



SRINIVASA EDUCATIONAL SOCIETY'S

PACE INSTITUTE OF TECHNOLOGY & SCIENCES **(AUTONOMOUS)**

Approved by AICTE and Govt. of Andhra Pradesh, Accredited by NAAC(A Grade) | Recognized under 2(f) & 12(B) of UGC
Permanently Affiliated to JNTUK, Kakinada. A.P. | ISO 9001:2015, ISO 14001:2015 and ISO 50001:2018 Certified Institution
NH-16, Near Valluramma Temple, ONGOLE - 523 272, A.P., India, Ph.: 08592 278315, 9581456310 | www.pace.ac.in

IMAGE FORGERY DETECTION USING MACHINE LEARNING

A Mini Project Report is submitted to

PACE INSTITUTE OF TECHNOLOGY AND SCIENCES
(AUTONOMOUS)

In partial fulfillment of the requirements for the award of the degree of
Bachelor of Technology

In

INFORMATION TECHNOLOGY

By

20KQ1A1212
21KQ5A1204
20KQ1A1254
20KQ1A1218
20KQ1A1256

JAGARLAMUDI POOJITHA
PATCHABATI CHANDRASEKHAR
SHAIK YASAR ARAFATH
NEELAM DIVYA
SUNKARA UDAY KIRAN REDDY

Under the esteemed guidance of

Dr. A. Seshagiri Rao M. Tech, Ph .D

Department of Information Technology

2020-2024



SRINIVASA EDUCATIONAL SOCIETY'S

PACE INSTITUTE OF TECHNOLOGY & SCIENCES **(AUTONOMOUS)**

Approved by AICTE and Govt. of Andhra Pradesh, Accredited by NAAC(A Grade) | Recognized under 2(f) & 12(B) of UGC
Permanently Affiliated to JNTUK, Kakinada. A.P. | ISO 9001:2015, ISO 14001:2015 and ISO 50001:2018 Certified Institution
NH-16, Near Valluramma Temple, ONGOLE - 523 272, A.P., India, Ph.: 08592 278315, 9581456310 | www.pace.ac.in

Certificate

This is to certify that the project work entitled as **“Image Forgery Detection Using Machine Learning”** is being submitted by, **JAGARLAMUDI POOJITHA (20KQ1A1212)**, **PATCHABATI CHANDRASEKHAR (21KQ5A1203)**, **SHAIK YASAR ARAFATH (20KQ1A1254)**, **NEELAM DIVYA (20KQ1A1218)**, **SUNKARA UDAY KIRAN REDDY(20KQ1A1256)** in the partial fulfillment for the Award of the degree of Bachelor of Technology in Information Technology in the academic during 2023-2024.

PROJECT GUIDE

Dr. A. Seshagiri Rao M.Tech, Ph.D

Professor & HOD

Information Technology Department

HEAD OF THE DEPARTMENT

Dr. A. Seshagiri Rao M.Tech, Ph.D

Professor & HOD

Information Technology Department

Internal Examiner

External Examiner



SRINIVASA EDUCATIONAL SOCIETY'S

PACE INSTITUTE OF TECHNOLOGY & SCIENCES **(AUTONOMOUS)**

Approved by AICTE and Govt. of Andhra Pradesh, Accredited by NAAC(A Grade) | Recognized under 2(f) & 12(B) of UGC
Permanently Affiliated to JNTUK, Kakinada. A.P. | ISO 9001:2015, ISO 14001:2015 and ISO 50001:2018 Certified Institution
NH-16, Near Valluramma Temple, ONGOLE - 523 272, A.P., India, Ph.: 08592 278315, 9581456310 | www.pace.ac.in

DECLARATION

We JAGARLAMUDI POOJITHA(20KQ1A1212),PATCHABATI CHANDRASEKHAR (21KQ5A1204), SHAIK YASAR ARAFATH(20KQ1A1254),NEELAM DIVYA(20KQ1A1218) ,SUNKIREDDY UDAY KIRAN REDDY(20KQ1A1256) are hereby declare that the project report titled “**IMAGE FORGERY DETECTION USING MACHINE LEARNING**” under the guidance of **Dr. A. Seshagiri Rao, M.Tech, Ph.D, Professor of Information Technology Department** is submitted in the partial fulfillment for the Award of the degree of Bachelor of Technology in Information Technology in the academic during 2022-2023.

This is a record of bonafied work carried out by us and the results embodied in this project report have not been reproduced or copied from any source. The results embodied in this project report have not been submitted to any other University or Institute for the award of any other degree or diploma.

PLACE: VALLUR

DATE :

TEAM MEMBERS

20KQ1A1212	JAGARLAMUDI POOJITHA
21KQ5A1204	PATCHABATI CHANDRASEKHAR
20KQ1A1254	SHAIK YASAR ARAFATH
20KQ1A1218	NEELAM DIVYA
20KQ1A1256	SUNKARA UDAY KIRAN REDDY

ACKNOWLEDGEMENT

At the outset, we thank the Lord Almighty for the grace, strength and hope to make our endeavor a success.

We would like to place on record the deep sense of gratitude to **Dr. M. VENU GOPAL**_{B.E., MBA, D.M.M.}, Chairman of PACE Institute of Technology & Sciences for providing necessary facilities to carry the concluded project work.

We express our gratitude to **Dr. M. SRIDHAR** _{M. Tech, MBA.}, Secretary & Correspondent of PACE Institute of Technology & Sciences for providing us with adequate facilities, ways and means by which we were able to complete this project work.

Our sincere thanks to **Dr. G.V.K. MURTHY** _{M tech, Ph. D.}, the Principal of PACE Institute of Technology & Sciences to carry out a part of the work outside the campus and hence providing us an utmost congenial atmosphere.

We were highly indebted to **Dr. A. SESHAGIRI RAO** _{M. Tech, Ph. D} the Head of the Department, IT of PACE Institute of Technology & Sciences for providing us the necessary expertise whenever necessary.

We thank our Project Guide **Dr. A. SESHAGIRI RAO** _{M. Tech, Ph. D} For his outstanding support throughout the project for the successful completion of the work.

Last but not least, we thank the **Project Coordinator, Teaching and Non-teaching staff** of the department and especially our team members and parents who in one way or another helped us in the successful completion of this work.

BATCH MEMBERS

20KQ1A1212	JAGARLAMUDI POOJITHA
21KQ5A1204	PATCHABATICHANDRASEKHAR
20KQ1A1254	SHAIK YASAR ARAFATH
20KQ1A1218	NEELAM DIVYA
20KQ1A1256	SUNKARA UDAY KIRAN REDDY

Image Forgery Detection with Machine Learning

TABLE OF CONTENTS

CHAPTER NO	TITLE	PAGE NO
1.	INTRODUCTION 1.1 About Image Forgery Detection 1.2 Related Work	9-12 9-10 11-12
2.	THEORITICAL BACKGROUND 2.1 Convolutional Neural Network 2.2 Support Vector Machines 2.3 Random Forests 2.4 Image Pre-Processing	13-22 13-15 15-16 17-18 18-22
3.	Error Level Analysis Algorithm 3.1 Overview 3.2 The Algorithm 3.3 Algorithm Output 3.4 Further Discussion On The Error Level Analysis Algorithm 3.5 Quality Dependence	23-27 23 23-24 25-26 26 26-27
4.	Experiments With Error Level Analysis And Results 4.1 The CASIA Dataset 4.2 System specifications 4.3 Convolutional Neural Network With ELA 4.4 Support Vector Machine With ELA 4.5 Random Forests with ELA	28-32 28 28-29 29-31 31 31-32
5.	VGG16 Experiment And Results 5.1 Image Pre -Processing Approach 5.2 VGG16 Pre-Trained Model	33-38 33-34 34-38
6.	Future Work And Conclusions 6.1 Future Work: Ground Truth Masks 6.2 Conclusions	39-40 39 40

