IMAGE FORGERY DETECTION AND AUTHENTICATION

By

KARIM TAREK YASSIEN ELSAYED AZEB ELGAZAR

195322

A Dissertation submitted to the Faculty of Engineering, The British University in Egypt, in Partial Fulfillment of the requirements for the bachelor's degree in Computer Engineering

Supervisor(s): Dr. Motasem Mahmoud Elshourbagy

June, 2024

Electrical and Communication Engineering Department

Faculty of Engineering

IMAGE FORGERY DETECTION AND AUTHENITCATION

Dissertation Approved:

Dr.

Committee member

Committee member

External committee member

Affiliation:

Dean Faculty of Engineering

Acknowledgments

I would like to thank Dr. Motasem Mahmoud Elshourbagy for his great effort and Support that he gave to me.

Abstract

In the modern digital age, the rise of image manipulation tools raises significant concerns about the authenticity of digital images. This dissertation addresses the challenge of detecting forged images by utilising deep learning techniques. The main focus of this research is developing and evaluating a Convolutional Neural Network (CNN) based architecture that utilises Error Level Analysis (ELA) and Spatial Rich Model (SRM) filtering for image forgery detection.

The proposed dual-branch CNN model utilises images processed through ELA and SRM filters, enhancing the neural network's ability to identify subtle discrepancies indicative of manipulation in image data. Extensive experimentation was conducted on the most prominent datasets in image forgery detection field, including CASIA V2, CoMoFoD (small) and MICC-F2000.

Evaluation metrics such as accuracy, precision, recall and F1 score were used to benchmark the performance of the proposed model on the datasets mentioned. Notably, the model achieved near state-of-art-results with an accuracy of 98.00% on the MICC-F2000 dataset and the approach exhibits superior generalization capability across different types of image forgery and outperforming traditional methods. The findings highlight the potential of integrating ELA and SRM in deep learning frameworks for efficient image forgery detection.

This dissertation contributes to the field of digital forensics by providing a comprehensive analysis of existing image forgery techniques and proposing a novel model architecture that sets a new benchmark for future research.

Table of Contents

Acknowledgments ii

Abstract iii

List of figures viii

List of Graphs x

Chapter 1: Introduction 1

1.1. Background 1

1.1.1. Brief introduction 1

1.1.2. Image forgery detection approaches 1

1.1.3. Traditional image forgery detection methods 2

1.1.4. Deep learning-based methods 3

1.1.5. Hybrid methods 4

1.1.6. Image forgery types 5

1.1.7. Evaluation metrics. 8

1.1.8. Detailed tasks of semester 1 10

1.1.9. Anticipated problems and how to solve them 11

1.1.10. Plan of tasks during semester 2 13

1.1.11. Limitations and Comments on tasks planned for semester 2 14

1.1.12. Popular Datasets available 14

1.2. Problem Statement 21

1.3. Research Objectives 21

1.4. Research Questions 22

1.5. Research Methodology 22

Chapter 2: Related work 23

2.1. Literature Review 23

Chapter 3: Methodology 34
3.1. Choice of datasets 34
3.2. Convolutional Neural Networks (CNNs) 36
3.3. Error Level Analysis 37
3.4. Spatial Rich Model (SRM) Filters 38
3.5. Proposed Model Architecture 40
Chapter 4: Setup and Experimental Results 42
4.1. System Configuration and Environment Setup 42
4.2. Software Tools used 43
4.3. Dataset preprocessing 44
4.3.1. Error level analysis (ELA) 44
4.3.2. Spatial Rich Model (SRM) Filtering 44
4.3.3. Processing Real and Forged Images 45
4.3.4. Training and evaluation results 46
4.4. Evaluating the first algorithm 47
4.4.1. Evaluation results on CASIA V2 47
4.4.2. Evaluation results on MICC-F2000 48
4.4.3. Evaluation results on CoMoFod 49
4.4.4. Comparison of the single branch CNN with the state-of-the-art methods 50
4.5. Evaluating the second algorithm 51
4.5.1. Evaluation results on CASIA V2 52
4.5.2. Evaluation results on MICC-F2000 53
4.5.3. Evaluation results on CoMoFod (small) 54
4.5.4. Evaluation conclusion 55
4.6. Comparing the proposed dual branch CNN with the state-of-the-art Methods 56

Chapter 5: Conclusion 57
References 58
Appendix A: Code 62
Preprocessing (CASIA V2 example) 62
Model Training and evaluation (CASIA V2 example) 65
Executive Summary 70
ABSTRACT 70
1. Problem Definition 71
2. Objectives 71
3. Brief Introduction 71
3.1. Background 71
3.2 Deep Learning-Based Image forgery detection 72
4. Image forgery types 72
5. Summary of Previous Work 72
6. Research Methodology 73
6.1. Dataset choice 73
6.2. Error Level Analysis (ELA) 73
6.3. SRM (Spatial Rich Model) Filters 73
6.4. Convolutional Neural Networks (CNNs) 74
6.5. Proposed Model Architecture 74
7. Experimental Work and Main Results 75
8. Conclusion 76
9. Future work 76
Appendix A: Code 79
Preprocessing (CASIA V2 example) 79

Model Training and evaluation (CASIA V2 example).....83

List of figures

Figure 1: Image manipulated through copy-move forgery from MICC-F2000 dataset (P1000293tamp2.JPG).....5
Figure 2: A splicing example where zebras are foreign to the image 5
Figure 3: Example where multiple aspects of the image where manipulated through image retouching 6
Figure 4: Object removal where the woman was removed from the image.....7
Figure 5: Example of an image processed with ELA (original left ELA right) 37
Figure 6: The 30 SRM high pass filters.....38
Figure 7: An example of an image processed with SRM (original left SRM right).....38
Figure 8: Architecture of the proposed model.....40

List of tables

Table 1: A comparison of image forgery detection datasets 20
Table 2: The architecture of a single branch CNN 46
Table 3. Comparison of the single branch ELA architecture with State-of-the-art methods 50
Table 4: Evaluating the improvement from the dual branch implementation 55

List of Graphs

Graph 1: CASIA V2 validation data confusion matrix 47
Graph 2: CASIA V2 training and validation loss and accuracy history 47
Graph 3: MICC-F2000 validation data confusion matrix 48
Graph 4: MICC-F2000 training and validation loss and accuracy history 48
Graph 5: CoMoFod validation data confusion matrix 49
Graph 6: CoMoFod training and validation loss and accuracy 49
Graph 7: Dual branch CASIA V2 validation data confusion matrix 52
Graph 8: Dual branch CASIA V2 training and validation loss and accuracy history 52
Graph 9: MICC-F2000 validation data confusion matrix 53
Graph 10: MICC-F2000 training and validation loss and accuracy history 53
Graph 11: CoMoFoD validation data confusion matrix 54
Graph 12: CoMoFoD training and validation loss and accuracy history 54

Chapter 1: Introduction

1.1. Background

1.1.1. Brief introduction

Digital images have been deeply embedded into our lives as a common medium for exchanging information. Every day, an estimated few billion images are being uploaded to the internet on social media websites. The large userbase of various social media applications such as Facebook, Instagram, X (formerly known as twitter) and more recently TikTok are being constantly presented with tens of millions of images and videos every day.
Image manipulation software is widely available on the internet and accessible allowing anyone to produce manipulated images that are impossible to differentiate from authentic images by the naked eye. The ease and spread of image manipulation software has raised concerns about the authenticity of images that are exchanged. Images that have been forged with malicious intent can be presented or might be perceived to be authentic. Thus, leading to deception and the spread of misinformation.

These concerns around the authenticity of digital images have resulted in the emergence of the field of digital multimedia forensics. Within digital media forensics, digital Image forgery detection and authentication can be a challenging but a vital mission for the insurance of the authenticity of information exchange via digital images. The creation of reliable digital image forgery detection and authentication algorithms improves the quality of life and dependability for workers of jobs that rely on information embedded in digital images such as lawyers, judges, law enforcement and health professionals.

Reliable and efficient image forgery algorithms may improve the credibility of communication media such as journals and social media feeds. The lack of presentable reliability for images today in most social media indicates the lack of awareness of the tech companies and the importance of research and work to progress in this field.

.

1.1.2. Image forgery detection approaches

Forgery detection may be classified into two different approaches active and passive. The active image forgery detection approach is focused on more traditional techniques and usually performs worse than the more recent passive image forgery detection approach.

Active forgery detection approach has two commonly used techniques. Digital signature, which could be embedded into the image during its capture or at a later point and during the forgery detection process the image image's digital signature is checked for corruption. And digital watermarking which is usually invisible and if the image's contents undergo manipulation the image's watermark will be manipulated, which is then checked during the watermark extraction process to verify its authenticity and if the watermark does not match the original watermark, then the image is deemed to have undergone manipulation(Gill et al., 2017).

Nevertheless, Passive forgery doesn't rely on the information that is already embedded in the image or its metadata which is an advantage over the active approach due to the nature of the circulated digital images that are lacking the metadata due to the image going through a process of removing the metadata after being posted on social media. Thus, the information needed for active forgery detection is lost, to protect the uploader's privacy(Gill et al., 2017).

1.1.3. Traditional image forgery detection methods

Many traditional methods have been established to determine if a photograph is manipulated including statistical analysis to investigate the statistical properties of an image to check for abnormalities that might be indicative of manipulation, these can be inconsistencies in noise patterns such as camera noise patterns which can be a sign that a region has been manipulated if the camera noise patterns have any abnormalities.

Traditional image forgery detection methods rely on analysing inconsistencies in an image's physical and geometric properties to identify manipulation. These methods include pixel-based techniques, which analyse the authenticity of each pixel and its relationship with others. Camera-based techniques detect forgeries through metadata or patterns unique to the camera that captured the photo, such as sensor noise patterns. Format-based techniques analyse compression artifacts, which can differ in parts of images that have been saved after editing in forged images. Geometric-based methods rely on lighting, shadows, and the perspective of the image to identify inconsistencies indicative of forgery.
There may be abnormalities in the colour distribution of the image or JPEG compression artifacts from manipulated regions undergoing compression multiple times from saving images after they have been edited. Pixel based Techniques like Error Level Analysis

(ELA)(Krawetz, 2007) can highlight these inconsistencies and reveal the abnormal variation in compression levels.

There are also Transform-based techniques which involves applying mathematical transformations to images and then the information present in the transformed domain is analysed to identify any signs of image manipulation. Moreover, two techniques that are commonly used for image forgery detection are Discrete Wavelet Transform (DWT) (Lai & Tsai, 2010)and Discrete Cosine Transform (DCT) these techniques highlight inconsistencies in the data within the frequency or spatial domain (Ahmed et al., 1974).

Another common approach to detect if image data has been manipulated or changed is through metadata analysis, where the meta-data is examined such as the creation data or the timestamp of an image or the geographical location in the image was captured in and notably the specific camera settings which the photo was taken with. Analysis of these might highlight a discrepancy in the metadata that is suggestive of manipulation (Farid, 2009).

These traditional methods have proven to be effective in detecting image forgery but with the increase in complexity in image editing tools and with the emergence of machine learning methods that have advanced pattern recognition, the traditional methods have been struggling to produce comparable results to deep learning methods. Thus, the popular approaches to image forgery detection have been strictly machine learning based for the past decade or so.

1.1.4. Deep learning-based methods

Machine learning methods don't extract methods from the images colour data exclusively they utilize any data that can be extracted from the image through any preprocessing that can enhance the correct classification of forged and real images. Classifiers that are commonly used in this field are Support Vector Machines (SVMs)(Palanivel, N., 2008) and Random forests. However, deep learning methods specifically Convolutional Neural Networks (CNNs) have proven their dominance and superiority in their capability to classify images correctly. These neural networks learn to extract features from image data that make them significantly more capable of identifying subtle hints of image manipulation that traditional methods are not able to detect (Cun et al., n.d.).

The development of a reasonably precise and balanced digital image forgery detection algorithm is best achieved through deep learning. Furthermore, the modern emergence of

deep learning allowed for the creation of techniques that far surpass the metrics of traditional methods. Which do not depend on neural networks for the classification of images as real or forged.

Passive forgery detection techniques are numerous, but this paper will focus on the methodologies utilizing deep learning, convolutional neural networks (CNNs) to be more specific and their modern variations, as CNNs yield the best evaluation metrics when it comes to detecting whether an image is forged or authentic (Zhou et al., n.d.).

1.1.5. Hybrid methods

In addition, hybrid methods have also demonstrated their capabilities in detecting image forgeries. Certain features extracted from a traditional method such as statistical or transform based methods may be used as data to train Convolutional Neural Networks on.

By combining the traditional methods ability to over emphasize patterns in raw image data and deep learning methods superior pattern recognition capabilities hybrid systems are capable of producing state of the art accuracy in the classification of forged and real images. Papers such as “Hybrid deep learning and machine learning approach for passive image forensic” provide evidence of the efficiency of hybrid models (A. Thakur & Jindal, 2020).

1.1.6. Image forgery types

There are several types of images forgeries some of the forgeries addressed in this paper are:

1.1.6.1. Copy-move forgery.

This forgery is done by taking a part of an image and that part is then pasted another location on the image that the copied part comes from. This is done to hide sensitive or essential data or add incorrect information to raw image data. This type of digital image forgery is usually not limited to this, and other transformations such as changing brightness and applying blurring, are usually preformed on the image to hide the fact that it has been edited to the viewer (Chauhan et al., 2016; Mehrjardi et al., 2023). Figure 1 shows an example of this type of specific forgery.

Figure 1: Image manipulated through copy-move forgery from MICC-F2000 dataset (P1000293tamp2.JPG).

1.1.6.2. image splicing

Image splicing involves taking the data of a region of an image and pasting it onto another location on a different image resulting in a single forged image. This typically results in

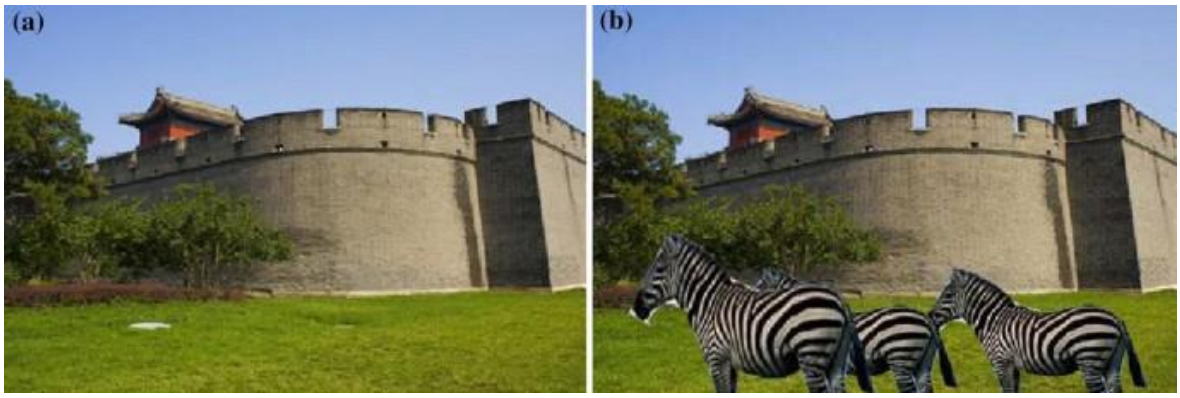


Figure 2: A splicing example where zebras are foreign to the image

visible edges and blurring at the region of the splicing, however image manipulation tools might be used to hide this so that the viewer is unaware of the forgery the image has undergone(Mehrjardi et al., 2023; Thakur & Rohilla, 2020). An example is shown in Figure 2.

1.1.6.3. Image retouching

This is usually the least harmful type of digital image forgery. Image retouching involves enhancing or minimizing certain features of an image such as changing the colours of pixels(Mehrjardi et al., 2023). This technique is popularly used in enhancing the perceived attractiveness of a person in an image such as in a magazine or an advertisement(Kaur & Rani, 2016). Figure 3 shows an example of this specific type of forgery.

Figure 3: Example where multiple aspects of the image where manipulated through image retouching

1.1.6.4. Object removal

Object removal is when a region of the image is deleted entirely from the image. Object removal is a type of destructive forgery. This type of forgery has two types of techniques: inpainting and copy-move (Elharrouss et al., 2019).

The copy-move removal technique is achieved through taking the data at a region of the image and placing a copy of it onto another region of the image thus the original region that the new image data is placed onto is deleted. The copy-move technique is more common than the inpainting technique due to the simplicity of the technique.

Image inpainting is a technique utilized in repairing photographs that were damaged or removing unwanted details such as an object in the background. This method changes the values of a region in the image to the values of the surrounding pixels so that the region appears to blend in with the background to the naked eye. The woman was removed from the image in the example in Figure 4.

Figure 4: Object removal where the woman was removed from the image

1.1.7. Evaluation metrics.

There are a few metrics that are commonly used to evaluate the performance of a model on a benchmark dataset (Mehrjardi et al., 2023). The higher the values are in these metrics the better the metrics of the model on the validation or test data. Along with these metrics there is usually a confusion matrix represents the model's classification performance on the validation data. These metrics use 4 fundamental values that are the components of a confusion matrix, these values are:

TP (True Positive): the number of manipulated images detected correctly as manipulated images.

• FP (False Positive): Real or authentic images detected wrongly as manipulated images

• FN (False Negative): manipulated images falsely missed as authentic or real images.

• TN (True Negative): authentic or real Images detected correctly as authentic or real images.

These values can also be used on a on a pixel level instead of images if the model has localization(M & M.N, 2015).

The metrics commonly used in the field of image forensics are:

Accuracy: this is defined as the ratio of correctly classified images to the overall number of images, and this is the equation used to calculate accuracy

$$Accuracy = \frac{(TP + TN)}{(TP + FP + TN + FN)}$$

Recall: this is a ratio of correctly classified manipulated images (TP) divided by the overall number of images that are a forgery and this is the equation used to calculate recall

$$Recall = \frac{TP}{(TP + FN)}$$

Precision: This metric is defined as the ratio of correctly classified manipulated images (TP) to the number of images overall predicted to be the manipulated, and this is the equation used to calculate precision

$$Precision = \frac{TP}{(TP + FP)}$$

F1-score: This is defined as two into P and R which is then divided by the sum of P, and this is the equation used to calculate the F1-score.

$$F1 = \frac{2 * (P * R)}{(P + R)}$$

These are the metrics that will be used in this paper to compare the performance of different model as other metrics are usually not available in some research papers.

