Enhancing Digital Image Forgery Detection Using Transfer Learning

Vegesna Naga Rishita Varma
PG scholar, Department of MCA, DNR college, Bhimavaram, Andhra Pradesh.
K.Sri Devi
(Assistant Professor), Master of Computer Applications, DNR college, Bhimavaram, Andhra Pradesh.

Abstract: Nowadays, digital images are a main source of shared information in social media. Meanwhile, malicious software can forge such images for fake information. So, it's crucial to identify these forgeries. This problem was tackled in the literature by various digital image forgery detection techniques. But most of these techniques are tied to detecting only one type of forgery, such as image splicing or copy-move that is not applied in real life. This paper proposes an approach, to enhance digital image forgery detection using deep learning techniques via CNN to uncover two types of image forgery at the same time, The proposed technique relies on discovering the compressed quality of the forged area, which normally differs from the compressed quality of the rest of the image. A deep learning-based model is proposed to detect forgery in digital images, by calculating the difference between the original image and its compressed version, to produce a featured image as an input to the pre-trained model to train the model after removing its classifier and adding a new fine-tuned classifier. A comparison between eight different pre-trained models adapted for binary classification is done. The experimental results show that applying the technique using the adapted eight different pre-trained models outperforms the state-of-the-art methods after comparing it with the resulting evaluation metrics, charts, and graphs. Moreover, the results show that using the technique with the pre-trained model MobileNetV2 has the highest detection accuracy rate (around 95%) with fewer training parameters, leading to faster training time.

INDEX TERMS: image compression, image forgery detection (IFD), pretrained model, CNN

I. INTRODUCTION

1.1 Aim and Purpose

The aim of this research is to enhance digital image forgery detection by leveraging CNN, a powerful machine learning technique. The purpose is to improve the accuracy, efficiency, and robustness of forgery detection methods, especially in the context of increasingly sophisticated image manipulation techniques. By utilizing pre-trained deep learning models, the study seeks to reduce the need for large labeled datasets and computational resources, while achieving high detection performance for both copy-move and image splicing forgeries.

With the rapid advancement of image editing tools and technologies, digital image forgeries have become increasingly difficult to detect. From **copy-move forgeries**, where portions of an image are copied and repositioned, to **splicing forgeries**, where multiple images are merged into one, image forensics is facing a growing challenge. Detecting such manipulations is crucial, especially in areas like law enforcement, journalism, and digital security, where authenticity is paramount.

Traditional image forgery detection methods often rely on handcrafted features or simple machine learning models, which can struggle with complex manipulations and lack generalization across different image types. In contrast, **deep learning** methods, especially **convolutional neural networks (CNNs)**, have demonstrated impressive capabilities in image analysis tasks, including forgery detection. However, training CNNs from scratch requires vast amounts of labeled data and significant computational power, which is often impractical.

Transfer learning addresses these limitations by leveraging pre-trained models that have already learned robust feature representations from large datasets. These models can be fine-tuned on smaller datasets, improving detection accuracy and reducing the need for extensive training data. In the context of digital image forgery detection, transfer learning has the potential to significantly enhance detection performance, especially when dealing with real-world forgeries, which may vary in their manipulation techniques.

This research aims to harness the power of CNN to enhance digital image forgery detection, providing a more efficient and accurate solution to the growing problem of digital image manipulation. By combining deep learning with transfer learning, the goal is to push the boundaries of what is currently achievable in the field of image forensics.

The tampering of a digital image is called digital image forgery, these forged images cannot be detected by the naked eye. Such images are the primary sources of spreading fake news and misleading information in the context of society with the aid of diverse social media platforms like Facebook, Twitter, etc. [1]. The editing software tools that can make these forgeries are available for free with some advanced features that are used for image tampering such as GNU, GIMP, and Adobe Photoshop [2]. Such forgeries can be detected using digital image forgery algorithms and techniques, these algorithms are used in The associate editor coordinating the review of this manuscript and approving it for publication was Rajeeb Dey . image security especially when the original content is not available [3].

Digital image forgery means adding unusual patterns to the original images that create a heterogeneous variation in image properties and an unusual distribution of image features[3]. Figure 1 shows the classification of digital image forgery. Active approaches require essential information about the image for the verification process. The inserted information within the picture is employed to observe the modification in that picture. The active approach consists of two types: digital signatures which insert some additional data obtained from an image by the end of the acquisition process, and digital watermarking which is inserted into images either during the acquisition phase or during the processing phase.

The passive image forgery detection methods benefit from the features retained by the image allocation processes achieved in different stages of digital image acquisition and storage. Passive methodologies do not require past information about the image. These approaches exploit that the tampering actions modify the contents of information of the image that can facilitate tampering detection [4].