LIST OF FIGURES

FIG NO.	NAME	PAGE NO.	
1.	Error Level Analysis Model Architecture	14	
2.	Model Architecture	18	
3.	VGG16 Pre -trained Model Architecture	20	
4.	Error Level Analysis Algorithm: Python Implementation	23	
5.	Pristine Image with corresponding error level analysis	24	
6.	Manipulated Image With Corresponding Error Level Analysis	24	
7.	The Same Image Saved With Different Qualities	26	
8.	Testing With Various Number Of Epoch Sand Batches	29	
9.	Support Vector Machines	30	
10.	Random Forests Results	31	
11.	Command to install CG vs Photo Package	32	
12.	Patch accuracy	33	
13.	Patch Validation Accuracy	34	
14.	Validation Accuracy is 94.5%	34	
15.	Whole Image Training And Validation Accuracy	35	
16.	Whole Image Training And Validations	35	
17.	Generating Random spliced Images	36	
18.	Detecting Spliced Area	37	
19.	Benchmark1: personal image with spliced region	37	
20.	Benchmark1: personal image with low Brightness	37	

ABSTRACT

IMAGEFORGERYDETECTIONWITHMACHINELEARNING

The issue of forged images is currently a global issue that spreads mainly via social networks. Image forgery has weakened Internet users confidence in digital images. In recent years, extensive research has been devoted to the development of new technique to combat various image forgery attacks. Detecting fake images prevents counterfeit photos from being used to deceive or cause harm to others. In this thesis, we propose methods using the error level analysis algorithm to detect manipulated images. We show that our combination of image pre-processing and machine learning techniques is an efficient approach to detecting image forgery attacks.

CHAPTER -I

INTRODUCTION

With the advent of digital cameras and other smart devices, it has become easy for anyone to manipulate an image. Some manipulations are not harmful, such as changing the brightness of an image or converting it to black and white. On the other hand, some manipulations can cause harm to others and defame them, especially politicians and celebrities.

Image forgery is the process of manipulating a digital image to hide valuable or essential content or to force the viewer to believe an idea [1]. It has been defined as the process of manipulating an original digital image to either conceal its original identity or create an entirely different image than what was originally intended by the user of the digital platform [2]. Forged images can cause disappointment and emotional distress and affect public sentiment and behavior [3]. Images can transmit much more information than text. People tend to believe what they can see, and this affects their judgment, which leads to a series of unwanted responses. Because fabrications have become wide spread, the urgency to detect forgeries has significantly increased. The copy move approach is one of the most widely used forgery techniques. It copies apart of the image and paste it onto another part of the image. The technique itself is not harmful, but it can lead to critical situations if someone uses it with malicious intent.

Image forgery is done mainly for malicious reasons. It serves to distort information, spread immorality and fake news, obtain money fraudulently from an unsuspecting audience, ruin the reputation of a popular celebrity or any other public figure, and Spread adverse political influence among the users of a digital platform. Therefore, clear authentication of images and videos uploaded to digital media platforms, before they are

Used in anyway, makes it more difficult for digital information users to share information[4]. Image forgery is often used by malicious people to ruin the reputation of public figures. Image forgery, especially through Photoshop, can be used to display unethical behavior in public figures. It is also sometimes an attempt to influence consumers of the goods produced or the services offered by these public figures to shift to other markets[5]. This forgery could also be used for political reasons against opponent politicians or their agents, spreading images that portray their immoral side. This aims to convey a message to the public regarding the lack of integrity of the subject. Image forgery often leads to emotional problems for those whose images are released to public websites in disregard for their privacy. There have been reports of suicide due to the leaking of private images to public digital platforms, after which the victims undergo significant rebuke. These deaths negatively affect society.

Image forgery is also sometimes used to cheat victims of their money in increasingly common fraud schemes. The forged images are uploaded with embedded text, purportedly from the owner of the original image, with instructions that end in innocent people losing money. This is also done with images portraying people who are in dire need of help, with intentions of fraudulently acquiring money from unsuspecting members of the public. Society then ceases helping even those who are in genuine need of help because of the fear of being swindled.

For all of these reasons, it is vital to develop methods of detecting whether an image is forged and to locate the region of manipulation.

Related Work

Scientific studies on image forgery have provided various approaches to detecting whether an image is manipulated. In [6], Bun et al. Discuss how re sampling is a vital signature of manipulated images. They proposed a method of detecting and locating the area of manipulation on an image using resampling detectors with deep learning classifiers. In [7], Abdalla et al. Offered an approach to classifying whether an image is forged that involved transfer learning. Bappy et al. [8] applied a long-short term memory network with encoder decoder architecture to detect and localize the area of manipulation. Wan et al. [9] employed a feature pyramid network based on Res Net with Mask R – CNN to identify and locate manipulated regions.

In [10], Rahmouni et al. classified an image by dividing it into 100 x 100 patches and passing these patches into a patch classifier (image pre-processing). They Subsequently trained the resulting complete images with a convolutional neural network (CNN). In [11], Kaur and Manro pre – processed the images first. They then altered the images to grayscale space and performed Gaussian pyramid decomposition from time to time. Afterward, they detected image forgery using the block-based approach. Amit Doegar [12] proposed a method involving Alex Nets with support vector machines (SVMs) to classify whether an image was forged without specifying the exact area of manipulation. Zhou et al. [13] employed a pre trained model, VGG16, with a steganalysis rich model and CNNs.

In [14], Gupta et al. examined several block-matching algorithms, such as exact match and robust match, and compared their performances. In [15], Shiva kumar and Baboo proposed an approach using the speeded-up robust features algorithm in parallel with the K-dimensional tree algorithm to identify the manipulated region. Sallow et al. [16] proposed using fully convolutional networks. In the beginning, they tried using single-task fully convolutional networks. However, they noticed that multi-task fully

convolutional networks obtained better results than single-task fully convolutional and background areas in the manipulation-detection stage before the image-redemption stage to improve the accuracy of the redemption. They suggested a hybrid approach that could easily retrieve images using Zernike moment features and features found by a scale-invariant feature transform.

CHAPTER II

THEORETICAL BACKGROUND

Convolutional Neural Networks

Convolutional Neural Networks (CNNs or Conv Nets) are a type of neural network that are used effectively in image recognition classification and applications. Specifically, these neural networks are effective in facial, traffic-sign, and object identification. They also help power vision in robots and remote-controlled cars (self-driving) [18]. They evolved from the Le Net architecture, which was the initial CNN that was useful in the development of deep learning [19]. There are four operations that form the foundation of every CNN:

1. An image composed of a matrix of pixel values

From computer graphics concepts, every image can be represented as pixel value matrices. Certain components of an image are referred to as channels. In digital camera images, three channels are present: red, green, and blue [20]. These three channels are stacked in layers in the form of 2-dimensional matrices, and each channel has a pixel value that is in the range of 0 to 255. Grayscale images are distinguished by the presence of only one channel.

2. The convolution step

The convolution step entails extracting features from the input image. This operation maintains the patterns between the pixels with the help of small squares of input data that learn the features of the image. The operation involves sliding one channel matrix, such as orange, by one pixel over the original image, which could be green. This sliding step is referred to as a stride. Element-wise multiplication is Performed for every position. Finally, these products are added to generate the final integer that

represents a single element of the desired output matrix, such as pink. The operations in this block involve a filter and a convolved feature, which interaction the input image to detect features from it.

