**A Major Project**

**on**

**IMAGE FORGERY DETECTION BASED ON FUSION OF LIGHTWEIGHT DEEP LEARNING MODELS**

***Submitted to***

***JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY, HYDERABAD***

*In Partial fulfilment of the requirement for the award of degree of the*

## BACHELOR OF TECHNOLOGY

## IN

**COMPUTER SCIENCE AND ENGINEERING**

*By*

**K. VISHNUVARDHAN REDDY 19RA1A0553**

**N. SAI 19RA1A0532**

**M. UDAY 19RA1A0547**

*Under the guidance of*

**Mrs.** **G. SUJATHA**

**HOD, CSE Dept.**

**Department of Computer Science and Engineering**

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

KOMMURI PRATAP REDDY INSTITUTE OF TECHNOLOGY

(Affiliated to JNTUH, Ghanpur(V), Ghatkesar(M), Medchal(D)-506345)

### *2019-2023*

**KOMMURI PRATAP REDDY INSTITUTE OF TECHNOLOGY**

(Affiliated to JNTUH, Ghanpur(V), Ghatkesar(M), Medchal(D)-506345)



# **CERTIFICATE**

This is to certify that the Major Project entitled “ Image Forgery Detection Based On Fusion Of Lightweight Deep Learning Models ” is submitted by Mr. K. Vishnuvardhan Reddy, Mr. N. Sai, Mr. M. Uday, bonafided student of **Kommuri Pratap Reddy Institute of Technology** in partial fulfilment of the requirement for the award of the degree of Bachelor of Technology in **Computer Science and Engineering** of the **Jawaharlal Nehru Technological University Hyderabad** during the year 2022-23.

## Internal Guide HOD

## Mrs. G. Sujatha Mrs. G. Sujatha

**Project Coordinator External Examiner**

Dr. C. Veena

**DECLARATION**

We hereby declare that this project work entitled “**Image Forgery Detection Based On Fusion Of Lightweight Deep Learning Models**” in partial fulfilment of requirements for the award of degree of **Computer Science and Engineering** is a bonafided work carried out by us during the academic year 2022 - 23.

We further declare that this project is a result of our effort and has not been submitted for the award of any degree by us to any institute.

By

K. Vishnuvardhan Reddy (19RA1A0553)

N. Sai (19RA1A0532)

M. Uday (19RA1A0547)

**ACKNOWLEDGEMENT**

It gives us immense pleasure to acknowledge with gratitude, the help and support extended throughout the project from the following:

We will be very much grateful to almighty our **Parents** who have made us capable of carrying out our job.

We express our profound gratitude to **Dr. D. Eshwar, Principal of Kommuri Pratap Reddy Institute of Technology**, who has provided with infrastructure and necessary resources in completing our project successfully.

We are grateful to **Mrs. G. Sujatha** who is our **Head of the Department**, **CSE** for her amiable ingenious and adept suggestions and pioneering guidance during the project.

We express our gratitude and thanks to the Project Coordinator **Dr. C. Veena, Professor** of our department for her contribution for making it success with in the given time duration.

We express our deep sense of gratitude and thanks to **Internal Guide Mrs. G. Sujatha,** **Assistant Professor** for her guidance during the project.

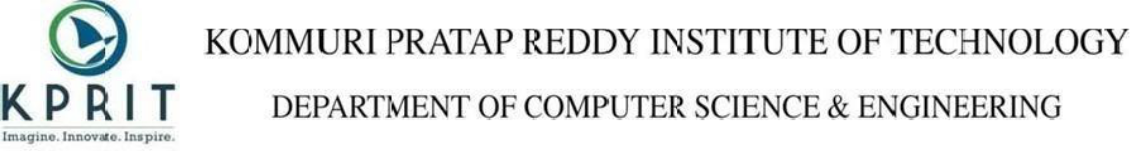
We are also very thankful to our Management, Staff Members and all Our Friends for their valuable suggestions and timely guidance without which we would not have been completed it.

*By*

K. Vishnuvardhan Reddy (19RA1A0553)

N. Sai (19RA1A0532)

M. Uday (19RA1A0547)

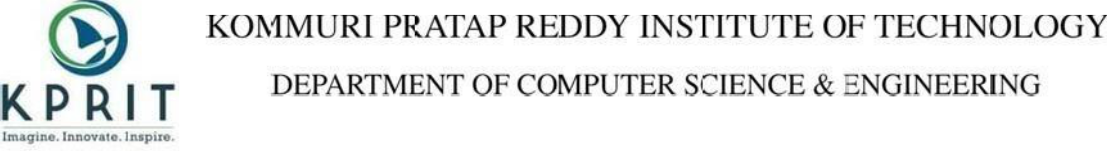
****

**Vision of the Institute**

To emerge as a premier institute for high quality professional graduates who can contribute to economic and social developments of the Nation.

**Mission of Institute**

|  |  |
| --- | --- |
| **Mission** | **Statement** |
| **DM1** | Laying the path for rich skills in Computer Science through The basic knowledge of mathematics and fundamentals of engineering |
| **DM2** | Provide latest tools and technology to the students as a part of learning Infrastructure |
| **DM3** | Training the students towards employability and entrepreneurship to meet the societal needs. |
| **DM4** | Grooming the students with professional and social ethics. |

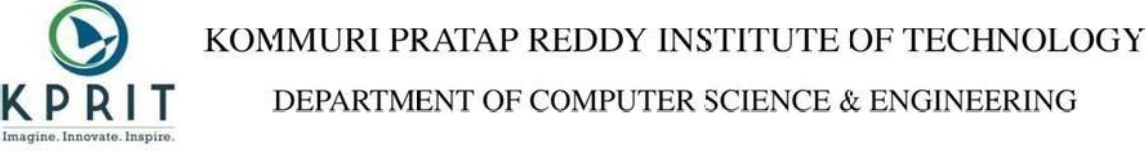
******

**Vision of the Department**

To Provide Quality Education in Computer Science for the innovative professionals to work for the development of the nation.

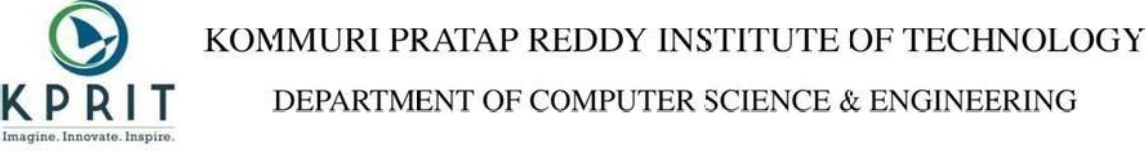
**Mission of the Department**

|  |  |
| --- | --- |
| **Mission** | **Statement** |
| **DM1** | Laying the path for rich skills in Computer Science through The basic knowledge of mathematics and fundamentals of engineering |
| **DM2** | Provide latest tools and technology to the students as a part of learning Infrastructure |
| **DM3** | Training the students towards employability and entrepreneurship to meet the societal needs. |
| **DM4** | Grooming the students with professional and social ethics. |

******

**Program Educational Objectives (PEOs)**

|  |  |
| --- | --- |
| **PEO’s** | **Statement** |
| **PEO1** | The graduates of Computer Science and Engineering will have successful career in technology. |
| **PEO2** | The graduates of the program will have solid technical and professional foundation to continue higher studies. |
| **PEO3** | The graduate of the program will have skills to develop products, offer services and innovation |
|
| **PEO4** | The graduates of the program will have fundamental awareness of industry process, tools and technologies. |

******

**Program Outcomes**

|  |  |
| --- | --- |
| **PO1** | **Engineering Knowledge:** Apply the knowledge of mathematics, science, Engineering fundamentals, and an engineering specialization to the solution of complex engineering problems. |
| **PO2** | **Problem Analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences. |
| **PO3** | **Design/development of Solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations. |
| **PO4** | **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions. |
| **PO5** | **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations. |
| **PO6** | **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice. |
| **PO7** | **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental context, and demonstrate the knowledge of, and need for sustainable development. |

|  |  |
| --- | --- |
| **PO8** | Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice. |
| **PO9** | Individual and team network: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings. |
| **PO10** | Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, being able to comprehend and write effective reports and design documentation, make Effective presentations, and give and receive clear instructions. |
| **PO11** | Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one’s own work, as a member and leader in a team, to manage projects and in a multidisciplinary environment. |
| **PO12** | Life-Long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change. |

**PROGRAM SPECIFIC OUTCOMES**

|  |  |
| --- | --- |
| **PSO1** | Foundation of mathematical concepts: To use mathematical methodologies to crack problems using suitable mathematical analysis, data structure and suitable algorithms. |
| **PSO2** | Foundation of Computer Science: The ability to interpret the fundamental concepts and methodology of computer systems. Students can understand the functionality of hardware and software aspects of computer systems. |
| **PSO3** | Foundation of Software development: The ability to grasp the software development lifecycle and methodologies of software systems. Possess competent skills and knowledge of the software design process. |

**ABSTRACT**

In this digital era, images and videos are being used as influential sources of evidence in a variety of contexts like evidence during trials, insurance fraud, social networking, etc. The easy adaptability of editing tools for digital images, especially without any visual proof of manipulation, give rise to questions about their authenticity. It is the job of image forensics authorities to develop technological innovations that would detect the forgeries of images. There are three primary classes of manipulation or forgery detectors studies until now: those supported features descriptors, those supported inconsistent shadows and eventually those supported double JPEG compression.

Image forgery detection is one of the key challenges in various real time applications, social media and online information platforms. The conventional methods of detection based on the traces of image manipulations are limited to the scope of predefined assumptions like hand-crafted features, size and contrast. In this paper, we propose a fusion based decision approach for image forgery detection. The fusion of decision is based on the lightweight deep learning models namely SqueezeNet, MobileNetV2 and ShuffleNet. The fusion decision system is implemented in two phases. First, the pretrained weights of the lightweight deep learning models are used to evaluate the forgery of the images. Secondly, the fine-tuned weights are used to compare the results of the forgery of the images with the pre-trained models. The experimental results suggest that the fusion-based decision approach achieves better accuracy as compared to the state-of-the-art approaches.