Copy move forgery involves duplicating a section or object within an image and pasting it again in a different location within the same image to replicate (or move) a specific scene in the image. Copy-move forgery is the most common technique used to manipulate images, it is also the most challenging type of forgery to detect due to the complexity of copying and replicating an object or section of the image with identical properties and feature distributions and pasting it within the same image [3].

some post-processing techniques can be added after CMF processes such as rotation, scaling, JPEG compression, etc. which makes the detection further difficult and complex [2]. Splicing forgery can be generated by adding or blending two images or set of images to produce an unprecedented image [3]. The source images used to generate a spliced image may include dissimilar color temperatures, illumination conditions, and noise levels based on various factors. Average filtering or some other related image processing operation can be applied as postprocessing like resizing, cropping, rotating, and retouching each of the source images to match the visual attributes, shape, and size of the target image so that the forged image can look realistic [5].

II. LITEARTURE SURVEY

A. DEEP NEURAL NETWORK-BASED IMAGE FORGERY DETECTION TECHNIQUES DNNs can autonomously learn an extensive number of features. Over the past few years, a variety of image forgery detection methods have been proposed, for detecting image forgery, where many of which relied on deep learning [5]. By constructing an appropriate neural network, deep learning networks can identify complex hidden patterns in data and effectively distinguish the forged parts from the original image [9]. Deep learning technique has proven to be effective in resolving many activities or issues that machine learning algorithms were previously unable to address [8].

When considering splicing detection, a scheme was proposed in [10] based on the local feature descriptor which is learned by a DNN. An improved initialization based on the (SRM) was proposed and developed a splicing localization scheme based on the proposed CNN model and fully connected conditional random field (CRF) with SVM which is robust against JPEG compression.

TABLE 1. Summary of deep learning-based image forgery detection techniques.

In [11], a (CNN) model was developed using a relatively small number of parameters that can be used as an on-time detection model. For splicing and copy-move separately, an end-to-end fully CNN that combines multi-resolution hybrid features, from RGB and noise streams was introduced in [12], where a tamper-guided dual self-attention (TDSA) module was designed to capture the difference between tampered and non-tampered areas and segments them from the image. A proposed hybrid features and semantic reinforcement network (HFSRNet) for IFD at the pixel level was proposed in [13], where the network employs an encoding and decoding approach and utilizes Long-Short Term Memory (LSTM) technology.

For copy move, [14] introduced a copy-move forgery detection and localization model based on super boundaryto-pixel direction (super-BPD) segmentation and deep CNN (DCNN). Starting with employing the segmentation technique that is used to enhance the connection among identical image blocks, thereby improving the accuracy of forgery detection, the DCNN is used to extract image features, ending by using image BPD information to optimize the edges of the rough detected image and obtain the final detected image.

[15] developed a deep learning CNN model which used multi-scale input and multiple stages of convolutional layers, with two different parts, encoder, and decoder. In [16], a simple and lightweight convolutional neural network (CNN) has been proposed for the automatic detection of copy-move forgery detection, which has a high detection accuracy rate. For copy-move and splicing together, [9] used a new image segmentation model U-Net by adding L2 regularization.

Reference [17] introduced a system for IFD using double image compression, in which the difference between an original image and recompressed one was used in training the model, the method is capable of detecting both image splicing and copy-move together.

B. PRETRAINED NETWORK-BASED IMAGE FORGERY DETECTION TECHNIQUES Different IFD techniques based on transfer learning will be discussed in this section. For splicing, [18] presented multiple image-splicing forgeries using Mask R-CNN and MobileNetV1 backbone. A novel approach utilizing ResNet50v2 was introduced in [19], that considered image batches as an input and used YOLO CNN weights with ResNet50v2 architecture. For splicing and copy-move separately, [20] proposed a multi-task learning network called FBI-Net based on (DCT).

The network employs a fully convolutional encoder-decoder architecture, and the Dilated Frequency Self-Attention Module (DFSAM) in the bridge layer adjusts fused features. Reference [21] introduced a lightweight model using mask R-CNN with MobileNet to detect copy-move and imagesplicing forgeries. For copy move, [22] used SmallerVGGNet and MobileNetV2, time- and memory-saving deep learning models. In [23] an Optimal Deep Transfer Learning based Copy Move Forgery Detection (ODTLCMFD) technique was presented that derived a DL model for the classification of target images and then localized the copy moved regions.

They used the MobileNet model with a political optimizer (PO) for feature extraction and the least square support vector machine (LS-SVM) model with an enhanced bird swarm algorithm (EBSA) for classification. They utilized the EBSA algorithm to modify the parameters in the Multiclass Support Vector Machine (MSVM) technique to enhance the classification performance. Reference [24] provided an automated deep learning-based fusion model for detecting and localizing copy-move forgeries (DLFM-CMDFC), that combined models of generative adversarial networks (GANs) and densely connected networks (DenseNet). The two outputs were merged in the DLFM-CMDFC technique to create a layer for encoding the input vectors with the first layer of an extreme learning machine (ELM) classifier.

III. PROPOSED METHOD

The proposed approach considers the fact shown in [17], that copying a part of an image from one to another may impose some changes in the image properties due to the different sources of the images. Although these changes may not be detectable to the human eye, they can be detected by CNNs in manipulated images. The proposed model aims to avoid all of the forementioned drawbacks, by adapting the idea of calculating the difference in compression qualities to produce the featured image as an input to a deep neural network with the assistance of a pretrained model to benefit from the power of transfer learning. As a result, the evaluation matrix will be improved including the accuracy rate that will get better than that which was recorded when using CNN in [17].

