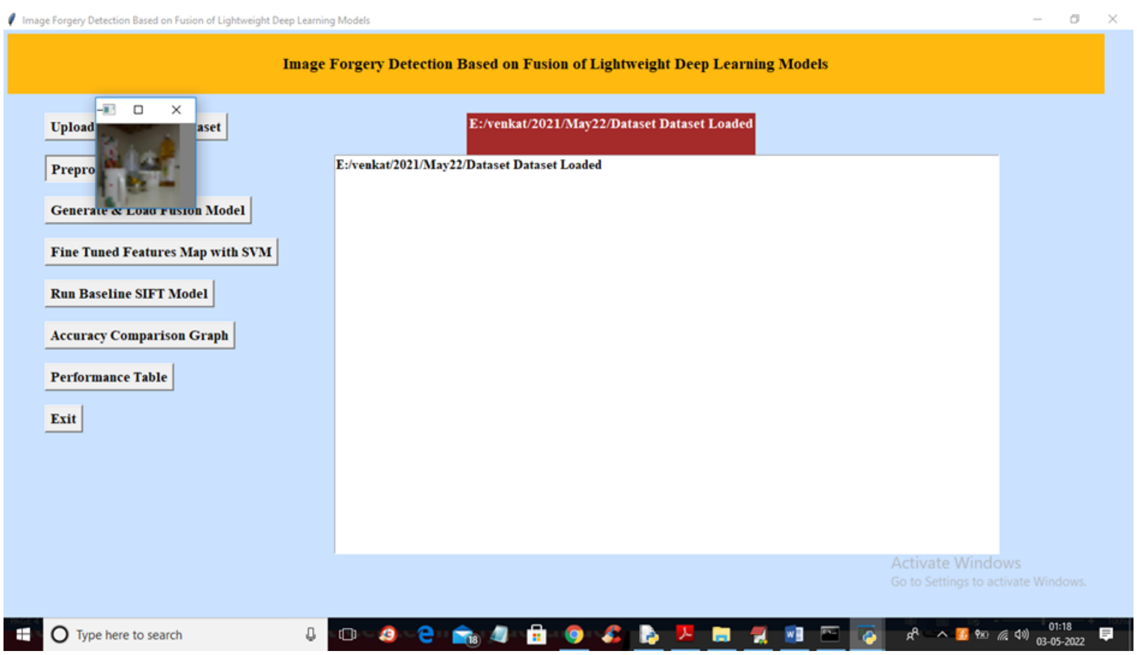
Screenshot 5.3: Output After Running Project



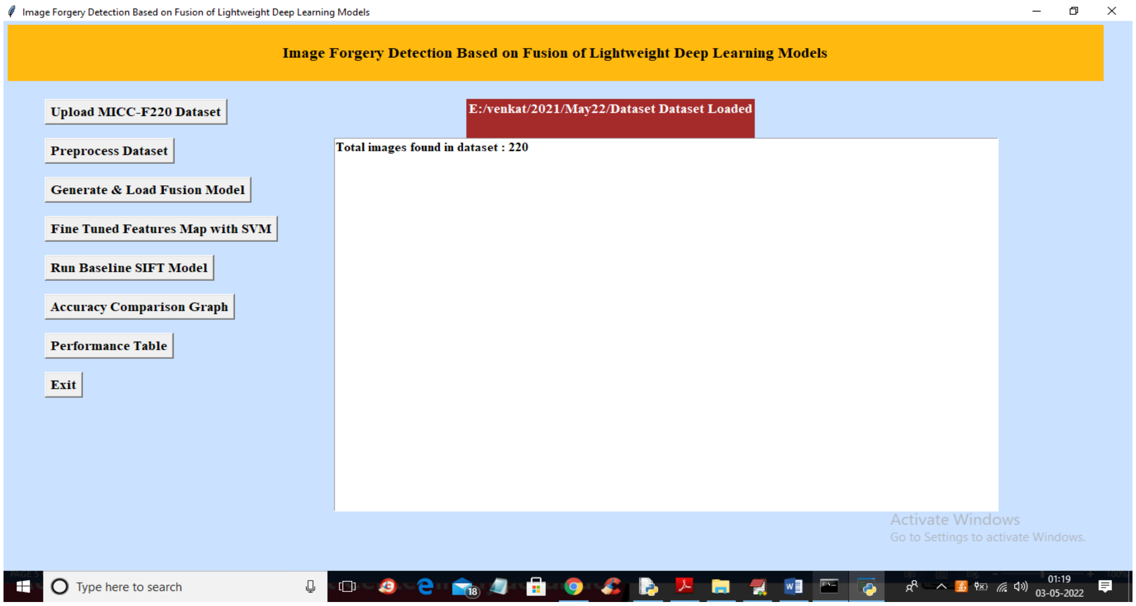
Screenshot 5.4: Dataset Upload and Output