3. Non-linearity(rectified linear unity [Re LU])

After every convolution operation, the Re Lu involves a non-linear operation. The Re Lu operates on every pixel and replaces all negative pixel values in the feature map with zero. The Re Lu aims to introduce non-linearity to the CNN, as almost every datum learned in the CNN has a linear property.

4. The pooling step

The pooling step retains the most significant information of the feature map while reducing the dimensionality of every feature through spatial pooling [21]. The pooling step involves pooling of different types such as average, sum, and max. For example, if the operation involves max pooling, then the spatial neighborhood must be defined, and subsequently, the largest element from the rectified feature map within the window is selected[21].The operation uses the average instead of the largest element for average pooling and the sum for sum pooling. Therefore pooling, convolution, and Re LU are the foundation blocks for the effective implementation of CNN.

Figure 1 shows our model architecture of the ELA algorithm as a pre-processing step

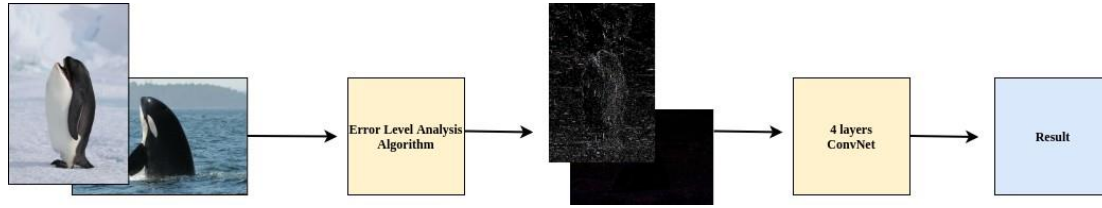


FIGURE1: Error Level Analysis model architecture

Support Vector Machines

The Support Vector Machine (SVM) [22] is a model used in classification and regression. It can solve linear and non-linear problems and performs well on various practical challenges. The SVM algorithm generates a hyperplane that divides the data into categories [23]. It is best applied in regression and classification problems, and it produces the highest accuracy while using less computational power [24]. This algorithm can be applied in classification, where the hyper plane in an N-dimensional space classifies distinct data points. The SVM is classified as a supervised machine learning model. It categorizes sets of training data into one or two other categories, and then a training algorithm model is built to assign the categories to the irrespective groups. The SVM is a non-probabilistic binary linear classifier that employs method such as Platt's scaling.

When working with textual analysis classification tasks, the SVM process involves refining training data while employing other forms such as Naïve Bayes algorithms.

A confidence score is generated for each recognized text or digit. When confidence is achieved in the dataset, the SVM continues the classification by applying a classification algorithm that is suitable when in situations with limited data. The algorithm involves separating two classes of data points with various choices of hyperplane. The SVM focuses on finding the plane with the maximum margin that represents the distance

between two data points in both classes. Classifying future data points becomes effective with reinforcement from the maximization of the margin distance.

In SVMs, hyperplanes represent decision boundaries that are essential for datapoint classification. The allocation of a data point to a certain class is based on the side of the hyperplane that the point falls on. There are various features that form the basis of the dimension of the hyperplane. Data points at the hyper plane determine the orientation of the hyper plane to define support vectors. The margin of the classifier is Maximized through the support vectors. Support vector machines use the kernel trick to perform linear classification while implicitly mapping inputs into feature spaces of high dimension. The main goal of the SVM is to help classify data in most of the statistical problems presented to machine learning experts. Understanding the correct position of data points on the hyper plane makes it possible to apply the SVM effectively.

Support vector machines provide solutions to real problems in a wide range of applications. A main application is text and hypertext categorization, which reduces the requirement for labeled training in transductive and inductive settings. The classification of images is another major area employing SVMs. The SVM is believed to achieve the highest search accuracy compared to traditional query refinement techniques[25].

Random Forests

A random Forest(RF)[26] is an ensemble algorithm, meaning that a decision is made using the results from various models. In most cases, the outcome from an ensemble model is better than that of any individual model [27]. Several decision trees are generated by RFs, and the decision is determined based on the outcome of all the decision trees[28]. An RF is a learning algorithm that randomly collects decision trees. Each decision tree consists of several decisions, and a combination of them forms the RF [29]. An RF integrates a collection of decision models from individual decision trees in the forest to improve the accuracy of the results. This prevents relying on a single learning model from a single decision tree in the RF. The merging of individual decision trees, each with its own set of algorithms constituting the RF, therefore creates a classification algorithm. This classification algorithm is independent from the algorithms of the individual decision trees. This forms a basis of prediction that is more accurate than that prediction that could have been made by a single decision tree or by a combination of independent decision trees.

Decision trees are the building blocks of an RF, and the individual trees are used to differentiate various events based on their most unique aspects[30].An RF consists of many trees that make the work more straightforward when complex sets of data are involved. They work on the principle that many uncorrelated decision trees, if made to operate as a group, will yield clearer and more accurate results than any individual decision tree. This is possible because the decision trees protect each other from the independent errors they make[31]. For an RF to work effectively, the decisions made By the individual decision trees should have little or no correlation with each other. There should also be unique signs in the differentiating features so that the models perform better than random guess.

Random forests utilize the idea of bagging, a process that allows decision trees to sample from the dataset while making replacements, which result in different trees [32]. This is possible because individual decision trees are highly sensitive to the data that they are trained on. Bagging therefore produces results where the individual trees not only train on different sets of data but can also use different characteristics to make informed decisions.

Image Pre-Processing

Image pre-processing is the process of improving image data by performing various operations on images and suppressing unwanted distortions in them. It can also be used to enhance certain unique features in an image that are crucial to further processing. Image processing may be a basic task, such as resizing[33]. For example, to feed an image data set into a deep learning model, all images must be of the same size. Other pre-processing tasks include geometric and color conversion, or the transition of color to gray scale; standardization; and data augmentation[34].

In this thesis, I used the pre-processing technique presented in [10]. The pre-processing step takes every image in the dataset and divides it into 100 x 100 patches. Subsequently, the patches are passed into a CNN classifier that classifies whether the given patch belongs to a raw image (green) or a computer graphics image (red). After the classification of patches, the complete images are generated again from the classified patches. The authors of [10] passed the resulting images into another CNN to classify whether an image is spliced. I used the VGG16 pre-trained model instead to accomplish better accuracy than a standard CNN. Figure 2 shows our model architecture. The pre-processing subfigure is from [10], and the VGG16 subfigure is from [35]. The data sets and experiments are discussed in Chapter 5