This will be elaborated and discussed in the following section. In a forged image, if the image is compressed, the forged section of the image will be compressed differently than the rest of the image. This is because the source of the original image differs from the source of the forged section. When analyzing the difference between the original image and its compressed version, the forgery component becomes more distinguished. Therefore, this aspect can be utilize

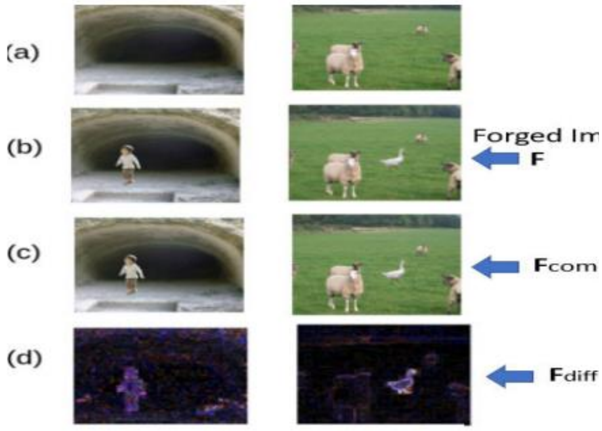


FIGURE 2. Set of images created in the proposed work.

The set of images created in the proposed work can be shown in Figure 2. The first image, (a) represents the original image without forgery, (b) represents the forged image that is denoted as F, (c) represents the compressed version of (b) that is denoted as Fcomp, (d) represents the mathematical difference between F and Fcomp denoted as Fdiff.

In the proposed model, the preprocessing phase starts with the forged image (input image), denoted as F, as shown in Figure 3, which is compressed to get a compressed version of the input image, denoted as Fcomp. The difference between the forged image and its compressed version is then calculated by mathematical subtraction, denoted as (Fdiff), as shown in Equation (1). Fdiff = F − −Fcomp (1) As a result, the forged part of the image appears in (Fdiff) due to the difference between the source of the forged and original parts. Fdiff is then reshaped to 160 × 160 pixels to fit as an input feature image for training a pre-trained model (M), which is then used to classify images as forged or authentic.

Figure 3 shows the overall architecture of the proposed system. In Figure 3, the pre-trained model, shown as block (M), is used to extract features from input images (Fdiff) and classify them as authentic images or forged images. In this block (pre-trained model), eight different pre-trained models are considered (one at a time) namely, VGG16 [27], VGG19 [28], ResNet [29], DenseNet [30], Xception [31], and MobileNet [32] for fine training with input images (Fdiff), to nominate the model with the best performance among them. Each model of the forementioned eight pre-trained models has its own architecture which consists of a set of convolutional layers with activation functions and ends with a set of fully connected layers that can classify up to 1000 classes of images.

So, each model architecture has to be modified to fit the binary classification problem with only two classes (authentic or forged images) as in the case of image forgery detection problems. Therefore, the native fully connected layers in each model are replaced with a new set of fully connected classification layers able to handle the binary classification problem at hand. The convolutional layers in every model should remain untouched since they contain all the trainable parameters used in transfer learning. Figure 4 shows the detailed architecture of the proposed model classifier with the newly added layers. After removing the fully connected layers of the pre-trained model, a flatten layer is added to convert the input data, which is typically a multi-dimensional array, into a one-dimensional vector that can be fed to the next layers.

The next two (new) layers are fully connected layers added with the ReLU activation function. The two layers have 1024 and 256 neurons, respectively. After each layer, a dropout 0.5 was added to prevent overfitting by randomly dropping out (setting to zero) about 50% of the output values of the previous layer will be randomly set to zero during the training phase. The last fully connected layer with a sigmoid activation function is added, which is the common activation function used in binary classification problems.

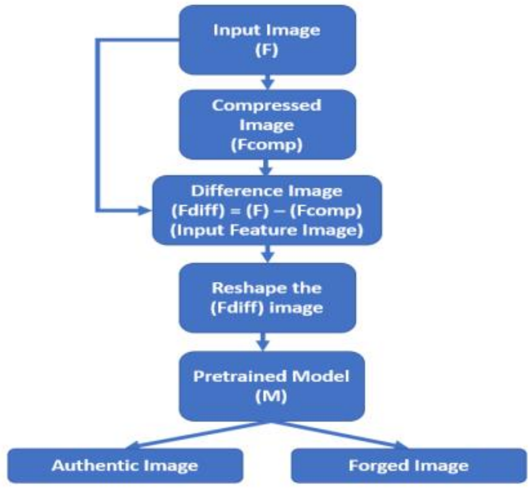


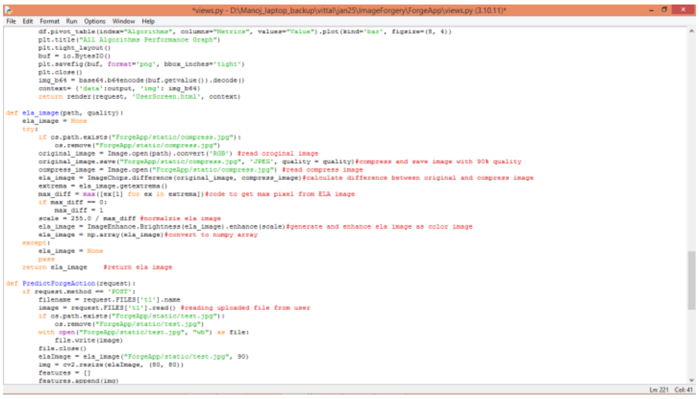
FIGURE 3. Flowchart of the proposed system (System Architecture).