1.1.8. Detailed tasks of semester 1

1.1.8.1. Introduction to deep learning

I had no experience in deep learning prior to this project but I had worked for years with multiple programming languages including python. Which is the best language for deep learning. In November, I decided to enroll in the Deep learning specialization course of Andrew Ng on Coursera, and I had completed the course of the specialization and was halfway through the second course when I decided to implement what I learned into developing an algorithm for image forgery detection.

1.1.8.2. Finding appropriate literature

at the same time, I enrolled onto the course I was searching using google scholar, semantic scholar, and tools like them to make an extensive list of image forgery detection forgery detection papers that are relevant and applicable to be included in my research project. and by taking the deep learning course I had enough knowledge to be able to comprehend the literature I collected, and the next step was to summarize it.

1.1.8.3. Going through the literature

I began filtering the papers I would use for my project, and everything was coming into place. The papers I started summarizing the papers and comparing the methods used by the authors so I would find the most optimal methodology for my research. Without extensive knowledge of deep learning, I was not able to replicate most of the methods used as they were not open source and needed deep knowledge of the subject matter to be able to replicate and improve on.

1.1.8.4. Drafting and writing the interim report and reflecting on my findings

After I had summarized each paper and the advantages drawbacks of each method, I began writing drafts of the report to externalize and reflect on my findings to be able to assess which method I should use for image forgery detection and the error analysis method used by (Gupta et al., 2022) and (Sudiatmika et al., 2019) was the most fitting as their implementations had flaws such as letting the model overfit and not implementing class weights during training to make the model handle the dataset imbalance appropriately. I noticed the potential in improving their implementations and the next step was to implement this method using python along with the TensorFlow library.

1.1.8.5. Coding the algorithm

I began implementing the architecture of the model by (Gupta et al., 2022)as code to test it and improve on it. I chose the CASIA V2 dataset as it is the most comprehensive dataset with over nine thousand images in JPEG and PNG image formats, I added class weights and implemented early stopping along with using the Adam optimizer varying learning rates and began iterating to reach the optimal model. I ran about fifty iterations of modeling and testing until I achieved a satisfactory result and remarkable improvements over the existing CCNs utilizing ELA. My model on the CASIAV2 dataset classified images with an accuracy of 94.58% on the validation dataset and state-of-the-art performance on the rest of the evaluation metrics. When tested on the whole dataset the model achieves an impressive 98.47% accuracy proving the possibility of creating models with state-of-the-art performance from scratch by utilizing existing architecture.

1.1.9. Anticipated problems and how to solve them

Problem 1- datasets

The publicly training datasets are insufficient for training the model well enough to be able to classify whether a novel image is forged or real with the remarkable performance.

Possible solutions:

1. Find substantial amounts of data to train on with datasets that are not commonly trained from scratch on. Such as ImageNet which is a large dataset that contains a one thousand object classes and over a million training images that can be used for training.
2. Expand existing datasets using data augmentation techniques that are easy to implement such as the ImageDataGenerator function that is available in TensorFlow that can create massive amounts of similar images with certain transformations such as Rotation in minutes.
3. Add a pretrained model to the algorithm such as EfficientNet or ResNet50

Problem 2- generalization

A single model will not achieve high accuracy for multiple forgery types simultaneously due to the difference in patterns of different forgery types which lower the model's accuracy.

Possible solutions:

1. Create a different model for each forgery type.
2. Employ a pretrained model for feature extraction to improve general accuracy.

Problem 3- Computational resources

Training multiple models will be computationally heavy as I am working on a laptop with a NVidia GTX 1650 and the training process requires experimentation and retraining with different parameters to get the best possible model.

possible solutions:

1. Move the training from locally on visual studio code to a cloud hosted Jupyter Notebook service such as Google colab or Kaggle Notebooks
2. Aim for a less accurate model and reduce training iterations.

Problem 4- Localization:

So far, the error level analysis convolutional neural network only detects forgeries on an image level and outputs either a 0 or a 1 the location of the forgery and what has been manipulated exactly in the image will not be describable in natural language using a large language model.

Possible solutions:

1. Develop a model from scratch that is able detect forged pixels.
2. Apply a pretrained model to the existing forgery algorithm as the location of the forgery is not the main objective of the research but rather whether the image is forged or not. Detecting forged pixels accurately is a way more challenging task than detecting forged images accurately. Therefore, it is wiser to use transfer learning or have a pretrained model detect the location of forged pixels.

Problem 5- Natural language description

The large language model to be used to describe the forgery and its location has not yet been chosen.

Possible solutions:

1. There are thousands of open source or openly available models that describe images using text with varying levels of. Indecisiveness about the model comes from the rapidly evolving landscape of large language models and not wanting to tie the project to a language model that would be outdated or significantly worse than the language models released in the next few months.

1.1.10. Plan of tasks during semester 2

1. Try to train the model on more than one dataset with near state-of-the-art evaluation metrics which can be challenging.

I have found that training the model on more than dataset at a time reduced the accuracy significantly (below 90%) and the more variation the training data has whether it be because it is from different datasets, or it is for different forgery types, The model's accuracy decreases significantly. To make a model that has excellent performance to novel images I need to implement a large dataset with great variation yet great accuracy. I must look further into the literature to find a methodology that achieves this and can be reverse engineered into my research project. I should try to achieve a satisfactory model within the first month and iterate to improve for the rest of the semester.

2. Find an appropriate cloud service for training.

The model needs to be trained on an extremely large number of images. To achieve a model with great generalization that is not overfitting to a specific dataset or style of images or a specific forgery type. Thus, I am going to try Google Colab and Kaggle Notebooks and compare between them and check if there are any other alternatives to local training on visual studio code. This should be done within the next couple of weeks to start training on large data early.

3. Find more relevant literature for localization and generalization.

I have yet to implement localization and I have not researched well enough about it, so I need to read more into the literature to know what is possible to implement and improve on. As for generalization I have not found a way to implement it successfully therefore I need to keep searching for more papers to analyze and reverse engineer the right methodology. Time plan for this task is the whole semester, however I am going to totally overhaul my existing list of papers to improve for the dissertation within the first month of the semester.

4. Stay up to date with image to text model releases.

To describe the type of forgery using heatmaps and natural language. Localization must be implemented and then a large language model can infer the region of localization and type of localization through image data. I need to stay up to date with images to text models to know when a good enough model is released that can be implemented into my research. There might be good enough open-source models currently available, but I do not want to sell myself short and decide early when a better alternative might be available soon. I will be monitoring new model releases for the whole semester or until I find a satisfactory model.

1.1.11. Limitations and Comments on tasks planned for semester 2

I had plans during semester one to use a model that was trained on ImageNet data as a feature extractor and improve the generalization utilizing ImageNet weights and use a model to describe image forgery in natural language. However, upon experimenting with pretrained models locally the kernel of Jupyter notebook kept crashing during training, trying smaller batch sizes did not help and after trying a to work with a few pretrained models as feature extractors such as EfficientNetB0 and Resnet50 and loading YOLO or ImageNet weights the jupyter kernel always crashed during training. I had to improve the model so I researched and found the 30 SRM high pass filters and after experimenting with different architectures I decided that this is the method I will use for my research.

Moving the model training and evaluation to cloud services was not feasible due to how large image data is, the cloud services would have been extremely expensive and there was no funding provided for this research. Thus, cloud computing was not an option, and I was not able to use any pretrained model to computational limitations. The ELA and SRM filters dual branch CNN was the best architecture I could achieve with these limitations.

1.1.12. Popular Datasets available

There are many datasets available in this field, yet most datasets are outdated and don’t effectively evaluate deep learning image forgery detection models. A comparison of the most prominent datasets is done in Table 1. A more detailed exploration of the datasets present in the table is found below.

1.1.12.1. CASIA V2 dataset

Type: Image Forgery Detection

Size: 7491 authentic images, 5123 manipulated images.

Resolution: 240 × 160 to 900 × 600

Image format: JPEG, BMP, TIFF

Forgery Types: copy-move forgery, splicing forgery.

The CASIA V2 (Chinese Academy of Sciences Institute of Automation) dataset is an image forgery detection dataset that is commonly used for evaluation and benchmarking of image forgery detection algorithms. It there are 12614 images in the dataset which is significantly more images than most image forgery detection datasets. The forged images in the dataset have been through post processing techniques such as copy and move transformations.

The dataset contains compressed and uncompressed images, the compressed images can be used for image tampering detection based on jpeg compression artifacts which is the method used in this paper and which is further enhanced using noise residual images from filtering the images through 30 high pass SRM (spatial rich model) filters.

The CASIA V2 dataset is more challenging than the CASIA v1 dataset as the cassia v2 dataset has been intentionally post processed to manipulate the boundary regions of forged images to make the detection more difficult. One major drawback in certain scenarios, of using the CASIA V2 dataset is the lack of ground truth images which are used for localization of forgeries in image forgery detection algorithms. Thus, the CASIA V2 dataset can only be used for image forgery detection on an image level rather than at a pixel level like many other datasets.

1.1.12.2. CoMoFod (small) dataset

- Type: Image Forgery Detection.
- Size: 5000 authentic images and 5000 forged images
- Resolution: 512x512 pixels.
- Forgery Types: copy-move forgery

The CoMoFod small dataset is widely used for benchmarking and evaluating image forgery detection algorithms specifically algorithms focused on detecting copy-move forgeries. There are 10000 images total in this dataset. In this dataset, there are 260 image sets of which there are 200 images in the small image category and 200 images in the large image category. The forged images in this dataset have gone through the following transformations:

1. Translation – This is forgery where a region of the image data is pasted on another area of the image without performing and other transformation
2. Rotation – This is a forgery here a region of the image data is rotated and is then pasted on another area of the image and in the image
3. Scaling – This is where a region of the image data is scaled and is then pasted on another region.
4. Distortion- This is a where a region is distorted and is then pasted on another area in the image.
5. Combination – This is a where a certain area of the image undergoes at least two transformations and is then pasted onto a different area.

A major advantage of the CoMoFod dataset over the CASIA V2 dataset is the presence of ground truth images. Every image set of CoMoFod consists of:

1. The original image- the image without any transformations performed on it-------
2. The coloured mask- this is the mask that indicates which regions are original and which are forged. Black regions are the regions that are not forged, and the coloured regions are the forged regions
3. The binary mask- a binary mask of the image for the evaluation of the detection algorithm. The black regions are the original regions, and the black regions are the forged regions.
4. The forged image- this is the original image after going through a forgery transformation

The postprocessing operations that were applied on the images are JPEG compression, noise adding, brightness change, colour reduction, contrast adjustments and image blurring. The dataset also has convenient image naming for indicating which category the image belongs to they are in the format of N1_M1. Where N1 is three (this is the small image category) or two (this is the big image category) digits indicating the image id in the category. Furthermore, M1 categorizes the image in of four different marks:

1. “O” – this mean that the image is original
2. “B” – this is the black and white binary mask of the image
3. “M” – this the coloured mask of the image
4. “F” – this is the forged image

Post processed images are also named with an additional mark M2. The images are name in N1_M1_M2 naming scheme where M2 indicates the postprocessing method applied on the image:

1. “CA” – the image has undergone contrast adjustments
2. “JC” – the image has undergone compression
3. “NA”- the image has noise added to it with an averaging filter
4. “CR” – the image has undergone colour reduction
5. “IB” – the image has undergone blurring
6. “BC” – the image has undergone a brightness change

1.1.12.3. IMD2020

- Type: Image Forgery Detection
- Size: 35,000 authentic images, 35,000 forged images (Total: 70,000)
- Resolution: Varies, generally high resolution
- Formats: JPEG, PNG
- Forgery Types: Splicing, copy-move, removal

The IMD2020 is a large dataset that was introduced by the paper “Extended IMD2020: a large-scale annotated dataset tailored for detecting manipulated images” by Adam Novozámský et al. This dataset was designed with the aim of addressing the need for novel image forensic techniques that can generalize to novel data. However, the dataset’s large size is a trade off with computational resources as a dataset this large would need a powerful computer to be able to train and evaluate deep learning models on it (Novozámský et al., 2021).

On the other hand, this dataset has images generated by advanced methods such as Generative Adversarial Networks (GANs), which other datasets like CASIA V2 and MICC-F2000 lack. One big standout feature in this dataset is the inclusion of 2000 “real-life” manipulated images that the authors sourced from the internet. This addition is unique in the design of image forgery detection datasets. Additionally, the dataset contains a controlled set of 2579 real images that were taken by 32 different cameras containing natural scenes. This is a controlled subset that is valuable for image forgery detection methods that work on analysing sensor noise and camera-specific artifacts, these details are crucial for methods that rely on high pass filters and noise residuals to enhance the patterns that are indicative of image manipulation (Novozámský et al., 2021).

1.1.12.4. MICC-F2000

- Type: Image Forgery Detection
- Size: 1300 authentic images, 700 forged images.
- Resolution: 2048 × 1536 pixels.
- Image Format: JPEG, PNG
- Forgery Types: copy-move forgery

The MICC-F2000 data was developed by researchers at the Media Integration and Communication Center (MICC) which is located at the University of Florence. This dataset contains 2000 JPEG images making it ideal for jpeg compression artifacts-based classification of forged and real images. The resolution of the images is large enough for detailed features to be present in the image which will help the feature extraction layers produce detailed features for the images. It is the largest dataset out of all the Media Integration and Communication Center (MICC) datasets.

The forged images in MICC-F2000 dataset have been created to closely mimic real-world forged images. Just like the CASIA V2 dataset the forgeries have undergone post processing and various transformations to make them less detectable to the naked eye and to forgery detection algorithms. This dataset has great variety in image types and scenes which include natural landscapes and indoors, this variety is beneficial for the neural networks ability to generalize of novel data.

The post processing techniques used to forge images here include scaling rotation and compression techniques which were performed with also with blending of the forged region to the background to make the forgery detection task more challenging for the algorithm.

Feature	CASIA V2	CoMoFoD (small)	MICC-F2000	IMD2020
Size	7491 authentic images, 5123 manipulated images	5000 authentic images, 5000 forged images	1300 authentic images, 700 forged images	35,000 authentic images, 35,000 forged images
Total Images	12614	10000	2000	70000
Resolution	240 Ã— 160 to 900 Ã— 600 pixels	512 Ã— 512 pixels	2048 Ã— 1536 pixels	Varies, generally high resolution
Image Format	JPEG, BMP, TIFF	JPEG	JPEG, PNG	JPEG, PNG
Forgery Types	Copy-move, Splicing	Copy-move	Copy-move	Splicing, Copy-move, Object Removal
Post-processing	Blurring, Copy-move transformations, Compression	Translation, Rotation, Scaling, Distortion, Combination	Scaling, Rotation, Compression, Blending	Various transformations including rotation, scaling, compression
Ground Truth available	No	Yes	No	Yes
Advantages	Large dataset, challenging due to post-processing	Detailed ground truth, easy-to-use naming conventions	High resolution, variety in image types and scenes	Large and diverse dataset with a wide range of forgeries and detailed ground truth
Disadvantages	No ground truth for localization, focuses on image level	Limited to copy-move forgeries	Smaller size compared to other datasets, no ground truth	Requires substantial computational resources for processing

Table 1: A comparison of image forgery detection datasets

1.2. Problem Statement

It is challenging to develop an algorithm that accurately detect and identify digital image forgeries in an era where image editing software is widely available and easy to use, leading the spread of massive quantities of forged images. This situation is a significant threat to the reliability of digital images, leading to the spread of misinformation and general distrust in the information presented in digital images. This problem persists due to the limitations of the current detection methods, which struggle with novel images and the several types of techniques and forgeries used in image manipulation. Additionally, the lack of solutions to the nontechnical person has been the biggest obstacle towards establishing trust in digital image forgery detection methods and widespread availability of these methods. The main issue is the obscurity or lack of existing forgery detection methods that can accurately detect various forms of image forgery. Thus, there is a need for an established methodology that can extract clues for image forgery effectively and detect forged images with a high accuracy.

1.3. Research Objectives

My aim is to develop an innovative image forgery detection algorithm that is easily replicable with near state-of-the-art performance. The algorithm will be able to extract image data that is indicative of forgery detection effectively and input it into a model that will have remarkable accuracy in detecting image forgery, and the models should have an excellent generalizing ability to novel images.

The algorithm will handle feature extract from raw input data and image forgery detection according to the extracted features. The algorithm will provide a detailed explanation of the classification decision in natural language through an open-source large language model to increase the user’s confidence in the decision and analyse the reasoning behind it so that the user can then determine whether the decision is justified.

1.4. Research Questions

- How effective is the integration of Error level Analysis (ELA) and Spatial Rich Model (SRM) filtering in enhancing the performance of a CNN-based image forgery detection system?
- What is the improvement gained from a dual branch CNN utilising ELA and SRM filters rather than a single branch CNN utilising ELA exclusively for data preprocessing?
- What are the performance benchmarks of the proposed dual branch CNN on the prominent image forgery detection datasets (CASIA V2, CoMoFoD (small), and MICC-F2000)?
- What are the potential improvements or future work than can be built upon the work done in this research?
- How does the proposed model dual branch CNN model compare to other state-of-the-art deep learning models?

1.5. Research Methodology

The research methodology of this dissertation involves the following steps in order:

1. Literature review: an extensive and comprehensive review of the existing literature on image forgery detection techniques was conducted and the findings were recorded in the literature review section of this dissertation.
2. Dataset selection: the prominent datasets in the image forgery detection field were compared and reviewed to select the most appropriate datasets to train and evaluate the deep learning algorithm.
3. Data preprocessing: data preprocessing techniques were reviewed and ELA and SRM high pass filter were chosen and implemented for the best preprocessing possible with the research limitations
4. Model Development: a dual branch CNN model was developed with the branches taking an ELA input and an input filtered with 30 SRM filters respectively,
5. Models were trained models, and their performance was evaluated with metrics such as precision, recall, accuracy and F1-score. The results were then compared with the state-of-the-art methods results to benchmark the proposed model.
6. The findings of the research were discussed and the effectiveness of implementing ELA and SRM filtering in the proposed model was established

Chapter 2: Related work

2.1. Literature Review

There are many ways to create a model that detects image forgery. (“Velmurugan, S., & Subashini, T. S.,” 2022) proposed a k-fold cross-validation approach with five as the value of k for dividing the dataset into training and testing, and Google Net was used to extract the features from images to train a random forest algorithm. The accuracy of the proposed model is 89.55%. The metrics of the proposed model were satisfactory yet unremarkable.