**TABLE OF CONTENTS**

ABSTRACT……………………………………………………………………………………I

LIST OF FIGURES………………...………………………………………………….……..IV

LIST OF SCREENSHOTS………………………………………………………………........V

1. INTRODUCTION…………………………………………………………………………..1

2. LITERATURE SURVEY…………………………………………………………………...3

3. SYSTEM ANALYSIS…………………………………………………………………........6

3.1 SOFTWARE DEVELOPMENT LIFE CYCLE…………………………………...6

3.1.1 REQUIREMENTS……………………………………………………….6

3.1.2 DESIGN………………………………………………………………….7

3.1.3 IMPLEMENTATION………………………………………………........7

3.1.4 TESTING………………………………………………………………...7

3.2 EXISTING SYSTEM…..………………………………………………………….7

3.2.1 SUPPORT VECTOR MACHINE (SVM)……………………………….7

3.2.2 APPLICATIONS………………………………………………………...9

3.2.3 DISADVANTAGES……………………………………………………..9

3.3 PROPOSED SYSTEMS……………………………………………………….......9

3.3.1 LIGHTWEIGHT DEEP LEARNING MODELS …...…………………10

3.3.2 SQUEEZE NET………………………………………………………...11

3.3.3 MOBILE NETv2…..……...……………………………………….........11

3.3.4 SHUFFLE NET…………………..……………………………………..12

3.3.5 FUSION MODEL……………………………………………………….12

3.3.6 ADVANTAGES…………………………………………....……….......14

4. SYSTEM DESIGN………………………...………………………….……………….......15

4.1 UML DIAGRAM…………………………………………………………………15

4.1.1 CLASS DIAGRAM………………..…………………………………………...15

4.1.2 USE CASE DIAGRAM………………………………………………………...16

4.1.3 SEQUENCE DIAGRAM……………..………….……………………….…….17

4.1.4 ACTIVITY DIAGRAM………….…..…………………….…………………...18

4.1.5 DEPLOYMENT DIAGRAM…………………………………………………..19

4.1.6 COMPONENT DIAGRAM………………….……………….………………...20

5. SOFTWARE ENVIRONMENT…………………….……………………………………. 22

5.1 MACHINE LEARNING………………………..………………………………...22

5.2 PYTHON………………………………………….…………………………...…28

6. SYSTEM REQUIREMENTS SPECIFICATION…………….…………………………...37

6.1 SYSTEM REQUIERMENTS………………………….…………………………37

6.1.1 SOFTWARE REQUIREMENT………………….…………………….37

6.1.2 HARDWARE REQUIREMENT…………………….…………………37

6.2 FUNCTIONAL REQUIREMENTS……………………………….……………..37

6.2.1 OUTPUT DESIGN………………………………………….………….37

6.2.2 INPUT DESIGN……………………………………………….……….38

7. SYSTEM IMPLEMENTATION..…………………………………………………….…...42

7.1 MODULES USED IN PROJECT…………………………………………….…..42

7.2 CODING….……………………………………………………………………....43

8. SYSTEM TESTING……………………………………………………………………….55

8.1 TESTING METHODOLOGIES………………………………………………….55

8.2 TYPES OF TESTING…………………………………………………………….55

8.3 TEST STRATEGY AND APPROACH………………………………………….57

9. RESULTS……………………….……………………………………………………........58

10. CONCLUSION……..…………………………………………………………………….65

11. FUTURE SCOPE……………………………………………………………………........66

12. REFERENCES..…………………………………………………………………………..67

**LIST OF FIGURE**

FIGURE NO. FIGURE NAME PAGE NUMBER

Figure 3.1 Software Development Life Cycle 6

Figure 3.2.1a Hyperplane 8

Figure 3.2.1b Basic classification using SVM 8

Figure 3.3 Fusion based decision model for forgery detection 10

Figure 3.3.2 SqueezeNet 11

Figure 3.3.3 MobileNetV2 11

Figure 3.3.4 ShuffleNet 12

Figure 3.3.5 Data Flow Diagram 14

Figure 4.1.1 Class Diagram 16

Figure 4.1.2 Use Case Diagram 17

Figure 4.1.3 Sequence Diagram 18

Figure 4.1.4 Activity Diagram 19

Figure 4.1.5 Deployment Diagram 20

Figure 4.1.6 Component Diagram 21

**LIST OF SCREENSHOTS**

SCREENSHOT NO. SCREENSHOT NAME PAGE NUMBER

Screenshot 8.1 Upload MICC F220 Dataset 1 58

Screenshot 8.2 Upload MICC F220 Dataset 2 58

Screenshot 8.3 Upload MICC F220 Dataset 3 59

Screenshot 8.4 Pre-processing Dataset 1 59

Screenshot 8.5 Pre-processing Dataset 2 60

Screenshot 8.6 Generate and Load Fusion Model 60

Screenshot 8.7 Fine Tuned Features Map with SVM 61

Screenshot 8.8 Run Baseline Shift Model 62

Screenshot 8.9 Accuracy Comparison Graph 62

Screenshot 8.10 Performance Table 63

Screenshot 8.11 Uploading Image to Predict 64

Screenshot 8.12 Predicting Result 64

1. **INTRODUCTION**

In this digital era, images and videos are being used as influential sources of evidence in a variety of contexts like evidence during trials, insurance fraud, social networking, etc. The easy adaptability of editing tools for digital images, especially without any visual proof of manipulation, give rise to questions about their authenticity. It is the job of image forensics authorities to develop technological innovations that would detect the forgeries of images. There are three primary classes of manipulation or forgery detectors studies until now: those supported features descriptors, those supported inconsistent shadows and eventually those supported double JPEG compression.

With sophisticated software, it is easy to tamper the contents of the image to influence the opinions of others. Image forgery techniques are broadly classified into two categories namely copy-move and splicing. For copy-move forgery, elements of the image content area are traced and smudge inside a similar image, whereas for splicing forgery, parts of the image content smudge from alternative pictures. To reconstruct the trust in pictures, various image forgery detection techniques have been proposed over the past few years. Many previous studies have tried to extract totally different properties from the image to spot the copy-paste or splicing of forged areas, such as the lighting, shadows, sensing element noise, and camera reflections.

Researchers determined the credibility of the image wherever it is known either as authentic or forged. Currently, there are many techniques to spot forged regions that exploits the artefacts left by multiple JPEG compression and other techniques of image manipulation to sight the forged regions. Camera primarily based ways have additionally analyzed where the detection relies on demosaicing regularity or sensing element pattern noise wherever the irregularities of the sensing element pattern area unit extracted and compared for anomalies. Forged or manipulated pictures can mislead people and may threaten individuals’ life. This paper aims to find the manipulated pictures by automating the method of feature extraction instead of feature engineering or feature extraction through the manual process. Deep learning to make use of highly correlated pixels in a vicinity, thus considering grouped native connections.

The motivation to use lightweight models in favour to prevent overfitting of the convolutional neural network (CNN) architectures and can be easily deployed on resource constrained hardware and can learn enriched representations. ShuffleNet makes more feature map channels for a given computation complexity budget, which helps to encode more information and is especially important to the efficiency of small networks. MobileNet, makes use of deep-separable convolutions and gains state-of-the-art results and demonstrated the effectiveness of MobileNet when applied to a broad range of tasks. SqueezeNe, optimizing the architecture for fast processing speed CNN system with 50×, fewer parameters than AlexNet and retains standard accuracy. The lightweight models can be deployed effectively on resource-restricted hardware and can learn enriched representation.

**Objective**

In this work, the decision fusion of lightweight deep learning-based models is proposed for the detection of image forgery. The proposed approach consists of two phases on the pretrained and fine-tuned lightweight deep learning models namely SqueezeNet, MobileNetV2, ShuffleNet. In the first phase, features from the images are extracted using lightweight deep learning models without regularization. In the second phase, fine-tuned lightweight deep learning models with fusion and regularization are used to detect image forgery. The main contributions of this work are:

* An approach of decision fusion-based system is proposed using the lightweight for the image forgery detection. The lightweight models used for the fusion decision are SqueezeNet, MobileNetV2, and ShuffleNet.
* The fusion of the decision system is implemented in two phases. First, the pretrained weights for the lightweight models are used to evaluate the forgery detection of the images. Second, the fine-tuned weights are used to compare the results of the forgery detection of the images with the pre-trained models.
* The utilization of the lightweight models leads to the reduction of the number of false matches, thereby reduce the false positive rate and ultimately increase the accuracy of the approach.

1. **LITERATURE SURVEY**

Amerini et al. proposed a step forward in this direction by analyzing how a single or double JPEG compression can be revealed and localized using convolutional neural networks (CNNs). Different kinds of input to the CNN have been taken into consideration, and various experiments have been carried out trying also to evidence potential issues to be further investigated.

Xiao et al. proposed a splicing forgery detection method with two parts: a coarse-to-refined convolutional neural network (C2RNet) and diluted adaptive clustering. The proposed C2RNet cascades a coarse convolutional neural network (C-CNN) and a refined CNN (R-CNN) and extracts the differences in the image properties between un-tampered and tampered regions from image patches with different scales. Further, to decrease the computational complexity, an image-level CNN is introduced to replace patch-level CNN in C2RNet. The proposed detection method learns the differences of various image properties to guarantee a stable detection performance, and the image-level CNN tremendously decreases its computational time.

Zhang et al. studied the first stage; this paper utilized a Stacked Autoencoder model to learn the complex feature for each individual patch. For the second stage, this paper integrated the contextual information of each patch so that the detection can be conducted more accurately.

Goh et al. proposed a hybrid evolutionary framework to perform a quantitative study to evaluate all features in image tampering for the best feature set. Upon feature evaluation and selection, the classification mechanism must be optimised for good performance. Therefore, in addition to being able to determine an optimal set of features for a classifier, the hybrid framework can determine the optimal multiple classifier ensembles while achieving the best classification performance in terms of low complexity and high accuracy for image tampering detection.

Sutthiwan et al. proposed image statistical features are generated by applying Markovian rake transform to image luminance component. Markovian rake transform is the application of Markov process to difference arrays which are derived from the quantized block discrete cosine transform 2-D arrays with multiple block sizes. The efficacy of thus generated features has been confirmed over a recently established large-scale image dataset designed for tampering detection, with which some relevant issues have been addressed and corresponding adjustment measures have been taken. The initial tests by using thus generated classifiers on some real-life forged images available in the Internet show signs of promise of the proposed features as well as the challenge encountered by the research community of image tampering detection.

He et al. proprosed a Markov based approach to detect this specific artifact. Firstly, the original Markov features generated from the transition probability matrices in DCT domain by Shi et al. is expanded to capture not only the intra-block but also the inter-block correlation between block DCT coefficients. Then, more features are constructed in DWT domain to characterize the three kinds of dependency among wavelet coefficients across positions, scales, and orientations. After that, feature selection method SVM-RFE is used to fulfill the task of feature reduction, making the computational cost more manageable. Finally, support vector machine (SVM) is exploited to classify the authentic and spliced images using the final dimensionality-reduced feature vector.

Change et al. proposed a novel forgery detection algorithm to recognize tampered inpainting images, which is one of the effective approaches for image manipulation. The proposed algorithm contains two major processes: suspicious region detection and forged region identification. Suspicious region detection searches the similarity blocks in an image to find the suspicious regions and uses a similarity vector field to remove the false positives caused by uniform area. Forged region identification applies a new method, multi-region relation (MRR), to identify the forged regions from the suspicious regions. The proposed approach can effectively recognize if an image is a forged one and identify the forged regions, even for the images containing the uniform background. Moreover, this paper proposed a two-stage searching algorithm based on weight transformation to speed up the computation speed.

Rhee et al. presented a short feature vector that is made up of three types of feature sets. The first set is defined by the variation to be the 3-D length in the gradient difference of the intensity values of the adjacent row and column line pairs in the image, respectively. The second set is defined by the variation in the coefficient difference of the Fourier transform to be the 3-D length in the adjacent line pairs. The last set is defined by the residual image between an image and its reconstructed image by the gradient based on solving Poisson’s equation, which is also the 3-D length. Two of the sets are extracted in the spatial and spectral domains of an image, respectively, and the last set is extracted from the residual image. The totally formed 9-D feature vector is subsequently trained in the support vector machine classifier for MFD.

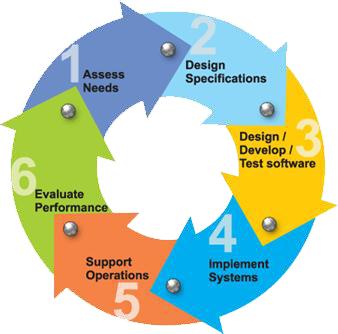
Lamba et al. proposed a discrete fractional wavelet transform-based scheme for identification of duplicated regions in the image. The test image is split into overlapping image blocks with fixed dimensions. Then, on each image block, discrete fractional wavelet transform is employed for the extraction of their features. All the feature vectors are systematized in lexicographical manner followed by the block matching and block filtering steps to obtain the replicated blocks, if any. The proposed method can detect single and multiple duplicated regions successfully.

Lin et al. proposed detecting tampered images by examining the double quantization effect hidden among the discrete cosine transform (DCT) coefficients. This paper is the only one to date that can automatically locate the tampered region, while it has several additional advantages: fine-grained detection at the scale of DCT blocks, insensitivity to different kinds of forgery methods (such as alpha matting and inpainting, in addition to simple image cut/paste), the ability to work without fully decompressing the JPEG images, and the fast speed. Experimental results on JPEG images are promising.

**3. SYSTEM ANALYSIS**

**3.1 SOFTWARE DEVELOPMENT LIFE CYCLE**

There are various software development approaches defined and designed which are used/employed during development process of software, these approaches are also referred as "Software Development Process Models". Each process model follows a particular life cycle in order to ensure success in process of software development.



**Figure 3.1 SOFTWARE DEVELOPMENT LIFE CYCLE**

**3.1.1** **Requirements**

Business requirements are gathered in this phase. This phase is the main focus of the project managers and stake holders. Meetings with managers, stake holders and users are held in order to determine the requirements. Who is going to use the system? How will they use the system? What data should be input into the system? What data should be output by the system? These are general questions that get answered during a requirements gathering phase. This produces a nice big list of functionality that the system should provide, which describes functions the system should perform, business logic that processes data, what data is stored and used by the system, and how the user interface should work. The overall result is the system as a whole and how it performs, not how it is actually going to do it.

**3.1.2** **Design**

The software system design is produced from the results of the requirements phase. Architects have the ball in their court during this phase and this is the phase in which their focus lies. This is where the details on how the system will work is produced. Architecture, including hardware and software, communication, software design (UML is produced here) are all part of the deliverables of a design phase.

**3.1.3** **Implementation**

Code is produced from the deliverables of the design phase during implementation, and this is the longest phase of the software development life cycle. For a developer, this is the main focus of the life cycle because this is where the code is produced. Implementation my overlap with both the design and testing phases. Many tools exists (CASE tools) to actually automate the production of code using information gathered and produced during the design phase.

**3.1.4** **Testing**

During testing, the implementation is tested against the requirements to make sure that the product is actually solving the needs addressed and gathered during the requirements phase. Unit tests and system/acceptance tests are done during this phase. Unit tests act on a specific component of the system, while system tests act on the system as a whole. So in a nutshell, that is a very basic overview of the general software development life cycle model. Now let‘s delve into some of the traditional and widely used variations.

**3.2 EXISTING** **SYSTEM**

**3.2.1 Support Vector Machine Algorithm**

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning. The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine. Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane:



**Figure 3.2.1a Hyperplane**

**Example:** SVM can be understood with the example that we have used in the KNN classifier. Suppose we see a strange cat that also has some features of dogs, so if we want a model that can accurately identify whether it is a cat or dog, so such a model can be created by using the SVM algorithm. We will first train our model with lots of images of cats and dogs so that it can learn about different features of cats and dogs, and then we test it with this strange creature. So as support vector creates a decision boundary between these two data (cat and dog) and choose extreme cases (support vectors), it will see the extreme case of cat and dog. On the basis of the support vectors, it will classify it as a cat. Consider the below diagram:



**Figure 3.2.1b Basic classification using SVM**

**3.2.2 Applications**

* Face recognition
* Weather prediction
* Medical diagnosis
* Spam detection
* Age/gender identification
* Language identification
* Sentimental analysis
* Authorship identification
* News classification

**3.2.3 Disadvantages**

* Support vector machine algorithm is not acceptable for large data sets.
* It does not execute very well when the data set has more sound i.e. target classes are overlapping.
* In cases where the number of properties for each data point outstrips the number of training data specimens, the support vector machine will underperform.
* As the support vector classifier works by placing data points, above and below the classifying hyperplane there is no probabilistic clarification for the classification.
* For this project it only gives below 70 % accuracy.