IV. RESULTS

In social media images are one of the major sources to spread useful information but some malicious users will utilize advance tools to generate forge image which can spread wrong message in the social media. To fight against such forgery many existing algorithms like slicing and copy move are introduced but their detection accuracy is not accurate.

To overcome from existing drawbacks author of this paper introducing novel concept called Transfer Learning based Forgery detection. Propose model compute difference between original and compress image to generate new image and this new image features will get trained with pre-trained models to detect forgery. In propose paper author has used 8 different pre-trained models such as DenseNet121, Resnet50, VGG19, MobileNetV2 and many more. Among all algorithms MobileNetV2 got highest accuracy with least training parameters.

In below screen showing python code to calculate compress and difference image called ELA image.

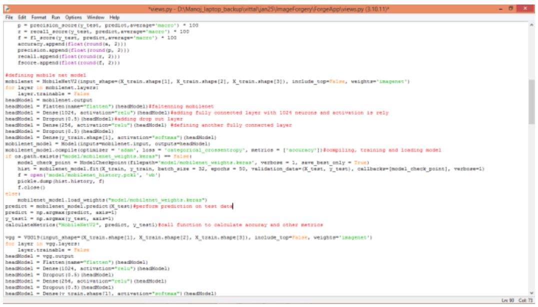

views.py - D:\Manoj_laptop_backup\vittal\jan25\ImageForgery\ForgeApp\views.py (3.10.11)
File
Edit
Format
Run
Options
Window
Help
Ln: 221 Col: 41

```
        df.pivot_table(index="Algorithms", columns="Metrics", values="Value").plot(kind='bar', figsize=(8, 4))
        plt.title("All Algorithms Performance Graph")
        plt.tight_layout()
        buf = io.BytesIO()
        plt.savefig(buf, format='png', bbox_inches='tight')
        plt.close()
        img_b64 = base64.b64encode(buf.getvalue()).decode()
        context= {'data':output, 'img': img_b64}
        return render(request, 'UserScreen.html', context)

def ela_image(path, quality):
    ela_image = None
    try:
        if os.path.exists("ForgeApp/static/compress.jpg"):
            os.remove("ForgeApp/static/compress.jpg")
        original_image = Image.open(path).convert('RGB') #read oroginal image
        original_image.save("ForgeApp/static/compress.jpg", 'JPEG', quality = quality)#compress and save image with 90% quality
        compress_image = Image.open("ForgeApp/static/compress.jpg") #read compress image
        ela_image = ImageChops.difference(original_image, compress_image)#calculate difference between original and compress image
        extrema = ela_image.getextrema()
        max_diff = max([ex[1] for ex in extrema])#code to get max pixel from ELA image
        if max_diff == 0:
            max_diff = 1
        scale = 255.0 / max_diff #normalzie ela image
        ela_image = ImageEnhance.Brightness(ela_image).enhance(scale)#generate and enhance ela image as color image
        ela_image = np.array(ela_image)#convert to numpy array
    except:
        ela_image = None
        pass
    return ela_image    #return ela image

def PredictForgeAction(request):
    if request.method == 'POST':
        filename = request.FILES['t1'].name
        image = request.FILES['t1'].read() #reading uploaded file from user
        if os.path.exists("ForgeApp/static/test.jpg"):
            os.remove("ForgeApp/static/test.jpg")
        with open("ForgeApp/static/test.jpg", "wb") as file:
            file.write(image)
        file.close()
        elaImage = ela_image("ForgeApp/static/test.jpg", 90)
        img = cv2.resize(elaImage, (80, 80))
        features = []
        features.append(img)
```

In above screen read red colour comments to calculate difference image called ELA image. Each ELA features will be input to pre-train models to train algorithms. In below screen showing MobileNetV2 code getting trained on ELA features and to this MobileNetV2 algorithm we are adding extra layers