(Katiyar & Bhavsar, 2022) used an in-house convolutional neural network (CNN) for classification, and the authors decided to employ Grad-CAM to understand the results and for localization within the results. The reported accuracy for a copy and move COCO dataset is 70%, 80% for inpainting, 70% for combined datasets. An observation to be made is that neural networks tend to perform worse with varied forgery types in the dataset, an approach with multiple neural networks for each forgery type would be ideal. However, the Grad-CAM results were insightful for the location of the forgery, the exact forged pixels were not clear in the heatmap but rather a rough location of where the forgery could have occurred.

A new CNN architecture was proposed by (Bayar & Stamm, 2018) which they called MISLnet, which can detect distinct types of image forgery with up to 99.97% accuracy. The innovation in this paper is the constrained convolutional layer that suppresses image content while adaptively learning manipulation detection features. However, this model was trained using an extremely large dataset and is more complex than other methods. Nevertheless, this novel approach addresses the challenge of designing forensic detectors for multiple image manipulations, making it a notable contribution to the field of digital image forensics. The replication of this study would be computationally expensive and time consuming.

The limitations of current forgery detection methods were highlighted by (Sharma et al., 2023) where they offered an overview over existing techniques including early detection methods and latest deep learning methods. The challenges that were identified were the need for more standardized datasets, benchmarks, and evaluation criteria to improve the reliability and accuracy of forgery detection models. The paper also highlights the need for more general models that are not overfitted to the same popular datasets, and the need for more robust models that can detect different forgeries under different conditions. The problem of

generalization is a common repeated theme in digital image forgery detection where models have underwhelming metrics when tested on novel data.

For copy and move forgery detection, (Nazir et al., 2022) introduced an advanced approach for detection copy and move forgery using a custom Mask-RCNN model. This method enhances the detection and segmentation of edited regions in images by utilizing the strength of DenseNet for feature extraction and with the use of Mask-RCNN framework for localization, segmentation, and classification of images. This model shows superior accuracy over the CASIA, CoMoFod and MICC-F2000 datasets. Over the CoMoFod dataset this method achieved a precision of 98.12%, recall of 95.85% and F1-score of 96.97% which is state of the art performance according to the authors. The authors also demonstrated the model's high accuracy when detecting with different post processing attacks such as brightness variation, colour reduction and noise addition attacks. However, when tested with images with huge light variations the model falls short.

(Qazi et al., 2022) proposed a methodology that leverages the architecture of ResNet50v2 enhanced with YOLO CNN weights for detecting image forgery. CASIA_V1 and CASIA_V2 were used as benchmark datasets with an 80/20 train test split. Without transfer learning an accuracy of 81% was obtained. However, using transfer learning an accuracy of 99.3% was obtained. However, the precision, f1-score and recall of this proposed model were not mentioned. Moreover, this study highlights the benefits of utilizing transfer learning in image forgery detection as an accuracy gap of 18.3% was observed when transfer learning was implemented. The increase in the complexity and size of the model might be a drawback and prevent the utilization of transfer learning even if it yields better results.

An efficient approach was used by (Gupta et al., 2022), where Error Level analysis was used as a preprocessing step for image forgery detection before inputting the image into the convolutional neural network. This method works by identifying the discrepancies in compression levels across different areas of an image which is often indicative of manipulation. Due to the image being saved again after editing which introduces compression artifacts into the image.

The process of converting image to an Error level analysis image involves recompressing a PNG or a JPEG image to 90% quality in this paper and comparing it with the original image to highlight any areas that have varying error levels. These areas are usually more

pronounced in tampering regions, as a result the tampering region is highlighted. This method works best with PNG and JPEG image formats as the tiff format in the dataset CASIA V2 has different compression artifacts, which when introduced into the dataset lowers the performance of the model significantly. However, the methodology used in this paper yields a near state-of-the-art results and is easily replicable as the exact architecture of the model is mentioned and is a relatively simple CNN.
(Sudiatmika et al., 2019) used a similar approach of using Error Level Analysis to highlight compression artifact and use a convolutional neural network for the detection of forged images. However instead of training from scratch an end-to-end neural network, the authors resorted to using a pretrained neural network architecture called VGG16. This pretrained model is widely used commonly used in deep learning for image recognition tasks. This architecture was originally trained on the famous ImageNet dataset which has millions of images across thousands of categories. However, pretrained architecture can encounter the problem of the lack of generalization in the field of digital image forensics. The study reports a high accuracy rate in identifying forgeries, with a validation accuracy of 88.46% after one hundred epochs of training. On the other hand, In the paper of (Gupta et al., 2022) a higher training accuracy and validation accuracy was achieved. Where the values for accuracy over one hundred epochs with training and validation sets were 96.13% and 92.20%.

These two studies can indicate that the use of transfer learning in digital image forensics is not always beneficial over training from neural networks from scratch on a known dataset. The accuracy discrepancy of models trained using transfer learning and models trained from scratch can be explained with the efficiency of architecture used. Pretrained models might have a lot of unnecessary information that can be a drawback when training for new information. Models trained on a single dataset usually perform better than models that are trained in one more than one dataset when validating images from the same training dataset. However, when testing models on images with unfamiliar forgeries even if it was from the same forgery type the models that were developed with transfer learning techniques perform better as indicated in this recent survey of digital image forgery detection techniques (Mehrjardi et al., 2023) and from the comparison of pretrained models and models trained from scratch in existing literature,

Another paper (Koul et al., 2022) trained a neural network from scratch on the MICC and tested its accuracy. And the authors compared the results with existing techniques such as

VGGNet features, Scale-Invariant Feature Transform (SIFT), and Stationary Wavelet Transform (SWT) based methods and it was concluded that that the proposed CNN-based approach is superior in all aspects. The authors also encouraged future work to use to employ more advanced deep learning architectures such as RCNN As this approach achieved an accuracy of 97.52% on the MICC-F2000 dataset, outperforming existing methods by a significant proving once again the validity of non-transfer learning approaches in the field of digital image forensics. If computational power and time is not a constraint a neural network trained from scratch on existing datasets can have state-of-the-art evaluation metrics with correct implementation and an efficient architecture.

(Tahir & Bal, 2024) had an interesting approach to addressing the dataset drawbacks in the field of image forgery detection, they acknowledged the shortcomings of the existing image datasets and that the datasets are outdated relative to the current image forgery detection algorithms.

The state-of-the-art deep learning algorithms preform extremely well on the current datasets that it has become extremely difficult to know which deep learning algorithms excel when all the recent deep learning methods fall a couple percentages of a one hundred percent accuracy on the most popular image forgery detection datasets. And when the neural networks trained on these datasets are presented with novel data the performance of the neural networks is unremarkable. To overcome the shortcoming of the image forgery detection datasets a novel framework was designed by the authors that automatically creates realistic spliced images, 24964 images are in the dataset created by the authors.

This new dataset could be used in future work to train and evaluate deep learning models on image forgery detection techniques. The authors intentionally increased the complexity and the realism of images within the dataset to make it even more difficult to detect by deep learning methods. When tested with the early fusion manipulation detection model the model's classification accuracy of the author's dataset was 71.899%, while the model had an 84.5% accuracy on the Casiav1+ dataset. This proves the greater efficacy of the new dataset for model training and evaluation it is challenging to classify this dataset correctly with high accuracy, and the author's dataset should be considered for future research.

The approach (Joshi et al., 2022) took was Similar to (Gupta et al., 2022)and (Sudiatmika et al., 2019) as the authors used error as a preprocessing step to enhance the inconsistencies in the image so that the image detection neural network has more useful features for

classification of the images. The ELA method was applied in a neural network utilizing transfer learning with state of the art pretrained models trained on ImageNet, including Inception-V3, ResNet152-V2, XceptionNet and EfficientNet-V2L. This network was then used for binary classification of forged and authentic images.
The highest precision was achieved with the VGG-19 model; the model achieved 98.25% precision. The other Inception-V3, ResNet152-V2, XceptionNet and EfficientNet-V2L models achieved precisions of 94.11%, 90.38%, 94.61%, 85.20% respectively. The other models outperformed VGG-19 in other metrics such as recall but the most important quality of image forgery detection networks is precision as it indicates a low false positive rate in the classification of the model and if the model was used in a court of law or for any important matter where it is important to know whether an image is real or false the false positive rate should be as low as possible.

This study shows that VGG-19 should be used for low positive rate and should be used in any future work that needs a low positive rate the VGG-19 pretrained image net forgery classification model should be among the models considered for any low positive rate critical work.

In another paper "Deep Learning-Based Image Forgery Detection: A Comprehensive Analysis with ManTra-Net CNN," (Chavan et al., 2024) used a method for classifying forged images by enhancing the capabilities of the Manipulation Tracing Network (ManTra-Net). The authors emphasized how the need for accurate image forgery detection models is crucial in this era as digital image manipulation is rampant and easily accessible for anyone with malicious intent. The Authors used the Error Level Analysis method that is commonly used for the preprocessing step to high inconsistencies and any indications of potential image manipulation in the raw image data.

The authors used a simplified version of Mantra-Net, the architecture stars with convolution layers which is typical of image forgery detection neural networks as these convolution layers extract useful data from visual data. The authors used pooling layers and Adam optimizer with a dense layer at the end of the neural network to classify the data into a final SoftMax layer. This architecture was aiming to improve the classification capabilities of the Mantra-Net.

The models were then trained and evaluated on the CASIA2C dataset. The authors model demonstrated its effective and high precision and recall in the task of classifying tampered

images on the CASIA2C dataset. The main forgery types that were detected were splicing and copy and move. The authors model achieved an accuracy of 96% a precision of 91%, a recall of 95%. This proves the effectiveness and reliability of using Error level analysis along with a MantTra-Net CNN as an algorithm or image forgery detection.

The VGG-19 model trained on image net achieved impressive results in previous work. The authors (Mohan et al.) of "Image Forgery Detection Using Ensemble of VGG-16 and CNN Architecture" Used a transfer learning approach with the VGG-16 architecture that was trained on ImageNet and the authors expanded on the model by using a custom CNN. The authors used Error Level Analysis as a preprocessing step. The authors resized the image to the uniform size of 224x224 pixels before using them as input into the neural network. The combined approach of Using VGG-19 with ImageNet weights and a custom CNN to further extract details achieved a training accuracy of 96.8% along with a validation accuracy of 88.46% on the Kaggle forgery detection dataset. These results are acceptable alone but with the VGG-19 achieving a precision of 98.25% in earlier studies the use of VGG-16 as a pretrained feature extractor instead of the VGG-19 architecture yield lower metrics and VGG-19 should always be used instead of VGG-16.

Another paper that explored using pretrained models as feature extractor is "Image Forgery Detection based on Fusion of Lightweight Deep Learning Models" by (Ramya et al., 2023). In this paper the authors presented an innovative approach for image forgery detection and classification of forged and real images. The authors combined the SqueezeNet, MobileNetV2, and ShuffleNet models using a decision fusion method, this boosts the detection capabilities of the network without sacrificing too much computational efficiency. The author's approach is best suited for systems where computational resources are constrained or are scarce and using more complex models might not be feasible.

The author's method consists of two phases, the feature extraction phase and the forgery detection phase. Features are extracted from the images by utilizing the pretrained models. lightweight models as feature extractors without regularization in the first phase, in the second phase the models are finetuned to the data while regularization is implemented. This fusion technique utilizes the strengths of each model forming a solid system that is capable of classifying whether an image is forged or not with a high accuracy. The accuracy that was achieved is 95,4%. Moreover, the precision and recall were 95% and 96.1% respectively which prove the systems reliability in image forgery detection.

This a significant improvement over each of the individual model's accuracy. The accuracy result proves the effectives of integrating multiple models to improve the metrics of the image forgery detection system.

In a recent paper "Entropy-Based Image Copy-Move Forgery Detection" the authors (Jiang & Lu, 2023) used the approach of detecting Copy-Move forgeries by using entropy information present in the image to enhance the keypoint-based detection method in smooth regions of the image. The traditional keypoint-based detection algorithms struggle to generate matches in the "smooth" areas of the image or the areas that can be described as having low-texture. This is especially prominent in the algorithms employing Scale Invariant Feature Transform. The problem of struggling to find matches in "smooth" regions of the image can be addressed with the concept of entropy as the authors claim.

This quantifies the texture contrast in the local regions in a better manner which leads to a more beneficial distribution of keypoints. The author's method consists of three stages. The preprocessing stage in which entropy image are generated then and keypoints are detected on those images. The author's keypoint generation method generates up to 1.5 times more keypoints in the "smooth" regions of the image than the traditional method, then there is the hierarchical keypoint clustering stage which matches the keypoints using an overlapped entropy level clustering technique that minimizes the complexity and enhances the accuracy of the process of matching keypoints, and finally the post processing stage. Where the algorithm utilizes the information of the matched keypoints to detect tampered regions. The authors achieved a balance between computational efficiency and performance.

This proposed method achieves a true positive rate (TPR) of 99.07% and a false positive rate (FPR) of 99.07% on the CMH+GRIPori dataset. This high precision is remarkable, and this method should be considered for future work in image forgery detection using keypoint-based techniques.

The machine learning methods still vastly outperform humans and traditional methods as demonstrated by the studies above. To understand human performance on these images (Frank et al., 2023) conducted an important study "A Representative Study on Human Detection of Artificially Generated Media Across Countries" to understand the human capabilities in detecting images, audio and text that were generated by artificial intelligence models. The author's finding that artificially generated images were extremely difficult for people to classify correctly as humans had an accuracy rate below 50% when classifying real

and artificially generated images. This insight emphasizes the need for machine learning models that are able to correctly classify images generated by artificial intelligence, as humans and traditional methods fall short as outlined by studies above.

For the task of detecting artificial intelligence generated images present Martin-Rodriguez et al. (2023) a sophisticated method of detecting artificial intelligence generated images using pixel wise feature extraction in combination with convolutional neural networks (CNNs). The paper "Detection of AI-Created Images Using Pixel-Wise Feature Extraction and Convolutional Neural Networks" focuses on distinguishing realistic AI-generated images from real images by using two main feature extraction techniques. The Photo Response Non-Uniformity technique (PRNU) and the Error Level Analysis (ELA) technique. The author's approach addresses the growing challenges created by the advancement of artificial intelligence image generators which produce highly realistic artificial intelligence generated images with the right prompts (Martin-Rodriguez et al., 2023).

In the experiments of Martin-Rodriguez et al. (2023), the authors used a dataset of 918 images of which 459 are AI generated and 459 are real images. The dataset is balanced yet the number of images is small and might not evaluate the algorithm effectively. However, the detection system based on ELA achieved an impressive accuracy of 98%, with a precision of 97%, a recall of 99%and an F1-score of 98%. Moreover, the PRNU-based system achieved an accuracy of 95%, precision of 93%, recall of 97%, and an F1-score of 95%. These accuracy rates emphasize the effectiveness of the approach that the authors implemented. To be certain that the author's results are reliable, they tested the robustness of their method using an extended dataset and alternative artificial intelligence image generators. The System maintained an accuracy of 99% for Error Level Analysis features (Martin-Rodriguez et al., 2023).

This study highlights the effectiveness of Error Level Analysis as a preprocessing step for convolutional neural networks input in detecting artificial intelligence generated images. Given the increase in artificial intelligence image generators and their exponentially increasing realistic output, this approach will be crucial in designing neural networks that can detect artificial intelligence generated images in the future. This study is a valuable addition to the field of image forensics and this approach should be considered in future work for the detection of photorealistic artificial intelligence generated images (Martin-Rodriguez et al., 2023).

Kumari et al. (2020) provided a review of modern techniques for image tampering detection. Their paper “A review of image detection, recognition and classification with the help of machine learning and artificial intelligence” highlights that the positive progress in deep learning algorithms convolutional neural networks have significantly improved the capabilities of image forgery detection algorithms. The authors elaborated on how CNNs are extremely effective in identifying and classifying image forgery detection and CNNS manage to achieve impressive results and accuracy in real-world applications(Kumari et al., n.d.).

This review also elaborated on the practical implementation of deep learning techniques using the popular deep learning frameworks such as Keras and Tensorflow. The author’s emphasized the utility of Tensorflow’s multi-dimensional array handling and keras’s streamlined process of neural network training. These two tools are valuable and indispensable in the development of sophisticated digital image forgery detection systems. The authors also compared the custom-built CNN object detectors with models that were pretrained (Kumari et al., n.d.).

The author’s concluded that custom-built CNN object detectors provide higher accuracy, but they demand extensive data and training time. Alternatively, pretrained models that utilize transfer learning approaches deliver a quick and typically sufficiently accurate alternative to custom-built CNN object detectors by refining networks that were already trained on large datasets (Kumari et al., n.d.). The custom-built CNN object detectors offer greater accuracy they should be utilized whenever accuracy is important, and the computational resource are not scarce. However, transfer learning approaches should be used when computational resources are scarce, and it would be difficult to train the neural network from scratch.(Kumari et al., n.d.)

An interesting addition of Error Level Analysis preprocessing was brought to my attention with the paper "Detection of Image Tampering Using Deep Learning, Error Levels and Noise Residuals”, in this paper Chakraborty et al. (2024) discusses the growing challenging of detecting image tampering. Thus, the authors propose an innovative dual branch CNN approach that utilizes ELA and noise residuals from the SRM to detect image manipulation with remarkable precision (Chakraborty et al., 2024).

Their approach starts typically by preprocessing the image using ELA to highlight the discrepancies during image manipulation. However, the author paired with that SRM which focuses on noise patterns altered through tampering. SRM was first introduced in the paper

“Rich Models for Steganalysis of Digital Images” by Fridrich & Kodovsky (2012), this dual preprocessing method prepares the data for the CNN, which is designed to automatically extract and classify any feature that are indicative of image manipulation. Their dual branch approach not only enhances the model’s accuracy and precision, but it also ensures generalization to various forms of image forgery, such as splicing and copy-move forgeries(Fridrich & Kodovský, 2012), (Chakraborty et al., 2024),

The results of this dual branch approach were impressive, the model managed to achieve a near-perfect accuracy of 98.55%. This surpasses many existing techniques in efficiency and efficacy. This performance is attributed to the hybrid approach of traditional methods and deep learning methods that enable the model to learn image tampering patterns that are indicative of manipulation. The performance of this model combination surpasses any traditional methods. The work of Chakraborty et al. showcases the potential of hybrid approaches in digital image forensics, this study not only sets a new standard for tampering detection, but it also paves the way for more future work and research to be done to explore and expand upon the integration of SRM and ELA with other deep learning frameworks and sophisticated processing methods (Chakraborty et al., 2024).