**3.3 PROPOSED SYSTEM**

The architecture of the proposed decision fusion is based on the lightweight deep learning models as shown in Figure 4.1. The lightweight deep learning models chosen are SqueezeNet, MobileNetV2, and ShuffleNet. The proposed system is implemented in two phases i.e., with pre-trained and fine-tuned deep learning models. In the pre-trained model’s implementation, regularization is not applied, and the pre-trained weights are used and for the fine-tuned implementation, regularization is applied to detect image forgery. Each phase consists of three stages namely, data pre-processing, classification, and fusion. In the data pre-processing stage, the image in the query is pre-processed based on the dimensions required by the deep learning models. SVM is used for the classification of the image as forged or non-forged. Initially, we discuss the lightweight deep learning models and then the strategy used for the regularization is discussed in the further sections.

Diagram, schematic

Description automatically generated

**Figure 3.3 Fusion based decision model for forgery detection.**

**Data pre-processing**

In this stage, the image in a query that needs to be identified whether it is forged or not is subjected to pre-processing. The height and width of the image required for SqueezeNet is 227×227. The height and width of the image required for MobileNetV2 is 224×224. The height and width of the image required for ShuffleNet is 224×224. The input image is pre-processed first based on the dimensions required for each of the models. Each model then takes the input image to produce feature vector in further stages.

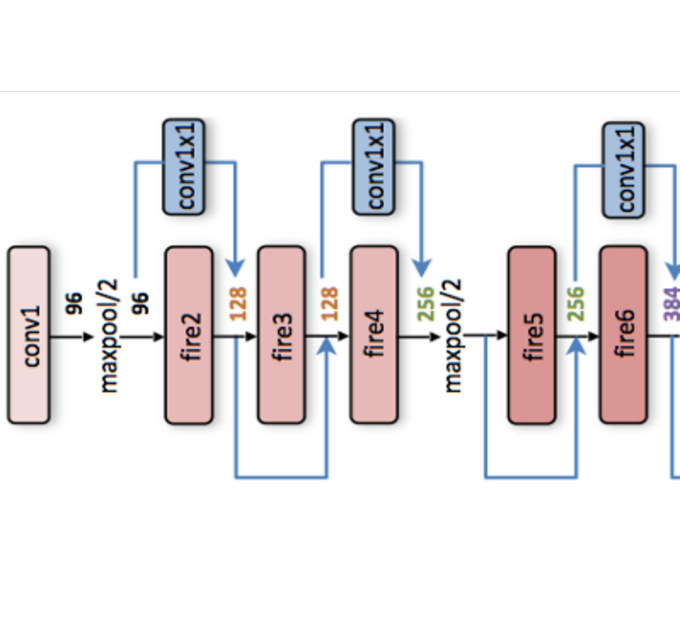
**3.3.1 Lightweight deep learning models**

The different lightweight deep learning models that are considered for fusion are SqueezeNet, MobileNetV2, and ShuffleNet. These models are used for the image classification problems numerously. In this section, these models are discussed briefly. The lightweight models1 considered are summarized as shown in the Table 1. It represents the depth, parameters and the image input size required for the lightweight models namely, SqueezeNet, MobileNetV2, and ShuffleNet.

**3.3.2 SqueezeNet**

It is a CNN trained on the ImageNet dataset with 18 layers deep and can classify the images up to 1000 categories. The network has learned rich representations of the images with 1.24 million

parameters. It requires only a few floating-point operations for the image classification.



**Figure 3.3.2 SqueezeNet.**

**3.3.3 MobileNetV2**

It is a CNN trained on the ImageNet dataset with 53 layers deep and can classify the images up to 1000 categories. The performance of the classification is improved based on the learning of the rich representations of the images.

Diagram

Description automatically generated

**Figure 3.3.3 MobileNetV2.**

**3.3.4 ShuffleNet**

It is a CNN that is also trained on the ImageNet dataset with 50 layers deep and can classify the images up to 1000 categories. Table 1. Parameters of lightweight deep learning models. (Depth represents the largest number of sequential convolutional or fully connected layers on a path from the input layer to the output layer, parameter represents the total number of learnable parameters in each layer and image input size represents the required input image size).

Table 1. Models description.

Table

Description automatically generated

Diagram

Description automatically generated

**Figure 3.3.4 ShuffleNet.**

**3.3.5 Fusion model**

The proposed system is first implemented with lightweight deep learning models using pretrained weights for the image forgery detection, afterward, the proposed system is implemented as a fusion of the decision of lightweight models as discussed in the previous section. Initially, the input image is passed to the lightweight models to obtain their feature maps respectively. The feature map from the SqueezeNet is denoted as, the feature map from the MobileNetV2 is denoted as , the feature map from the ShuffleNet is denoted as . For the fusion model, the pretrained lightweight deep learning model’s output feature mapping is used. This feature map is a combination of the feature maps obtained from the lightweight models as shown in Equation (1).

(1)

The fusion model uses feature map as a local descriptor for an input patch to extract the features of the image. The image for the fusion model is represented as a function where is the patch in the input image. For a test image size , a sliding window of size p×p is used to compute the local descriptor is computed as shown in the equation (2) where represents the descriptors of the patches of the image obtained from the deep learning models. It is obtained as a concatenation of all the input patches and the new image representation is given by equation (3) where s is the size of the stride used for transforming the input patch, this new image representation is used as the feature map for the classification by the SVM as forged or nonforged. (2)

(3)

For fine tuning of the parameters of the fusion model, the initialization of the weight kernels is used as shown in Equation (4). In this equation Wf represents the weights of the fusion model, represents the weights of the SqueezeNet model, represents the weights of the MobileNetV2 model and represents the weights of the ShuffleNet model. The weight of the fusion model is initialized as shown in Equation (5). The initialization of the weights acts as a regularization term and facilitates the fusion model to learn the robust features of detecting the forgery rather than the complex image representations.

= [] j = 1, 2, 3 (4)

= [−2 4k−2 4k] where (5)

**Dataflow diagram**

A data flow diagram (DFD) maps out the flow of information for any process or system. It uses defined symbols like rectangles, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination.

Diagram

Description automatically generated

**Figure 3.3.5 Data Flow Diagram**

**Classifier**

SVM is used as a classifier. SVM is popular and efficient for binary classification. The performance of the proposed approach is evaluated at the image level by calculating the performance metrics like precision, recall also known as true positive rate (TPR), false positive rate (FPR), F-score and accuracy.

**3.3.6 Advantages**

* It accepts large and huge data
* It can even work on noisy data
* There will be clear probabilistic classification of data points
* It shows more then 95 % accuracy.

**4 SYSTEM DESIGN**

**4.1 UML DIAGRAMS**

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group. The goal is for UML to become a common language for creating models of object-oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems. The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

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

* Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
* Provide extendibility and specialization mechanisms to extend the core concepts.
* Be independent of particular programming languages and development process.
* Provide a formal basis for understanding the modeling language.
* Encourage the growth of OO tools market.
* Support higher level development concepts such as collaborations, frameworks, patterns and components.
* Integrate best practices.

**4.1.1 CLASS DIAGRAM**

In software engineering, a class diagram in the Unified Modeling 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.

Table

Description automatically generated

**Figure 4.1.1 CLASS DIAGRAM**

**4.1.2 USE CASE DIAGRAM**

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral 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.

Diagram

Description automatically generated

**Figure 4.1.2 USE CASE DIAGRAM**

**4.1.3 SEQUENCE DIAGRAM**

A sequence diagram represents the interaction between different objects in the system. The important aspect of a sequence diagram is that it is time-ordered. This means that the exact sequence of the interactions between the objects is represented step by step. Different objects in the sequence diagram interact with each other by passing "messages".

Table

Description automatically generated with medium confidence

**Figure 4.1.3 SEQUENCE DIAGRAM**

**4.1.4 ACTIVITY DIAGRAM**

The process flows in the system are captured in the activity diagram. Similar to a state diagram, an activity diagram also consists of activities, actions, transitions, initial and final states, and guard conditions.

Diagram

Description automatically generated

**Figure 4.1.4 ACTIVITY DIAGRAM**

**4.1.5 DEPLOYMENT DIAGRAM**

A deployment diagram represents the physical deployment of a system, showing the hardware components, software components, and their relationships. It is used to model the distribution of a system and to identify potential deployment issues.

Diagram

Description automatically generated

**Figure 4.1.5 DEPLOYMENT DIAGRAM**

**4.1.6 COMPONENT DIAGRAM**

It is used to model the physical components of a system and the relationships between them. It is a type of structural diagram that shows the organization and dependencies of the components within a system.

A component diagram consists of several important elements:

* Component: A physical or logical part of a system that performs a specific function. Components are represented as rectangles with the component name in the top compartment, and the component type in the bottom compartment.
* Interface: A contract that specifies the operations and services that a component provides or requires. Interfaces are represented as circles with the interface name inside.
* Port: A point of interaction between a component and its environment. Ports are represented as small squares attached to the edges of components.
* Connector: A link between two components or between a component and an interface that represents the communication or dependency between them. Connectors are represented as lines with arrows showing the direction of communication.
* Dependency: A relationship between two components in which one component depends on the other to perform its function. Dependencies are represented as dashed lines with an arrow pointing from the dependent component to the component it depends on.
* Provided interface: An interface that a component provides to its environment. A provided interface is represented as a solid circle attached to a port.
* Required interface: An interface that a component requires from its environment. A required interface is represented as an empty circle attached to a port.

Component diagrams are useful for modeling the high-level structure of a system and for identifying the major components and their relationships. They can also help in designing and implementing systems by identifying the interfaces between components and the dependencies between them.

Diagram

Description automatically generated

**Figure 4.1.6 COMPONENT DIAGRAM**

**5. SOFTWARE ENVIRONMENT**

**5.1 MACHINE LEARNING**

**What is Machine Learning**

Before we 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 Leaning**

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

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 have not 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.

**Challenges in Machines Learning**

While Machine Learning is rapidly evolving, making significant strides with cybersecurity and autonomous cars, this segment of AI as whole still has a long way to go. The reason behind is that ML has not been able to overcome number of challenges. The challenges that ML is facing currently are −

1. Quality of data − Having good-quality data for ML algorithms is one of the biggest challenges. Use of low-quality data leads to the problems related to data preprocessing and feature extraction.
2. Time-Consuming task − Another challenge faced by ML models is the consumption of time especially for data acquisition, feature extraction and retrieval.
3. Lack of specialist persons − As ML technology is still in its infancy stage, availability of expert resources is a tough job.
4. No clear objective for formulating business problems − Having no clear objective and well-defined goal for business problems is another key challenge for ML because this technology is not that mature yet.
5. Issue of overfitting & underfitting − If the model is overfitting or underfitting, it cannot be represented well for the problem.
6. Curse of dimensionality − Another challenge ML model faces is too many features of data points. This can be a real hindrance.
7. Difficulty in deployment − Complexity of the ML model makes it quite difficult to be deployed in real life.

**Applications of Machines Learning**

Machine Learning is the most rapidly growing technology and according to researchers we are in the golden year of AI and ML. It is used to solve many real-world complex problems which cannot be solved with traditional approach. Following are some real-world applications of ML

* Emotion analysis
* Sentiment analysis
* Error detection and prevention
* Weather forecasting and prediction
* Stock market analysis and forecasting
* Speech synthesis
* Speech recognition
* Customer segmentation
* Object recognition
* Fraud detection
* Fraud prevention
* Recommendation of products to customer in online shopping

**How to Start Learning Machine Learning?**

Arthur Samuel coined the term “Machine Learning” in 1959 and defined it as a “Field of study that gives computers the capability to learn without being explicitly programmed”.

And that was the beginning of Machine Learning! In modern times, Machine Learning is one of the most popular (if not the most!) career choices. According to Indeed, Machine Learning Engineer Is The Best Job of 2019 with a 344% growth and an average base salary of $146,085 per year.

But there is still a lot of doubt about what exactly is Machine Learning and how to start learning it? So, this article deals with the Basics of Machine Learning and also the path you can follow to eventually become a full-fledged Machine Learning Engineer. Now let’s get started!!!

**How to start learning ML?**

This is a rough roadmap you can follow on your way to becoming an insanely talented Machine Learning Engineer. Of course, you can always modify the steps according to your needs to reach your desired end-goal!

Step 1 – Understand the Prerequisites

In case you are a genius, you could start ML directly but normally, there are some prerequisites that you need to know which include Linear Algebra, Multivariate Calculus, Statistics, and Python. And if you don’t know these, never fear! You don’t need a Ph.D. degree in these topics to get started but you do need a basic understanding.

(a) Learn Linear Algebra and Multivariate Calculus

Both Linear Algebra and Multivariate Calculus are important in Machine Learning. However, the extent to which you need them depends on your role as a data scientist. If you are more focused on application heavy machine learning, then you will not be that heavily focused on maths as there are many common libraries available. But if you want to focus on R&D in Machine Learning, then mastery of Linear Algebra and Multivariate Calculus is very important as you will have to implement many ML algorithms from scratch.

(b) Learn Statistics

Data plays a huge role in Machine Learning. In fact, around 80% of your time as an ML expert will be spent collecting and cleaning data. And statistics is a field that handles the collection, analysis, and presentation of data. So it is no surprise that you need to learn it!!!  
Some of the key concepts in statistics that are important are Statistical Significance, Probability Distributions, Hypothesis Testing, Regression, etc. Also, Bayesian Thinking is also a very important part of ML which deals with various concepts like Conditional Probability, Priors, and Posteriors, Maximum Likelihood, etc.