views.py - D:\Manoj_laptop_backup\vittal\jan25\ImageForgery\ForgeApp\views.py (3.10.11)
File
Edit
Format
Run
Options
Window
Help
Ln: 90 Col: 73

```
    p = precision_score(y_test, predict,average='macro') * 100
    r = recall_score(y_test, predict,average='macro') * 100
    f = f1_score(y_test, predict,average='macro') * 100
    accuracy.append(float(round(a, 2)))
    precision.append(float(round(p, 2)))
    recall.append(float(round(r, 2)))
    fscore.append(float(round(f, 2)))

#defining mobile net model
mobilenet = MobileNetV2(input_shape=(X_train.shape[1], X_train.shape[2], X_train.shape[3]), include_top=False, weights='imagenet')
for layer in mobilenet.layers:
    layer.trainable = False
headModel = mobilenet.output
headModel = Flatten(name="flatten")(headModel)#faltenning mobilenet
headModel = Dense(1024, activation="relu")(headModel)#adding fully connected layer with 1024 neurons and activation is rely
headModel = Dropout(0.5)(headModel)#adding drop out layer
headModel = Dense(256, activation="relu")(headModel) #defining another fully connected layer
headModel = Dropout(0.5)(headModel)
headModel = Dense(y_train.shape[1], activation="softmax")(headModel)
mobilenet_model = Model(inputs=mobilenet.input, outputs=headModel)
mobilenet_model.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics = ['accuracy'])#compiling, training and loading model
if os.path.exists("model/mobilenet_weights.keras") == False:
    model_check_point = ModelCheckpoint(filepath='model/mobilenet_weights.keras', verbose = 1, save_best_only = True)
    hist = mobilenet_model.fit(X_train, y_train, batch_size = 32, epochs = 50, validation_data=(X_test, y_test), callbacks=[model_check_point], verbose=1)
    f = open('model/mobilenet_history.pckl', 'wb')
    pickle.dump(hist.history, f)
    f.close()
else:
    mobilenet_model.load_weights("model/mobilenet_weights.keras")
predict = mobilenet_model.predict(X_test)#perform prediction on test data
predict = np.argmax(predict, axis=1)
y_test1 = np.argmax(y_test, axis=1)
calculateMetrics("MobileNetV2", predict, y_test1)#call function to calculate accuray and other metrics

vgg = VGG19(input_shape=(X_train.shape[1], X_train.shape[2], X_train.shape[3]), include_top=False, weights='imagenet')
for layer in vgg.layers:
    layer.trainable = False
headModel = vgg.output
headModel = Flatten(name="flatten")(headModel)
headModel = Dense(1024, activation="relu")(headModel)
headModel = Dropout(0.5)(headModel)
headModel = Dense(256, activation="relu")(headModel)
headModel = Dropout(0.5)(headModel)
headModel = Dense(y_train.shape[1], activation="softmax")(headModel)
```

In above screen showing mobilenetv2 code utilizing as transfer learning to train forge detection model.

Note: training all 8 models along with existing CNN algorithm is difficult so we have trained MobileNetV2, VGG19, DenseNet121 and existing CNN.

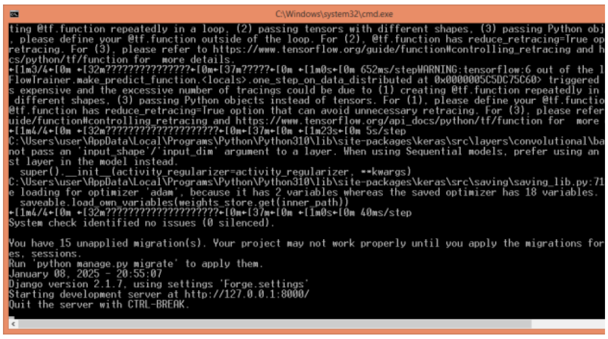
To implement this project we have designed following modules

1) User Login: user can login to system using username and password as admin and admin.
2) Load & Process Dataset: after login user can run this module to load and process dataset and then split dataset images into train and test where application using 80% images for training and 20% for testing
3) Train Pre-Train Models: 80% training images will be input to all algorithms to train a model and this model will be applied on 20% test images to calculate prediction accuracy and other metrics

4) Forgery Detection: using this module user can upload test image and then application extract ELA features and input to MobileNetV2 algorithm to detect as Forge or authentic image.

SCREEN SHOTS

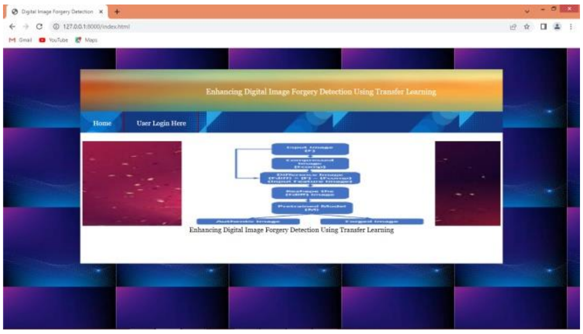
To run project install python 3.10 and then install all packages given in requirements.txt file and then double click on 'runServer.bat' file to start python server and get below page