To conclude, the techniques used in the above studies and the reviews, and the surveys indicate that the most efficient forgery detection techniques are the techniques that utilize transfer learning for training a new neural network for detection. And when transfer learning is employed, it utilizes existing neural network architectures that have been trained on enormous datasets.

While the above studies suggested that transfer learning can significantly boost performance especially in datasets like those the model was pre-trained on, custom trained models from scratch can offer remarkably superior performance in more specialized tasks such as detecting a certain type of forgery so that the patterns the model is trained on are all similar and have higher accuracy. Additionally, certain techniques like error level analysis can offer impressive metrics with the right implementation on a neural network trained from scratch. in multiple studies, Error level analysis has proven its effectiveness enhancing the model’s detection ability by highlighting the discrepancies indicative of manipulation before they are fed into the neural network for a higher chance of detection. Moreover, the combination of ELA and SRM implemented by Chakraborty et al. (2024) achieves the most impressive results.

On the other hand, the popular way to design a neural network for image forgery detection is the use of pretrained model mainly for extracting features from the image data. However, there is a notable lack of models pretrained on image forgery detection classification specifically as most pretrained image models have been designed on for ImageNet Classification which classifies images into one of a thousand classes. This classification task is different and extracts different features for classification. Papers mentioned above suggest that the use of models pretrained on ImageNet might not be the best approach for image forgery as the model architecture varies depending on the task needed. If a neural network has enough parameters and understanding of the input data, it can classify the images correctly regardless if it was pretrained or an end-to-end neural network trained from scratch. The ELA and SRM dual branch CNN method is interesting and recent and should further be expanded upon with future research. Thus, it will be implemented in this dissertation to benefit the field of digital image forensics as a whole.

Chapter 3: Methodology

3.1. Choice of datasets

One of the most critical choices in the development and evaluation of image forgery detection algorithms is the choice of datasets. The datasets should be well balanced and designed for the neural network to be able to learn and extract useful features and patterns. After considering the most popular choices of datasets in research papers I selected the CASIA V2 dataset, the CoMoFoD (Small) dataset and the MICC-F2000 dataset to be the datasets that I will train and evaluate a convolutional neural network on, and I would recommend any future research in this field use these datasets for their models. The following is my rationale for choosing these datasets over other datasets:

1. CASIA V2 Dataset

- Number of images: the number of images in the CASIA V2 dataset are 12,614 which is the largest dataset used in this research. Many other datasets have only a few hundred to a few thousand images which are less effective for extensive training and evaluation of the neural network
- Image forgery type variety: The dataset has forged images of two major types of digital image forgery, copy and move forgery and splicing forgeries which ensures that the neural network has better generalization ability for more forgery types as they are exposed to more forgery techniques.
- Post processing complexity: The creators of this dataset ensure that the forged images undergo further post processing to intentionally hide and manipulate boundaries of forged regions
- Presence of compression artifacts: This dataset has over 9000 images in JPEG format. This will be crucial in feature extraction as these artifacts are the indicators of manipulation in forged images.

2. CoMoFoD (small) dataset:

- Exclusively focused on Copy and Move forgery: the only type of image forgery present in this dataset is Copy and Move forgery, with a large number of images (10000) this dataset is convenient for training the neural network on copy and move forgery with great generalization

- Wide variety of transformations: this dataset has 6 different types of transformations with variations in the intensity within each transformation this ensure the models ability to generalize to a wide variety of image transformations.
- Post processing complexity: This dataset
- has forged images of two major types of digital image forgery, copy and move forgery and splicing forgeries which ensures that the neural network has better generalization ability for more forgery types as they are exposed to more forgery techniques.
- Post processing complexity: The creators of this dataset ensure that the forged images undergo further post processing to intentionally hide and manipulate boundaries of forged regions
- Presence of compression artifacts: This dataset has over 9000 images in JPEG format. This will be crucial in feature extraction as these artifacts are the indicators of manipulation in forged images.

3. MICC-F2000 Dataset:

This dataset has high resolution images up to 2048x1536, these image sizes are ideal for feature extraction for the neural networks and will result in better generalization in the trained model. Moreover, the forgery transformations are varied including scaling, rotation and compression which define the images to be similar to real world forged images.

Additionally, after these transformations were applied post processing was done to blur the boundaries and make it more difficult which helps the neural network extract features for manipulated images that were manipulated with the intention of presenting to the viewer of the image that is original. Thus, manipulated images that were maliciously manipulated now can be detected as

3.2. Convolutional Neural Networks (CNNs)

The methodology that was chosen for the classification task is a Convolutional Neural Networks. Convolutional Neural Networks (CNNs) are neural networks that contain at least one convolution layer. They are the most used type of neural networks for extracting features from visual data such as images. They were first introduced by LeCunn et. al. (1989) and CNNs are the state of the art when it comes to visual data extraction (Cun et al., 1989).

The types of layers usually found in these networks are:

- Convolution layer: this layer applies filters or kernels to an image that is inputted to the network. This layer helps the network understand features such as edges, patterns and or textures that are present in visual data.
- Pooling layer: this layer reduces the spatial dimensions of the input to reduce computational complexity and prevent overfitting while not losing the important feature data extracted from the convolution layer
- Fully connected layer: this layer receives the features passed from the convolution layer and pooling layer and based on these features then next layer decides which class the data belongs to
- Softmax layer: this layer is usually found at the end of the neural network and has as many neurons as there are classes. In the case of image forgery, it is either a 1 or a 0 for whether the image is classified as real or forged
- Dropout layer: dropout layer implements a regularization technique to reduce overfitting in neural network. The dropout layer drops a percentage of input values so that the network doesn't overfit to it. The dropout layer is very effective in minimizing overfitting thus it has become a standard practice to use it in image forgery detection neural networks.

However, to correctly classify images whether they are forged or not using, there needs to be clear patterns for the convolutional neural network to extract and make correct classifications upon. The raw image data (the values in each colour channel of all pixels) was insufficient for high accuracy classification. When the raw image data of the CASIA V2 dataset was used as input into the neural network, the accuracy of the trained model was didn't not exceed 65%. This was a clear indication that a feature extraction technique needed to be implemented. Some paper used the raw image data as input and got remarkable accuracy but

that was in combination with utilizing transfer learning which was not feasible due to computational constraints in my research

3.3. Error Level Analysis

This technique was introduced by (Krawetz, 2007).Error level analysis is a technique that is well documented and proved its efficacy in emphasizing features that were indicative of tampering, multiple papers achieved impressive results using this technique. It enhances the differences in error levels that are produced when an image is compressed in a JPEG format. Thus, if the differences in error levels are enhanced there is more useful data indicative of image forgery in the input of the neural network if ELA is performed on the input image rather than using the raw image data as input. Error level analysis works by performing the following steps on an image:

An image that has been compressed before (saved in JPEG format) is then saved again at a lower quality which compresses it even further. In my method I resaved images to 90% quality. This step produces new compression artifacts that are distributed uniformly across the image. After this step it is expected that the compression artifacts present in the image are uniform across the image however if the image was manipulated then there will be areas with more compression artifacts which are the manipulated regions.

An error map is then generated by comparing the saved version with the original image. This error map highlights any compression artifacts that exist in regions of the image and the areas that high any abnormal compression levels are highlighted accordingly. This error map with the compression levels is used as the input into one branch of the neural network

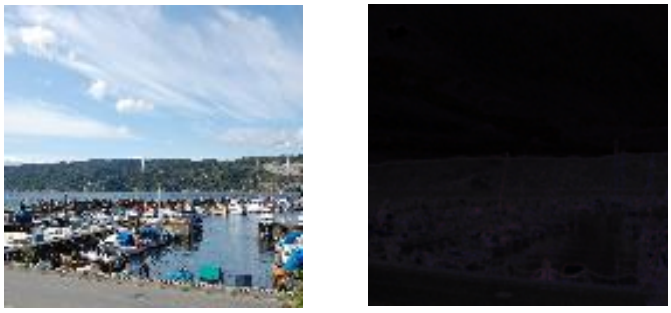


Figure 5: Example of an image processed with ELA (original left ELA right)

3.4. Spatial Rich Model (SRM) Filters

Spatial Rich Model SRM filters (Fridrich & Kodovský, 2012) are exceptional at extracting subtle noise residuals that exist in the data of images that have undergone certain manipulations such as copy-move or splicing forgeries. The SRM filter were originally developed for steganalysis purposes, however due to their ability to outline hidden data in an image they can be utilized to preprocess images for image forgery detection algorithms. The SRM filters highlight the noise components that may be indicative of manipulation that the neural network may then utilise in the classification decision.

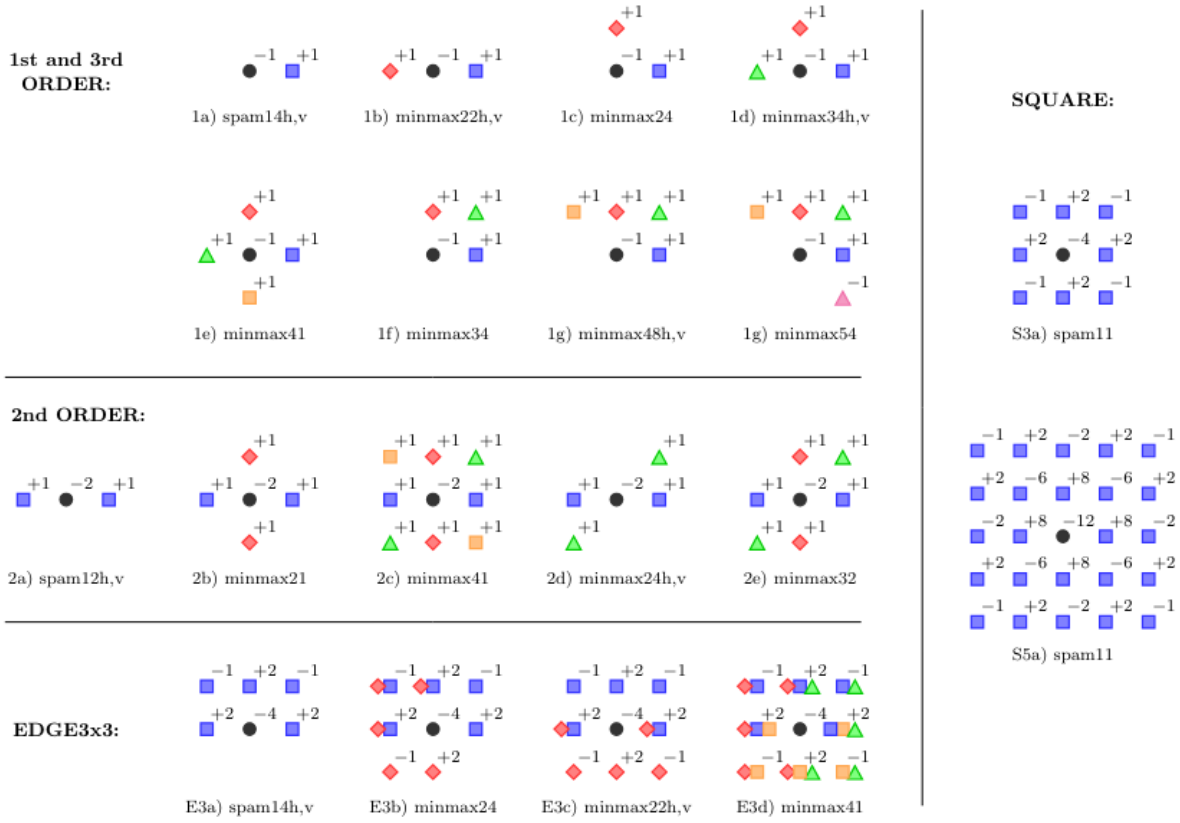


Figure 6: The 30 SRM high pass filters

The SRM filters isolate the noise components of the image data by processing the data through 30 high pass filters (fig), the result of this filtering is the isolation of the discrepancies that are present in the image data but are not visible in the image data. Then data is produced that enhances the detection of various types of forgeries even if the SRM filters have not been intended to be used for the purpose of preprocessing images in image forgery detection tasks (Fridrich & Kodovský, 2012).

In my approach I employ SRM filters that is utilised for creation noise residuals data from the raw image data and use it as input it into the second branch of the CNN. This approach

allows the neural network to receive more features than the features present only in the ELA data producing higher accuracy and generalization capabilities. The addition of the SRM data branch was the only approach that increased the accuracy of the neural network, multiple approaches were tried and did not improve the neural network’s accuracy. This approach was designed by Chakraborty et al. (2024) in their remarkable paper "Detection of Image Tampering Using Deep Learning, Error Levels and Noise Residuals" (Chakraborty et al., 2024).

3.5. Proposed Model Architecture

The CNN architecture I propose contains two main branches, each processing different types of input images (ELA and SRM), these two branches are later merge and form a single model that is capable of extracting features from ELA and SRM high pass filtering techniques. Thus, extracting a huge amount of hidden data in the image and classifying the images whether they are real or forged based on the hidden data.

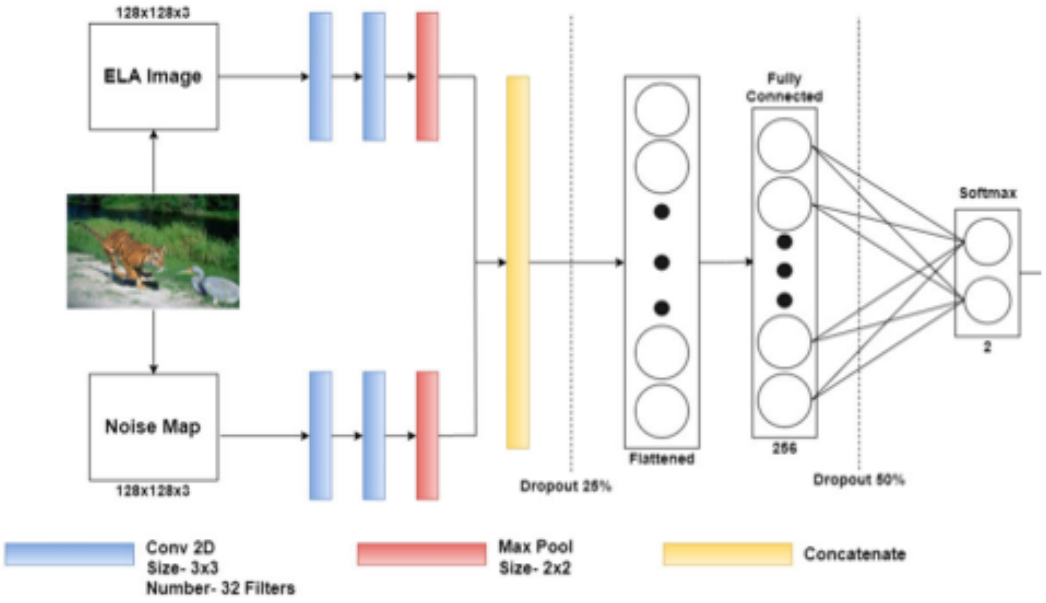


Figure 8: Architecture of the proposed model

This proposed architecture has 2 input layers as shown in Figure 8. The input shape of each layer is (128,128,3). The input goes in each branch through 2 convolution layers each containing 32 filters of size 3x3 and same padding is applied in each convolution layer. These layers extract specific features from the ELA and SRM filtered image. Then the data is processed with a Max Pooling Layer. Utilising a 2x2 max pooling layer reduces the spatial dimensions of the input by half to down sample the feature maps while preserving the important features.

Then both branches are concatenated. This is crucial to merge the two feature maps from both branches into one output combining the detailed information from ELA and SRM processed images. Then the data goes through a dropout layer this layer works by randomly setting 25% of the input values to zeros in each training step this reduces overfitting and improves the

generalization ability of the model. The next layer is a flatten layer that flattens combines the features maps into a single one-dimensional vector. Thus, the output has become suitable to become an input for a fully connected layer.

The next layer is a dense layer with 256 units and ReLU as its activation function. This Dense layer is used to learn the complex patterns from the combined feature vector. Then the data goes through a dropout layer this layer works by randomly setting 50% of the values to 0, the values are selected randomly. The final layer is a layer containing 2 units with its activation function being SoftMax. This dense layer uses the SoftMax function to produce probabilities for both class (real and forged). This layer outputs the final classification result which is either a 1 which indicates that the image is real or a 0 which indicates the image is forced as it has been implemented in the python code.

The model was built in visual studio code using python with Tensorflow. The model contains a total of 67,129,922 parameters which are all trainable. When training the models, I added class weights for imbalanced datasets which were CASIA V2 and MICC-F2000.

Chapter 4: Setup and Experimental Results

4.1. System Configuration and Environment Setup

For the implementation of the image forgery detection algorithms, I used my personal computer. The computer used for the training and evaluation of models is an Acer Nitro 5 AN515-54 model. The computer contains an intel® Core™ i7-9750H CPU with six cores and twelve logical processors. The operating system on the computer is Microsoft Windows 11. Which supports hardware acceleration features essential for high-performance computing tasks that deep learning requires.

The system has 16 Gigabytes of ram installed as dual channel memory. The Graphics Processing Unit (GPU) of the system that the deep learning frameworks utilised is the Nvidia GTX 1650 with 4 Gigabytes of virtual ram (VRAM) available.

4.2. Software Tools used

Python was the programming Language used to program the deep learning image forgery detection algorithms. Python was chosen for its ease of use and availability of popular frameworks and libraries that makes running deep learning programs easy and effortless.

The python version used is 3.9.6, this version was chosen for the availability of libraries and frameworks on it. The more recent versions of python lacked important frameworks and libraries for my algorithms to work.

For the Deep learning frameworks to work on the GPU I needed to install the Tensorflow-Directml plugin. The popular Tensorflow library by google and the popular Pytorch library by Meta works performs computation exclusively on the central processing unit (CPU) of the system. Which is significantly slower at training and evaluating models. To avoid having to install Linux on my system which was not feasible as this is my personal computer, the only choice for using deep learning on windows is the Tensorflow-Directml plugin from Microsoft (Abadi et al., 2016).

