Master's thesis

Malene Vassenden

Copy-Move and Splicing Forgery Detection using Deep Learning

Image Forgery Detection

Master's thesis in Information Security
Supervisor: Lasse Øverlier
Co-supervisor: An Thi Nguyen
June 2022

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Dept. of Information Security and Communication Technology

NTNU
Norwegian University of Science and Technology

Malene Vassenden

Copy-Move and Splicing Forgery Detection using Deep Learning

Image Forgery Detection

Master's thesis in Information Security
Supervisor: Lasse Øverlier
Co-supervisor: An Thi Nguyen
June 2022

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Dept. of Information Security and Communication Technology

NTNU
Norwegian University of
Science and Technology

Copy-Move and Splicing Forgery Detection using Deep Learning

Malene Vassenden

CC-BY 2022/06/01

Abstract

In a world with rapid change in technology and the use of social media (SM), a growing attack surface with a fast distribution of fake images to a vast audience is an increasing problem. The two most used image tampering techniques are copy-move forgery (CMF), where copied objects are from within the same image, and splicing forgery (SF) where the copied objects originate from another image. Many attempts to develop deep learning models for image tampering detection have been proposed in the literature. However, many of them have not considered the possibility that tampered images can be additionally post-processing in image editing programs and be compressed when transferred on social media (SM). This thesis explores the impact of using test images post-processed with rotation, JPEG compression, blurring, and brightening. We test the impact on a convolutional neural network (CNN), and the transfer learning models ResNet-50 and ResNet-101, and error level analysis (ELA) is used as a pre-processor to highlight tampering regions. We propose an improved ResNet-50 model with an F1 score of 98% when presented with images not post-processed. We address the performance drop in experiments with post-processed images, and we address that using ResNet-101, a deeper model than ResNet-50, does not result in increased performance. Another variable that caused the performance drop was ELA, which depends on analyzing images that have not lost data due to additional JPEG compression. This research can conclude that ELA is not a suitable post-processing technique to rely on when investigating images that may be post-processed or transferred online. The depth and complexity of deep learning models have an impact on the accuracy and performance. Good detection accuracy on testing images, similar to training images, does not equal good detection accuracy on new types of image transformations.

Sammendrag

I en verden som endrer seg i stor hastighet innen teknologi, og bruk av sosiale medier, en ny angrepsflate hvor distribuering av forfalskede bilder kan spre seg til et stort publikum på bare sekunder. De to mest brukte teknikkene for bildemanipulering er copy-move forgery (CMF) hvor kopierte objkekter kommer fra det samme bildet, og splicing forgery (SF) hvor kopierte objekter er fra andre bilder. Mange forsøk på å utvikle dype maskinlæringsmodeller for deteksjon av manipulerte bilder har vært presentert i litteraturen. Det som ikke addresseres i mange av disse modellene er mulighetene for at bilder kan bli retusjert og komprimert under overføring på sosiale medier. Denne oppgaven utforsker utforsker effekten av testing med manipulerte bilder som har blitt ytterligere prosessert med rotasjon, JPEG kompresjon, blur og lysendring. Vi testet effecten på et convolutional neural network (CNN), og de to *transfer learning* modellene ResNet-50 og ResNet-101, og vi brukte error level analysis (ELA) som en preprosessor til å lyse opp manipulerte objekter i bildene. Vi foreslår en ResNet-50 model som oppnådde en F1 skår på 98% på bilder uten ytterligere prossesering. Vi diskuterer ytelsesnedgangen i eksperimentene hvor bildene har vært ytterligere prossesert, og vi diskuterer at bruk av ResNet-101 som er et dypere nettverk sammenlikned med ResNet-50 ikke fører til bedre ytelse. En annen variabel som kan ha forårsaket ytelsesnedgangen er ELA som ikke egner seg som preprosesseringsteknikk på bilder som er blitt komprimert med JPEG. Dette prosjektet konkluderer at ELA ikke er egnet som preprosesseringsteknikk når bilder som kan ha blitt komprimert under overførelse på sosiale medier. Dybden og kompleksiteten i en dyp læringsmodell spiller også inn på treffsikkerheten og ytelse. Bra deteksjonstreffsikkerhet på testbilder som er lik til treningsbilder vil ikke automatisk bety at modellen er robust for nye typer transformasjoner.