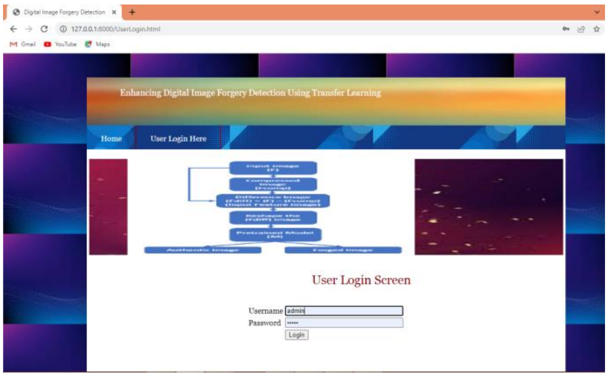
```
ting @tf.function repeatedly in a loop, (2) passing tensors with different shapes, (3) passing Python obj
, please define your @tf.function outside of the loop. For (2), @tf.function has reduce_retracing=True op
retracing. For (3), please refer to https://www.tensorflow.org/guide/function#controlling_retracing and h
cs/python/tf/function for  more details.
←[1m3/4←[0m ←[32m???????????????←[0m←[37m?????←[0m ←[1m0s←[0m 652ms/stepWARNING:tensorflow:6 out of the l
FlowTrainer.make_predict_function.<locals>.one_step_on_data_distributed at 0x0000005C5DC75C60> triggered
s expensive and the excessive number of tracings could be due to (1) creating @tf.function repeatedly in
 different shapes, (3) passing Python objects instead of tensors. For (1), please define your @tf.functio
@tf.function has reduce_retracing=True option that can avoid unnecessary retracing. For (3), please refer
uide/function#controlling_retracing and https://www.tensorflow.org/api_docs/python/tf/function for  more
←[1m4/4←[0m ←[32m????????????????????←[0m←[37m←[0m ←[1m23s←[0m 5s/step
C:\Users\user\AppData\Local\Programs\Python\Python310\lib\site-packages\keras\src\layers\convolutional\ba
not pass an 'input_shape'/'input_dim' argument to a layer. When using Sequential models, prefer using an
st layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
C:\Users\user\AppData\Local\Programs\Python\Python310\lib\site-packages\keras\src\saving\saving_lib.py:71
e loading for optimizer 'adam', because it has 2 variables whereas the saved optimizer has 18 variables.
  saveable.load_own_variables(weights_store.get(inner_path))
←[1m4/4←[0m ←[32m????????????????????←[0m←[37m←[0m ←[1m0s←[0m 40ms/step
System check identified no issues (0 silenced).

You have 15 unapplied migration(s). Your project may not work properly until you apply the migrations for
es, sessions.
Run 'python manage.py migrate' to apply them.
January 08, 2025 - 20:55:07
Django version 2.1.7, using settings 'Forge.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

In above screen python server started and now open browser and then enter URL as http://127.0.0.1:8000/index.html and then press enter key to get below page


Digital Image Forgery Detection
127.0.0.1:8000/index.html
Gmail
YouTube
Maps
Enhancing Digital Image Forgery Detection Using Transfer Learning
Home
User Login Here
Enhancing Digital Image Forgery Detection Using Transfer Learning

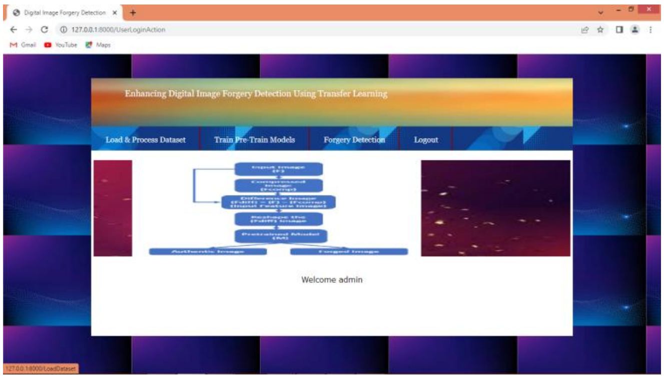
In above screen click on 'User Login Here' link to get below page


Digital Image Forgery Detection
127.0.0.1:8000/UserLogin.html
Gmail
YouTube
Maps
Enhancing Digital Image Forgery Detection Using Transfer Learning
Home
User Login Here
Login

User Login Screen

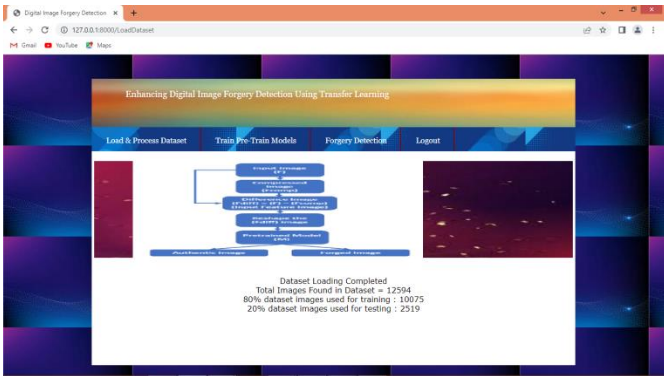
Username admin
Password •••••

In above screen user is login by entering username and password as 'admin and admin' and then press button to get below page


Digital Image Forgery Detection
127.0.0.1:8000/UserLoginAction
Gmail
YouTube
Maps
Enhancing Digital Image Forgery Detection Using Transfer Learning
Load & Process Dataset
Train Pre-Train Models
Forgery Detection
Logout
127.0.0.1:8000/LoadDataset

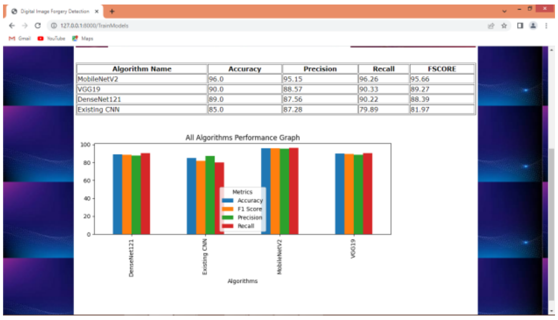
Welcome admin

In above screen click on 'Load & Process Dataset' link to get below page


Digital Image Forgery Detection
127.0.0.1:8000/LoadDataset
Gmail
YouTube
Maps
Enhancing Digital Image Forgery Detection Using Transfer Learning
Load & Process Dataset
Train Pre-Train Models
Forgery Detection
Logout

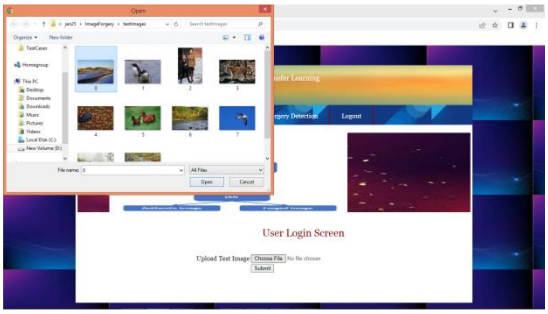
Dataset Loading Completed
Total Images Found in Dataset = 12594
80% dataset images used for training : 10075