Here is a list of libraries that were used in the python files and their versions cv2

- (OpenCV): 4.10.0.82
- numpy: 1.23.5
- Pillow (PIL): 10.3.0
- random: Standard library (no version needed)
- pickle: Standard library (no version needed)
- os: Standard library (no version needed)
- matplotlib: 3.9.0
- scikit-learn (sklearn): 1.5.0
- keras: 2.10.0
- tensorflow: 2.10.0
- itertools: Standard library (no version needed)
- time: Standard library (no version needed)
- tqdm: 4.66.4

4.3. Dataset preprocessing

The dataset preparation methods were implemented seamlessly using python. Two methods were implemented to preprocess the data to prepare them to be the input for each of the branch of the convolutional neural network architecture. The data of each image of a given dataset was processed using ELA for the first branch and the data was processed using 30 SRM high pass filters for the second branch. See Error Level Analysis and Spatial Rich Model (SRM) Filters to understand how these techniques process images.

4.3.1. Error level analysis (ELA)

To convert the images to ELA data the function convert_to_ela_image() was used. This preforms the following operations sequentially:

1. The function converts input image data to ELA image data
2. The image is then saved into the same path as a temporary image at 90% quality
3. The ELA image is generated by getting the difference of the values of data in the compressed image and the values of data in the original image
4. The result is the enhanced JPEG compression inconsistencies

4.3.2. Spatial Rich Model (SRM) Filtering

The SRM filter weights exist in a file with the name “SRM_Kernels.npy”. These filter weights are loaded and normalized with the highest absolute value in each filter respectively. This normalization is done to ensure consistent feature extraction across images

The 30 SRM filters that were loaded were applied to the input image data to discover the features that are indicative of image manipulation. These filters were applied in using the apply apply_srm_filters() function. This function performs the following operations sequentially:

1. The function applies the 30 SRM high pass filters to the raw image data, each colour channel has the filters applied to it separately
2. The combined residual image is then normalized and then clipped to enhance the visibility of noise patterns in the image data

4.3.3. Processing Real and Forged Images

After the images have been through the ELA technique and the SRM filters separately, the image data of each is then resized to a size of 128 by 128 pixel to reduce computational complexity of the training and evaluation process. The images from the dataset that are real are present in a path and the image that are forged in the dataset are present in another path. All the images that are real that are present in the real images path are assigned the label value 1 and all the images that are in the forged images path are assigned the label 0.

The processed ELA and SRM images, along with their labels, were combined and shuffled so that the CNN doesn't overfit to a certain order of images and learn an order-bias during training.

The final data is then saved using python's pickle module. The data is saved in a single file that can be loaded and be ready to use for training and evaluating the dual branch CNN architecture

4.3.4. Training and evaluation results

The algorithm was coded in two phases. The first phase was accomplished in semester one the second phase was accomplished in semester two. The first phase was a single branch ELA, and this was the architecture of the single branch convolutional neural network

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 124, 124, 32)      2432

 conv2d_1 (Conv2D)           (None, 120, 120, 32)      25632

 max_pooling2d (MaxPooling2D  (None, 60, 60, 32)       0
 )

 dropout (Dropout)           (None, 60, 60, 32)        0

 flatten (Flatten)           (None, 115200)            0

 dense (Dense)               (None, 256)               29491456

 dropout_1 (Dropout)         (None, 256)               0

 dense_1 (Dense)             (None, 2)                 514

=================================================================
Total params: 29,520,034
Trainable params: 29,520,034
Non-trainable params: 0
_________________________________________________________________
```

Table 2: The architecture of a single branch CNN

4.4. Evaluating the first algorithm

I split each dataset into an 80-20 train-test split. These were the evaluation metrics for each Dataset.

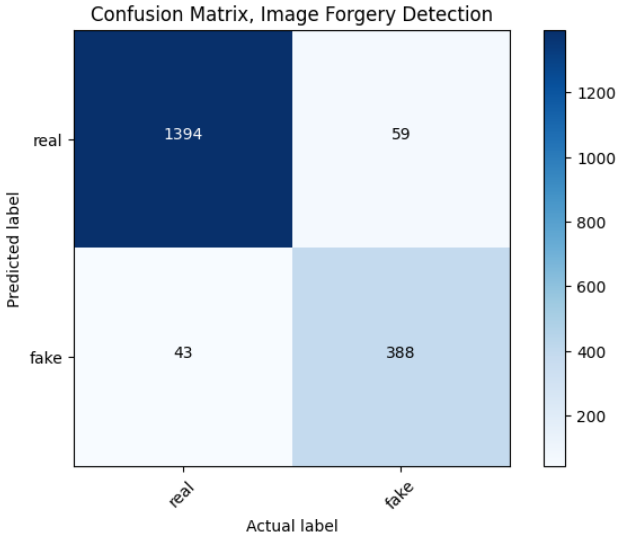
4.4.1. Evaluation results on CASIA V2

Accuracy: 94.59%

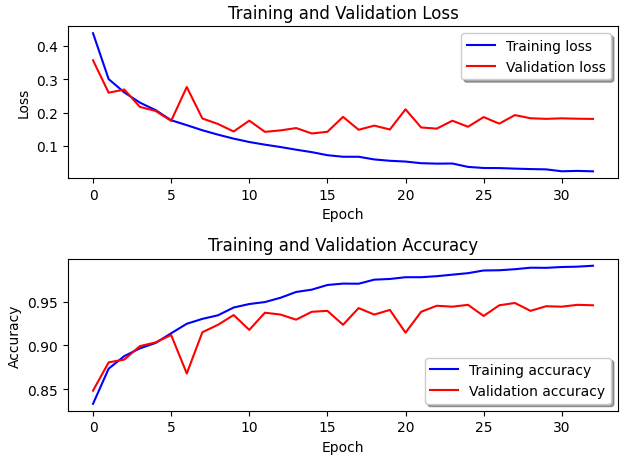
Precision: 97.01%

Recall: 95.94%

F1 Score: 96.47%



Graph 1: CASIA V2 validation data confusion matrix



Graph 2: CASIA V2 training and validation loss and accuracy history

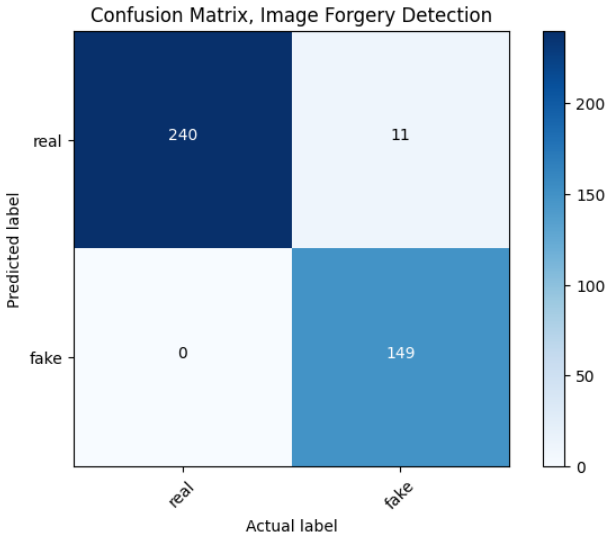
4.4.2. Evaluation results on MICC-F2000

Accuracy: 97.25%

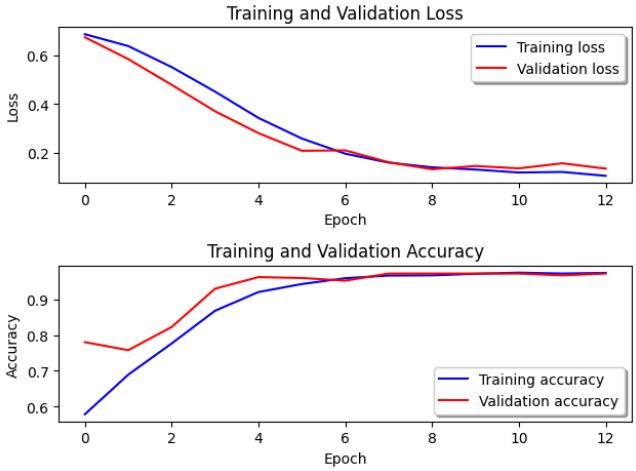
Precision: 100.00%

Recall: 95.62%

F1 Score: 97.76%



Graph 3: MICC-F2000 validation data confusion matrix



Graph 4: MICC-F2000 training and validation loss and accuracy history

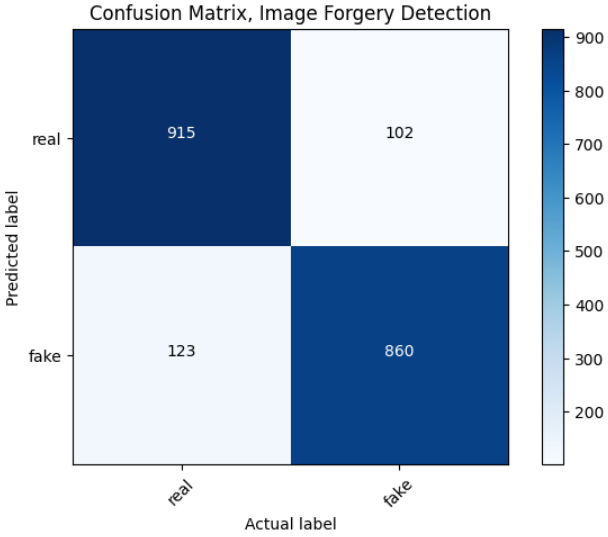
4.4.3. Evaluation results on CoMoFod

Accuracy: 88.75%

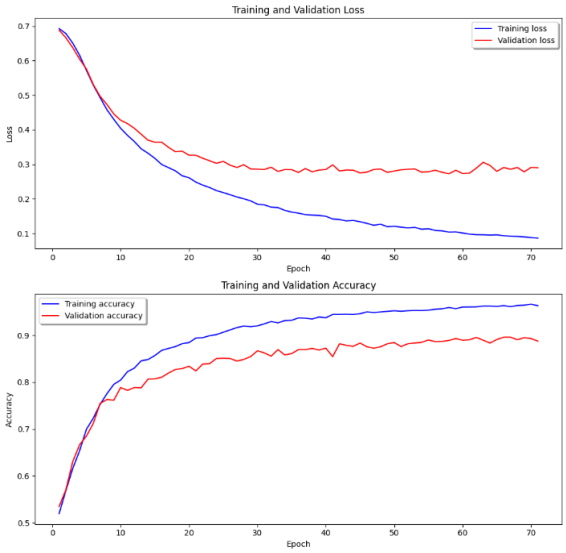
Precision: 88.15%

Recall: 89.97%

F1 Score: 89.05%



Graph 5: CoMoFod validation data confusion matrix



Graph 6: CoMoFod training and validation loss and accuracy

4.4.4. Comparison of the single branch CNN with the state-of-the-art methods

Technique	Dataset	Precision	Recall	F1-score	Accuracy
SEGMENTATION-BASED	MICC-F2000	86.0	88.0	87.0	73.9
SIFT	MICC-F2000	72.0	91.14	80.4	89.93
BRISK	MICC-F2000	99.0	94.6		95.98
(Koul et al., 2022)	MICC-F2000	97.0	96.0	97.0	97.5
Proposed ELA CNN	MICC-F2000	100.00	95.62	97.76	97.25

Table 3. Comparison of the single branch ELA architecture with State-of-the-art methods

As shown in Table 3 the proposed ELA CNN method closely matches or even outperforms state of the art methods on various metrics, it is the methodology with the best overall performance as proven with this comparison. Notably it has a precision of 100% that indicates that all tampered images in the MICC_F2000 validation dataset were identified correctly as forged images.

4.5. Evaluating the second algorithm

In the second semester, I added a second branch to the ELA CNN architecture I established in the first semester. The second branch received data processed by 30 SRM high-pass filters. The second branch improved the performance on all metrics. Note that the second algorithm had a much shorter testing period the results could be significantly higher. Each dataset was split into an 80-20 train-test split except for CASIA V2, the train test split for CASIA V2 was 90-10 to match the method used by (Chakraborty et al., 2024).

In the training process I used an initial learning rate of 0.001 except for the CoMoFod dataset as it appeared to not improve from 50% accuracy at a learning rate of 0.001, I reduced the learning rate to 0.0001. The decay rate was set to 0.0001. The formula the effective learning rate is:

$$\text{Effective Learning Rate} = 1 + (\text{decay} \times \text{iterations})\ \text{Initial Learning Rate}$$

I set the training to 200 epochs with early stopping implementing with patience 7 over no improvement over the validation accuracy except for the MICC-F2000 dataset I set the patience to 10 as the dataset was small and the learning process was too quick, I needed to check if there was room for improvement.

In this research I employed the Adam optimizer for its efficiency in training deep neural networks. Adam combines the benefits of momentum and RMSProp. Adam adjusts the learning rates for each parameter dynamically that accelerates convergence and improves the model's performance. With the initial learning rate of 0.001 and an adaptive decay, the Adam optimizer effectively handled the complex and sparse gradients in the dual branch CNN model (Kingma & Ba, 2014).

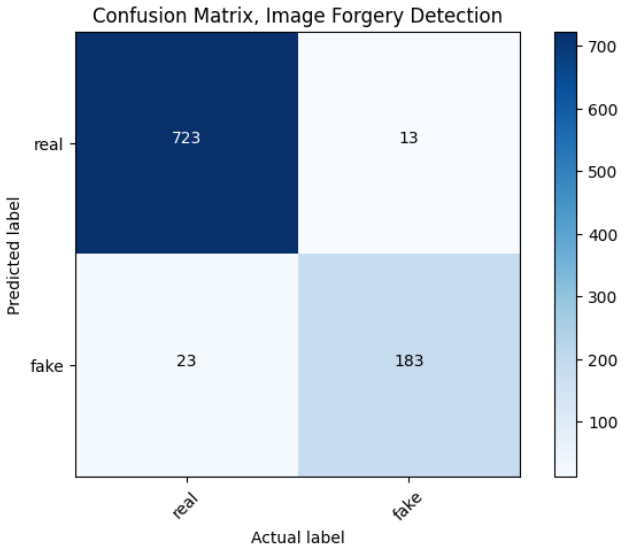
4.5.1. Evaluation results on CASIA V2

Accuracy: 96.18%

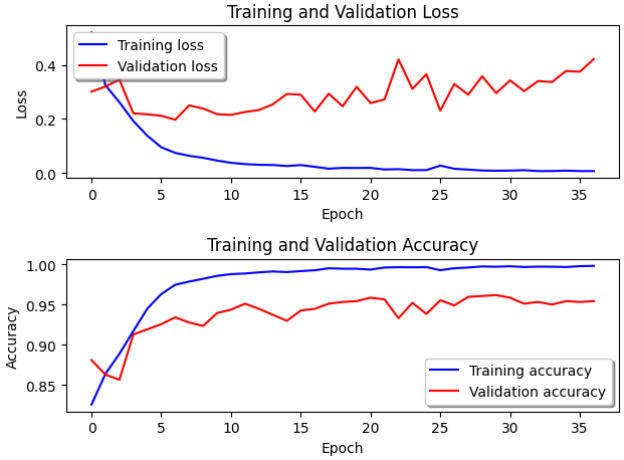
Precision: 96.92%

Recall: 98.23%

F1 Score: 97.57%



Graph 7: Dual branch CASIA V2 validation data confusion matrix



Graph 8: Dual branch CASIA V2 training and validation loss and accuracy history

4.5.2. Evaluation results on MICC-F2000

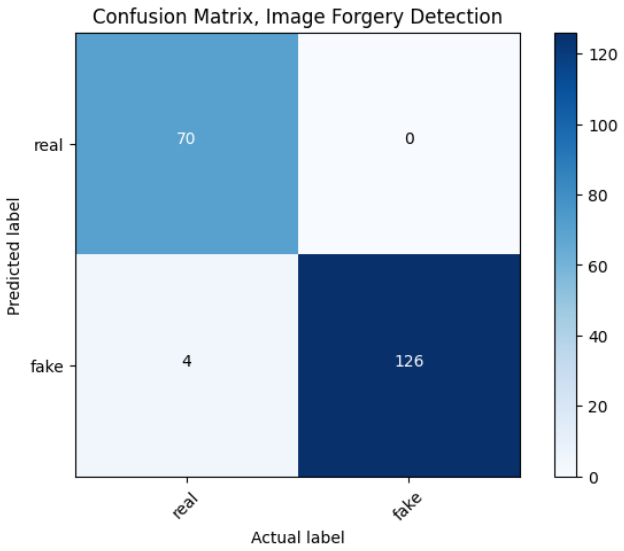
Learning rate used

Accuracy: 98.00%

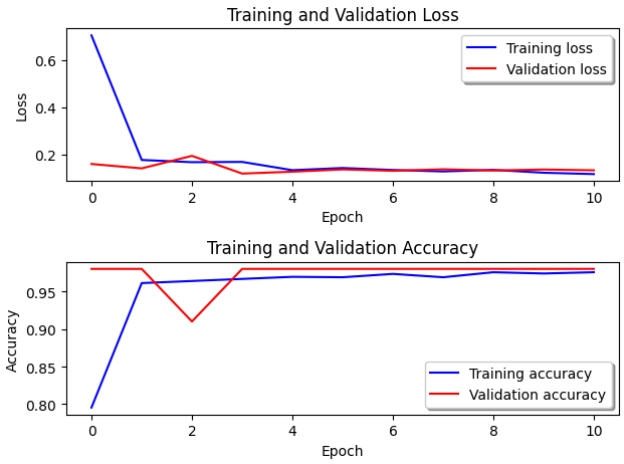
Precision: 94.59%

Recall: 100.00%

F1 Score: 97.22%



Graph 9: MICC-F2000 validation data confusion matrix



Graph 10: MICC-F2000 training and validation loss and accuracy history

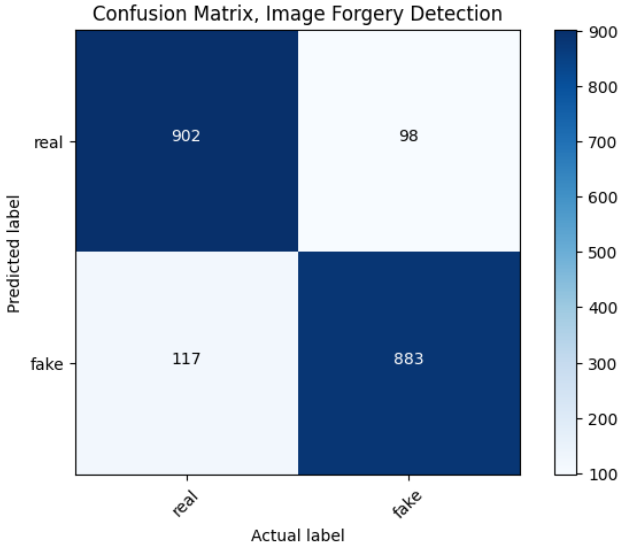
4.5.3. Evaluation results on CoMoFod (small)

Accuracy: 89.25%

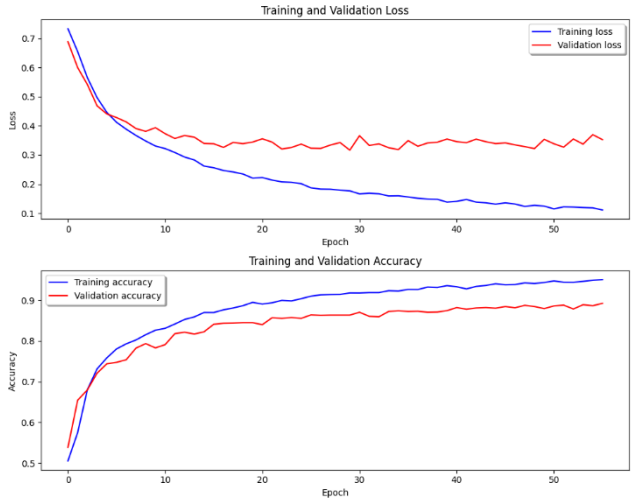
Precision: 88.52%

Recall: 90.20%

F1 Score: 89.35%



Graph 11: CoMoFoD validation data confusion matrix



Graph 12: CoMoFoD training and validation loss and accuracy history

4.5.4. Evaluation conclusion

Metric	Dataset	ELA+SRM Accuracy (%)	ELA Accuracy (%)	Improvement (ELA+SRM - ELA) (%)
Accuracy	CASIA V2	96.18	94.59	1.59
Precision	CASIA V2	96.92	97.01	-0.09
Recall	CASIA V2	98.23	95.94	2.29
F1 Score	CASIA V2	97.57	96.47	1.1
Accuracy	CoMoFod (small)	89.25	88.75	0.5
Precision	CoMoFod (small)	88.52	88.15	0.37
Recall	CoMoFod (small)	90.2	89.97	0.23
F1 Score	CoMoFod (small)	89.35	89.05	0.3
Accuracy	MICC-F2000	98	97.25	0.75
Precision	MICC-F2000	94.59	100	-5.41
Recall	MICC-F2000	100	95.62	4.38
F1 Score	MICC-F2000	97.22	97.76	-0.54

Table 4: Evaluating the improvement from the dual branch implementation

4.6. Comparing the proposed dual branch CNN with the state-of-the-art Methods

Note that the ELA+SRM accuracy reached 97% on CASIA V2 on the validation set but the kernel crashed during training, so this is the second-best accuracy achieved as the 97% model is not available. The state-of-the-art results for CASIA V2 accuracy according to (Zanardelli et al., 2022), is 99.07%. My proposed model got around 97% accuracy which is almost state of the art if more deep learning good practices were implemented the model could achieve higher accuracy, but time and computational power were limited. As for the MICC-F2000 dataset the model outperformed the state-of-the-art metrics mentioned by (Koul et al., 2022) making this approach the state of the art for the MICC-F2000 image forgery detection dataset.

Chapter 5: Conclusion

In the advancing landscape of digital media, the existence and advancement of sophisticated image manipulation tools necessitates advanced deep learning methodologies to ensure the authenticity of visual content circulating between gullible viewers. This dissertation presents a solid solution to the challenge of image forgery detection by developing and evaluating a dual branch Convolutional Neural Network (CNN) model that utilizes Error Level Analysis (ELA) and Spatial Rich Model (SRM) filtering.

This innovative approach was brought into my attention by Chakraborty et al. (2024), integrating ELA and SRM filters within a deep learning framework significantly enhances the model's capability to detect image forgeries in comparison with the use of ELA exclusively. The processing of the image by both ELA and SRM filters the compression discrepancies are amplified and can then be easily identified by the dual branch CNN. Moreover, this approach does not only capture more detail but also it improves performance on novel images due to picking up more detail indicative of image manipulation in the hidden data of the noise residual image (Chakraborty et al., 2024).

Extensive experimentation was conducted on prominent datasets, including CASIA V2, CoMoFoD (small), and MICC-F2000. These experiments concluded the efficacy of the proposed model as the performance of the dual branch CNN model achieves near state-of-the-art performance with accuracy rates reaching up to 98.00% on the MICC-F2000 dataset and 96.18% on the CASIA V2 dataset. This is particularly significant because it highlights how underutilised the combination of SRM and ELA in deep learning frameworks.

To conclude, the work in this dissertation not only find the addresses to the challenge of digital image forgery with a highly effective innovative solution but it also should shed light on the effectiveness of the use of ELA and SRM in image forgery detection. This paves the way for future advancements utilizing this method with other deep learning frameworks. The methodologies and insights presented in this paper will be instrumental in maintaining the originality of images in the digital world in the future.

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D. G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., … Zheng, X. (2016). *TensorFlow: A system for large-scale machine learning*. http://arxiv.org/abs/1605.08695

Ahmed, N., Natarajan, T., & Rao, K. R. (1974). Discrete Cosine Transform. *IEEE Transactions on Computers*, *C–23*(1), 90–93. https://doi.org/10.1109/T-C.1974.223784

Associate Professor, P. N. (2008). Image forgery detection using support vector machine. *International Research Journal of Engineering and Technology*, 4221. www.irjet.net

Bayar, B., & Stamm, M. C. (2018). Constrained Convolutional Neural Networks: A New Approach Towards General Purpose Image Manipulation Detection. *IEEE Transactions on Information Forensics and Security*, *13*(11), 2691–2706. https://doi.org/10.1109/TIFS.2018.2825953

Chakraborty, S., Chatterjee, K., & Dey, P. (2024). Detection of Image Tampering Using Deep Learning, Error Levels and Noise Residuals. *Neural Processing Letters*, *56*(2). https://doi.org/10.1007/s11063-024-11448-9

Chauhan, D., Kasat, D., Jain, S., & Thakare, V. (2016). Survey on Keypoint Based Copy-move Forgery Detection Methods on Image. *Procedia Computer Science*, *85*, 206–212. https://doi.org/10.1016/j.procs.2016.05.213

Chavan, R. B., Ghanashyam Kathavate, N., Jadhav, A. N., Hagir, A. D., & Rampurkar, V. (2024). International Journal of Research Publication and Reviews Deep Learning-Based Image Forgery Detection : A Comprehensive Analysis with ManTra-Net CNN. In *International Journal of Research Publication and Reviews* (Issue 5). www.ijrpr.com

Cun, L., Henderson, J., Le Cun, Y., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (n.d.). *Handwritten Digit Recognition with a Back-Propagation Network.*

Digital Image Forgery Detection Using Deep Learning Models. (2022). *International Journal of Emerging Trends in Engineering Research*, *10*(4), 236–240. https://doi.org/10.30534/ijeter/2022/091042022

Elharrouss, O., Almaadeed, N., Al-Maadeed, S., & Akbari, Y. (2019). *Image inpainting: A review*.

Farid, H. (2009). Image forgery detection. *IEEE Signal Processing Magazine*, *26*(2), 16–25. https://doi.org/10.1109/MSP.2008.931079

Frank, J., Herbert, F., Ricker, J., Schönherr, L., Eisenhofer, T., Fischer, A., Dürmuth, M., & Holz, T. (2023). *A Representative Study on Human Detection of Artificially Generated Media Across Countries*. http://arxiv.org/abs/2312.05976

Fridrich, J., & Kodovský, J. (2012). *Rich Models for Steganalysis of Digital Images*.

Gill, N. K., Garg, R., & Doegar, E. A. (2017, December 13). A review paper on digital image forgery detection techniques. *8th International Conference on Computing, Communications and Networking Technologies, ICCCNT 2017*. https://doi.org/10.1109/ICCCNT.2017.8203904

Gupta, A., Joshi, R., & Laban, R. (2022). *Detection of Tool based Edited Images from Error Level Analysis and Convolutional Neural Network*. http://arxiv.org/abs/2204.09075

Jiang, L., & Lu, Z. (2023). *An Effective Image Copy-Move Forgery Detection Using Entropy Information*. http://arxiv.org/abs/2312.11793

Joshi, R., Gupta, A., Kanvinde, N., & Ghonge, P. (2022). *Forged Image Detection using SOTA Image Classification Deep Learning Methods for Image Forensics with Error Level Analysis*. https://doi.org/10.1109/ICCCNT54827.2022.9984489

Katiyar, A., & Bhavsar, A. (2022). *Image Forgery Detection with Interpretability*. http://arxiv.org/abs/2202.00908

Kaur, A., & Rani, J. (2016). *Digital Image Forgery and Techniques of Forgery Detection: A brief review*. www.ijtrs.com

Kingma, D. P., & Ba, J. (2014). *Adam: A Method for Stochastic Optimization*. http://arxiv.org/abs/1412.6980

Koul, S., Kumar, M., Khurana, S. S., Mushtaq, F., & Kumar, K. (2022). An efficient approach for copy-move image forgery detection using convolution neural network. *Multimedia Tools and Applications*, *81*(8), 11259–11277. https://doi.org/10.1007/s11042-022-11974-5

Krawetz, N. (2007). *A Picture's Worth... Digital Image Analysis and Forensics*. www.hackerfactor.com

Kumari, R., Nikki, S., Beg, R., Gope, S. K., Mallick, R. R., Ranjan, S., & Dutta, A. (n.d.). *International conference on Recent Trends in Artificial Intelligence, IOT, Smart Cities & A REVIEW OF IMAGE DETECTION, RECOGNITION AND CLASSIFICATION WITH THE HELP OF MACHINE LEARNING AND ARTIFICIAL INTELLIGENCE*. https://ssrn.com/abstract=3611339

Lai, C.-C., & Tsai, C.-C. (2010). Digital Image Watermarking Using Discrete Wavelet Transform and Singular Value Decomposition. IEEE Transactions on Instrumentation and Measurement, 59, 3060-3063. *Instrumentation and Measurement, IEEE Transactions On*, *59*, 3060–3063. https://doi.org/10.1109/TIM.2010.2066770

M, H., & M.N, S. (2015). A Review on Evaluation Metrics for Data Classification Evaluations. *International Journal of Data Mining & Knowledge Management Process*, *5*(2), 01–11. https://doi.org/10.5121/ijdkp.2015.5201

Martin-Rodriguez, F., Garcia-Mojon, R., & Fernandez-Barciela, M. (2023). Detection of AI-Created Images Using Pixel-Wise Feature Extraction and Convolutional Neural Networks. *Sensors*, *23*(22). https://doi.org/10.3390/s23229037

Mehrjardi, F. Z., Latif, A. M., Zarchi, M. S., & Sheikhpour, R. (2023). A survey on deep learning-based image forgery detection. *Pattern Recognition*, *144*. https://doi.org/10.1016/j.patcog.2023.109778

N G T Mohan Kumar Manjushree H C Abhishek M Nallabothula Sneha, S. B. (n.d.). *Image Forgery Detection Using Ensemble of VGG-16 and CNN Architecture*. https://doi.org/10.5281/zenodo.8210388

Nazir, T., Nawaz, M., Masood, M., & Javed, A. (2022). Copy move forgery detection and segmentation using improved mask region-based convolution network (RCNN). *Applied Soft Computing*, *131*. https://doi.org/10.1016/j.asoc.2022.109778

Novozámský, A., Mahdian, B., & Saic, S. (2021). Extended IMD2020: a large-scale annotated dataset tailored for detecting manipulated images. *IET Biometrics*, *10*(4), 392–407. https://doi.org/10.1049/bme2.12025

Qazi, E. U. H., Zia, T., & Almorjan, A. (2022). Deep Learning-Based Digital Image Forgery Detection System. *Applied Sciences (Switzerland)*, *12*(6). https://doi.org/10.3390/app12062851

Ramya, Mrs. S., Panathukula, S. C., Kamtam, K., & Praharshith, G. S. (2023). Image Forgery Detection based on Fusion of Lightweight Deep Learning Models. *IJARCCE*, *12*(4). https://doi.org/10.17148/ijarcce.2023.124148

Sharma, P., Kumar, M., & Sharma, H. (2023). Comprehensive analyses of image forgery detection methods from traditional to deep learning approaches: an evaluation. *Multimedia Tools and Applications*, *82*(12), 18117–18150. https://doi.org/10.1007/s11042-022-13808-w

Sudiatmika, I. B. K., Rahman, F., Trisno, & Suyoto. (2019). Image forgery detection using error level analysis and deep learning. *Telkomnika (Telecommunication Computing Electronics and Control)*, *17*(2), 653–659. https://doi.org/10.12928/TELKOMNIKA.V17I2.8976

Tahir, E., & Bal, M. (2024). *Deep Image Composition Meets Image Forgery*. http://arxiv.org/abs/2404.02897

Thakur, A., & Jindal, N. (2020). Hybrid deep learning and machine learning approach for passive image forensic. *IET Image Processing*, *14*(10), 1952–1959. https://doi.org/https://doi.org/10.1049/iet-ipr.2019.1291

Thakur, R., & Rohilla, R. (2020). Recent advances in digital image manipulation detection techniques: A brief review. In *Forensic Science International* (Vol. 312). Elsevier Ireland Ltd. https://doi.org/10.1016/j.forsciint.2020.110311

Zanardelli, M., Guerrini, F., Leonardi, R., & Adami, N. (2022). Image forgery detection: a survey of recent deep-learning approaches. *Multimedia Tools and Applications*, *82*, 17521–17566. https://doi.org/10.1007/s11042-022-13797-w

Zhou, P., Han, X., Morariu, V. I., & Davis, L. S. (n.d.). *Learning Rich Features for Image Manipulation Detection*.

Appendix A: Code

Preprocessing (CASIA V2 example)