Contents

Figures

Tables

Acronyms

AUC Area Under the receiver operating characteristic Curve. 7, 11, 43

CM copy-move. 3, 4, 18, 47

CMF copy-move forgery. iii, v, xi, 1–3, 5, 10, 11, 15, 17, 42–44, 47

CNN convolutional neural network. iii, v, 3, 7, 8, 11, 15, 47, 48

DCT Discrete Cosine Transform. 5, 14, 48

DNN deep neural network. 13

DWT Discrete Wavelet Transform. 5, 48

ELA error level analysis. iii, v, 10, 14, 15, 19, 22, 33, 37–41, 44, 47, 48

FN false negative. 26, 40

FP false positive. 26, 40

GAP global average pooling. 24, 25

JPEG Joint Photographic Experts Group. iii, v, 11, 14, 15, 19, 22, 37–42, 47

ML machine learning. 7–10, 41

SF splicing forgery. iii, v, xi, 1–4, 10, 11, 15, 17, 18, 42–44, 47

SGD Stochastic Gradient Descent. 9, 24

SM social media. iii, 1, 2, 47

SVM Support Vector Machine. 8, 10, 11

TF transfer learning. 11

TN true negative. 26

TP true positive. 26

Chapter 1

Introduction

In a digital age where thoughts, opinions, and life updates are shared with friends, family, and the public on social media (SM) platforms, the possibility of reaching a massive audience is just a click away. Not only is it possible to broadcast content on different platforms, as long as terms of services are withheld, but the content may not be checked for its integrity by a third party before it is published, so it depends on people to be able to distinguish real and fake images and news articles [1]. People's knowledge of distinguishing between real and altered truth may also vary because of training and source criticism. These skills have become more critical because there is a growing tendency for people to access news mainly on SM sites. Studies on how teenagers in Norway consume news reveals that they get access to news mainly from social media sites. As much as 63% of teenagers of age 9-18 years state that they get access to news mainly on Snapchat, Facebook, and YouTube, and 88% of Teenagers of age 17-18 years say that they also read online news articles from established newspapers [2]. Using images or fake images in distributing untruthful information can amplify the message's visual effects. Humans tend to be better at recalling articles and news if they contain images than remembering content in pure text articles [3].
A fake image can be defined as a photograph that has been altered with an image editing program that changes the message in the image. Over the last couple of years, image tampering techniques have increased in popularity, and it is used to transform an image from representing its actual content to represent a modified version of its true self. More sophisticated scams may also utilize more advanced image manipulation techniques, such as color correction, blurring lines, and smoothing the tampered area to make it harder for the human eye to detect the scam. Image tampering can be done with widely available editing tools such as Paint.NET [4], GIMP [5] and Adobe Photoshop [6].

copy-move forgery (CMF) is one of the most used and most researched image tampering techniques in the research field. It is the art of copying objects or regions from the same image and pasting it to another location in the image. Another popular image tampering technique is splicing forgery (SF), where objects from other images are pasted in the image. A motivation for using copied areas from the same image is because it may appear more natural. After all, colors and objects may appear more valid than the image's composition and context [7]. The copied object can be used to hide elements in an image or cover up evidence which can give deceptive information.

One example of CMF is shown in figure Figure 1.1. The Iranian military published the tampered image of a successful missile launching, but the original image reveals that one of the missiles never left the ground [8]. Image tampering has also been done in academic research. For example, Tijdink et al. discovered that 15% of their survey responders reported that they had tampered with their research results within the last three years. Image manipulation can be one of the methods they use to manipulate research [9].

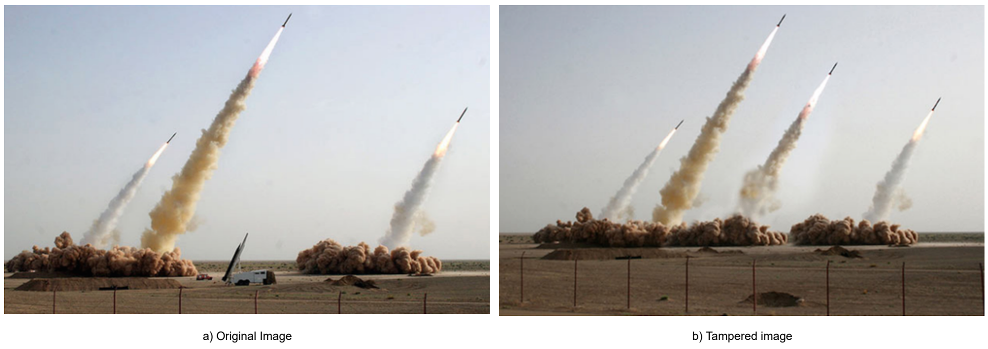


Figure 1.1: Iranian missile launching [8]