(c) Learn Python

Some people prefer to skip Linear Algebra, Multivariate Calculus and Statistics and learn them as they go along with trial and error. But the one thing that you absolutely cannot skip is Python! While there are other languages you can use for Machine Learning like R, Scala, etc. Python is currently the most popular language for ML. In fact, there are many Python libraries that are specifically useful for Artificial Intelligence and Machine Learning such as Keras, TensorFlow, Scikit-learn, etc.

So if you want to learn ML, it’s best if you learn Python! You can do that using various online resources and courses such as Fork Python available Free on GeeksforGeeks.

Step 2 – Learn Various ML Concepts

Now that you are done with the prerequisites, you can move on to actually learning ML (Which is the fun part!!!) It’s best to start with the basics and then move on to the more complicated stuff. Some of the basic concepts in ML are:

**(a) Terminologies of Machine Learning**

* Model – A model is a specific representation learned from data by applying some machine learning algorithm. A model is also called a hypothesis.
* Feature – A feature is an individual measurable property of the data. A set of numeric features can be conveniently described by a feature vector. Feature vectors are fed as input to the model. For example, in order to predict a fruit, there may be features like color, smell, taste, etc.
* Target (Label) – A target variable or label is the value to be predicted by our model. For the fruit example discussed in the feature section, the label with each set of input would be the name of the fruit like apple, orange, banana, etc.
* Training – The idea is to give a set of inputs(features) and it’s expected outputs(labels), so after training, we will have a model (hypothesis) that will then map new data to one of the categories trained on.
* Prediction – Once our model is ready, it can be fed a set of inputs to which it will provide a predicted output(label).

**(b) Types of Machine Learning**

* Supervised Learning – This involves learning from a training dataset with labeled data using classification and regression models. This learning process continues until the required level of performance is achieved.
* Unsupervised Learning – This involves using unlabelled data and then finding the underlying structure in the data in order to learn more and more about the data itself using factor and cluster analysis models.
* Semi-supervised Learning – This involves using unlabelled data like Unsupervised Learning with a small amount of labeled data. Using labeled data vastly increases the learning accuracy and is also more cost-effective than Supervised Learning.
* Reinforcement Learning – This involves learning optimal actions through trial and error. So the next action is decided by learning behaviors that are based on the current state and that will maximize the reward in the future.

**Advantages of Machine learning**

1. Easily identifies trends and patterns -

Machine Learning can review large volumes of data and discover specific trends and patterns that would not be apparent to humans. For instance, for an e-commerce website like Amazon, it serves to understand the browsing behaviors and purchase histories of its users to help cater to the right products, deals, and reminders relevant to them. It uses the results to reveal relevant advertisements to them.

2. No human intervention needed (automation)

With ML, you don’t need to babysit your project every step of the way. Since it means giving machines the ability to learn, it lets them make predictions and also improve the algorithms on their own. A common example of this is anti-virus softwares; they learn to filter new threats as they are recognized. ML is also good at recognizing spam.

3. Continuous Improvement

As ML algorithms gain experience, they keep improving in accuracy and efficiency. This lets them make better decisions. Say you need to make a weather forecast model. As the amount of data you have keeps growing, your algorithms learn to make more accurate predictions faster.

4. Handling multi-dimensional and multi-variety data

Machine Learning algorithms are good at handling data that are multi-dimensional and multi-variety, and they can do this in dynamic or uncertain environments.

5. Wide Applications

You could be an e-tailer or a healthcare provider and make ML work for you. Where it does apply, it holds the capability to help deliver a much more personal experience to customers while also targeting the right customers.

**Disadvantages of Machine Learning**

1. Data Acquisition

Machine Learning requires massive data sets to train on, and these should be inclusive/unbiased, and of good quality. There can also be times where they must wait for new data to be generated.

2. Time and Resources

ML needs enough time to let the algorithms learn and develop enough to fulfill their purpose with a considerable amount of accuracy and relevancy. It also needs massive resources to function. This can mean additional requirements of computer power for you.

3. Interpretation of Results

Another major challenge is the ability to accurately interpret results generated by the algorithms. You must also carefully choose the algorithms for your purpose.

4. High error-susceptibility

Machine Learning is autonomous but highly susceptible to errors. Suppose you train an algorithm with data sets small enough to not be inclusive. You end up with biased predictions coming from a biased training set. This leads to irrelevant advertisements being displayed to customers. In the case of ML, such blunders can set off a chain of errors that can go undetected for long periods of time. And when they do get noticed, it takes quite some time to recognize the source of the issue, and even longer to correct it.

**5.2 PYTHON**

**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

**Advantages of Python**

Let’s see how Python dominates over other languages.

1. Extensive Libraries

Python downloads with an extensive library and it contain code for various purposes like regular expressions, documentation-generation, unit-testing, web browsers, threading, databases, CGI, email, image manipulation, and more. So, we don’t have to write the complete code for that manually.

2. Extensible

As we have seen earlier, Python can be extended to other languages. You can write some of your code in languages like C++ or C. This comes in handy, especially in projects.

3. Embeddable

Complimentary to extensibility, Python is embeddable as well. You can put your Python code in your source code of a different language, like C++. This lets us add scripting capabilities to our code in the other language.

4. Improved Productivity

The language’s simplicity and extensive libraries render programmers more productive than languages like Java and C++ do. Also, the fact that you need to write less and get more things done.

5. IOT Opportunities

Since Python forms the basis of new platforms like Raspberry Pi, it finds the future bright for the Internet Of Things. This is a way to connect the language with the real world.

6. Simple and Easy

When working with Java, you may have to create a class to print ‘Hello World’. But in Python, just a print statement will do. It is also quite easy to learn, understand, and code. This is why when people pick up Python, they have a hard time adjusting to other more verbose languages like Java.

7. Readable

Because it is not such a verbose language, reading Python is much like reading English. This is the reason why it is so easy to learn, understand, and code. It also does not need curly braces to define blocks, and indentation is mandatory. This further aids the readability of the code.

8. Object-Oriented

This language supports both the procedural and object-oriented programming paradigms. While functions help us with code reusability, classes and objects let us model the real world. A class allows the encapsulation of data and functions into one.

9. Free and Open-Source

Like we said earlier, Python is freely available. But not only can you download Python for free, but you can also download its source code, make changes to it, and even distribute it. It downloads with an extensive collection of libraries to help you with your tasks.

10. Portable

When you code your project in a language like C++, you may need to make some changes to it if you want to run it on another platform. But it isn’t the same with Python. Here, you need to code only once, and you can run it anywhere. This is called Write Once Run Anywhere (WORA). However, you need to be careful enough not to include any system-dependent features.

11. Interpreted

Lastly, we will say that it is an interpreted language. Since statements are executed one by one, debugging is easier than in compiled languages.

Any doubts till now in the advantages of Python? Mention in the comment section.

Advantages 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 can 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.

**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 client-side. 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.

**History of Python**

What do the alphabet and the programming language Python have in common? Right, both start with ABC. If we are talking about ABC in the Python context, it's clear that the programming language ABC is meant. ABC is a general-purpose programming language and programming environment, which had been developed in the Netherlands, Amsterdam, at the CWI (Centrum Wiskunde &Informatica). The greatest achievement of ABC was to influence the design of Python. Python was conceptualized in the late 1980s. Guido van Rossum worked that time in a project at the CWI, called Amoeba, a distributed operating system. In an interview with Bill Venners1, Guido van Rossum said: "In the early 1980s, I worked as an implementer on a team building a language called ABC at Centrum voor Wiskunde en Informatica (CWI). I don't know how well people know ABC's influence on Python. I try to mention ABC's influence because I'm indebted to everything I learned during that project and to the people who worked on it. "Later on in the same Interview, Guido van Rossum continued: "I remembered all my experience and some of my frustration with ABC. I decided to try to design a simple scripting language that possessed some of ABC's better properties, but without its problems. So I started typing. I created a simple virtual machine, a simple parser, and a simple runtime. I made my own version of the various ABC parts that I liked. I created a basic syntax, used indentation for statement grouping instead of curly braces or begin-end blocks, and developed a small number of powerful data types: a hash table (or dictionary, as we call it), a list, strings, and numbers."

**Python Development Steps**

Guido Van Rossum published the first version of Python code (version 0.9.0) at alt.sources in February 1991. This release included already exception handling, functions, and the core data types of list, dict, str and others. It was also object oriented and had a module system.

Python version 1.0 was released in January 1994. The major new features included in this release were the functional programming tools lambda, map, filter and reduce, which Guido Van Rossum never liked. Six and a half years later in October 2000, Python 2.0 was introduced. This release included list comprehensions, a full garbage collector and it was supporting unicode. Python flourished for another 8 years in the versions 2.x before the next major release as Python 3.0 (also known as "Python 3000" and "Py3K") was released. Python 3 is not backwards compatible with Python 2.x. The emphasis in Python 3 had been on the removal of duplicate programming constructs and modules, thus fulfilling or coming close to fulfilling the 13th law of the Zen of Python: "There should be one -- and preferably only one -- obvious way to do it."Some changes in Python 7.3:

Print is now a function.

* Views and iterators instead of lists
* The rules for ordering comparisons have been simplified. E.g., a heterogeneous list cannot be sorted, because all the elements of a list must be comparable to each other.
* There is only one integer type left, i.e., int. long is int as well.
* The division of two integers returns a float instead of an integer. "//" can be used to have the "old" behaviour.
* Text Vs. Data Instead of Unicode Vs. 8-bit

**Purpose**

We demonstrated that our approach enables successful segmentation of intra-retinal layers—even with low-quality images containing speckle noise, low contrast, and different intensity ranges throughout—with the assistance of the ANIS feature.

**Python**

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

* Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
* Python is Interactive − you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking.

**How to Install Python on Windows and Mac**

There have been several updates in the Python version over the years. The question is how to install Python? It might be confusing for the beginner who is willing to start learning Python but this tutorial will solve your query. The latest or the newest version of Python is version 3.7.4 or in other words, it is Python 3.

Note: The python version 3.7.4 cannot be used on Windows XP or earlier devices.

Before you start with the installation process of Python. First, you need to know about your System Requirements. Based on your system type i.e. operating system and based processor, you must download the python version. My system type is a Windows 64-bit operating system. So the steps below are to install python version 3.7.4 on Windows 7 device or to install Python 3. Download the Python Cheatsheet here.The steps on how to install Python on Windows 10, 8 and 7 are divided into 4 parts to help understand better.

Download the Correct version into the system

Step 1: Go to the official site to download and install python using Google Chrome or any other web browser. OR Click on the following link: https://www.python.org

Now, check for the latest and the correct version for your operating system.

Step 2: Click on the Download Tab.

Step 3: You can either select the Download Python for windows 3.7.4 button in Yellow Color or you can scroll further down and click on download with respective to their version. Here, we are downloading the most recent python version for windows 3.7.4

Step 4: Scroll down the page until you find the Files option.

Step 5: Here you see a different version of python along with the operating system.

* To download Windows 32-bit python, you can select any one from the three options: Windows x86 embeddable zip file, Windows x86 executable installer or Windows x86 web-based installer.
* To download Windows 64-bit python, you can select any one from the three options: Windows x86-64 embeddable zip file, Windows x86-64 executable installer or Windows x86-64 web-based installer.

Here we will install Windows x86-64 web-based installer. Here your first part regarding which version of python is to be downloaded is completed. Now we move ahead with the second part in installing python i.e. Installation

Note: To know the changes or updates that are made in the version you can click on the Release Note Option.

**Installation of Python**

Step 1: Go to Download and Open the downloaded python version to carry out the installation process.

Step 2: Before you click on Install Now, Make sure to put a tick on Add Python 3.7 to PATH.

Step 3: Click on Install NOW After the installation is successful. Click on Close.

With these above three steps on python installation, you have successfully and correctly installed Python. Now is the time to verify the installation.

Note: The installation process might take a couple of minutes.

Verify the Python Installation

Step 1: Click on Start

Step 2: In the Windows Run Command, type “cmd”.

Step 3: Open the Command prompt option.

Step 4: Let us test whether the python is correctly installed. Type python –V and press Enter.

Step 5: You will get the answer as 3.7.4

Note: If you have any of the earlier versions of Python already installed. You must first uninstall the earlier version and then install the new one.

**6. SYSTEM REQUIREMENTS SPECIFICATIONS**

**6.1 SYSTEM REQUIREMENTS**

**6.1.1 SOFTWARE REQUIREMENT**

The functional requirements or the overall description documents include the product perspective and features, operating system and operating environment, graphics requirements, design constraints and user documentation.

The appropriation of requirements and implementation constraints gives the general overview of the project in regard to what the areas of strength and deficit are and how to tackle them.

* Python IDLE 3.7 version (or)
* Anaconda 3.7 (or)
* Jupiter (or)
* Google colab

**6.1.2 HARDWARE REQUIREMENT**

Minimum hardware requirements are very dependent on the particular software being developed by a given Enthought Python / Canopy / VS Code user. Applications that need to store large arrays/objects in memory will require more RAM, whereas applications that need to perform numerous calculations or tasks more quickly will require a faster processor.

Operating system : Windows, Linux

Processor : minimum intel i3

Ram : minimum 4 GB

Hard disk : minimum 250GB

**6.2 FUNCTIONAL REQUIREMENTS**

**6.2.1 OUTPUT DESIGN**

Outputs from computer systems are required primarily to communicate the results of processing to users. They are also used to provides a permanent copy of the results for later consultation. The various types of outputs in general are:

* External Outputs, whose destination is outside the organization
* Internal Outputs whose destination is within organization and they are the
* User’s main interface with the computer.
* Operational outputs whose use is purely within the computer department.
* Interface outputs, which involve the user in communicating directly.