20% dataset images used for testing : 2519

In above screen dataset loaded and can see total images loaded from dataset along with training and testing size and now click on 'Train Pre-Train Models' link to train models and get below page

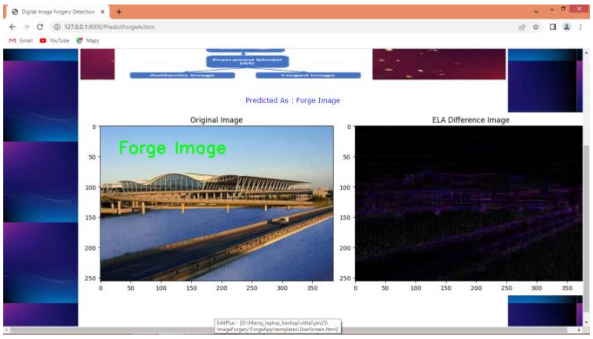


Algorithm Name	Accuracy	Precision	Recall	FSCORE
MobileNetV2	96.0	95.15	96.26	95.66
VGG19	90.0	88.57	90.33	89.27
DenseNet121	89.0	87.56	90.22	88.39
Existing CNN	85.0	87.28	79.89	81.97

In above screen in tabular format can see accuracy and precision, recall and FSCORE of all algorithms and in graph x-axis represents algorithm names and y-axis represents accuracy and other metrics in different colour bars and in all algorithms MobileNetV2 got high performance. Now click on 'Forgery Detection' link to get below page

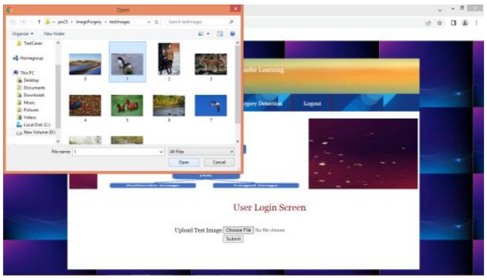


In above screen selecting and uploading test image from ‘test Images’ folder and then click on ‘Open and submit’ button to get below MobileNetV2 detection output

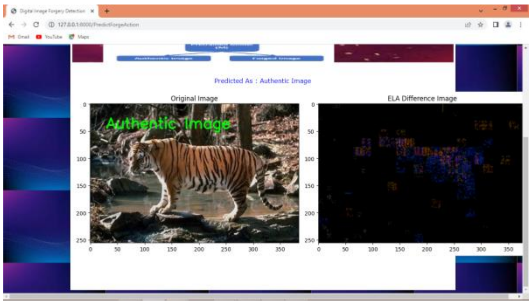


Predicted As : Forge Image

In above screen first image is the original image and second image is the difference computed image and then MobileNetV2 predicted as ‘Forge Image’. Similarly you can upload and test other images. Below showing another example



In above screen uploading another image and below is the output



Predicted As : Authentic Image

In above screen loaded image predicted as ‘Authentic image’. Similarly you van test any image.

V. CONCLUSION

In the proposed model, the preprocessing phase starts with the forged image (input image), denoted as F, as shown in Figure 3, which is compressed to get a compressed version of the input image, denoted as Fcomp. The difference between the forged image and its compressed version is then calculated by mathematical subtraction, denoted as (Fdiff), as shown in Equation (1). Fdiff = F − −Fcomp (1) As a result, the forged part of the image appears in (Fdiff) due to the difference between the source of the forged and original parts. Fdiff is then reshaped to 160 × 160 pixels to fit as an input feature image for training a pre-trained model (M), which is then used to classify images as forged or authentic. Figure 3 shows the overall architecture of the proposed system. In Figure 3, the pre-trained model, shown as block (M), is used to extract features from input images (Fdiff) and classify them as authentic images or forged images. In this block (pre-trained model), eight different pre-trained models are considered (one at a time) namely, VGG16 [27], VGG19 [28], ResNet [29], DenseNet [30], Xception [31], and MobileNet [32] for fine training with input images (Fdiff), to nominate the model with the best performance among them. Each model of the forementioned eight pre-trained models has its own architecture which consists of a set of convolutional layers with activation functions and ends with a set of fully connected layers that can classify up to 1000 classes of images. So, each model architecture has to be modified to fit the binary classification problem with only two classes (authentic or forged images) as in the case of image forgery detection problems. Therefore, the native fully connected layers in each model are replaced with a new set of fully connected classification layers able to handle the binary classification problem at hand. The convolutional layers in every model should remain untouched since they contain all the trainable parameters used in transfer learning. Figure 4 shows the detailed architecture of the proposed model classifier with the newly added layers. After removing the fully connected layers of the pre-trained model, a flatten layer is added