1.1 Problem Description

The use of SM platforms in people's everyday life has increased the importance of being able to distinguish between tampered and legitimate images. Distributors of tampered images can reach a greater audience in a short amount of time. As earlier described, many people get access to news through SM platforms. Therefore, people must distinguish between news from established media channels and shared opinions that can contain tampered images to affect people's beliefs, which have been done in political elections [10]. Facebook and Twitter are also two of the most popular sites used to distribute falsified information [1]. SM sites often use lossy compression on images to eliminate problems with bandwidth and give users faster uploading of web pages [11]. This means that images that have been uploaded and downloaded can be exposed to compression multiple times, which may lead to loss of information. Another aspect of images that have been manipulated with CMF or SF is that the distributor of the image may try to retouch the image to make it look as legitimate as possible, which means that images can be exposed to multiple manipulations such as blurring, color correction, and scaling.

1.2 Justification and Motivation

Many authors have proposed comprehensive models that detect CM and SF, and the models may seem appropriate to use in a forensics examination. However, many models are trained and tested on images from the same dataset which may indicate that the test images have been post-processed similarly to the training images. The creators of the CASIA datasets [12] have tried to create tampered images as close to real-world images as possible, but that does not mean that models that have been trained and tested on their datasets are suitable for real-world CM and SF image detection tasks. Some convolutional neural network (CNN) models are trained with augmented data to increase the accuracy of the model because it may perform better if it is trained on more samples than the original dataset included. Using augmentation may be advantageous because of the benefits of making the model robust for certain transformations. Still, the limitations lie in the developer's imagination to imagine which transformations the model is trained on.

1.3 Research Questions

1. Which combinations of backbone models, hyperparameters, and optimizers provide good accuracy and performance for the detection of CM and SF images in standard datasets?
2. Which image pre-processing techniques can assist in CMF and SF detection?
3. How do the models and pre-processing pipelines developed from RQ1 and RQ2 perform on tampered images that have been processed with other transformations than those present in the training images?

1.4 Contribution

This study proposes a CNN model that uses ResNet-101 as a transfer learning model for feature extraction, with two dense layers at the top of the model to predict the model. This research proposes a transfer learning model that uses ResNet-101 as a backbone model with two dense layers and a fully connected layer to classify tampered ad legitimate images. The model differs from previous works with the approach of training the model on the training data and testing the model on training data that have been augmented with augmentation the model has not trained on. The results will be compared with tests done on test data that have been augmented with augmentation that the model has trained on.

1.5 Outline

The thesis is structured accordingly; Chapter 2 introduces relevant background and theory regarding detection models for CM and SF, which are important for the reader to have knowledge about to understand the experiments and proposed results. Chapter 3 introduces the methods that are used in this thesis to answer the research questions. Chapter 4 introduces the reader to the experimental setup and model architectures. Chapter 5 presents the results achieved with the proposed method and experimental setup. Chapter 6 is a discussion and analysis of the results, before the final conclusion in Chapter 7.

Chapter 2

Background

This chapter provides relevant background information about theory, technologies, and related work to the research topic.

2.1 Copy-Move and Splicing Forgery

CMF is an image tampering technique that uses copied objects from an image and pastes it to another location within that image, as mentioned in section Chapter 1. The technique is simple because it mainly requires an image editing program. Image splicing is a similar tampering technique to CMF, but the copied object originates from another photograph. Tampered images can be made less noticeable if the image's creator applies post-processing by blurring sharp edges, color corrections, and flipping or rotating copied objects to hide more obvious traces of tampering.

Two of the techniques that have been used in the literature for CMF detection are passive and active detection. Active techniques involve the extraction of data from the image, digital watermarking, or digital signatures to verify the integrity of the image. The digital signature or watermark is inserted into the picture, and the recipient of the image may then verify it. Passive techniques use algorithms to detect tampered areas in the image that may seem invisible to the naked eye, but forged attempts can leave traces discovered by block- and key-point-based algorithms and machine learning approach [13].

Block-based methods divide images into overlapping or non-overlapping blocks. The technique can eliminate time complexity when images are processed suitably for detecting smoothed regions. Typical feature extractions that have been combined with block-based techniques are Discrete Cosine Transform DCT and Discrete Wavelet Transform DWT. Extracted features in each block are then compared to determine if two or more blocks contain a similarity score above a threshold to determine if there is a match between blocks.