```
import cv2
import numpy as np
from PIL import Image, ImageChops, ImageEnhance
import random
import pickle
import os

# Function to convert an image to an ELA image
def convert_to_ela_image(path, quality=90):
    temp_filename = 'temp_file_name.jpg'

    image = Image.open(path).convert('RGB')
    image.save(temp_filename, 'JPEG', quality=quality)
    temp_image = Image.open(temp_filename)

    ela_image = ImageChops.difference(image, temp_image)

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

    ela_image = ImageEnhance.Brightness(ela_image).enhance(scale)

    return ela_image

# Function to apply SRM filters using loaded weights
def apply_srm_filters(image, srm_weights):
    steps = srm_weights.shape[-1]  # Number of filters
    filtered_channels = []

    for step in range(steps):
        filtered_channels_step = []
        for channel in range(3):  # Apply each filter to each color channel
            filter_kernel = srm_weights[:, :, 0, step]
            filtered_image = cv2.filter2D(image[:, :, channel], -1,
filter_kernel)
            filtered_channels_step.append(filtered_image)

        combined_residual = np.stack(filtered_channels_step, axis=-1)
        filtered_channels.append(combined_residual)
```

```
    combined_residual = np.mean(filtered_channels, axis=0)  # Average over all
filters
    combined_residual = np.clip(combined_residual, -3, 3)  # Clipping

    # Normalize to the range [0, 255]
    combined_residual = ((combined_residual - combined_residual.min()) /
                         (combined_residual.max() - combined_residual.min()))
* 255.0

    return combined_residual.astype(np.uint8)

# Prepare image for ELA
def prepare_ela_image(image_path, image_size=(128, 128)):
    ela_image = convert_to_ela_image(image_path, 90).resize(image_size)
    return np.array(ela_image).flatten() / 255.0

# Prepare image for SRM using loaded weights (apply SRM before resizing)
def prepare_srm_image(image_path, srm_weights, image_size=(128, 128)):
    image = cv2.imread(image_path, cv2.IMREAD_COLOR)

    # Apply SRM filters before resizing
    combined_residual = apply_srm_filters(image, srm_weights)

    # Resize the SRM filtered image
    resized_residual = cv2.resize(combined_residual, image_size)

    return resized_residual.flatten() / 255.0

# Preprocess images with both ELA and SRM
def preprocess_images(base_path, label, srm_weights, image_size=(128, 128),
max_images=None):
    X_ela = []
    X_srm = []
    Y = []
    image_count = 0
    file_log = []

    for dirname, _, filenames in os.walk(base_path):
        filenames.sort()
        for filename in filenames:
            if filename.endswith(('jpg', 'png', 'jpeg')):
                full_path = os.path.join(dirname, filename)
                X_ela.append(prepare_ela_image(full_path,
image_size=image_size))
                X_srm.append(prepare_srm_image(full_path, srm_weights,
image_size=image_size))
                Y.append(label)
                file_log.append(full_path)
```

```
                image_count += 1
                if len(Y) % 500 == 0:
                    print(f'Processing {len(Y)} images')
                if max_images and image_count >= max_images:
                    return X_ela, X_srm, Y, file_log
    return X_ela, X_srm, Y, file_log

# Save preprocessed data
def save_preprocessed_data(X_ela, X_srm, Y, file_log,
filename='CASIAV2DATA.pkl'):
    with open(filename, 'wb') as f:
        pickle.dump((X_ela, X_srm, Y, file_log), f)

def main():
    image_size = (128, 128)

    real_path = 'C:\\Users\\Karim Tarek\\Downloads\\archive\\CASIA2\\au' #real
path of any dataset is insterted here CASIA V2 in inserted as an example
    fake_path = 'C:\\Users\\Karim Tarek\\Downloads\\archive\\CASIA2\\Tp' #fake
path of any dataset is insterted here CASIA V2 in inserted as an example

    # Load SRM weights
    srm_weights = np.load('SRM_Kernels.npy')

    # Normalize each filter by its maximum absolute value
    for i in range(srm_weights.shape[-1]):
        max_value = np.max(np.abs(srm_weights[:, :, 0, i]))
        srm_weights[:, :, 0, i] /= max_value

    X_real_ela, X_real_srm, Y_real, log_real = preprocess_images(real_path, 1,
srm_weights, image_size=image_size)
    X_fake_ela, X_fake_srm, Y_fake, log_fake = preprocess_images(fake_path, 0,
srm_weights, image_size=image_size)