**Output Definition**

The outputs should be defined in terms of the following points:

* Type of the output
* Content of the output
* Format of the output
* Location of the output
* Frequency of the output
* Volume of the output
* Sequence of the output

It is not always desirable to print or display data as it is held on a computer. It should be decided as which form of the output is the most suitable.

**6.2.2 INPUT DESIGN**

Input design is a part of overall system design. The main objective during the input design is as given below:

* To produce a cost-effective method of input.
* To achieve the highest possible level of accuracy.
* To ensure that the input is acceptable and understood by the user.

**Input Stages**

The main input stages can be listed as below:

* Data recording
* Data transcription
* Data conversion
* Data verification
* Data control
* Data transmission
* Data validation
* Data correction

**Input Types**

It is necessary to determine the various types of inputs. Inputs can be categorized as follows:

* External inputs, which are prime inputs for the system.
* Internal inputs, which are user communications with the system.
* Operational, which are computer department’s communications to the system?
* Interactive, which are inputs entered during a dialogue.

**Input Media**

At this stage choice has to be made about the input media. To conclude about the input media consideration has to be given to;

* Type of input
* Flexibility of format
* Speed
* Accuracy
* Verification methods
* Rejection rates
* Ease of correction
* Storage and handling requirements
* Security
* Easy to use
* Portability

Keeping in view the above description of the input types and input media, it can be said that most of the inputs are of the form of internal and interactive. As Input data is to be the directly keyed in by the user, the keyboard can be considered to be the most suitable input device.

**Error Avoidance**

At this stage care is to be taken to ensure that input data remains accurate form the stage at which it is recorded up to the stage in which the data is accepted by the system. This can be achieved only by means of careful control each time the data is handled.

**Error Detection**

Even though every effort is made to avoid the occurrence of errors, still a small proportion of errors is always likely to occur, these types of errors can be discovered by using validations to check the input data.

**Data Validation**

Procedures are designed to detect errors in data at a lower level of detail. Data validations have been included in the system in almost every area where there is a possibility for the user to commit errors. The system will not accept invalid data. Whenever an invalid data is keyed in, the system immediately prompts the user and the user has to again key in the data and the system will accept the data only if the data is correct. Validations have been included where necessary.

The system is designed to be a user friendly one. In other words the system has been designed to communicate effectively with the user. The system has been designed with popup menus.

**User Interface Design**

It is essential to consult the system users and discuss their needs while designing the user interface:

**User Interface Systems Can Be Broadly Classified As:**

* User initiated interface the user is in charge, controlling the progress of the user/computer dialogue.
* Computer initiated interfaces the computer selects the next stage in the interaction.

In the computer-initiated interfaces the computer guides the progress of the user/computer dialogue. Information is displayed and the user response of the computer takes action or displays further information.

**User Initiated Interfaces**

User initiated interfaces fall into two approximate classes:

* Command driven interfaces: In this type of interface the user inputs commands or queries which are interpreted by the computer.
* Forms oriented interface: The user calls up an image of the form to his/her screen and fills in the form. The forms-oriented interface is chosen because it is the best choice.

**Computer-Initiated Interfaces**

The following computer – initiated interfaces were used:

* The menu system for the user is presented with a list of alternatives and the user chooses one; of alternatives.
* Questions – answer type dialog system where the computer asks question and takes action based on the basis of the users reply.

Right from the start the system is going to be menu driven, the opening menu displays the available options. Choosing one option gives another popup menu with more options. In this way every option leads the users to data entry form where the user can key in the data.

**Error Message Design**

The design of error messages is an important part of the user interface design. As user is bound to commit some errors or other while designing a system the system should be designed to be helpful by providing the user with information regarding the error he/she has committed.

This application must be able to produce output at different modules for different inputs.

**Performance Requirements**

Performance is measured in terms of the output provided by the application. Requirement specification plays an important part in the analysis of a system. Only when the requirement specifications are properly given, it is possible to design a system, which will fit into required environment. It rests largely in the part of the users of the existing system to give the requirement specifications because they are the people who finally use the system. This is because the requirements have to be known during the initial stages so that the system can be designed according to those requirements. It is very difficult to change the system once it has been designed and on the other hand designing a system, which does not cater to the requirements of the user, is of no use.

The requirement specification for any system can be broadly stated as given below:

* The system should be able to interface with the existing system
* The system should be accurate
* The system should be better than the existing system
* The existing system is completely dependent on the user to perform all the duties.

**7. SYSTEM IMPLEMENTATIONS**

**7.1 Modules 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 datatypes can be defined using NumPy which allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

**Pandas**

Pandas is 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.

**7.2 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\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 predict():

global shufflenet

global labels

filename = filedialog.askopenfilename(initialdir="testImages")

img = cv2.imread(filename)

img = cv2.resize(img, (32,32))

im2arr = np.array(img)

im2arr = im2arr.reshape(1,32,32,3)

test = np.asarray(im2arr)

test = test.astype('float32')

test = test/255

preds = shufflenet.predict(test)

predict = np.argmax(preds)

img = cv2.imread(filename)

img = cv2.resize(img, (800,400))

cv2.putText(img, 'image Recognize as : '+labels[predict], (10, 25), cv2.FONT\_HERSHEY\_SIMPLEX,0.7, (0, 255, 0), 2)

cv2.imshow('image Recognize as : '+labels[predict], img)

cv2.waitKey(0)

def graph():

df = pd.DataFrame([['SqueezeNet','Precision',precision[0]],['SqueezeNet','Recall',recall[0]],['SqueezeNet','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><td>'+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 Existing 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)

predictButton = Button(main, text="Predict", command=predict)

predictButton.place(x=50,y=450)

predictButton.config(font=font1)

exitButton = Button(main, text="Exit", command=close)

exitButton.place(x=50,y=500)

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()

1. **SYSTEM TESTING**

**8.1 TESTING METHODOLOGIES**

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.

**8.2 TYPES OF TESTING**

**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 components.

**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 centred 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 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.

**8.3 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.

**a. 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.

**b. Acceptance Testing**

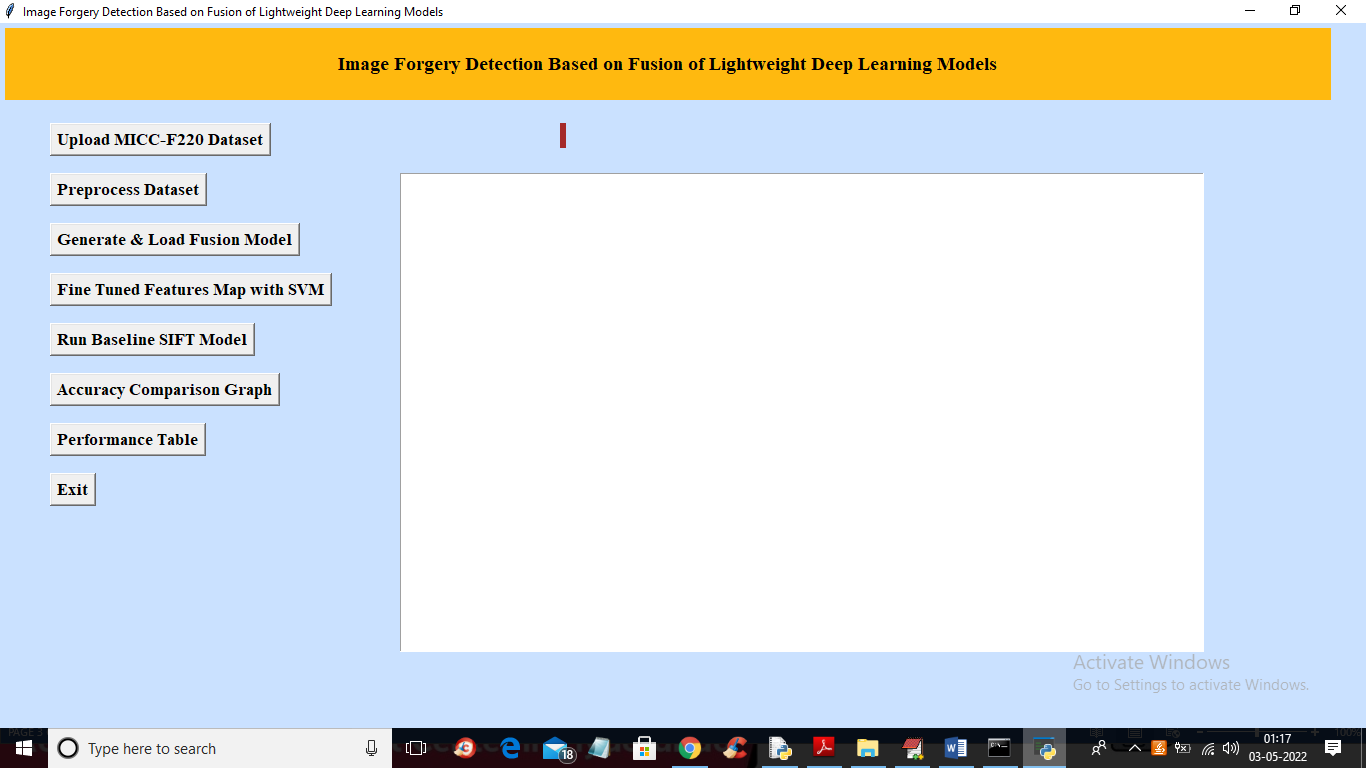
User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