One research project that used block-based methods is Liu et. al. [14]. The authors used the simple linear iterative clustering (SLIC) algorithm to perform a super-pixel segmentation to solve the problem with time complexity. The advantage of the technique is that the CMF model can analyze a smaller amount of pixels to extract features. The method involves segmenting the image into patches that consist of pixel groups with similar levels of brightness and color. Harris corner detection can then be applied to the super-pixels to detect tampered areas. The results showed that the model was robust towards rotation attacks and smoothed areas. Key Point-based algorithms have provided sufficient results for detecting geometrical transformed and scaled areas. The downside of the method is that it is less effective at detecting smoothed or small tampered areas. David Lowe's Scale Invariant Feature Transform (SIFT) [15] is a key-point based algorithm used in computer vision. SIFT is used for object matching and object comparison in computer vision by extracting key points from one image and comparing them with key points from another image. SIFT is robust for added noise, scaling, and rotation of objects, but the method has some drawbacks. The algorithm is computationally expensive due to the number of feature vectors computed in each image, and it does not solve the issue of lack of detection of small objects and smooth, tampered areas [16].

2.2 Metric Description

The performance can be evaluated by calculating precision, recall, and F1 score. The precision score, described in equation Equation (2.1) can give an indicator of how well the model is performing when it comes to predicting true positives. The formula calculates how many true positives were predicted and divides it by the number of predicted true positives plus false positives. A high precision score will indicate that the model can make predictions relevant to the study. A recall score can describe the test's sensitivity and indicates how well the model is at correctly predicting true positives out of all data samples. F1 score is used to determine the harmonic mean of precision and recall by combining the results of both of them. If the dataset is imbalanced, the F1 score can be used to evaluate the performance because false positive and false negative results are encountered. Weighted average is a practical method in classification problems where the dataset is imbalanced. The method encounters how much each data sample has contributed. In other words, data samples in a class with fewer samples in an imbalanced dataset will be weighted more compared with a data sample from the class in majority.

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives} \tag{2.1}$$

$$Recall = \frac{TruePositives}{TruePositives + FalseNegatives} \tag{2.2}$$

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{2.3}$$

$$Specificity = \frac{TrueNegatives}{TrueNegatives + FalsePositives} \tag{2.4}$$

Another method used to evaluate a test is Area Under the receiver operating characteristic Curve (AUC) which is used to measure the specificity and sensitivity score of a test. The sensitivity score is calculated with the same formula as recall score in formula Equation (2.2), and the specificity score is calculated with equation **??** **??**. An AUC score tells us the probability between 0 and 1 for a ML model to correctly predict classes in a test, according to a selected threshold. A high AUC score indicates that the model correctly predicted a larger selection of the data samples.

2.3 Dataset

Benchmark datasets used in the research field such as CASIA2.0 and CASIA1.0 are created by using the same image editing tool on all sample images [12]. The creators have tried to create more realistic tampered images by including scaling and rotation on copied objects, and some images in CASIA2.0 are also post-processed with blurring. However, it is up to the imagination of the image editor to determine how comprehensive each manipulation is.

2.4 Convolutional Neural Network

Convolutional neural networks are deep learning models that use convolutional layers to perform linear operations to learn useful features from input data [17]. The philosophy behind the CNN architecture is that the nodes in each hidden layer act similarly to neurons in the human body, where neurons are connected and can feed forward information. The architecture of CNNs improves traditional neural networks. For example, it creates smaller outputs than the original input because it creates smaller kernels, also called sparse interaction. In other words, the operation is less computational expensive compared to traditional architectures where all input nodes are talking to output nodes in the hidden layers. The kernels use tied weights, meaning that the network learns a set of weights instead of learning many parameters in separate nodes, so the weights in one input are connected with the weights applied in another input in the architecture.

CNN is a popular architecture in computer vision because it can learn useful features from images. After all, it is less computationally expensive, and it uses tied weights to connect different inputs, as described earlier in this section. Another advantage of CNN is that it does not involve as much manual preprocessing as traditional ML models such as Support Vector Machine (SVM) because it learns features, and it is useful when working with massive datasets that can provide a big hypothesis space for the model [18]. One variable that is critical when training CNN is to have many data samples for the model to train on. If not, the model may reach overfitting, which will be described later in section Section 2.4.3. Loss functions are used in CNN to minimize the training error so that the model will classify data correctly. The function is activated during back-propagation, and it returns the partial derivative value for each weight in the network. For binary classification problems, the loss function binary cross-entropy is an appropriate one to use, and it is given in formula Equation (2.5), taken from [19] "*where M is the number of training examples, ym is the target label for training example m, xm is the input for training example m, and hθ is the model with neural network weights θ*".

$$J_{bce} = -\frac{1}{M}\sum_{m=1}^{M}[y_m * log(h_\theta(x_m)) + (1 - y_m) * log(1 - h_\theta(x_m))] \tag{2.5}$$