    X_ela = X_real_ela + X_fake_ela
    X_srm = X_real_srm + X_fake_srm
    Y = Y_real + Y_fake
    log = log_real + log_fake

    combined = list(zip(X_ela, X_srm, Y, log))
    random.shuffle(combined)
    X_ela[:], X_srm[:], Y[:], log[:] = zip(*combined)

    print(f'Total images processed: {len(X_ela)}')

    save_preprocessed_data(X_ela, X_srm, Y, log, filename='CASIAV2DATA.pkl')

if __name__ == '__main__':
```

```
    main()
```

Model Training and evaluation (CASIA V2 example)

```
import numpy as np
import matplotlib.pyplot as plt
np.random.seed(2)
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from keras.utils import to_categorical
from keras.callbacks import EarlyStopping, ModelCheckpoint
from keras import layers, models
import tensorflow as tf
import pickle
# from keras.callbacks import ModelCheckpoint
from CoMoFodpreprocessing import save_preprocessed_data
import os
import itertools
from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_score
from keras.models import load_model
import time
from tqdm import tqdm
import miccpreprocessing as pp
from sklearn.model_selection import StratifiedShuffleSplit
import json
gpus = tf.config.experimental.list_physical_devices('GPU')
if gpus:
    try:
        # memory growth needs to be the same across GPUs
        for gpu in gpus:
            tf.config.experimental.set_memory_growth(gpu, True)
        logical_gpus = tf.config.experimental.list_logical_devices('GPU')
        print(f"{len(gpus)} Physical GPUs, {len(logical_gpus)} Logical GPUs")
    except RuntimeError as e:
        # Memory growth must be set before GPUs have been initialized
        print(e)
        os.environ['CUDA_VISIBLE_DEVICES'] = ''
def load_preprocessed_data(filename):
    with open(filename, 'rb') as f:
        X_ela, X_srm, Y, log = pickle.load(f)
    return np.array(X_ela), np.array(X_srm), np.array(Y), log
X_ela, X_srm, Y, log = load_preprocessed_data('CASIAV2DATA.pkl') #CASIA V2
DATASET LOCADED YOU CAN LOADE THE DATASET YOU WANT TO USE

X_ela = X_ela.reshape(-1, 128, 128, 3)
```

```
X_srm = X_srm.reshape(-1, 128, 128, 3)

# One-hot encode the labels
Y = to_categorical(Y, 2)

# Stratified split to ensure same class ratio in train and validation sets
sss = StratifiedShuffleSplit(n_splits=1, test_size=0.1, random_state=5)

for train_index, val_index in sss.split(X_ela, np.argmax(Y, axis=1)):
    X_ela_train, X_ela_val = X_ela[train_index], X_ela[val_index]
    X_srm_train, X_srm_val = X_srm[train_index], X_srm[val_index]
    Y_train, Y_val = Y[train_index], Y[val_index]

# Print the number of samples
print(f'Training samples: {len(X_ela_train)}, Validation samples:
{len(X_ela_val)}')

# Checking the distribution of classes in the training and validation sets
print('Class distribution in training set:', np.bincount(np.argmax(Y_train,
axis=1)))
print('Class distribution in validation set:', np.bincount(np.argmax(Y_val,
axis=1)))
def build_model():
    # ELA Branch
    input_ela = layers.Input(shape=(128,128,3), name='ELA_Input')
    x_ela = layers.Conv2D(32, (3, 3), activation='relu',
padding='same')(input_ela)
    x_ela = layers.Conv2D(32, (3, 3), activation='relu',
padding='same')(x_ela)
    x_ela = layers.MaxPooling2D((2, 2))(x_ela)

    # SRM Branch
    input_srm = layers.Input(shape=(128,128,3), name='SRM_Input')
    x_srm = layers.Conv2D(32, (3, 3), activation='relu',
padding='same')(input_srm)
    x_srm = layers.Conv2D(32, (3, 3), activation='relu',
padding='same')(x_srm)
    x_srm = layers.MaxPooling2D((2, 2))(x_srm)

    # Concatenate the outputs of both branches
    combined = layers.Concatenate()([x_ela, x_srm])
    combined = layers.Dropout(0.25)(combined)
    combined = layers.Flatten()(combined)

    # Fully connected layer
    fc = layers.Dense(256, activation='relu')(combined)
    fc = layers.Dropout(0.5)(fc)
```

```
    output = layers.Dense(2, activation='softmax')(fc)

    # Create the model
    model = models.Model(inputs=[input_ela, input_srm], outputs=output)
    return model

model = build_model()
model.summary()

epochs = 200
batch_size = 16

optimizer = optimizer = tf.keras.optimizers.Adam(
    learning_rate=0.001,
    beta_1=0.9,
    beta_2=0.999,
    epsilon=1e-08,
    decay=0.0001
)

model.compile(optimizer=optimizer, loss='binary_crossentropy',
metrics=['accuracy'])

early_stopping = EarlyStopping(monitor = 'val_accuracy',
                               min_delta = 0,
                               patience = 7,
                               verbose = 1,
                               mode = 'auto')

no_au = 7354
no_tp = 2064
total_images = no_au + no_tp

weight_for_au = (1 / no_au) * (total_images / 2.0)
weight_for_tp= (1 / no_tp) * (total_images / 2.0)
class_weights = {0: weight_for_tp, 1: weight_for_au}
checkpoint_filepath = 'finalmodelcasiatest5.h5'
checkpoint_callback = ModelCheckpoint(
    filepath=checkpoint_filepath,
    monitor='val_accuracy',
    save_best_only=True,            # Save only the best weights
    mode='max',
    verbose=1
)
hist = model.fit([X_ela_train, X_srm_train],
                    Y_train,
                    batch_size=batch_size,
                    class_weight=class_weights,
```

```
                    epochs=epochs,
                    validation_data=([X_ela_val, X_srm_val], Y_val),
                    callbacks=[early_stopping, checkpoint_callback])
    #early stopping implemented to prevent overfitting
model = load_model('finalmodelcasiatest5.h5')
history=hist.history
history_dict = hist.history
with open('finalmodelcasiatest5.pkl', 'wb') as file_pi:
    pickle.dump(history_dict, file_pi)
predictions = model.predict([X_ela_val, X_srm_val])
y_pred = np.argmax(predictions, axis=1)
y_true = np.argmax(Y_val, axis=1)
def generate_metrics(y_true, y_pred):
    print("Accuracy = " , accuracy_score(y_true, y_pred))
    print("Precision = " ,precision_score(y_true, y_pred))
    print("Recall = " ,recall_score(y_true, y_pred))
    print("F1 Score = " ,f1_score(y_true, y_pred))
    pass# Make predictions

generate_metrics(y_true, y_pred)

def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]

    cm_flipped = cm[::-1, ::-1]

    plt.imshow(cm_flipped, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes[::-1], rotation=45)
    plt.yticks(tick_marks, classes[::-1])

    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm_flipped.shape[0]),
range(cm_flipped.shape[1])):
        plt.text(j, i, cm_flipped[i, j],
                 horizontalalignment="center",
                 color="white" if cm_flipped[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('Predicted label')
    plt.xlabel('Actual label')
```

```
Y_pred = model.predict([X_ela_val, X_srm_val])
Y_pred_classes = np.argmax(Y_pred, axis=1)
Y_true = np.argmax(Y_val, axis=1)
confusion_mtx = confusion_matrix(Y_true, Y_pred_classes)
class_names = ['fake','real']
plot_confusion_matrix(confusion_mtx, classes=class_names, title='Confusion
Matrix, Image Forgery Detection')

fig, ax = plt.subplots(2, 1)
#code for visulating training and validation loss and accuracy history
ax[0].plot(hist.history['loss'], color='b', label="Training loss")
ax[0].plot(hist.history['val_loss'], color='r', label="Validation loss")
ax[0].set_title('Training and Validation Loss')
ax[0].set_xlabel('Epoch')
ax[0].set_ylabel('Loss')
ax[0].legend(loc='best', shadow=True)

ax[1].plot(hist.history['accuracy'], color='b', label="Training accuracy")
ax[1].plot(hist.history['val_accuracy'], color='r', label="Validation
accuracy")
ax[1].set_title('Training and Validation Accuracy')
ax[1].set_xlabel('Epoch')
ax[1].set_ylabel('Accuracy')
ax[1].legend(loc='best', shadow=True)
plt.tight_layout()
plt.show()
```

Executive Summary

IMAGE FORGERY DETECTION AND AUTHENTICATION

Karim Tarek (195322)

Dr. Motasem Mahmoud Elshourbagy

Faculty of Engineering, Mechanical Department, The British University in Egypt, El Sherouk City, Misr-Ismalia Road, Cairo, Egypt

Emails: karim195322@bue.edu.eg & motasem.elshourbagy@bue.edu.eg

ABSTRACT

In the modern digital age, the rise of image manipulation tools raises significant concerns about the authenticity of digital images. The focus of this research is developing and evaluating a Convolutional Neural Network (CNN) based architecture that utilises Error Level Analysis (ELA) and Spatial Rich Model (SRM) filtering for image forgery detection.

The proposed dual-branch CNN model utilises images processed through ELA and SRM filters, enhancing the neural network's ability to identify subtle discrepancies indicative of manipulation in image data. Extensive experimentation was conducted on the most prominent datasets in image forgery detection field, including CASIA V2, CoMoFoD (small) and MICC-F2000.

Evaluation metrics such as accuracy, precision, recall and F1 score were used to benchmark the performance of the proposed model on the datasets mentioned. Notably, the model achieved near state-of-art-results with an accuracy of 98.00% on the MICC-F2000 dataset and the approach exhibits superior generalization capability across different types of image forgery and outperforming traditional methods. The findings highlight the potential of integrating ELA and SRM filtering in deep learning frameworks to enhance image forgery detection and establish a solid methodology for future advancements in the field

1. Problem Definition

Image editing tools have become widespread and easily available for use, the authenticity of digital images is increasingly under threat. The decrease of confidence in the authenticity of digital images leads to misinformation and diminishes trust in digital media. Therefore, there is a critical need for advanced algorithms that reliably detects various types of image forgery to ensure the integrity of digital images. These algorithms are essential for purposes such as verification of the authenticity of images in a court setting.

2. Objectives

1. To develop an innovative method for image forgery detection using deep learning frameworks
2. Find the most appropriate preprocessing of images to improve the image forgery detection capabilities of deep learning methods.
3. Evaluate the performance of the developed algorithm across multiple prominent datasets, ensuring generalization capabilities to novel images
4. The developed algorithm should be able to achieve near state-of-the-art accuracy in detecting forgeries on the CASIA V2, CoMoFoD (small), and MICC-F2000 datasets.
5. Compare the proposed algorithm's performance on the on CASIA V2, CoMoFoD (small), and MICC-F2000 datasets with the existing state-of-the-art methods.
6. To provide insights and recommendations on future work that can done based on the developed algorithms.

3. Brief Introduction

3.1. Background

The ease and spread of image manipulation software has raised concerns about the authenticity of images that are exchanged. The concerns around the authenticity of digital images have resulted in the emergence of the field of digital image forgery. The creation of reliable digital image forgery detection and authentication algorithms improves the quality of life and dependability for workers of jobs that rely on information embedded in digital images such as lawyers, judges, law enforcement and health professionals.

Image Forgery detection approaches are classified into two primary approaches: active and passive approaches. Active image forgery detection relies on the pre-embedded information in the image, such as digital signatures or watermarks, which can be checked for the authenticity of the image. Digital signatures are embedded during the capture or processing stages of the image and are verified for any corruption during detection. Digital watermarking, often invisible, is altered if the image is manipulated, and any discrepancy during watermark extraction indicates forgery (Gill et al., 2017)

3.2. Traditional Image forgery detection

traditional image forgery techniques are the techniques that analyse inconsistencies in the physical and geometrics properties of an image these methods include:

- Pixel-Based Techniques: Techniques such as Error Level Analysis (ELA) emphasize the compression levels that suggest image manipulation (Krawetz, 2007).
- Camera-Based Techniques: These techniques identify image forgery by analysing the meta of the image or the camera noise patterns of the image
- Geometric-Based Methods: Techniques that rely on the geometric qualities of an image to detect inconsistencies such as detecting inconsistencies in the lighting, shadows, or the perspective of an image.

3.2 Deep Learning-Based Image forgery detection

Deep learning methods, especially Convolutional Neural Networks (CNNs), has proven to be superior at image forgery detection than traditional methods, CNNs can process visual data and learn to recognize patterns indicative of image manipulation.

Support Vector Machines (SVMs) have been also utilised in this field but CNNs have been proven superior in their image forgery detection capabilities.

4. Image forgery types

1. Copy-move: A region of an image is copied and pasted onto another region in the same image.
2. Splicing: a region of the image is copied and pasted onto a region in a different image
3. Image retouching: Retouching is subtly enhancing certain features in an image mainly for aesthetic appeal. This type of forgery is usually harmless

5. Summary of Previous Work

(Gupta et al., 2022) and (Sudiatmika et al., 2019)) highlighted the effectiveness of using ELA to preprocesses image to highlight inconsistencies indicative of image forgery detection. Thus, enhancing CNN's ability to detect forgery by emphasizing compression discrepancies in images.

(Joshi et al., 2022) demonstrated that preprocessing images using ELA and using pre-trained models as a feature extractor was an effective method of forgery detection. VGG-19 was the pre-trained model that produced the best performance metrics.

(Martin-Rodriguez et al., 2023) utilised pixel-wise feature extraction using ELA and Photo-Response Non-Uniformity (PRNU) for image preprocessing to differentiate artificial intelligence generated images from real ones achieving up to 99% accuracy.

(Mehrjardi et al., 2023) authored a very thorough survey of image forgery detection methods. Their work highlighted the advancements in the field and the various methods of image forgery detection

(Chakraborty et al., 2024) proposed an innovative method utilising a dual-branch CNN using ELA and SRM filtering to preprocess the input images. Achieving an impressive accuracy of 98.55%. The author's method is effective and efficient in detecting various types of image forgeries.

6. Research Methodology

6.1.Dataset choice

For training and evaluating the deep learning model, three datasets were selected: CASIA V2, CoMoFoD (Small), and MICC-F2000.

Figure 1: Example of an image processed with ELA (original left ELA right)

CASIA V2: This dataset includes 12,614 images, of which there are 7491 authentic images, 5123 manipulated images. To utilise ELA to detect compression artifacts the JPEG format is crucial. There are over 9000 JPEG images in this dataset making it a good fit for the requirements of the model. Contains Copy-Move and Splicing forgeries.

CoMoFoD (Small): exclusively focused on Copy-Move forgeries. 10000 JPEG images, 5000 authentic images and 5000 forged images. Images are forged through various forms of transformations such as blurring, rotation and scaling, excellent choice for generalization.

MICC-F2000: includes high-resolution images (up to 2048x1536), MICC-F2000 has various transformations like scaling and rotation, it is designed to simulate real-world forgeries. Post-processing blurs boundaries to challenge the model in detecting subtle manipulations. High resolution images are effective for feature extraction

6.2. Error Level Analysis (ELA)

ELA detects subtle signs of image forgery by analysing compression artifacts across an image. ELA highlights regions with unusual compression levels. ELA involves saving an image at a lower quality, 90% was used my method. Then finding the difference between the saved image and the original image. ELA images have details that indicate may manipulation.

6.3.SRM (Spatial Rich Model) Filters

An image is filtered through 30 high SRM high pass proposed by (Fridrich & Kodovský, 2012). This process extracts

subtle inconsistencies in the data that are not visible in the original image. The result is the noise residuals that contain isolated noise patterns of the image that may indicate manipulation.

Figure 2: An example of an image processed with SRM (original left SRM right)

6.4.Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) are neural networks that contain at least one convolution layer. They are the most used type of neural networks for extracting features from visual data such as images. They were first introduced by LeCunn et. al. (1989) and CNNs are the state of the art when it comes to visual data extraction.

My proposed method's architecture contains each of the following:

- A Convolution layer: this layer Applies filters or kernels to the input image, to allow the network to extract features like edges, patterns, and textures to classify images correctly.
- A Pooling layer: Reduces the spatial dimensions of the input to decrease computational complexity and prevent overfitting while preserving important feature data.
- A Fully connected layer: this Processes the features that are passed from the convolution and pooling layers to be able to decide the final class of the data.
- Softmax layer: Typically found at the end of the network, this layer outputs a 1 for real images or a 0 for forged images.
- Dropout layer: this layer implements a regularization technique to reduce overfitting and improve the generalization capabilities of the model.

6.5.Proposed Model Architecture

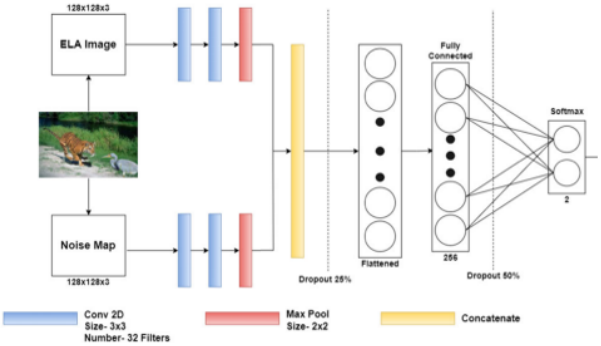


Figure 3: The architecture of the dual CNN utilizing ELA and SRM preprocessed images

The proposed CNN architecture is a dual branch CNN architecture, a branch receives a ELA analysis image resized to 128x128 pixels resolution, the other branch recieves an image filtered with 30 high pass SRM filters and resized to 128x128 pixels resolution. The images are processed by 2 convolution layers with 32 filters each with a kernel size of 3x3, then the images go through a max pooling layer to reduce the spatial dimensions of the image.

The output is concatenated and a dropout layer with a dropout rate of 0.25 follow the max pooling layer. The output then is flattened to then passes through a dense layer with 256 units using ReLU activation to learn complex patterns. The final layer is a 2-unit dense layer with a SoftMax function that outputs 1 if the image is real or 0 if the image is forged. The model was trained and evaluated using python with the TensorFlow library and optimized with class weights to address class balances,

7. Experimental Work and Main Results

Metric	Dataset	ELA+SRM Accuracy (%)	Improvement (ELA+SRM - ELA) (%)
Accuracy	CASIA V2	96.18	1.59
Precision	CASIA V2	96.92	-0.09
Recall	CASIA V2	98.23	2.29
F1 Score	CASIA V2	97.57	1.1
Accuracy	CoMoFod (small)	89.25	0.5
Precision	CoMoFod (small)	88.52	0.37
Recall	CoMoFod (small)	90.2	0.23
F1 Score	CoMoFod (small)	89.35	0.3
Accuracy	MICC-F2000	98	0.75
Precision	MICC-F2000	94.59	-5.41
Recall	MICC-F2000	100	4.38
F1 Score	MICC-F2000	97.22	-0.54

Table 1: Evaluating the improvement from the dual branch implementation

The model was trained and evaluated on three different prominent datasets with an 80-20 train-test split, the results are show

in Table 1. The state-of-the-art results for CASIA V2 accuracy according to (Zanardelli et al., 2022), is 99.07%. The proposed model achieved around 96.18% accuracy which is almost state of the art and within the margin of error of the state-of-the-art. the model could achieve higher accuracy, but time and computational power were limited. As for the MICC-F2000 dataset the model outperformed the state-of-the-art metrics mentioned by (Koul et al., 2022) making this approach the state of the art for the MICC-F2000 image forgery detection dataset.

Table 1 also showcases the improvement of the addition of SRM filtering to ELA for image preprocessing. There are improvements to metrics all around except for precision on the MICC-F2000 dataset which can be explained by the small size of the test data (200). Thus, there is a large margin of error for metrics MICC-F2000 as it is a small dataset

8. Conclusion

integrating ELA and SRM filters within a deep learning framework significantly enhances the model's capability to detect image forgeries in comparison with the use of ELA exclusively. The processing of the image by both ELA and SRM filters the compression discrepancies are amplified and can then be easily identified by the dual branch CNN. Moreover, this approach does not only capture more detail but also it improves performance on novel images due to picking up more detail indicative of image manipulation in the hidden data of the noise residual image.

Extensive experimentation was conducted on prominent datasets, including CASIA V2, CoMoFoD (small), and MICC-F2000. These experiments concluded the efficacy of the proposed model as the performance of the dual branch CNN model achieves near state-of-the-art performance with accuracy rates reaching up to 98.00% on the MICC-F2000 dataset and 96.18% on the CASIA V2 dataset. This is particularly significant because it highlights how underutilised the combination of SRM and ELA in deep learning frameworks.

9. Future work

The methodologies and insights presented in this paper will be instrumental in maintaining the originality of images in the digital world in the future.

Recommendations for Future Work:

1. Research the use of a pre-trained model with ImageNet weights or YOLO weights as a feature extractor from the ELA and SRM images, this can greatly enhance the model's accuracy and generalization ability. This could not be researched in this work due to computational limitations and

lack of funding for cloud computing.

2. Localization and description: I aimed to utilize ELA and SRM images to develop models that are capable of localization and description of image forgeries in natural language utilizing transformers. The features extracted from ELA and SRM enhance the learnability of localization greatly for deep learning models. This was not possible in this research due to
3. Exploration of other preprocessing techniques: ELA and SRM are a powerful combination to extract patterns that are indicative of image manipulation, the efficacy of other preprocessing techniques combined with these two techniques must be studied to establish if there are any improvements with another technique added to this combination.
4. Generalization: Testing the image forgery detection algorithm on large real-world images was not possible due to a lack of computational power. This should be done in the future with large datasets for training and testing Such as the ImageNet dataset which contains over a million images.

References

Chakraborty, S., Chatterjee, K., & Dey, P. (2024). Detection of Image Tampering Using Deep Learning, Error Levels and Noise Residuals. *Neural Processing Letters*, *56*(2). https://doi.org/10.1007/s11063-024-11448-9

Fridrich, J., & Kodovský, J. (2012). *Rich Models for Steganalysis of Digital Images*.

Gill, N. K., Garg, R., & Doegar, E. A. (2017, December 13). A review paper on digital image forgery detection techniques. *8th International Conference on Computing, Communications and Networking Technologies, ICCCNT 2017*. https://doi.org/10.1109/ICCCNT.2017.8203904

Gupta, A., Joshi, R., & Laban, R. (2022). *Detection of Tool based Edited Images from Error Level Analysis and Convolutional Neural Network*. http://arxiv.org/abs/2204.09075

Koul, S., Kumar, M., Khurana, S. S., Mushtaq, F., & Kumar, K. (2022). An efficient approach for copy-move image forgery detection using convolution neural network. *Multimedia Tools and Applications*, *81*(8), 11259–11277. https://doi.org/10.1007/s11042-022-11974-5

Krawetz, N. (2007). *A Picture's Worth... Digital Image Analysis and Forensics*. www.hackerfactor.com

Martin-Rodriguez, F., Garcia-Mojon, R., & Fernandez-Barciela, M. (2023). Detection of AI-Created Images Using Pixel-Wise Feature Extraction and Convolutional Neural Networks. *Sensors*, *23*(22). https://doi.org/10.3390/s23229037

Mehrjardi, F. Z., Latif, A. M., Zarchi, M. S., & Sheikhpour, R. (2023). A survey on deep learning-based image forgery detection. *Pattern Recognition*, *144*. https://doi.org/10.1016/j.patcog.2023.109778

Sudiatmika, I. B. K., Rahman, F., Trisno, & Suyoto. (2019). Image forgery detection using error level analysis and deep learning. *Telkomnika (Telecommunication Computing Electronics and Control)*, *17*(2), 653–659. https://doi.org/10.12928/TELKOMNIKA.V17I2.8976

Zanardelli, M., Guerrini, F., Leonardi, R., & Adami, N. (2022). Image forgery detection: a survey of recent deep-learning approaches. *Multimedia Tools and Applications*, *82*, 17521–17566. https://doi.org/10.1007/s11042-022-13797-w

Appendix A: Code

Preprocessing (CASIA V2 example)