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

1. **RESULTS**

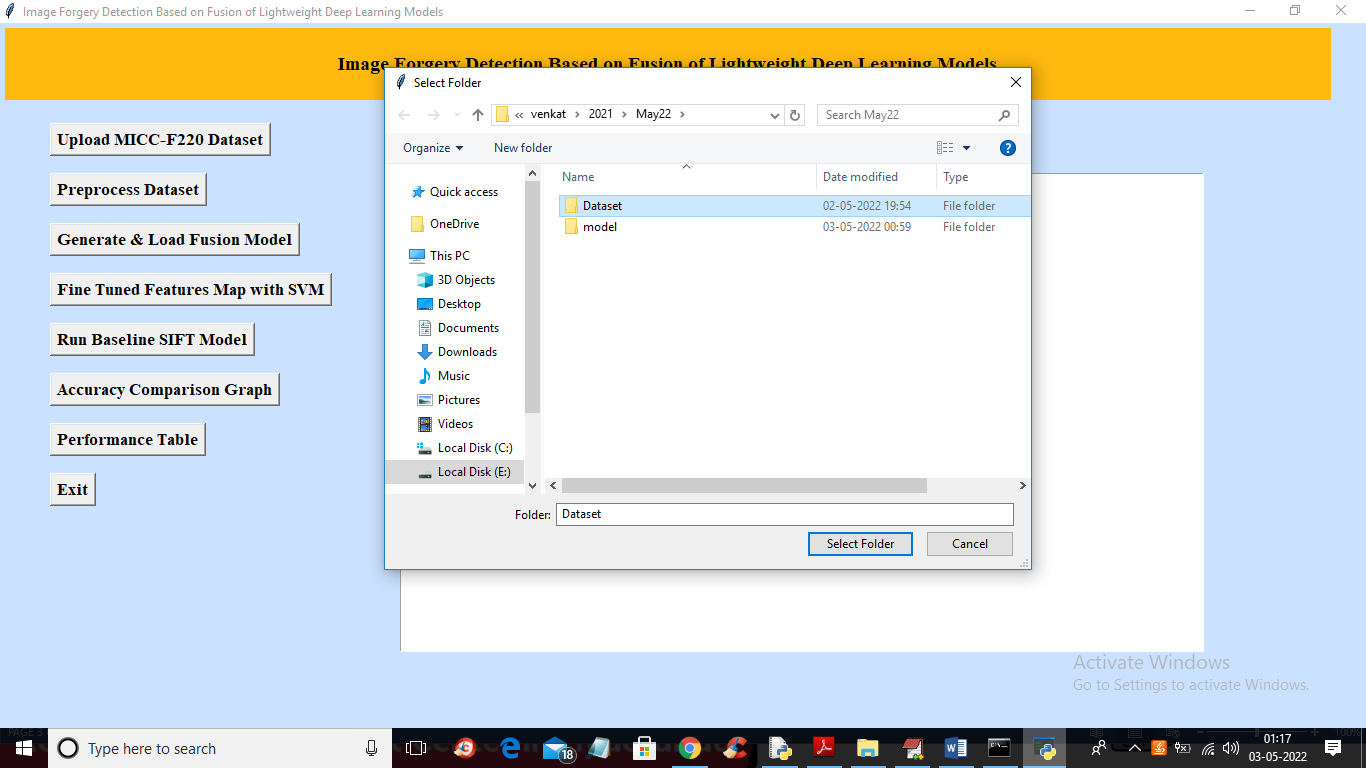
**UPLOAD MICC F220 DATASET**

To run project double click on ‘run.bat’ file to get below output



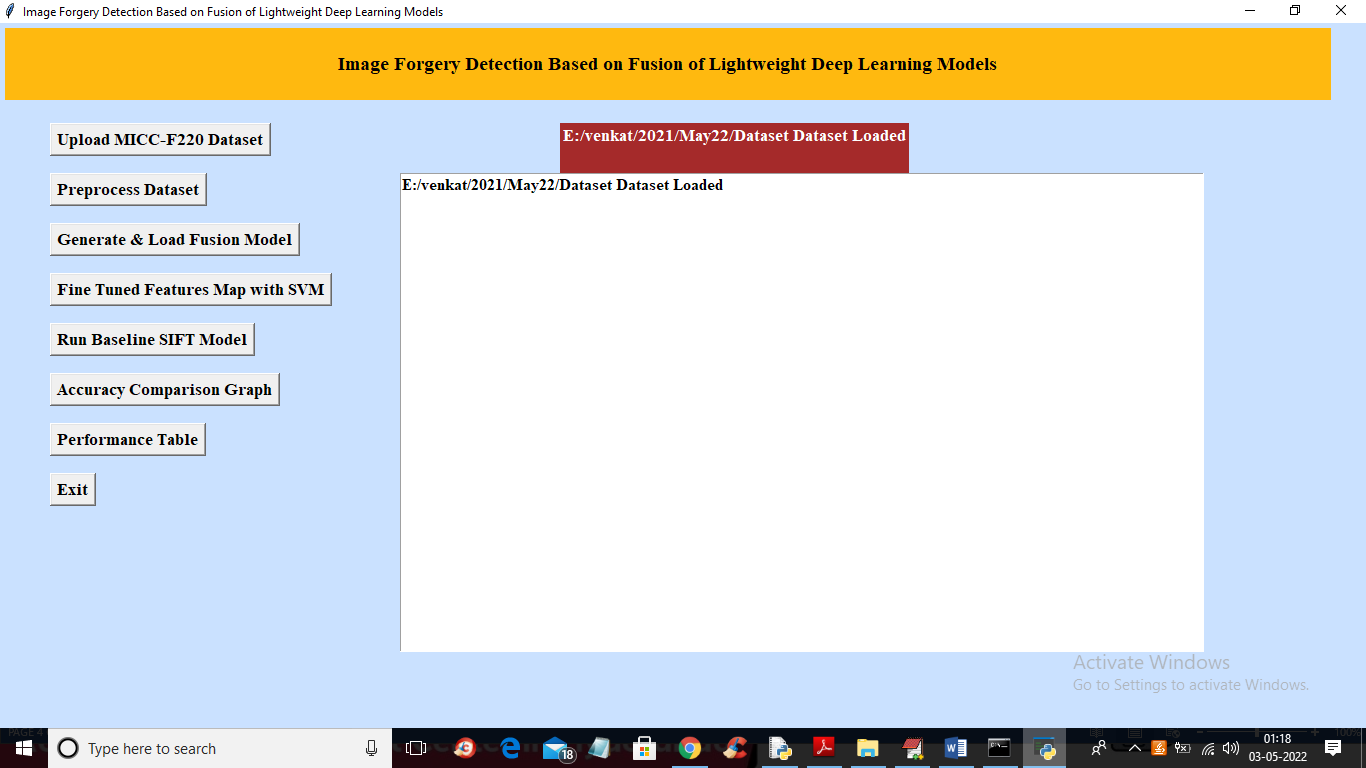
**Screenshot 8.1 Upload MICC F220 Dataset 1**

In above screen click on ‘Upload MICC-F220 Dataset’ button to upload dataset and get below output



**Screenshot 8.2 Upload MICC F220 Dataset 2**

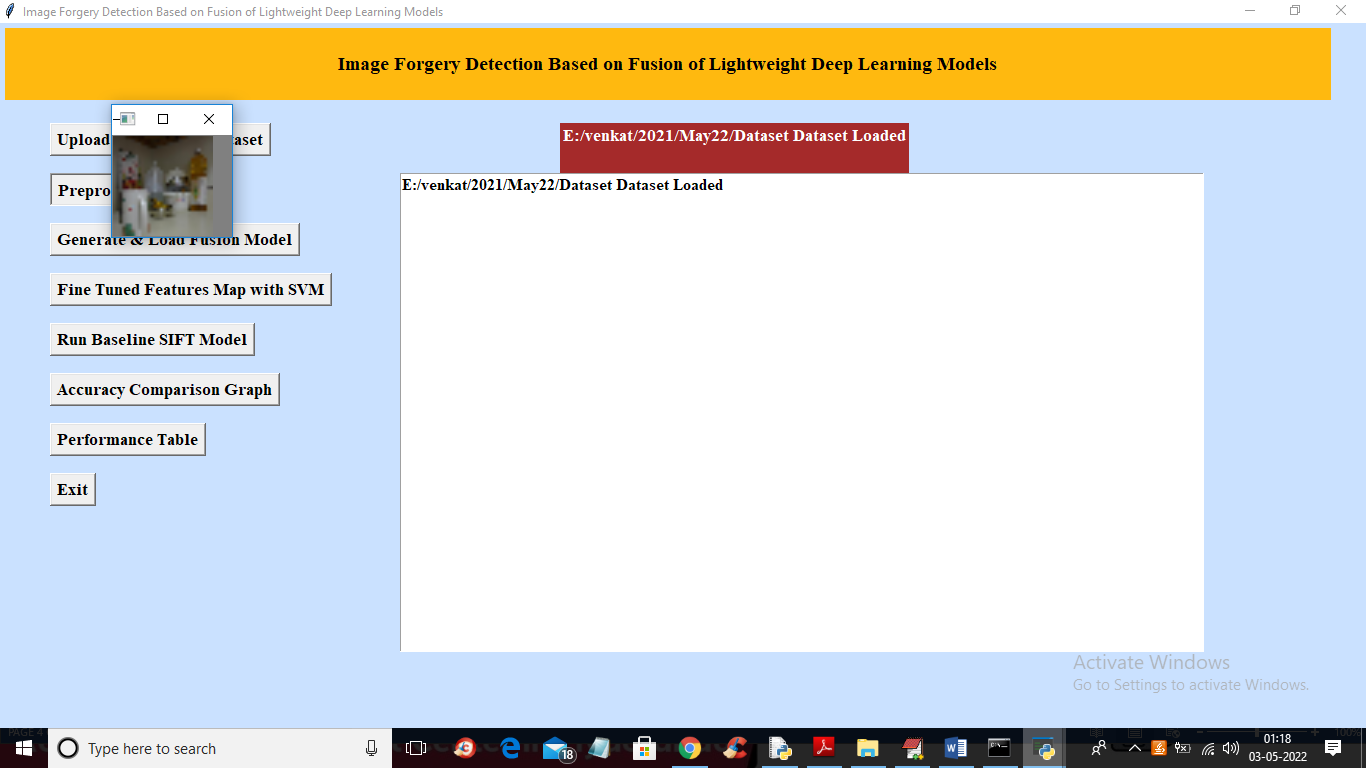
In above screen selecting and uploading ‘dataset’ folder and then click on ‘Select folder’ button to load dataset and get below output.



**Screenshot 8.3 Upload MICC F220 Dataset 3**

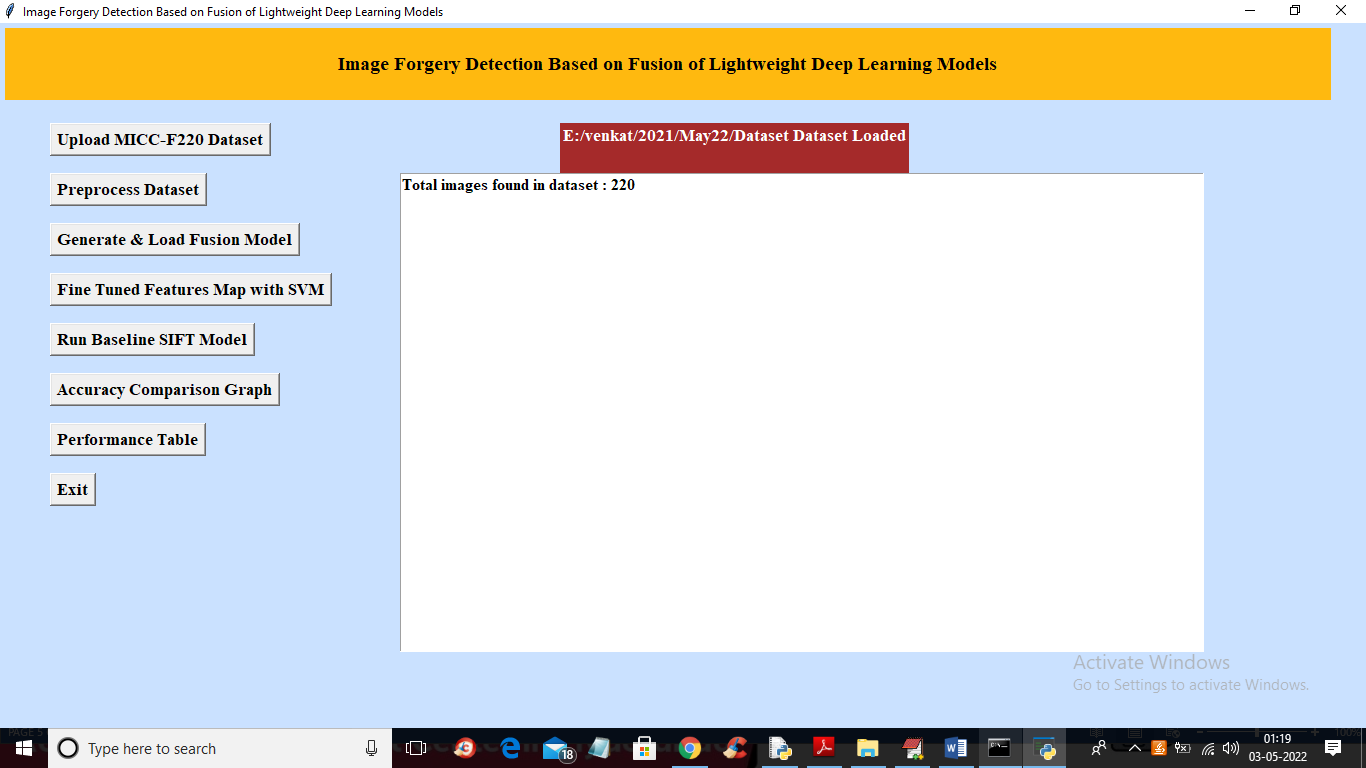
**PRE-PROCESS DATASET**

In above screen dataset loaded and now click on ‘Pre-process Dataset’ button to read all images and normalize them and get below output



**Screenshot 8.4 Pre-process Dataset 1**

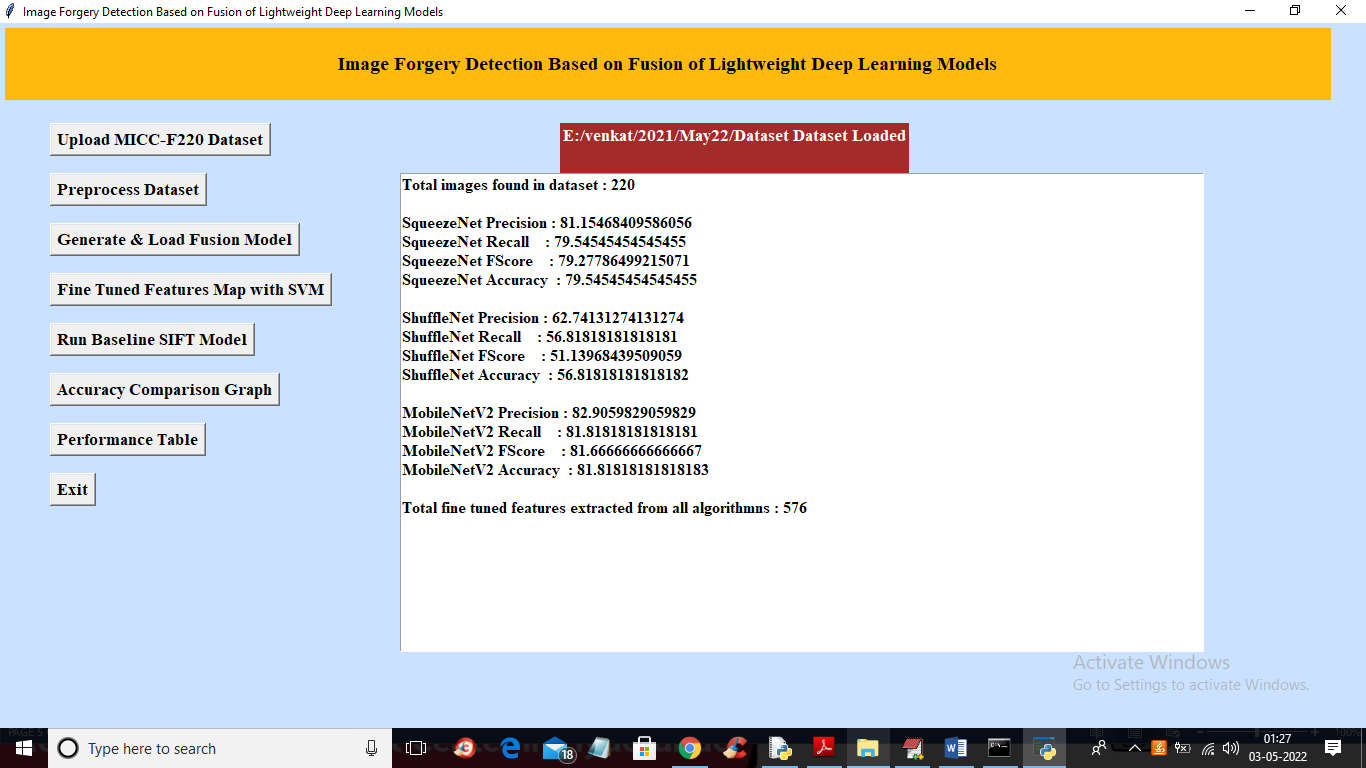
In above screen all images are processed and to check images loaded properly I am displaying one sample image and now close above image to get below output



**Screenshot 8.5 Pre-process Dataset 2**

**GENERATE AND LOAD FUSION MODEL**

In above screen we can see dataset contains 220 images and all images are processed and now click on ‘Generate & Load Fusion Model’ button to train all algorithms and then extract features from them and then calculate their accuracy. To get below output.