One optimization function used in CNN is the Adam optimization function [20], a gradient-based optimization used to optimize the learning objectives within classification problems with large datasets used in ML models. Adam is calculated by first taking the gradients for the stochastic objective with the formula Equation (2.6). The updated biased first moment is calculated with formula Equation (2.7). The second updated biased momentum is calculated with formula Equation (2.8). Bias corrected the first moment is calculated with formula Equation (2.9). The bias-corrected second raw moment is computed by formula Equation (2.10), and the last step is to update the parameters with formula Equation (2.11).

$$g_t = \Delta_\theta \int_t (\theta_{t-1}) \tag{2.6}$$

$$m_t = \beta_1 * m_{t-1} + (1 - \beta_1) * gt \tag{2.7}$$

$$v_1 = \beta_2 * v_{t-1} + (1 - \beta_2) * g_t^2 \tag{2.8}$$

$$\hat{m}_t = \frac{m_t}{(1 - \beta_1^t)} \tag{2.9}$$

$$\hat{v}_t = \frac{v_t}{(1 - \beta_2^t)} \tag{2.10}$$

$$\theta_t = \theta_{t-1} - \alpha * \frac{\hat{m}_t}{\sqrt{\hat{v}_t + \epsilon}} \tag{2.11}$$

The second optimization function vastly used for optimization in ML is Stochastic Gradient Descent (SGD) which is calculated with equation Equation (2.12) to get the weights updated during training [21].

$$w = w - \alpha \nabla L_i(w) \tag{2.12}$$

Over the past couple of years, computational neural network (CNN) models have become more popular due to the simplicity of image prepossessing, feature extraction, and fine-tuning between layers. Moreover, CNN models have shown to provide good performance in terms of accuracy without the trad off with time complexity, as proposed by Goel et. al [22] who created a novel dual branch CNN model by using Keras as backend. The model is robust for geometrical transformation in copied objects, but it is computationally expensive and tends to produce false positives when training on legitimate images. A support vector machine (SVM) is then used for the final classification of the test images.

2.4.1 Transfer Learning

Transfer learning has become a more relevant approach to machine learning over the past couple of years due to the possibility of reusing a trained model on new data. There are three options of how to used a ResNet transfer learning architecture [23]. The first approavh is to update all of the pre-trained weights during training. The second approach is to only update weights on specified layers of the ResNet model during training. Finally, the third approach is only to update the top layer of the ResNet architecture and set the rest of the pre-trained weights as not trainable during training to prevent them from being updated. One of the advantages of the third approach is that the model must not be trained from scratch but instead use the pre-trained weights to classify new data. [24] One argument that speaks for transfer learning is that training of a new model can be done much faster because the knowledge of the pre-trained model is reused. Another advantage of this method is that the model may provide better performance in classification problems where there are a limited amount of data samples for training and cross-validation [18]. The pre-trained models are capable of learning new problems in a much faster manner because they are capable of generalizing new data because they have trained on a vast selection of images, such as the ImageNet database that contains 1000 classes, and because of the convolutional layers, described earlier in section Section 2.4. The models can also extract new and helpful features faster from the new images that it is fed with, because there are similar classification problems that the model has already been trained on, which means that weights are generated based on training on other data samples.

There are many different transfer learning models out there, such as VGG16, VGG19, AlexNet, ResNet-50, and ResNet-101. The ResNet models are some of the deepest transfer learning models out there that are capable of training the deepest layers. This element is further described in section Section 2.5. One is not limited to only using the transfer learning models with pretrained weights, because it is possible to unfreeze the weights and train the model from scratch, if it suits the classification problem. Unfreezing weights means that the weights are updated during training, and extracted features from the ImageNet is not utilized during training of a new classification problem. Moreover, if a pretrained model suits the classification task, a pretrained model can be loaded with the ML framework we selected for the project. One example where a pre-trained model may be advantageous is in classification problems when there is a limited amount of data samples available for training a model.