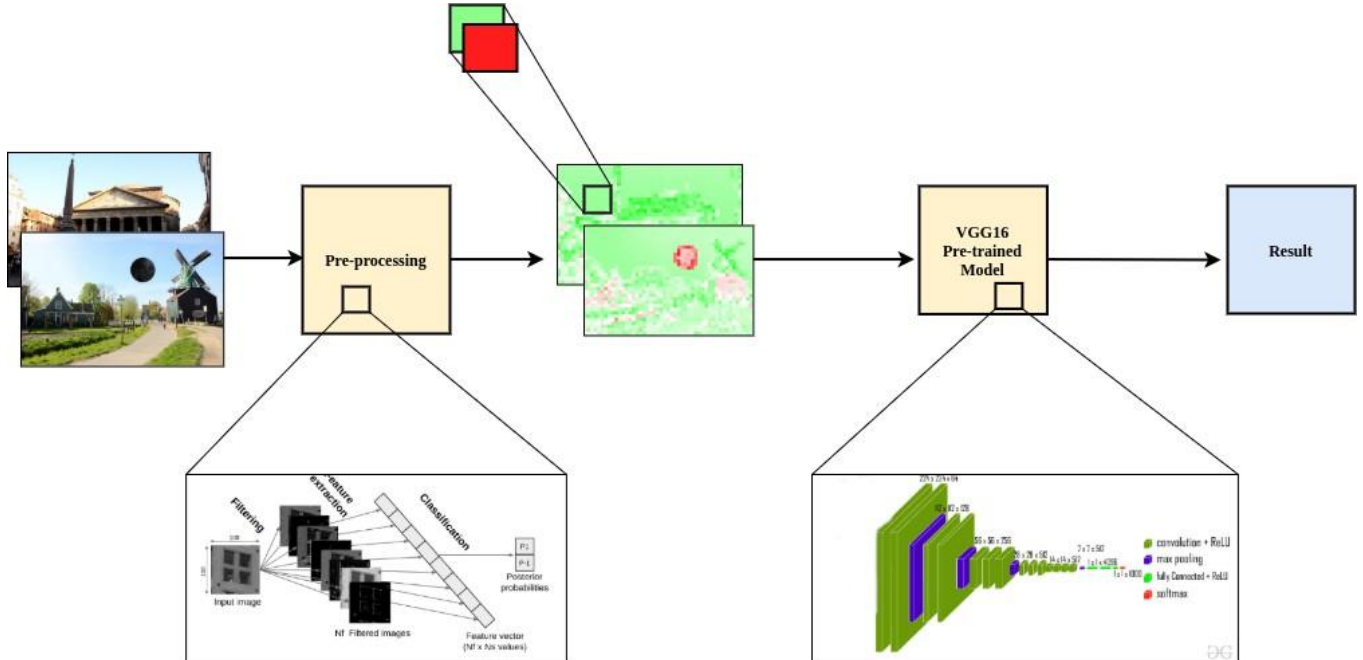


FIGURE2: Model architecture

Pre-Trained Image Net Models and Transfer Learning

A pre-trained ImageNet model is a model that has been trained on a significantly large dataset to solve a problem that is similar to the problem I want the model to solve[36]. I used a model pre-trained for a certain task on the Image Net dataset. The initial training of the model could have been done on a similar or very different domain, but the ability to solve problems remains useful [37]. Studies on modern computer vision have revealed that models that perform better on ImageNet usually perform better on other vision tasks as well. It is common practice to use imported models, such as Mobile Netsor VGG, due to the relatively high costs involved in training these models from scratch. The task of importing, usually referred to as transfer learning, is not only effective but also cost friendly to any profit-making institution. Transfer learning is also common because pre-training a model requires a relatively large data set for the model to extract

the characteristics required for the given task [19]. For instance, Image Net contains over one million images in 1000 categories [38]. A lack of data makes it difficult to train a model from scratch and makes it necessary to import a pre-trained model. Another reason for importing a pre-trained model instead of training one from scratch is the time required to train a model from scratch, depending on the experience of the trainer. This is because one must do many calculations and experiments before discovering a CNN architecture. Pre-trained models are also commonly used because training a model from scratch requires specific computational resources that might not be available. This also makes it necessary to import a pre-trained model. A pre-trained model that is imported is usually more efficient than a model that can be trained from scratch [39]. Pre-trained models are more accurate in most cases because they have been trained on a large number of classes, such as the 1000 classes in ImageNet. Employing a pre-trained model enhances their suitability to work on a wider range of issues compared to a model that is trained from scratch. Importing a pre-trained model is also advantageous because the most complex work of optimizing the parameters has usually been completed; what remains is only fine-tuning the model, a process that involves adjusting the hyperparameters to improve the pre-trained model. Another advantage of the pre-trained model is that it uses fewer steps before the convergence of the output [39]. This is because for a classification task, the features to be extracted will be similar, and it thus requires less time. Prior to choosing to import a pre-trained model, thorough research must be conducted on the problem in question, and the keywords should be determined based on the type of data set to be used. This is because, depending on the complexity of the dataset, some models usually work better than others. VGG16 is a CNN model that is used in large-scale image recognition. It provides high accuracy testing using Image Net, which consists of 100 classes of million images.

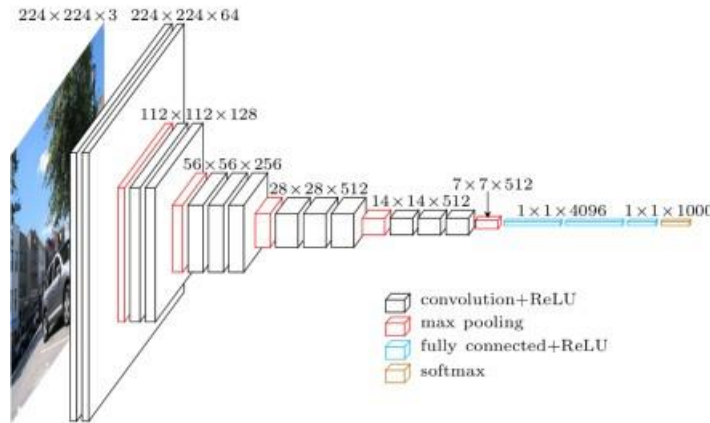


FIGURE3:VGG16 pre-trained model architecture

The input to the first convolutional layer of the VGG16 model is an RGB image of fixed-size, 224×224 . The image is passed through a stack of convolutional layers that represent the use of a receptive field of size 3×3 , the smallest size that can represent bottom, up, left, right, and center. The VGG16 also make suse of 1×1 convolution filters in several of its configurations. This configuration is the transformation of linear input channels, which is followed by non-linearity. In this model, the convolutional stride has a fixed value of 1 pixel. Therefore, any spatial padding of the convolutional layer input must be preserved during convolution. A stack of convolutional layers forms the foundation for fully connected layers. However, the stack is different from the fully connected layers in architecture and depth. Each of the top two stacks comprises 4096channels, and the third layer can perform a 1000-way ImageNet large-scale visual recognition challenge classification, giving it 1000 channels. The soft max layer forms the final layer that contributes to the configuration of the fully connected layers in the networks[41].In the presence of hidden layers in pre-trained models, the Re LU non- linearity is the basic

option that is applied. One of the challenges facing this model is the need for increased memory due to a high consumption of space. The VGG16 is a responsible model that assists machine learning experts in applying pre-trained networks to improve the level of learning[42].

CHAPTER III

ERROR LEVEL ANALYSIS ALGORITHM

Error Level Analysis (ELA) is a tool for exposing fabricated regions in JPEG images[41]. It can help recognize manipulations of compressed (JPEG) images By detecting the noise distribution present after resaving the image at a particular compression rate.

Overview

Neal Krawetz invented the concept of Error Level Analysis for images when he noticed how errors spread when a JPEG image is saved [43][44]. When cutting out a section of an image and pasting it into another image, the ELA for the pasted section often detects a more significant error, which means it is brighter than the rest of the image. There are several implementations of this algorithm, but they all follow the same steps.

The Algorithm

1. Compress the input image with a given compression rate and save it as a new image.
2. Calculate the pixel-by-pixel difference between the original image and the new image.
3. Store the difference in variable *elaImg*.
4. Compute the minimum and maximum pixel values for each band in the image and store the min variable *extrema*.

5. Compute the maximum pixel in *extrema* and store it in variable *max*.
6. Calculate the new scale by dividing 255 by the *max*.
7. Enhance the brightness of *elaImg* based on the resulting scale, then save and return the resulting ELA image.

Figure4 illustrates our Python implementation of this algorithm.