to convert the input data, which is typically a multi-dimensional array, into a one-dimensional vector that can be fed to the next layers. The next two (new) layers are fully connected layers added with the ReLU activation function. The two layers have 1024 and 256 neurons, respectively. After each layer, a dropout 0.5 was added to prevent overfitting by randomly dropping out (setting to zero) about 50% of the output values of the previous layer will be randomly set to zero during the training phase. The last fully connected layer with a sigmoid activation function is added, which is the common activation function used in binary classification problems. localization. The combination of the proposed approach with other known image localization techniques will improve the accuracy, but it may increase the time complexity so it will need more improvement. The detection of forged videos that may be created by merging several videos is an incredibly challenging task.

REFERENCES

[1] K. D. Kadam, S. Ahirrao, and K. Kotecha, “Multiple image splicing dataset (MISD): A dataset for multiple splicing,” *Data*, vol. 6, no. 10, p. 102, Sep. 2021.

[2] R. Agarwal, O. P. Verma, A. Saini, A. Shaw, and A. R. Patel, “The advent of deep learning-based,” in *Innovative Data Communication Technologies and Application*. Singapore: Springer, 2021.

[3] M. A. Elaskily, M. H. Alkinani, A. Sedik, and M. M. Dessouky, “Deep learning based algorithm (ConvLSTM) for copy move forgery detection,” *J. Intell. Fuzzy Syst.*, vol. 40, no. 3, pp. 4385–4405, Mar. 2021.

[4] A. Mohassin and K. Farida, “Digital image forgery detection approaches: A review,” in *Applications of Artificial Intelligence in Engineering*. Singapore: Springer, 2021.

[5] K. B. Meena and V. Tyagi, *Image Splicing Forgery Detection Techniques: A Review*. Cham, Switzerland: Springer, 2021.

[6] S. Gupta, N. Mohan, and P. Kaushal, “Passive image forensics using universal techniques: A review,” *Artif. Intell. Rev.*, vol. 55, no. 3, pp. 1629–1679, Jul. 2021.

[7] W. H. Khoh, Y. H. Pang, A. B. J. Teoh, and S. Y. Ooi, “In-air hand gesture signature using transfer learning and its forgery attack,” *Appl. Soft Comput.*, vol. 113, Dec. 2021, Art. no. 108033.

[8] Abhishek and N. Jindal, “Copy move and splicing forgery detection using deep convolution neural network, and semantic segmentation,” *Multimedia Tools Appl.*, vol. 80, no. 3, pp. 3571–3599, Jan. 2021.

[9] M. M. Qureshi and M. G. Qureshi, *Image Forgery Detection & Localization Using Regularized U-Net*. Singapore: Springer, 2021.

[10] Y. Rao, J. Ni, and H. Zhao, “Deep learning local descriptor for image splicing detection and localization,” *IEEE Access*, vol. 8, pp. 25611–25625, 2020.

[11] K. M. Hosny, A. M. Mortda, N. A. Lashin, and M. M. Fouda, “A new method to detect splicing image forgery using convolutional neural network,” *Appl. Sci.*, vol. 13, no. 3, p. 1272, Jan. 2023.

[12] F. Li, Z. Pei, W. Wei, J. Li, and C. Qin, “Image forgery detection using tamper-guided dual self-attention network with multiresolution hybrid feature,” *Secur. Commun. Netw.*, vol. 2022, pp. 1–13, Oct. 2022.

[13] C. Haipeng, C. Chang, S. Zenan, and L. Yingda, “Hybrid features and semantic reinforcement network for image,” *Multimedia Syst.*, vol. 28, no. 2, pp. 363–374, 2021.

[14] Q. Li, C. Wang, X. Zhou, and Z. Qin, “Image copy-move forgery detection and localization based on super-BPD segmentation and DCNN,” *Sci. Rep.*, vol. 12, no. 1, Sep. 2022, Art. no. 14987.

[15] A. K. Jaiswal and R. Srivastava, “Detection of copy-move forgery in digital image using multi-scale, multi-stage deep learning model,” *Neural Process. Lett.*, vol. 54, no. 1, pp. 75–100, Aug. 2021.