Transfer learning methods have increased their popularity in CMF and SF over the past couple of years. Wilsey et al. used the five pretrained transfer learning models VGG-16, VGG-19, GoogleNet, Inception-v3, and Alexnet to detect SF in the DSO-1 dataset. The outputs from each model were used to train a SVM for further classification of the images. The pre-trained models achieved better accuracy than state-of-the-art research using hand-crafted feature extractors. In [25], Wu et al. used the VGG16 architecture as the backbone model in a deep matching and validating network used to detect and locate CMF and SF in image pars from the sun2012 and Microsoft COCO datasets. In this context, image pairs indicate that the original and tampered image are compared to locate tampered objects. The research achieved an F1 score of 86%, precision score of 94%, and a recall score of 78%. Kadam et al. used the transfer learning models ResNet-101 and MobileNet v1 to detect CMF on the CASIA 2.0 database and SF on the CASIA 1.0 dataset [26]. Pandey et al. proposed a transfer learning model that used ResNet-101 as a backbone model without a custom layer on top of the transfer learning model. Instead, the network used ELA as a feature extractor to detect tampered areas in the CASIA 2.0 dataset [27]. The research achieved a weighted average F1 score of 88%, which is one of the best with ResNet-101 as the backbone model, but the researchers do not provide precision and recall scores. Ahmed et al. [28] used the pre-trained ResNet-50 and ResNet-101 models to compare their ability to detect image forgeries on a computer-generated dataset.

Liu et al. [29] proposed a constrained image splicing detection and localization model trained on the CASIA dataset. They used ResNet-50 and VGG16 as the feature extractors and a decoder architecture. The model is fed with pairs of authentic and spliced images and is processed with feature maps, before fine-grained masks are computed to locate the tampered object. Meena et al meena2021res50 used Resnet-50 as the backbone model to extract features to detect SF on the CUISIDE dataset. Feature extraction was done by generating a noise residual map for each image, the noise print is used to extract the fingerprint of the camera model that was used to take the picture. For the final classification, the author used a SVM classifier that achieved a detection accuracy of 97.24%. Pomari et al. used ResNet-50 with illuminant maps as a feature extractor to find lighting in small areas in images [30]. The authors also used a SVM for the final classification method instead of using a traditional softmax classifier. The model achieved an accuracy score of 96%. Nath et al. nath2021res50 used ResNet-50 as the backbone model for the detection of CMF and SF on the CASIA 2.0 dataset without using any extra mechanism for feature extraction. The model achieved an F1 score of 95.4%, a precision score of 96.6%, and a recall score of 94.1% Jaiswal et al. [31] proposed an image tampering detection model trained on the CASIA 2.0 dataset. The model used ResNet-50 as the backbone model with custom architecture, and the three different classifiers: k-nearest neighbors, Naïve Bayes, and SVM, for final image classification. They achieved the best results with the SVM classifier with an accuracy score of 0.7026, a specificity score of 0.7497, and a sensitivity score of 0.6339. Specificity and sensitivity measure how the model predicted "authentic" and "tampered" images.

Saba et al. used a pre-trained ResNet-101 and a custom CNN architecture on the six different datasets to detect CMF. The six datasets were; CG-1050-v1 [32], CG-1050-v2 [33], Coverage [34], MICC-F2000 [35] and Copy-move Forgery Dataset [36] and a unified dataset. Zhou et al. [37] proposed an R-CNN network to detect and locate tampered areas. The R-CNN network uses a two-stream architecture with one RGB stream that extracts manipulation artifacts, and one noise stream that detects differences in noise streams between tampered and authentic objects. They use a transfer learning architecture with ResNet-101 for the RGB stream for feature extraction. They used CASIA 2.0 for training and CASIA 1.0 for testing. They also tested how data augmentation affected the training results, and they concluded that implementing augmentation for flipping improved the accuracy of the model, but augmentation methods such as JPEG compression and adding noise did not improve the results in a significant way. They also tested the model on images from the NIST Nimble 2016 [38] dataset, compressed with JPEG compression of quality 70% and 50% to test if their model was robust to compression attacks, and the results show that the the model is robust. They measured the accuracy in Area Under the receiver operating characteristic Curve (AUC) whitch measures the specificity and sensitivity of the test.

2.4.2 Data Augmentation

One method used for preprocessing is data augmentation, and the method involves the inclusion of augmentations to data samples in the data set. A great variety of possible augmentation methods can be applied, and it depends on the classification problem of which augmentations should be applied or not. Augmentation is not always required to achieve good accuracy when using CNN and transfer learning (TF), as proven in [39] [40] [41] [42]. However, it is a technique that can prevent overfitting, described in section Section 2.4.3, increase accuracy, and provide the model with more data samples during training.

2.4.3 Overfitting

A problem that may occur when training a machine learning model is that the model can predict well during training. Still, validation and testing may result in high training errors, and the model cannot learn how to distinguish between classes [43]. This problem is called overfitting and may occur when there is too little training data. The model cannot learn the general features that distinguish the classes because the input data introduced a too large hypothesis space for the model to handle. For example, validation data is either too homogeneous or heterogeneous, which means that the data is either too similar or differs too much to let the model learn valuable features. This phenomenon can lead to learning of either noisy data or learning only a few features that will not be useful when the model is introduced to real-world data.