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

def ToEla(path, quality):
    filename = path
    resavedImageName = filename.split('.')[0] + '.resaved.jpg'
    ElaImageName = filename.split('.')[0] + '.ela.png'

    img = Image.open(filename).convert('RGB')
    img.save(resavedImageName, 'JPEG', quality=quality)
    imgResaved = Image.open(resavedImageName)

    elaImg = ImageChops.difference(img, imgResaved)

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

    elaImg = ImageEnhance.Brightness(elaImg).enhance(scale)
    os.remove(resavedImageName)

    return elaImg
```

FIGURE4: Error level analysis algorithm: Python implementation

Algorithm Output

Figure5 and Figure6 show pristine and manipulated images with their corresponding error level analysis.

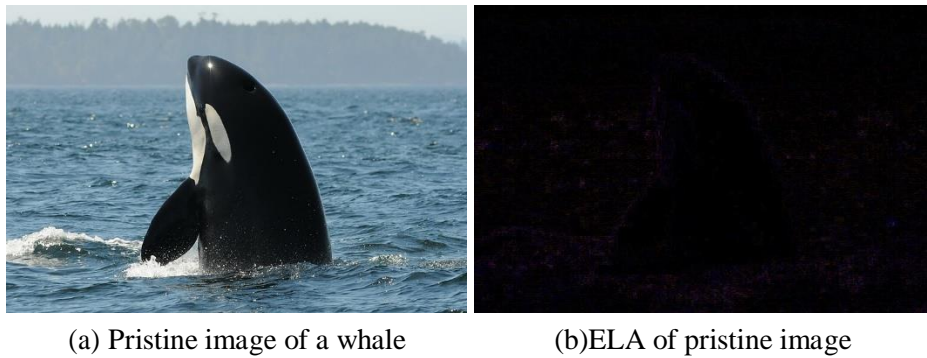


FIGURE5:Pristine image with corresponding error level analysis

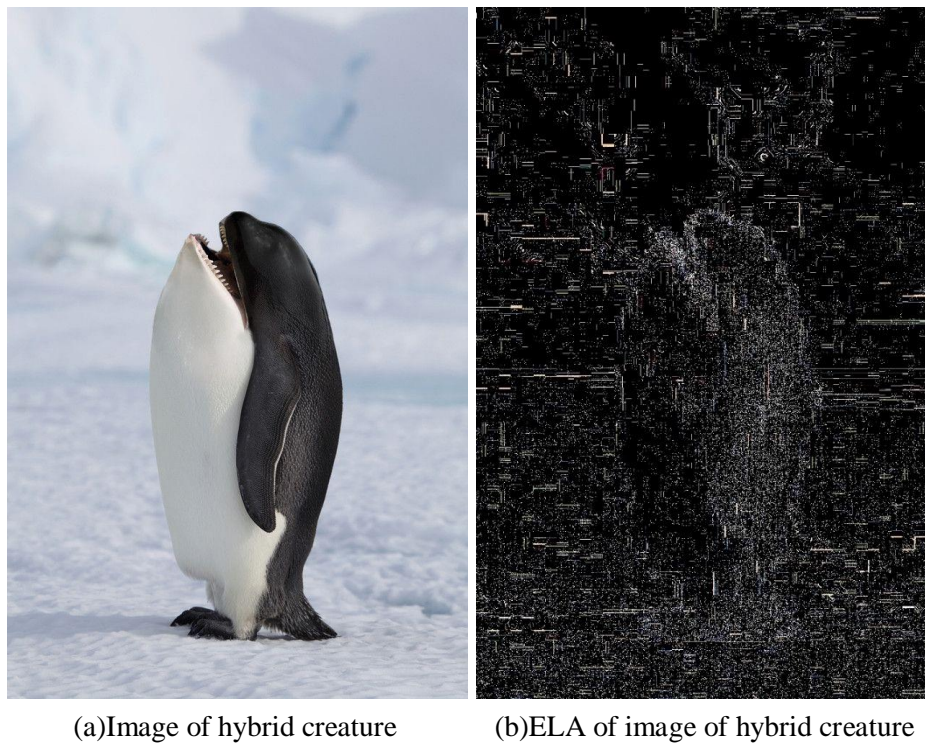


FIGURE 6: Manipulated image with corresponding error level analysis

I loaded Figure5a into the ELA algorithm, and the output is given in Figure5b. Figure5b shows an early entirely black box, indicating that there is no noise in the image, which means that the image is real and not manipulated.

I calculated the error level of Figure6a, as shown in Figure6b. The noise in the image is clear and indicates that the image of the hybrid animal is manipulated.

Further Discussion on the Error Level Analysis Algorithm

A variable quality level is used control the amount of compression of a JPEG image. The amount of JPEG compression is typically measured as a percentage of the quality level. An image at 100% quality has (almost) no loss, and a 1%-quality image is a very low-quality image. In general, quality levels of 90% or higher are considered high quality, 80-90% is medium quality, and 70-80% is low quality. Anything below 70% is typically very low quality [45]. Low-quality images can reduce the ability of analysis algorithms to detect modifications. The ELA algorithm works by resaving an image at a known quality level, such as 75%, and during Step3 in the algorithm, it then identifies the amount of error introduced [45].

Quality Dependence

Saving an image with different quality levels affects the ELA algorithm, and this leads to a distinct number of bright pixels. The lower the quality, the higher the number of bright pixels. An image can be modified and saved in a quality lower than the quality used in the ELA algorithm, and this makes it difficult to detect whether the image is manipulated. Figure 7 shows an example of a pristine image that is saved with different qualities. Subfigures (a) and (d) show the outcome of saving an image with 90% quality. Sub figures(b) and (e) show the outcome of saving an image with 70% quality. Sub figures

(c) and (f) show the outcome of saving an image with 48% quality. From Figure 7, I have concluded that the lower the quality, the greater the noise. This means that the ELA algorithm is not always accurate.

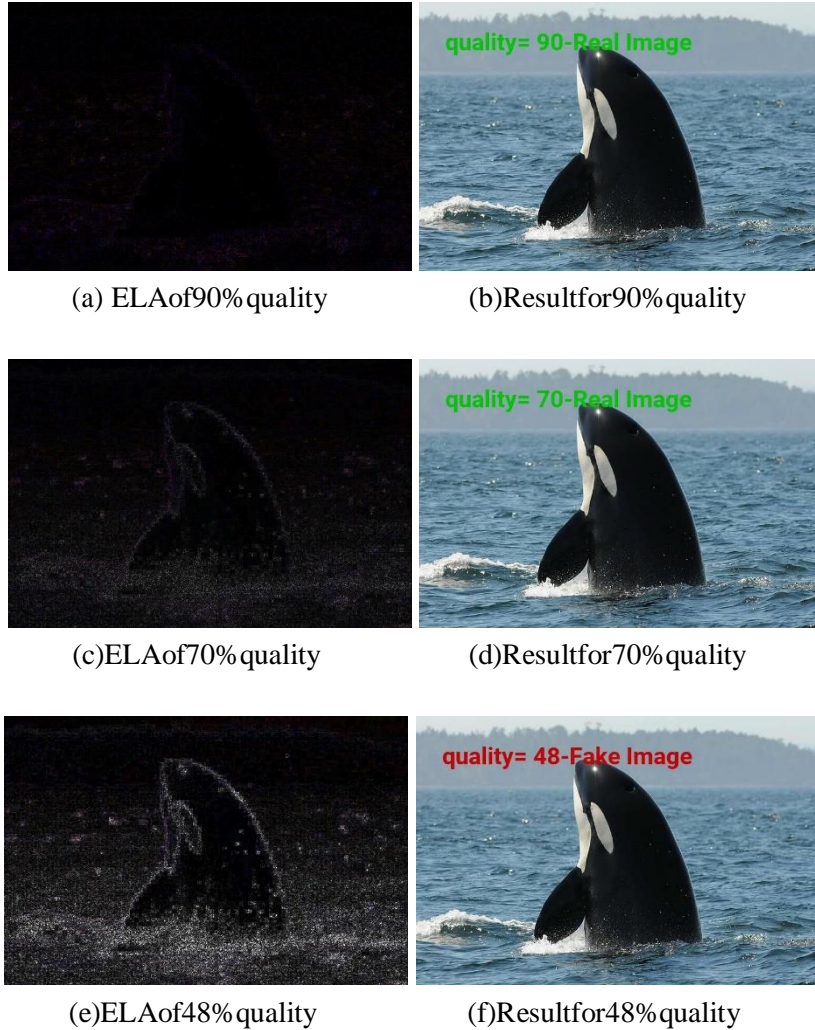


FIGURE 7: The same image saved with different qualities

CHAPTER IV

EXPERIMENTS WITH ERROR LEVEL ANALYSIS AND RESULTS

The CASIA Dataset

The CASIA dataset[46] contains three categories that make it an appropriate dataset for this research:

1. Pristine images: Unspoiled images in their original form. Shown in Appendix A.
2. Copy move images: The manipulated region has been copied from the same image and pasted on another area of the same image. Shown in Appendix A.
3. Spliced images: The manipulated region has been copied from a different image and pasted on this image. Shown in Appendix A.

System Specifications

The specifications of the computer I used to conduct these experiment sare the following:

- **Operating system:** Ubuntu18.04.4LTS.
- **CPU:** Intel®Core™i7.
- **RAM:**16GB.
- **GPU:** NVIDIAGeForce2060.
- **Driver:** NVIDIADriver430.
- **Externalharddrive:**6TBexternalharddrive.

I tested the ELA algorithm on the CASIA dataset with three different classifiers: CNNs[47], SVMs, and RFs. The following sections contain explanations of and results for each classifier with the ELA algorithm.

Convolutional Neural Networks with ELA