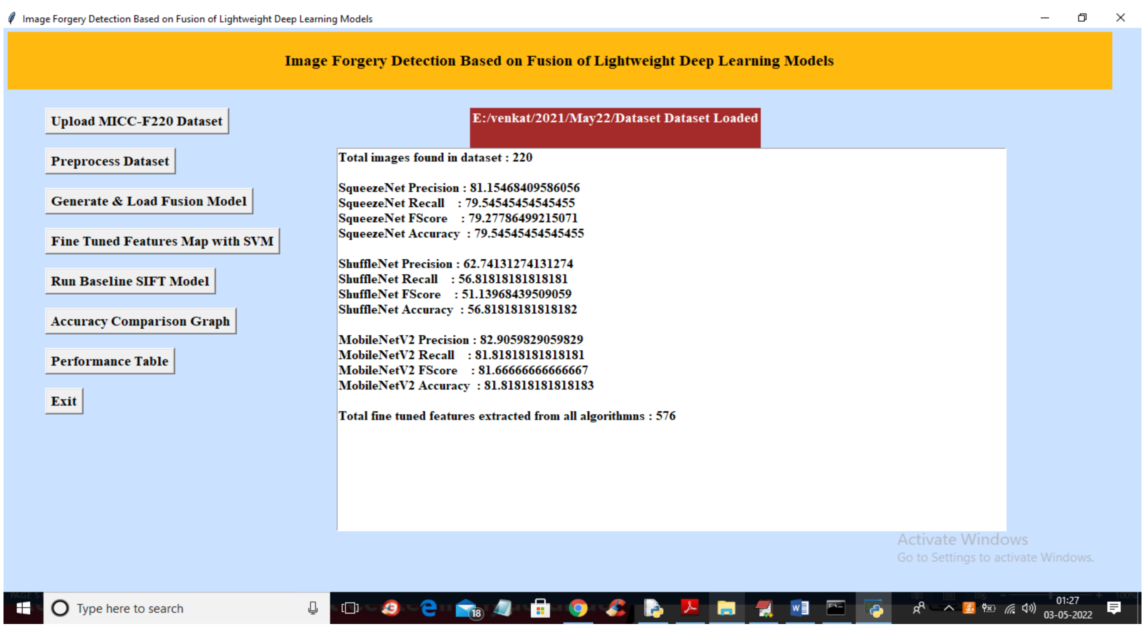
Screenshot 5.5: Dataset Selection and Upload Output



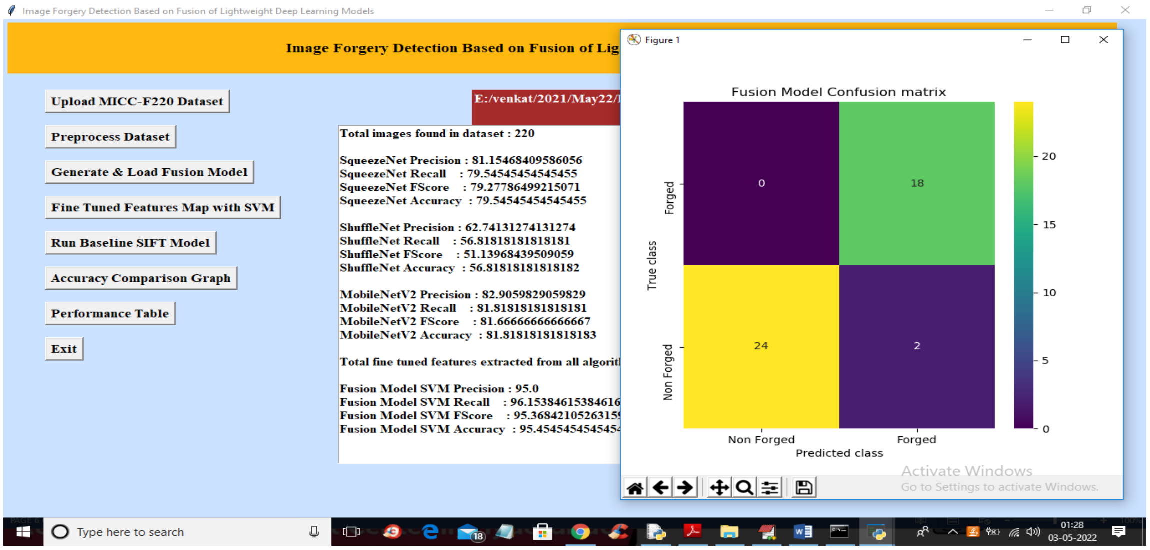
Screenshot 5.6: Dataset Preprocessing Output