```
import cv2
import numpy as np
from PIL import Image, ImageChops, ImageEnhance
import random
import pickle
import os

# Function to convert an image to an ELA image
def convert_to_ela_image(path, quality=90):
    temp_filename = 'temp_file_name.jpg'

    image = Image.open(path).convert('RGB')
    image.save(temp_filename, 'JPEG', quality=quality)
    temp_image = Image.open(temp_filename)

    ela_image = ImageChops.difference(image, temp_image)

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

    ela_image = ImageEnhance.Brightness(ela_image).enhance(scale)

    return ela_image
```

```
# Function to apply SRM filters using loaded weights
def apply_srm_filters(image, srm_weights):
    steps = srm_weights.shape[-1]  # Number of filters
    filtered_channels = []

    for step in range(steps):
        filtered_channels_step = []
        for channel in range(3):  # Apply each filter to each color channel
            filter_kernel = srm_weights[:, :, 0, step]
            filtered_image = cv2.filter2D(image[:, :, channel], -1,
filter_kernel)
            filtered_channels_step.append(filtered_image)

        combined_residual = np.stack(filtered_channels_step, axis=-1)
        filtered_channels.append(combined_residual)

    combined_residual = np.mean(filtered_channels, axis=0)  # Average over all
filters
    combined_residual = np.clip(combined_residual, -3, 3)  # Clipping

    # Normalize to the range [0, 255]
    combined_residual = ((combined_residual - combined_residual.min()) /
                         (combined_residual.max() - combined_residual.min()))
* 255.0

    return combined_residual.astype(np.uint8)

# Prepare image for ELA
def prepare_ela_image(image_path, image_size=(128, 128)):
    ela_image = convert_to_ela_image(image_path, 90).resize(image_size)
    return np.array(ela_image).flatten() / 255.0
```

```
# Prepare image for SRM using loaded weights (apply SRM before resizing)
def prepare_srm_image(image_path, srm_weights, image_size=(128, 128)):
    image = cv2.imread(image_path, cv2.IMREAD_COLOR)

    # Apply SRM filters before resizing
    combined_residual = apply_srm_filters(image, srm_weights)

    # Resize the SRM filtered image
    resized_residual = cv2.resize(combined_residual, image_size)

    return resized_residual.flatten() / 255.0

# Preprocess images with both ELA and SRM
def preprocess_images(base_path, label, srm_weights, image_size=(128, 128),
max_images=None):
    X_ela = []
    X_srm = []
    Y = []
    image_count = 0
    file_log = []

    for dirname, _, filenames in os.walk(base_path):
        filenames.sort()
        for filename in filenames:
            if filename.endswith(('jpg', 'png', 'jpeg')):
                full_path = os.path.join(dirname, filename)
                X_ela.append(prepare_ela_image(full_path,
image_size=image_size))
                X_srm.append(prepare_srm_image(full_path, srm_weights,
image_size=image_size))
                Y.append(label)
```

```
                file_log.append(full_path)
                image_count += 1
                if len(Y) % 500 == 0:
                    print(f'Processing {len(Y)} images')
                if max_images and image_count >= max_images:
                    return X_ela, X_srm, Y, file_log
    return X_ela, X_srm, Y, file_log


# Save preprocessed data
def save_preprocessed_data(X_ela, X_srm, Y, file_log,
filename='CASIAV2DATA.pkl'):
    with open(filename, 'wb') as f:
        pickle.dump((X_ela, X_srm, Y, file_log), f)


def main():
    image_size = (128, 128)


    real_path = 'C:\\Users\\Karim Tarek\\Downloads\\archive\\CASIA2\\au' #real
path of any dataset is insterted here CASIA V2 in inserted as an example
    fake_path = 'C:\\Users\\Karim Tarek\\Downloads\\archive\\CASIA2\\Tp' #fake
path of any dataset is insterted here CASIA V2 in inserted as an example


    # Load SRM weights
    srm_weights = np.load('SRM_Kernels.npy')


    # Normalize each filter by its maximum absolute value
    for i in range(srm_weights.shape[-1]):
        max_value = np.max(np.abs(srm_weights[:, :, 0, i]))
        srm_weights[:, :, 0, i] /= max_value


    X_real_ela, X_real_srm, Y_real, log_real = preprocess_images(real_path, 1,
srm_weights, image_size=image_size)
```

```
    X_fake_ela, X_fake_srm, Y_fake, log_fake = preprocess_images(fake_path, 0,
srm_weights, image_size=image_size)


    X_ela = X_real_ela + X_fake_ela

    X_srm = X_real_srm + X_fake_srm

    Y = Y_real + Y_fake

    log = log_real + log_fake


    combined = list(zip(X_ela, X_srm, Y, log))

    random.shuffle(combined)

    X_ela[:], X_srm[:], Y[:], log[:] = zip(*combined)


    print(f'Total images processed: {len(X_ela)}')


    save_preprocessed_data(X_ela, X_srm, Y, log, filename='CASIAV2DATA.pkl')


if __name__ == '__main__':

    main()
```

Model Training and evaluation (CASIA V2 example)

```
import numpy as np

import matplotlib.pyplot as plt

np.random.seed(2)

from sklearn.model_selection import train_test_split

from sklearn.metrics import confusion_matrix

from keras.utils import to_categorical

from keras.callbacks import EarlyStopping, ModelCheckpoint

from keras import layers, models

import tensorflow as tf

import pickle
```

```
# from keras.callbacks import ModelCheckpoint
from CoMoFodpreprocessing import save_preprocessed_data
import os
import itertools
from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_score
from keras.models import load_model
import time
from tqdm import tqdm
import miccpreprocessing as pp
from sklearn.model_selection import StratifiedShuffleSplit
import json
gpus = tf.config.experimental.list_physical_devices('GPU')
if gpus:
    try:
        # memory growth needs to be the same across GPUs
        for gpu in gpus:
            tf.config.experimental.set_memory_growth(gpu, True)
        logical_gpus = tf.config.experimental.list_logical_devices('GPU')
        print(f"{len(gpus)} Physical GPUs, {len(logical_gpus)} Logical GPUs")
    except RuntimeError as e:
        # Memory growth must be set before GPUs have been initialized
        print(e)
        os.environ['CUDA_VISIBLE_DEVICES'] = ''
def load_preprocessed_data(filename):
    with open(filename, 'rb') as f:
        X_ela, X_srm, Y, log = pickle.load(f)
    return np.array(X_ela), np.array(X_srm), np.array(Y), log
X_ela, X_srm, Y, log = load_preprocessed_data('CASIAV2DATA.pkl') #CASIA V2
DATASET LOCADED YOU CAN LOADE THE DATASET YOU WANT TO USE


X_ela = X_ela.reshape(-1, 128, 128, 3)
```

```
X_srm = X_srm.reshape(-1, 128, 128, 3)


# One-hot encode the labels
Y = to_categorical(Y, 2)


# Stratified split to ensure same class ratio in train and validation sets
sss = StratifiedShuffleSplit(n_splits=1, test_size=0.1, random_state=5)


for train_index, val_index in sss.split(X_ela, np.argmax(Y, axis=1)):
    X_ela_train, X_ela_val = X_ela[train_index], X_ela[val_index]
    X_srm_train, X_srm_val = X_srm[train_index], X_srm[val_index]
    Y_train, Y_val = Y[train_index], Y[val_index]


# Print the number of samples
print(f'Training samples: {len(X_ela_train)}, Validation samples:
{len(X_ela_val)}')


# Checking the distribution of classes in the training and validation sets
print('Class distribution in training set:', np.bincount(np.argmax(Y_train,
axis=1)))
print('Class distribution in validation set:', np.bincount(np.argmax(Y_val,
axis=1)))
def build_model():
    # ELA Branch
    input_ela = layers.Input(shape=(128,128,3), name='ELA_Input')
    x_ela = layers.Conv2D(32, (3, 3), activation='relu',
padding='same')(input_ela)
    x_ela = layers.Conv2D(32, (3, 3), activation='relu',
padding='same')(x_ela)
    x_ela = layers.MaxPooling2D((2, 2))(x_ela)


    # SRM Branch
    input_srm = layers.Input(shape=(128,128,3), name='SRM_Input')
```

```
    x_srm = layers.Conv2D(32, (3, 3), activation='relu',
padding='same')(input_srm)
    x_srm = layers.Conv2D(32, (3, 3), activation='relu',
padding='same')(x_srm)
    x_srm = layers.MaxPooling2D((2, 2))(x_srm)

    # Concatenate the outputs of both branches
    combined = layers.Concatenate()([x_ela, x_srm])
    combined = layers.Dropout(0.25)(combined)
    combined = layers.Flatten()(combined)

    # Fully connected layer
    fc = layers.Dense(256, activation='relu')(combined)
    fc = layers.Dropout(0.5)(fc)

    output = layers.Dense(2, activation='softmax')(fc)

    # Create the model
    model = models.Model(inputs=[input_ela, input_srm], outputs=output)
    return model

model = build_model()
model.summary()

epochs = 200
batch_size = 16

optimizer = optimizer = tf.keras.optimizers.Adam(
    learning_rate=0.001,
    beta_1=0.9,
    beta_2=0.999,
    epsilon=1e-08,
```

```
    decay=0.0001
)

model.compile(optimizer=optimizer, loss='binary_crossentropy',
metrics=['accuracy'])

early_stopping = EarlyStopping(monitor = 'val_accuracy',
                               min_delta = 0,
                               patience = 7,
                               verbose = 1,
                               mode = 'auto')

no_au = 7354
no_tp = 2064
total_images = no_au + no_tp

weight_for_au = (1 / no_au) * (total_images / 2.0)
weight_for_tp= (1 / no_tp) * (total_images / 2.0)
class_weights = {0: weight_for_tp, 1: weight_for_au}
checkpoint_filepath = 'finalmodelcasiatest5.h5'
checkpoint_callback = ModelCheckpoint(
    filepath=checkpoint_filepath,
    monitor='val_accuracy',
    save_best_only=True,            # Save only the best weights
    mode='max',
    verbose=1
)
hist = model.fit([X_ela_train, X_srm_train],
                    Y_train,
                    batch_size=batch_size,
                    class_weight=class_weights,
```

```
                    epochs=epochs,
                    validation_data=([X_ela_val, X_srm_val], Y_val),
                    callbacks=[early_stopping, checkpoint_callback])
    #early stopping implemented to prevent overfitting
model = load_model('finalmodelcasiatest5.h5')
history=hist.history
history_dict = hist.history
with open('finalmodelcasiatest5.pkl', 'wb') as file_pi:
    pickle.dump(history_dict, file_pi)
predictions = model.predict([X_ela_val, X_srm_val])
y_pred = np.argmax(predictions, axis=1)
y_true = np.argmax(Y_val, axis=1)
def generate_metrics(y_true, y_pred):
    print("Accuracy = " , accuracy_score(y_true, y_pred))
    print("Precision = " ,precision_score(y_true, y_pred))
    print("Recall = " ,recall_score(y_true, y_pred))
    print("F1 Score = " ,f1_score(y_true, y_pred))
    pass# Make predictions

generate_metrics(y_true, y_pred)

def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]

    cm_flipped = cm[::-1, ::-1]

    plt.imshow(cm_flipped, interpolation='nearest', cmap=cmap)
```

```
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes[::-1], rotation=45)
    plt.yticks(tick_marks, classes[::-1])

    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm_flipped.shape[0]),
range(cm_flipped.shape[1])):
        plt.text(j, i, cm_flipped[i, j],
                 horizontalalignment="center",
                 color="white" if cm_flipped[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('Predicted label')
    plt.xlabel('Actual label')

Y_pred = model.predict([X_ela_val, X_srm_val])
Y_pred_classes = np.argmax(Y_pred, axis=1)
Y_true = np.argmax(Y_val, axis=1)
confusion_mtx = confusion_matrix(Y_true, Y_pred_classes)
class_names = ['fake','real']
plot_confusion_matrix(confusion_mtx, classes=class_names, title='Confusion
Matrix, Image Forgery Detection')

fig, ax = plt.subplots(2, 1)
#code for visulating training and validation loss and accuracy history
ax[0].plot(hist.history['loss'], color='b', label="Training loss")
ax[0].plot(hist.history['val_loss'], color='r', label="Validation loss")
ax[0].set_title('Training and Validation Loss')
ax[0].set_xlabel('Epoch')
ax[0].set_ylabel('Loss')
```

```
ax[0].legend(loc='best', shadow=True)


ax[1].plot(hist.history['accuracy'], color='b', label="Training accuracy")
ax[1].plot(hist.history['val_accuracy'], color='r', label="Validation
accuracy")
ax[1].set_title('Training and Validation Accuracy')
ax[1].set_xlabel('Epoch')
ax[1].set_ylabel('Accuracy')
ax[1].legend(loc='best', shadow=True)
plt.tight_layout()
plt.show()
```