The image paths were passed to a function that converts the images into their ELA form, as discussed in Section 3.2. I then split the dataset into training and testing sets using the `train_test_split` method from `sklearn`. I subsequently created a CNN with three convolutional layers because it achieved better results than two layers. With four layers, the model became overfitted. The results are discussed in Section 4.3. The primary packages used include `pandas` to read images paths, `matplotlib.pyplot` to plot performance curves, `sklearn` and `keras` to create the neural network.

The `Image`, `ImageChops` and `ImageEnhance` modules were used in the ELA algorithm. Figure8 shows validation and training accuracy for various numbers of epoch sand batches.

Results

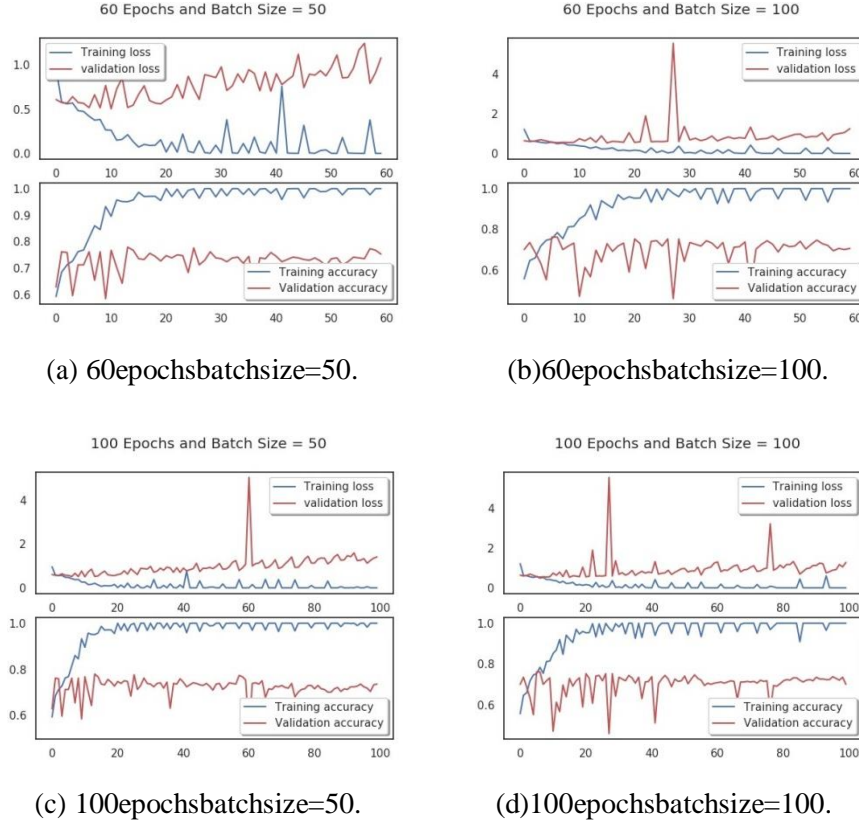


FIGURE8:Testing with various numbers of epoch sand batches

With the CNN, the model achieved a 79% accuracy. It had reached 80% when I initially ran it on PopOs 19.04. However, I were required to downgrade the system to Ubuntu18.04Bionic Beaver, since some Python packages and libraries are not yet supported by PopOs. Figure8 shows various performance measure curves: (a) shows the accuracy of training with 60 epochs and a batch size of 50, (b) shows the accuracy of training with 60 epochs and a batch size of 100, (c) shows the accuracy of training with 100 epochs and a batch size of 50, and (d) shows the accuracy of training with 100 epochs and a batch size of 100. Of the training dataset, 80% was from the CASIA dataset. The remaining 20% was for validation. Figure8 shows that training accuracy was high. With

the validation data used on the model to evaluate its performance, the accuracy dropped to 72%.

Support Vector Machine with ELA

I converted all the images to their ELA format. I then passed the resulting two folders (pristine image folder and manipulated image folder) to the SVM [48] with `rbf` kernel. The results are shown in Section 4.4. The primary packages used includes `kimage` to open images and `sklearn` and `keras` to use the SVM classifier.

Results

The SVM achieved 72% accuracy, as shown in Figure 9:

	precision	recall	f1-score	support
0	0.70	0.81	0.75	184
1	0.73	0.61	0.66	160
accuracy			0.72	344
macroavg	0.72	0.71	0.71	344
weightedavg	0.72	0.72	0.71	344

FIGURE 9: Support vector machine results

Random Forests with ELA

For feature extraction, I chose it based on the brightness of the pixels. I began counting bright pixels on the ELA form. I estimated 150 pixels after consulting the RGB table shown in Appendix C. According to the table, three zeros represent the color black. The combination (255,255,255) represents white, the brightest shade. The second row of the RGB table shows the combination (127,127,127) that represents gray. In my

opinion, the shade of gray provided by this combination was too dark. I there for echos
 ethe combination $(150,150,150)$ as a boundary. Pixels brighter than $(150,150,150)$
 were considered to be noise in the ELA images. The brighter the pixels in an ELA
 image, the greater the noise. Figure 6b shows the ELA for a fake image. It contains
 634 bright pixels. I created the condition that if there are more than 300 bright pixels
 on an ELA image, then the image is fake. I estimated 300 because some pristine
 images may have some noise, as discussed in Section3.4. The results are shown in
 Section4.5. The primary packages used include `pandas` to read images paths and
`sklearn` to create the RF.

Results

The RFs in this study achieved 91% accuracy, as shown in Figure10:

	precision	recall	f1-score	support
0	0.86	0.96	0.90	94
1	0.93	0.79	0.85	71
accuracy			0.91	165
macroavg	0.90	0.87	0.91	165
weightedavg	0.89	0.91	0.91	165

FIGURE10:Random forest results

CHAPTER V

VGG16 EXPERIMENTS AND RESULTS

RAISE Data set and Reference Database

RAISE is a demanding real world image dataset, developed primarily to test digital forgery detection algorithms. It consists of approximately 8000 high-resolution RAW images, which are uncompressed and never processed [49]. An image from this dataset is shown in Appendix B. Reference Database is a free set of tagged screen shots taken from games. It is a computer graphics database [50]. An image from this dataset is shown in Appendix B. I conducted our experiments on 1800 randomly chosen images to compare our results to those from [10]. The authors of [10] used CNNs to detect forgery in an image. I used the same approach for image pre-processing but with a pre-trained Image Net model, VGG16, instead of a standard CNN.

Image Pre- Processing Approach

The authors of [10] made it possible to use their pre-processing approach by providing the `CGvsPhoto` Python package.