The solution to overfitting and learning noisy features is to implement mechanisms such as early stopping, class weights, dropout, and data augmentation described earlier. Early stopping is used to stop the model from continuing training if the training or validation metrics do not improve or the accuracy of the metrics starts to decrease, which should be avoided. Class weight is implemented by defining a parameter that gives the class in the minority to be weighted more than the other class that is over-represented [44]. The reduced learning rate is a method used to schedule a reduction of the learning rate if a metric is not improving over time. Applying dropout layers after dense layers are done to “thin" out the model. A set of neurons and their connections to previous and forward layers are dropped [45] during each epoch. The set of neurons is selected with a given probability, and this is done during each epoch, so the neurons are only temporarily dropped.

2.5 Residual Networks

Previously, deep neural network (DNN) was created on the philosophy that deeper layers were equal to better performance and accuracy. However, the main problem with deeper layers at the time was that it was complicated to optimize deeper layers because deep layers were difficult to train. The work of He et al. [46] resulted in what we know today as residual networks, and their ideas are used in ResNet-50 and ResNet-101 architecture today. Their idea was that instead of letting input x be changed linearly through each layer in a layered model, they introduced skip connections where the model learns the difference between input x and the output of each skip function $f(x)$, instead of the function $H(x)$ that represents the calculated output of the input x after it has gone through the stacked layers, see Figure 2.1. This results in the residual function:

$$f(x) + x \tag{2.13}$$

Where $F(x)$ equals the change in parameters for x after each weight layer in the residual block, and x is the identity function, the residual block's default input. A residual network is a network consisting of multiple of these residual blocks.

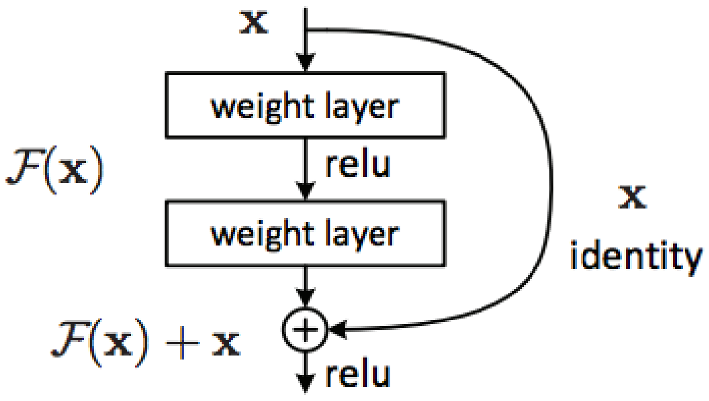

x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 2.1: Illustration of skip connections in the residual network, taken from [47]

ResNet-50 and ResNet-101 are made up of multiple residual blocks that creates two very deep networks of 50 and 101 layers. To reduce time complexity and the dimensions, because of the depth of the ResNet-50 and ResNet-101 networks, the authors of residual networks came up with the bottle neck building blocks solution that uses three layers in each residual block instead of two layers. The convolutional layers in the bottleneck block are built up of $1x1, 3x3$ and $1x1$ convolutions, see Figure 2.2. The first $1x1$ layer is used to lower the dimension from the input so that the second $3x3$ layer has a smaller bottleneck to manage, before the third $1x1$ layer reduce back the dimensions. ResNet-50 consists of 16 three-layered bottleneck residual blocks, and ResNet-101 consists of 33 bottleneck blocks.