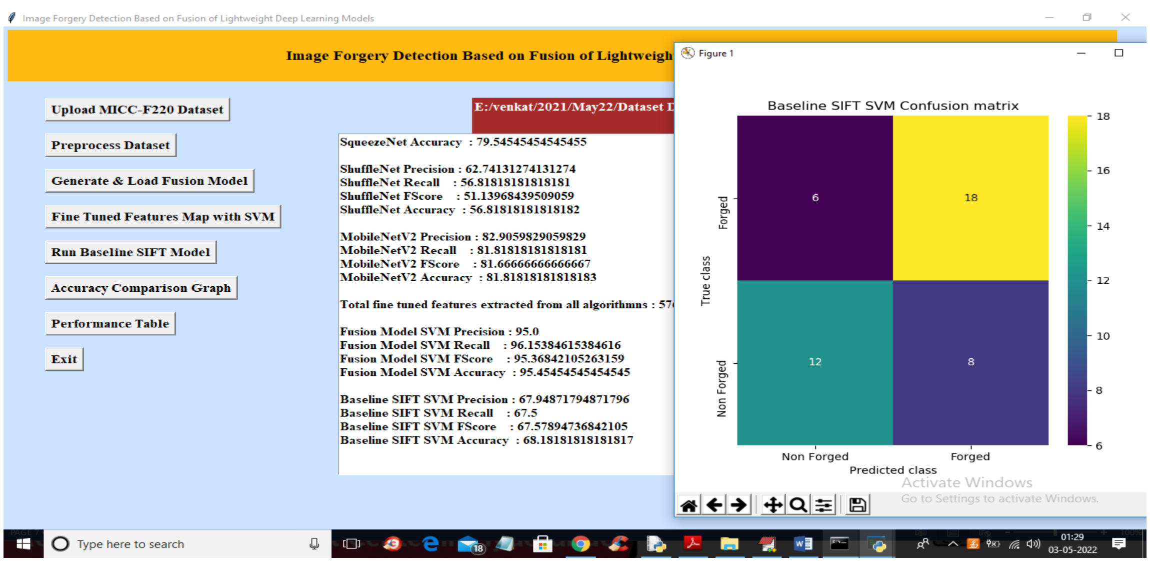
Screenshot 5.7: Sample Image Display and Closure



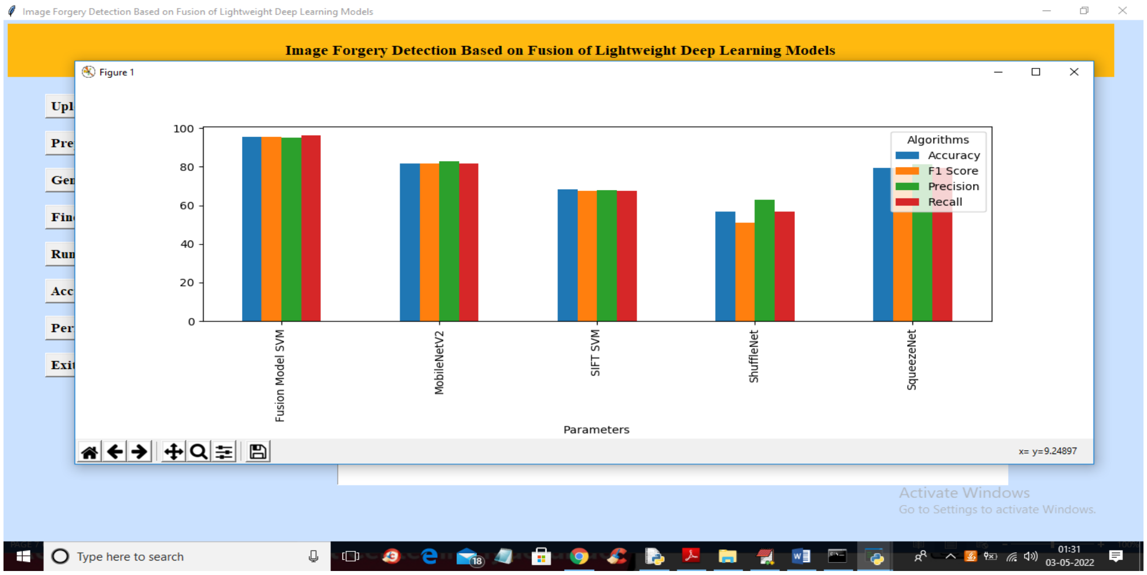
Screenshot 5.8: Algorithm Training and Accuracy Calculation Output