```
pip3installCGvsPhoto
```

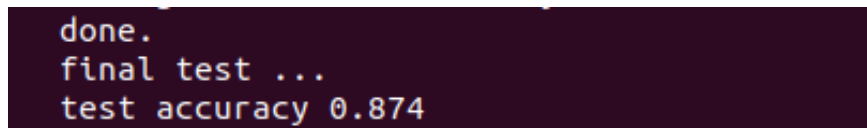
FIGURE 11: Command to install CGvs Photo package

As mentioned in Chapter II, the pre-processing function takes every image in the dataset and divides it into 100 x 100 patches. The patches are then passed into a CNN classifier that classifies whether the given patch belongs to a raw image (green) or a

Computer graphics image(red).After classifying 100 patches,the pre-processing function displays the accuracy of the classification. The classifier saves weights in a `.ckpt` file after classifying 500 patches. After the classification of patches, the complete images are regenerated from the classified patches. The authors of [10] passed the resulting images into another CNN to obtain a final result, and they achieved an accuracy of 93.4%. I used the vGG16 pre-trained model to accomplish higher accuracy than using a standard CNN. For the pre-processing step, the authors of[10] achieved an accuracy of 84.4%.

VGG16Pre-Trained Model

The authors of[10] achieved 84.4% accuracy on the pre-processing step. I obtained 87% on patch level. This higher accuracy was achieved because our weights were optimized. Figure12shows the obtained patch accuracy.

A terminal window with a dark background and light-colored text. The text shows the completion of a test process and the resulting accuracy.

```
done.  
final test ...  
test accuracy 0.874
```

FIGURE12:Patch accuracy

Figure13shows the curve of validation accuracy.

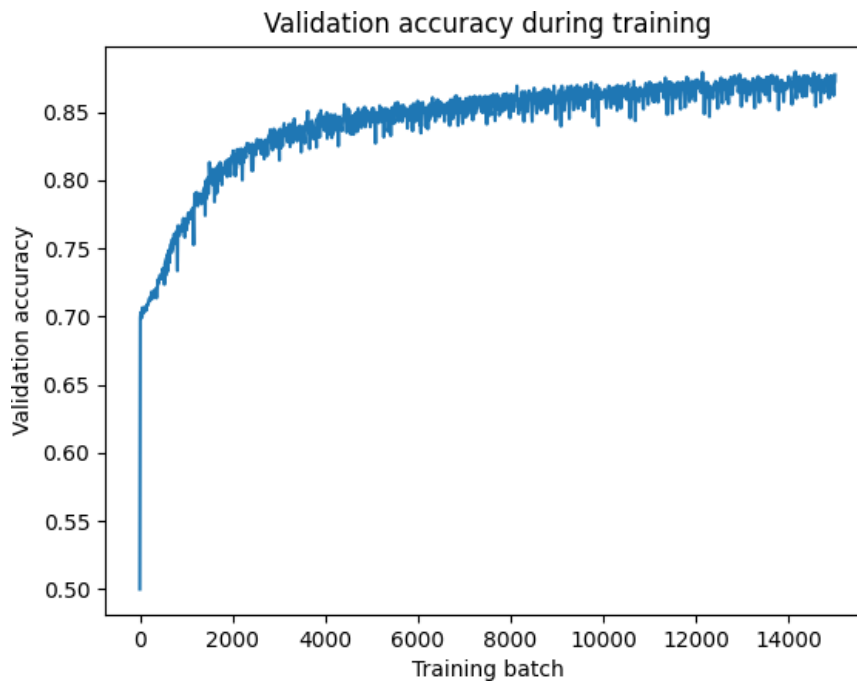


FIGURE13:Patchvalidationaccuracy

The authors of [10] passed the resulting complete images to another CNN and achieved 93.4% accuracy by training 1800 images. I obtained 94.5% accuracy by using a VGG16 pre-trained model. Figure14depicts our results.

```
Epoch 1995/2000
1000/1000 [=====] - 0s 323us/step - loss: 0.0093 - acc: 0.9970 - val_loss: 2.0044 - val_acc: 0.9579
Epoch 1996/2000
1000/1000 [=====] - 0s 322us/step - loss: 0.0118 - acc: 0.9940 - val_loss: 2.0064 - val_acc: 0.9579
Epoch 1997/2000
1000/1000 [=====] - 0s 324us/step - loss: 0.0140 - acc: 0.9950 - val_loss: 2.1425 - val_acc: 0.9450
Epoch 1998/2000
1000/1000 [=====] - 0s 327us/step - loss: 0.0164 - acc: 0.9920 - val_loss: 2.4732 - val_acc: 0.9450
Epoch 1999/2000
1000/1000 [=====] - 0s 327us/step - loss: 0.0139 - acc: 0.9930 - val_loss: 2.1794 - val_acc: 0.9450
Epoch 2000/2000
1000/1000 [=====] - 0s 323us/step - loss: 0.0118 - acc: 0.9960 - val_loss: 2.2077 - val_acc: 0.9450
```

FIGURE14:Validationaccuracyis94.5%

Figure15 and Figure16 show the curves of training and validation.

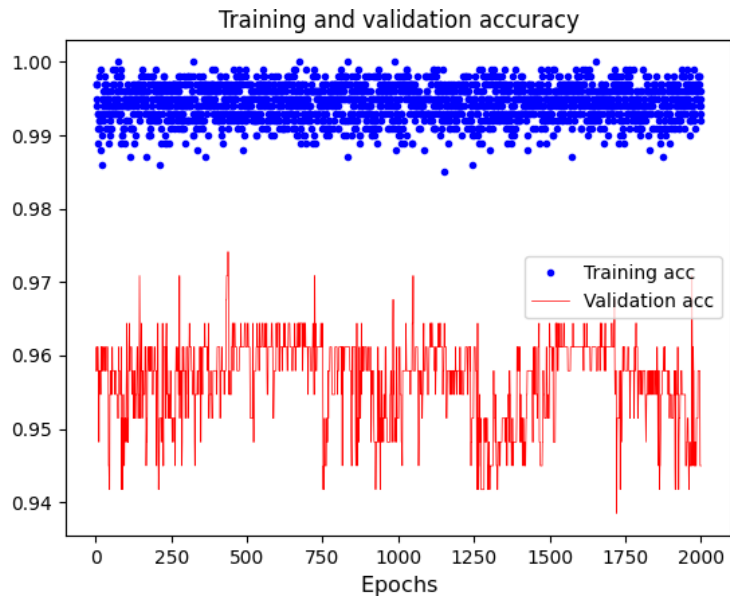


FIGURE15:Whole image training and validation accuracy

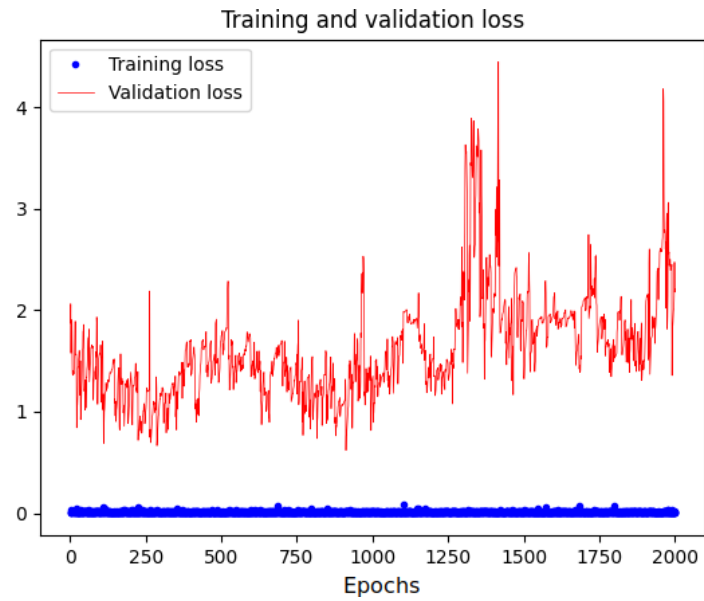


FIGURE16:Whole image training and validation loss

I wrote a program that generates randomly spliced images, illustrated in Figure17.