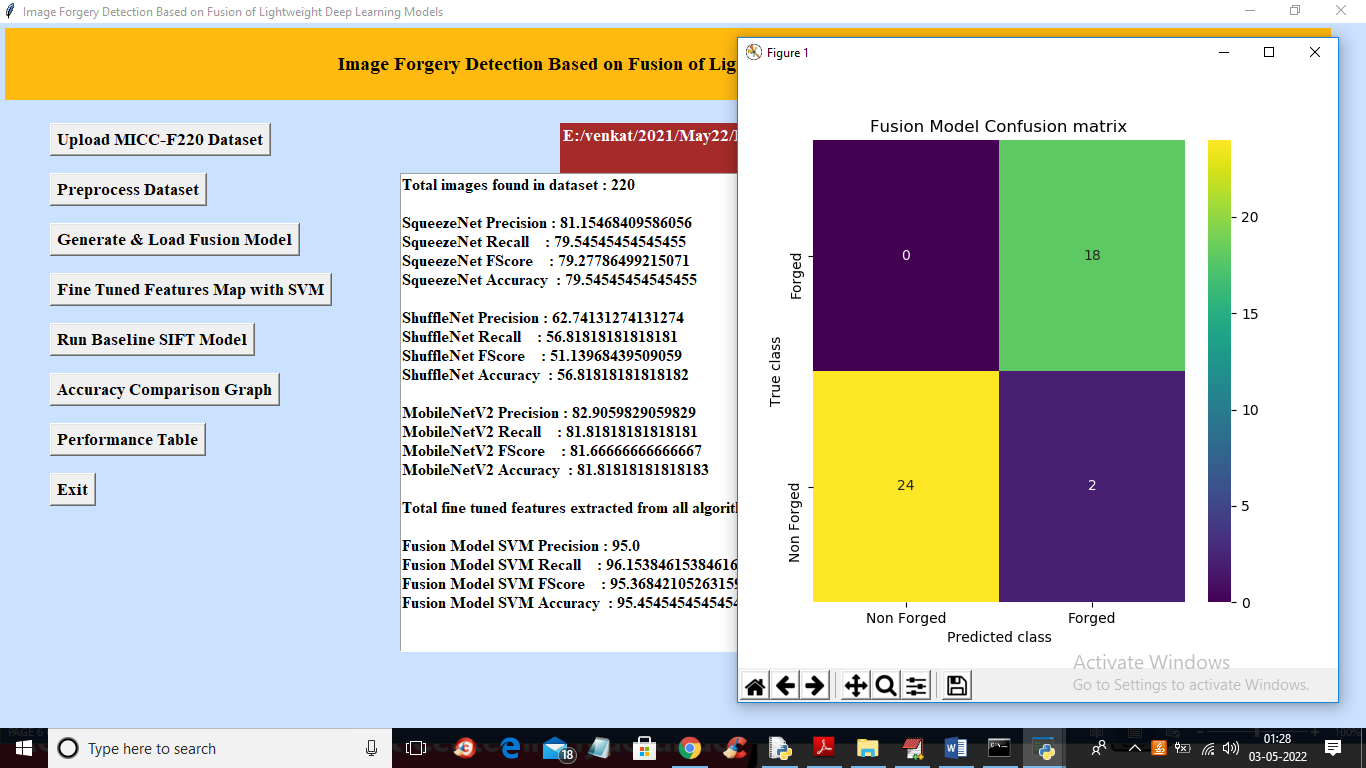


**Screenshot 8.6 Generate and Load Fusion Model**

In above screen we can see accuracy of all 3 algorithms and then in last line we can see from all 3 algorithms application extracted 576 features.

**FINE TUNED FEATURES MAP WITH SVM**

now click on ‘Fine Tuned Features Map with SVM’ to train SVM with extracted features and get its accuracy as fusion model.

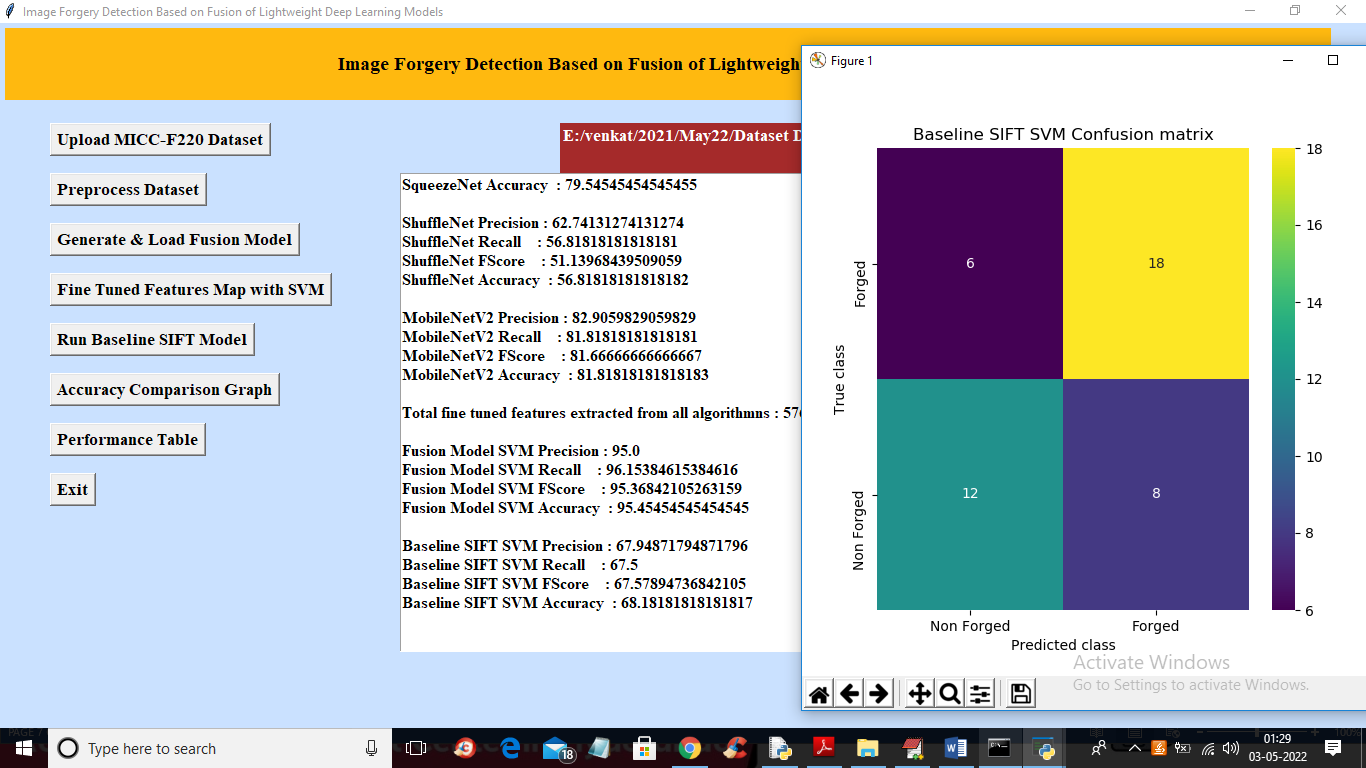


**Screenshot 8.7 Fine Tuned Features Map with SVM**

In above screen with Fine tune SVM fusion model we got 95% accuracy and in confusion matrix graph x-axis represents PREDICTED LABELS and y-axis represent TRUE labels and we can see both X and Y boxes contains a greater number of correctly prediction classes. In all algorithms we can see fine tune features with SVM has got high accuracy

**RUN BASELINE SIFT MODEL**

now close confusion matrix graph and then click on ‘Run Baseline SIFT Model’ button to train SVM with SIFT existing features and get its accuracy

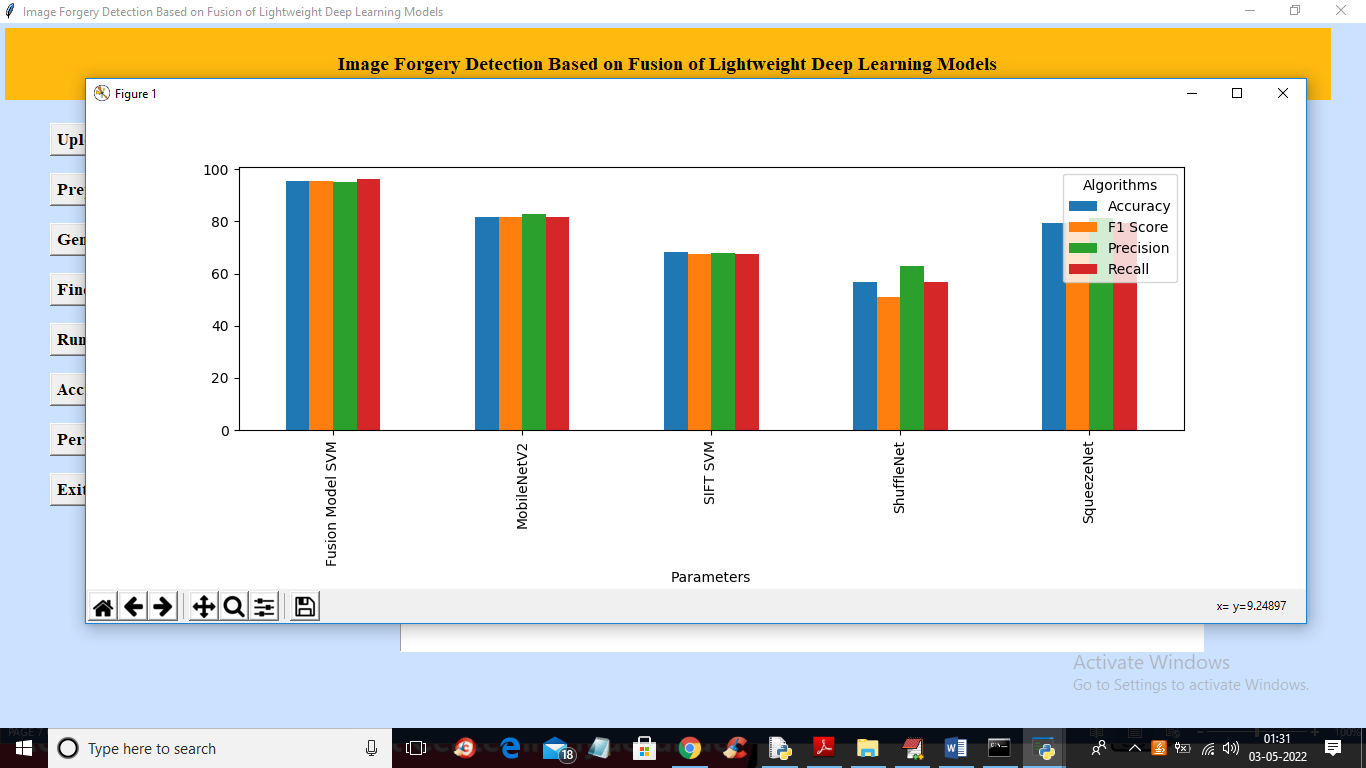


**Screenshot 8.8 Run Baseline Shift Model**

In above screen with existing SIFT SVM features we got 68% accuracy and in confusion matrix graph we can see existing SIFT predicted 6 and 8 instances incorrectly. So we can say existing SIFT features are not good in prediction

**ACCURACY COMPARISON GRAPH**

now close above graph and then click on ‘Accuracy Comparison Graph’ button to get below graph

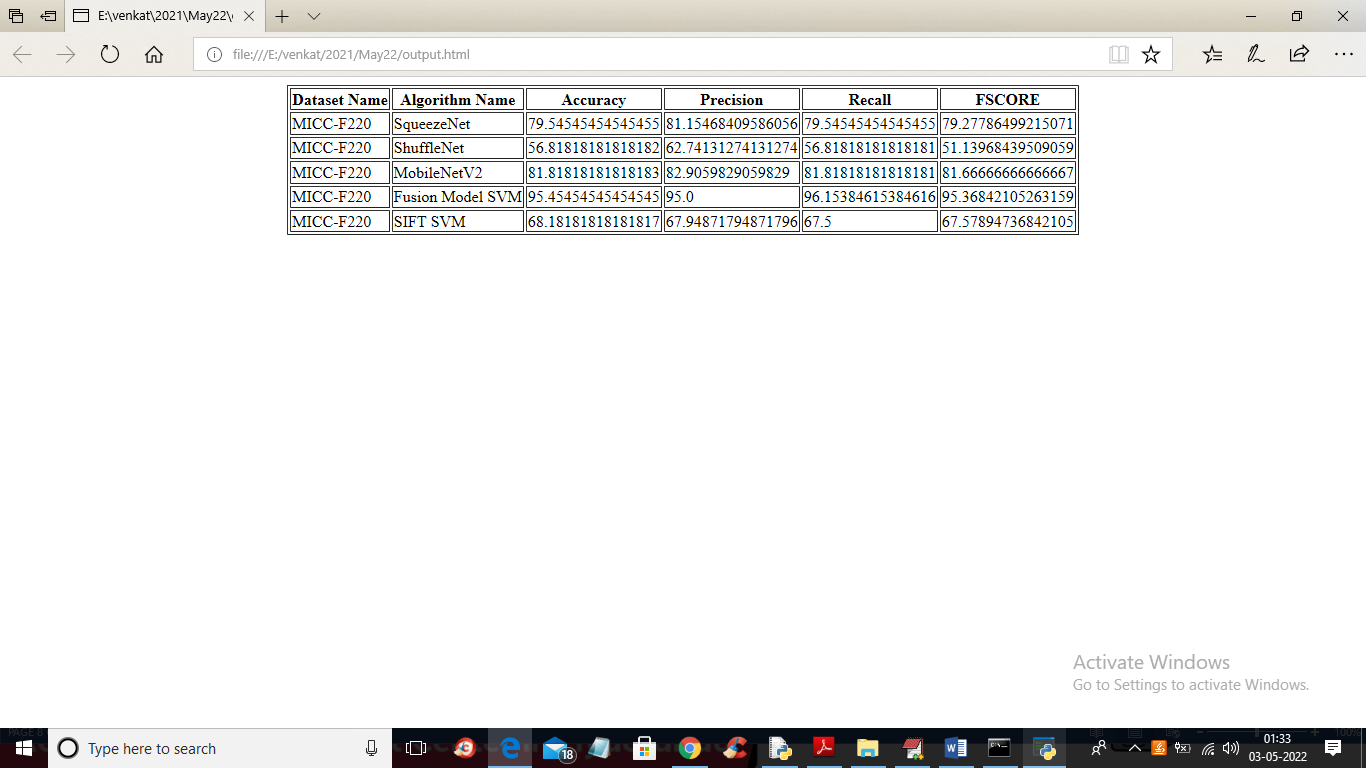


**Screenshot 8.9 Accuracy comparison graph**

In above graph x-axis represents algorithm names and y-axis represents accuracy and other metrics where each different colour bar represents different metrics like precision, recall etc.