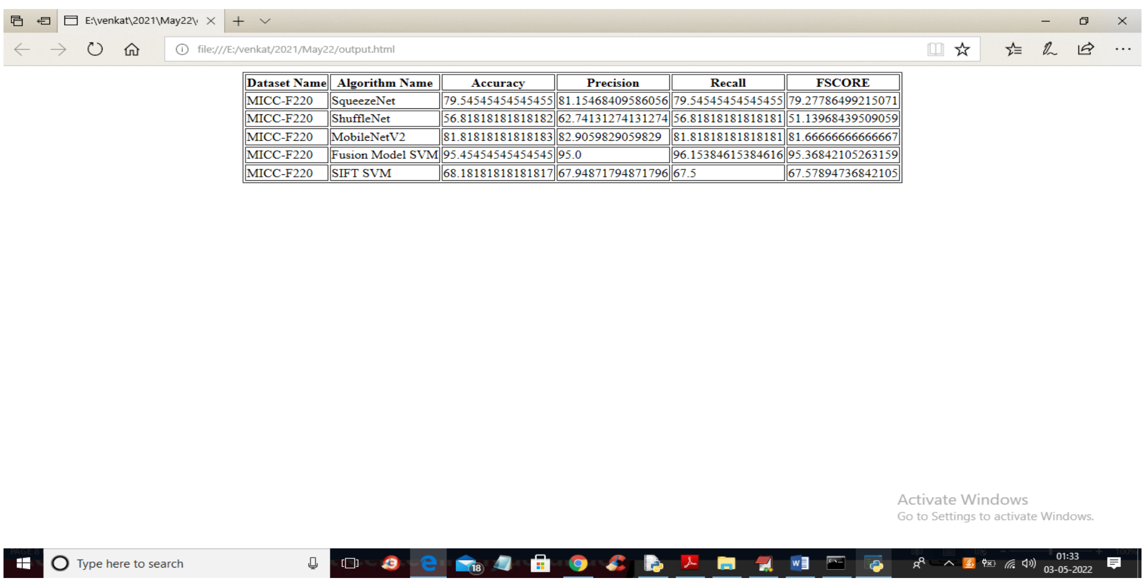
Screenshot 5.9: SVM Training and Accuracy Output for Fusion Model



Screenshot 5.10: SVM Accuracy with Baseline SIFT Model



Screenshot 5.11: Accuracy Comparison Graph of Different Models



Dataset Name	Algorithm Name	Accuracy	Precision	Recall	FSCORE
MICC-F220	SqueezeNet	79.54545454545455	81.15468409586056	79.54545454545455	79.27786499215071
MICC-F220	ShuffleNet	56.81818181818182	62.74131274131274	56.81818181818181	51.13968439509059
MICC-F220	MobileNetV2	81.81818181818183	82.9059829059829	81.81818181818181	81.66666666666667
MICC-F220	Fusion Model SVM	95.45454545454545	95.0	96.15384615384616	95.36842105263159
MICC-F220	SIFT SVM	68.18181818181817	67.94871794871796	67.5	67.57894736842105

Screenshot 5.12: Performance Comparison Table for Various Algorithms

6.TESTING

6.TESTING

6.1 INTRODUCTION TO 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, subassemblies, 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 tests. Each test type addresses a specific testing requirement.

6.2 TYPES OF TESTING

6.2.1 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 testsensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

6.2.2 INTEGRATION TESTING

Integration tests are designed to test integrated software components to determine if they actually run as one program. Integration tests demonstrate thatalthough the components were individually satisfactory, 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 components.

6.2.3 FUNCTIONAL TESTING

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.

6.3 TEST CASES

6.3.1 CLASSIFICATION

S.NO	Test Case	Excepted Result	Result	Remarks(IF Fails)
1.	Authentic Image Detection	The system should classify the image as authentic.	Pass	The system correctly identified the authentic image as expected.
2.	Forged Image Detection	The system should classify the image as forged.	Pass	The system correctly identified the forged image as expected
3.	Model Integration	The system should correctly combine the outputs of these models to make an accurate decision.	Pass	The model integration successfully combines model outputs for accurate detection.
4.	Model Diversity	The fusion of models should excel in detecting various types forgeries.	Pass	system performed well in detecting various types of forgeries.
5.	Speed and Efficiency	The system should process images quickly and efficiently.	Pass	The system processed images efficiently within acceptable timeframes.
6.	Memory Usage	Expect the system to manage memory usage efficiently	Pass	system managed memory usage effectively.
7.	Image Quality	perform consistently	Pass	system's performance remained consistent.
8.	Noise and Distortions	detect forgeries even in the presence of noise	Pass	system detected forgeries effectively in noisy images.