```
from PIL import Image, ImageDraw, ImageFilter, imageops, random

export = "/lubna/CWU/Thesis/generatingFakeImages/SpHuge/"
realPath = "/lubna/CWU/Thesis/generateFakeImages/realimSP/"
realfiles = os.listdir(realPath)
fakePath = "/lubna/CWU/Thesis/generatingFake/fakeimgSP/"
fakefiles = os.listdir(fakePath)

for i in range(1439):
    left = 155
    top = 65
    right = 600
    bottom = 600
    x = random.randint(0, 2300)
    y = random.randint(0, 2300)
    a = random.choice(realfiles)
    b = random.choice(fakefiles)
    im1 = Image.open(realPath+a)
    im2 = Image.open(fakePath+b)
    im3 = im2.crop((left, top, right, bottom))
    backim = im1.copy()
    backim.paste(im3, (x, y))
    backim.save(export+str(i)+'.jpg', quality=95)
```

FIGURE17: Generating random spliced images

By using the image pre-processing function found in the library, I could locate the spliced area in an image. I tested our model with a randomly generated spliced image, and it was successful, as shown in Figure18.



FIGURE18:Detecting spliced area

I also took a photograph from a personal camera and pasted an object on to it.

Figure19 shows the result of detecting a spliced area in this photograph.

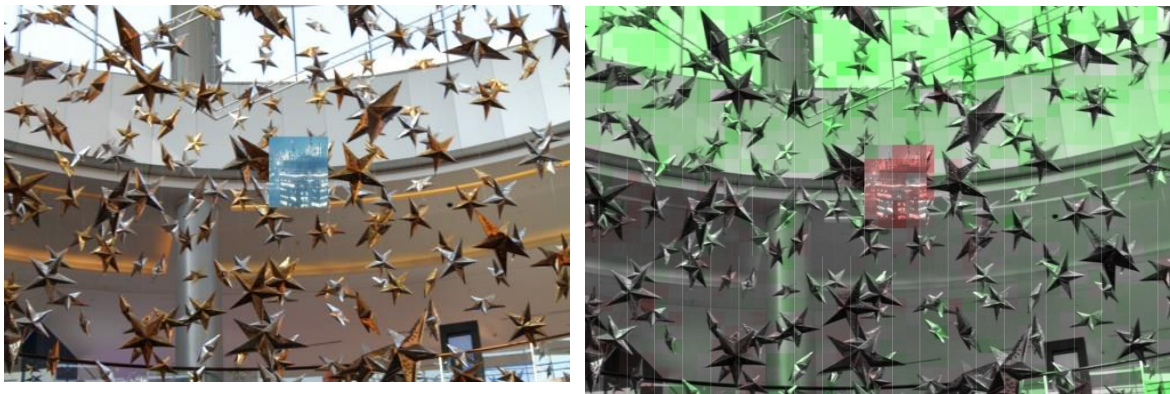


FIGURE19: Benchmark1: personal image with spliced region

I also attempted to input an image with lower brightness and a different object location, as shown in Figure20.

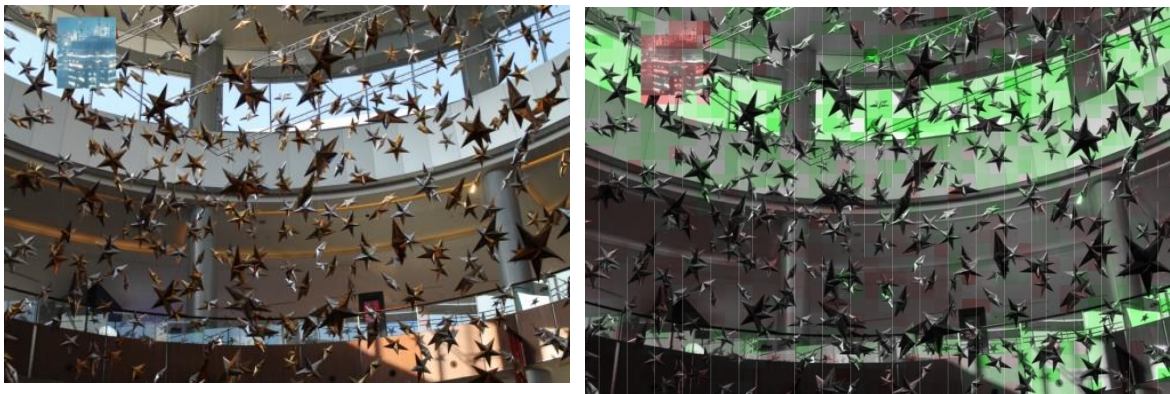


FIGURE20: Benchmark2: spliced personal image with low brightness

CHAPTER VI

FUTURE WORK AND CONCLUSIONS

Future Work: Ground Truth Masks

Generating the ground truth mask of a manipulated image is more reliable than the ELA algorithm, and it is not affected by the quality of the image. It also shows the exact area of manipulation. Obtaining the ground truth mask requires knowledge of image pre-processing and image segmentation techniques. Figure 21 shows a manipulated image with its ground truth mask.



(a) Manipulated image.



(b) Ground truth mask.

FIGURE21: Manipulated image with its ground truth mask

Conclusions

Image forgery involves distorting images, sometimes images of people, for malicious reasons. This involves a genuine image that had been displayed on a public website or a digital communication platform and is edited into an entirely different image. The new image will likely be immoral in nature or targeted to spread negative publicity.

The ELA algorithm shows whether an image is manipulated when the input images quality is close to the quality used in the algorithm. If there is a large difference between the quality of the image and the quality of the algorithm, then the result will always be incorrect. Further more, the algorithm does not show the exact area of manipulation.

A pre-trained model is a model that has been trained on a certain task on the Image Net dataset. It is a model that has been trained to solve issues that might be similar to the problem athand. A pre-trained model is preferred in most cases to training a model from scratch. The process of importing a pre-trained model is referred to as transfer learning.

Other approaches do not depend on the quality of the images and show the exact area of manipulation. The patch classification approach is not affected by the quality of the image and achieves more accurate results. Commonly imported models such as VGG and Mobile Nets have been trained on large sets of data and are therefore very efficient on any given dataset. The time required to train a model from scratch, depending on experience, is relatively high, which makes it necessary to consider using pre-trained models.