**PERFORMANCE TABLE**

Now close above graph and then click on ‘Performance Table’ button to get result in below tabular format.

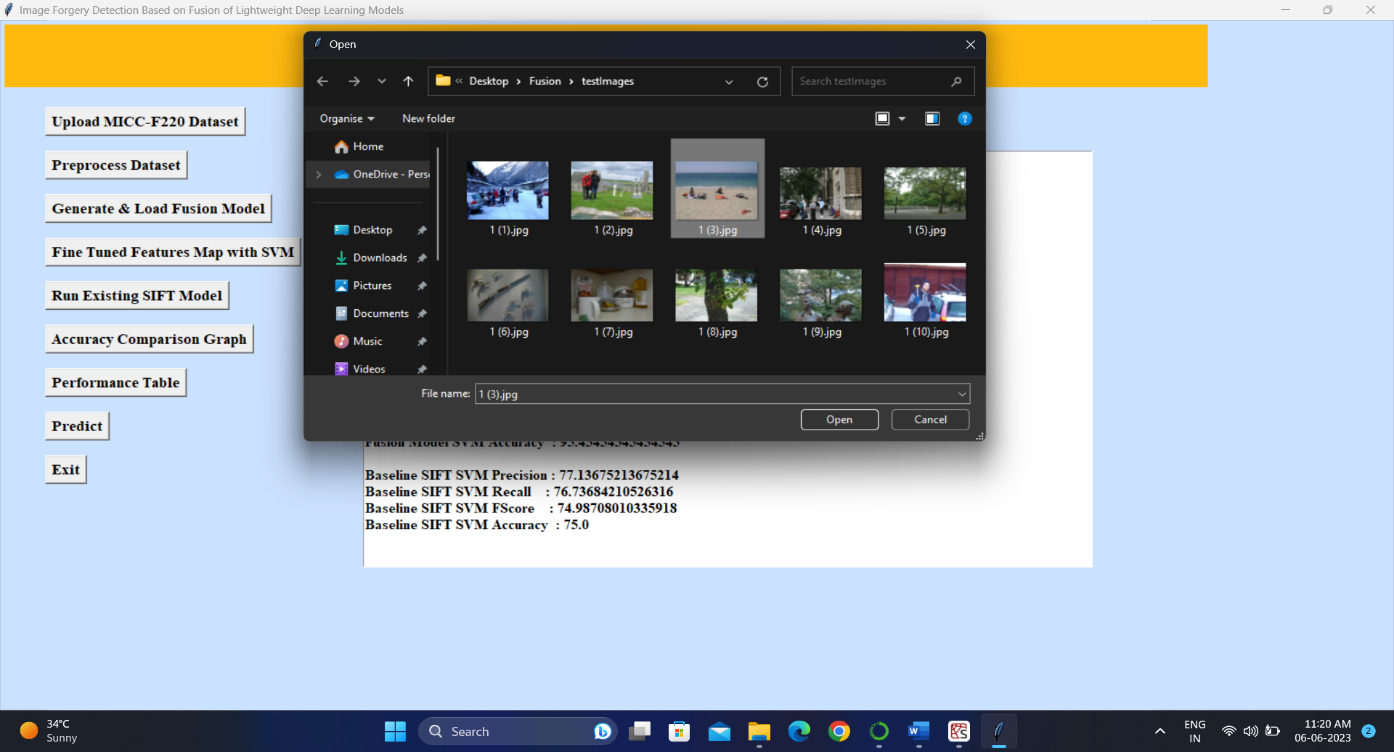


**Screenshot 8.10 Performance Table**

In above screen we can see propose fusion model SVM with fine tune features has got 95% accuracy which is better than all other algorithms.

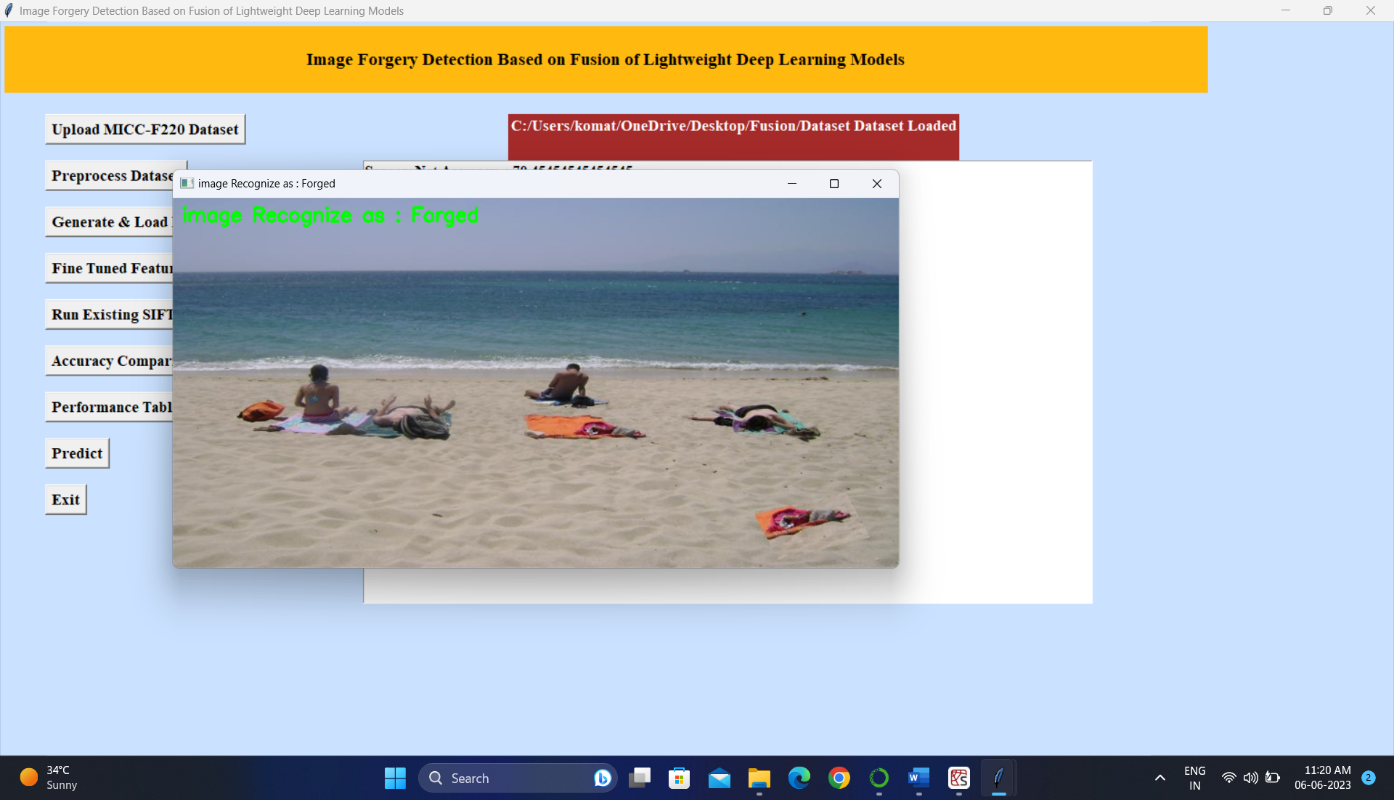
**PREDICT**

Now close above performance table and click on predict we get a below output.



**Screenshot 8.11 Uploading Image to Predict**

Now select an image and click on open to check its forged or not and we get below output of image saying it is forged image.



**Screenshot 8.12 Prediction Result**

**10. CONCLUSION**

Image forgery detection helps to differentiate between the original and the manipulated or fake images. In this work, a decision fusion of lightweight deep learning-based models is implemented for image forgery detection. The idea was to use the lightweight deep learning models namely SqueezeNet, MobileNetV2, and ShuffleNet and then combine all these models to obtain the decision on the forgery of the image. Regularization of the weights of the pretrained models is implemented to arrive at a decision of the forgery. The experiments carried out indicate that the fusion-based approach gives more accuracy than the state-of-the-art approaches. In the future, the fusion decision can be improved with other weight initialization strategies for image forgery detection.

**11. 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.

**12.REFERENCE**

1. Amerini, T. Uricchio, L. Ballan, and R. Caldelli, "Localization of JPEG Double Compression Through Multi-Domain Convolutional Neural Networks," 2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2017, pp. 1865-1871, doi: 10.1109/CVPRW.2017.233.
2. B Xiao, Y Wei, X Bi, W Li, J Ma. “Image splicing forgery detection combining coarse to refined convolutional neural network and adaptive clustering”, Information Sciences, Volume 511, Pages 172-191, 2020, ISSN 0020-0255, https://doi.org/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. Chang IC, Yu JC, Chang CC. A forgery detection algorithm for exemplar-based inpainting images using multi-region relation. Image and Vision Computing 2013; 31 (1): 57-71.
8. Rhee KH. Median filtering detection based on variations and residuals in image forensics. Turkish Journal of Electrical Engineering & Computer Science 2017; 25 (5): 3811-3826.
9. Lamba AK, Jindal N, Sharma S. Digital image copy-move forgery detection based on discrete fractional wavelet transform. Turkish Journal of Electrical Engineering & Computer Science 2018; 26 (3): 1261-1277.
10. Lin Z, He J, Tang X, Tang CK. Fast, automatic, and fine-grained tampered JPEG image detection via DCT coefficient analysis. Pattern Recognition 2009; 42 (11): 2492-2501.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Project Details** | | | | | |
| **Academic Year** | | | 2022-2023 | | |
| **Title of the Project** | | | **Image Forgery Detection Based on Fusion of Lightweight Deep Learning Models** | | |
| **Name of the Students and Hall Ticket No.** | | | K. Vishnuvardhan Reddy  (19RA1A0553)  N. Sai  (19RA1A0532)  M. Uday  (19RA1A0547) | | |
| **Name of the Guide** | | | Mrs. G. Sujatha | | |
| **Project PO Mapping** | | | | | |
| **Name of Course** | **Related** | **Description of the application** | | **Page** | **Attained** |
| **From which** | **Course** |  | | **Number** |  |
| **Principles are** | **Outcomes** |  | |  |  |
| **applied in** | **Number** |  | |  |  |
| **This Project** |  |  | |  |  |
| Python | C413.1, | Students described the basis for their | | 01 | PO2 |
| Programming | C313.1 | problem statement. | |  |  |
| Software |  |  | |  |  |
| Engineering |  |  | |  |  |
| (C413,C313) |  |  | |  |  |
| Machine Learning, Python Programming  (C413,C411) | C413.2, C413.3, | Students explained about Image forgery detection based on fusion of Lightweight deep learning models with python programming | | 22-28 | PO1 |
| Software Engineering, Python Programming (C313,C413) | C313.3, C413.2 | Students identified the existing system and its Drawbacks and proposed a Solution to it. | | 7-14 | P02,P03 |
| Software Engineering  (C313) | C313.1 | Students identified the hardware and software required for the project | | 37 | PO5 |
| Design Patterns, | C313.2, | Students explain the flow of the | | 15-20 | P03,P05, |
| Software | C322.3, | project using UML diagrams | |  | PO9, PSO3 |
| Engineering |  | designed in STAR UML, ER | |  |  |
| (C313,C322,) |  | diagram. | |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Python Programming | C413.2, C411.2 | Students explained about python programming language and designed the modules for the solution of the problem. | 42-43 | PO2, PO3,PO4 |
| (C413, C411) |
| Python Programming (C411) | C411.2 | Students Developed code for the problem Statement. | 43-54 | PO3,PO4,PO5 |
|  |
| Future Scope |  | Students explained about how they would like to further their project and | 63 | PO12,PS02 |
| Bibliography |  | Listed the references from which the literature was collected | 64 | PO8,PO12 |
| ENG |  | Prepared the thesis and intermediate progress reports and explained to the review panel. Also, continuously interact with guide and explain the progress. |  | PO9,PO10 |
|  |

**SIGNATURE OF STUDENTS SIGNATURE OF GUIDE**