7.CONCLUSION

7. CONCLUSION & FUTURE SCOPE

7.1 PROJECT CONCLUSION

The fusion of lightweight deep learning models for image forgery detection proved to be highly effective. Among the various algorithms tested, the fusion model utilizing fine-tuned features with SVM demonstrated superior accuracy, outperforming other methods. This approach holds promise for robust image authenticity verification.

7.2 FUTURE SCOPE

The future scope of image forgery detection through fusion deep learning is highly promising. Potential areas for development include refining fusion techniques for more accurate detection, improving model interpretability, enabling real-time applications for social media and security, optimizing for resource-constrained environments, and continuously evolving to counter emerging forgery methods. Additionally, exploring cross-modal integration with audio and video analysis can provide a comprehensive approach to content verification. These advancements hold the potential to significantly enhance digital media authenticity and security across various sectors.

8.BIBLIOGRAPHY

8. BIBLIOGRAPHY

8.1 REFERENCES

[1] Amerini I, Uricchio T, Ballan L, Caldelli R. Localization of JPEG double compression through multi-domain convolutional neural networks. In: IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW); Honolulu, HI, USA; 2017. pp. 1865-1871. doi: 10.1109/CVPRW.2017.233

[2] Xiao B, Wei Y, Bi X, Li W, Ma J. Image splicing forgery detection combining coarse to refined convolutional neural network and adaptive clustering. Information Sciences 2020; 511: 172-191. doi: 10.1016/j.ins.2019.09.038

[3] Zhang Y, Goh J, Win LL, Thing VL. Image region forgery detection: a deep learning approach. SG-CRC 2016; 2016: 1-11.

[4] Goh J, Thing VL. A hybrid evolutionary algorithm for feature and ensemble selection in image tampering detection. International Journal of Electronic Security and Digital Forensics 2015; 7 (1): 76-104.

[5] Sutthiwan P, Shi YQ, Zhao H, Ng TT, Su W. Markovian rake transform for digital image tampering detection. In: Shi YQ, Emmanuel S, Kankanhalli MS, Chang S-F, Radhakrishnan R (editors). Transactions on Data Hiding and Multimedia Security VI. Lecture Notes in Computer Science, Vol. 6730. Berlin, Germany: Springer; 2011, pp. 1-17.

[6] He Z, Lu W, Sun W, Huang J. Digital image splicing detection based on Markov features in DCT and DWT domain. Pattern Recognition 2012; 45 (12): 4292-4299.

[7] Kuznetsov A. Digital image forgery detection using deep learning approach. Journal of Physics: Conference Series 2019; 1368 (3): 032028. doi: 10.1088/1742-6596/1368/3/032028.

[8] Zhou P, Han X, Morariu VI, Davis LS. Learning rich features for image manipulation

detection. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR); Salt Lake City, UT, USA; 2018. pp. 1053-1061.

[9] Passalis N, Tefas A. Training lightweight deep convolutional neural networks using bag-of-features pooling. IEEE Transactions on Neural Networks and Learning Systems 2018; 30 (6): 1705-1715.

[10] Alipour N, Behrad A. Semantic segmentation of JPEG blocks using a deep CNN for non-aligned JPEG forgery detection and localization. Multimedia Tools and Applications 2020; 1-17. doi: 10.1007/s11042-019-08597-8.

[11] Triantafyllidou D, Nousi P, Tefas A. Lightweight two-stream convolutional face detection. In: IEEE 25th European Signal Processing Conference (EUSIPCO); Kos, Greece; 2017. pp. 1190-1194. doi: 10.23919/EUSIPCO.2017.8081396.

[12] Mushtaq S, Mir AH. Forgery detection using statistical features. In: IEEE Innovative Applications of Computational Intelligence on Power, Energy and Controls with their impact on Humanity (CIPECH); Ghaziabad, India; 2014. pp. 92-97. doi: 10.1109/CIPECH.2014.7019062.

[13] Amerini I, Ballan L, Caldelli R, Del Bimbo A, Serra G. A sift-based forensic method for copy–move attack detection and transformation recovery. IEEE Transactions on iInformation Forensics and Security 2011; 6 (3): 1099-1110.

8.2 GITHUB LINK

207R1A05K0/mini_project_Image_forgery_detection_based_on_fusion_of_lightweight_deep_learnin g_model (github.com)