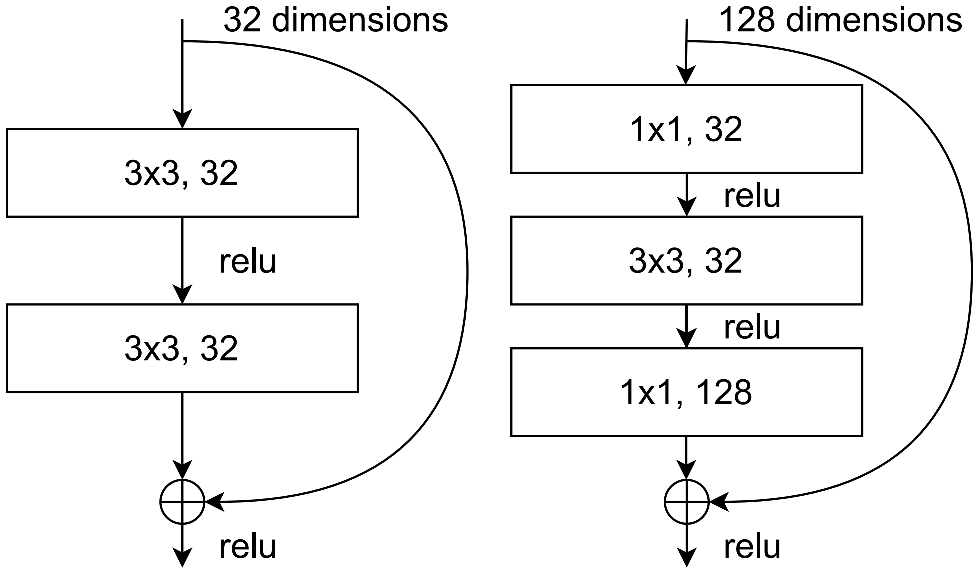


Figure 2.2: Illustration of the ResNet bottleneck architecture [46]

2.6 Error Level Analysis

error level analysis is a technique used in image forensics for the detection of manipulated areas by using JPEG compression to its advantage. JPEG compression is a standard method used to reduce the image size via a lossy compression technique. The process includes dividing the image into blocks of size $8x8$ that are further processed with color transformation and down-sampling before a quantization table is applied to calculate the DCT coefficients [48].
The practical aspect of using JPEG compression is to bring down the size of an image that will be transferred over the Internet. The compression can be almost unnoticed by the human eye without comparing the original image with the compressed image.

When performing an ELA on images, the first step is to save a copy of the image with another compression quality between 0 and 99. The second step is to calculate the extreme values between the original and compressed images. If the image has not been tampered with, the error levels in the $8x8$ blocks should be approximately at the same level. On the other hand, if the image has been tampered with in a certain way, the error levels in the $8x8$ blocks will be different because an image is compressed the first time it is saved with the JPEG format. The image had gone through a second compression when it was resaved in an image editing program. Editing programs such as Photoshop use their own JPEG quantization table for JPEG compression. A disadvantage of using ELA to detect manipulated images is that for each time an image is resaved, the error levels in the image may be reduced for each time. This indicates that the error levels will stand less out when error levels are computed on images that have been compressed multiple times, as can happen when it is up and downloaded on numerous websites.

Jabeen et al. [49] used ELA as a feature extractor of the input images in their proposed CNN model. The model was trained on CASIA 2.0 with a precision score of 81% and a recall score of 61%. Sari et al. [50] used a CNN architecture with ELA for preprocessing to detect CMF and SF in images. They tested how ELA with quality of 90%, 50% and 10% affected the detection rate of the model. They concluded that ELA with quality of 50% gave the best testing accuracy event though the artefacts in the tampered objects were not as visible compared with ELA of 90% quality.

Chapter 3

Method

3.1 Datasets

The three CMF and SF detection models must be trained and evaluated on a dataset with images containing copied objects from the same photo or other photos. Using datasets already used in academic research will ensure the possibility for the experiment to be repeated by others. It will also ensure the possibility of comparing our research with related work. The dataset should be as close to real-world examples as possible and contain a great variety of motives to feed the model. This will also ensure that one can be sure that the model can detect a forgery in different contexts, such as scaled, rotated, smoothed, and small areas.

CASIA v1.0 and CASIA v2.0 are two of the most used datasets for detecting image splicing and CM forgery and may be one of the most suitable datasets for this project. CASIA v2.0 is also one of the datasets with the highest number of data samples in one dataset, it is publicly available and can be downloaded for free from Kaggle [51].. The two datasets were created by [12], who used Adobe Photoshop to edit the images. CASIA v2.0 contains 7491 valid images and 5124 manipulated images in JPEG and tif format. The dataset provides a variety of images containing people, nature, buildings, and animals. The class distribution is 71.5% of the image samples are "*authentic*", and 28.5% are "*tampered*". We initially split the dataset into 80% for training, 10% for validation, and 10% for testing purposes for all the three models used in this paper. CASIA v2.0 is a more comprehensive dataset than CASIA v1.0, because the dataset creators have applied post-processing techniques such as as blurring in the manipulated images to create a more realistic dataset. In other words, some of the data samples have already been post-processed to a certain extent. However, for this research, we wanted to investigate the impact of extra post-processing that can be applied to images, for example if tampered images are transferred over social media, and therefore the CASIA 2.0 dataset fits the purpose of this research.

3.2 Experiment

To run experiments to answer the first research question, how a SF model will be at predicting tampered images that have gone through a transformation process it has not trained on, a CM and SF model must be trained on the respective dataset mentioned in section Section 3.1. For this particular experiment, we selected a machine learning approach. It is possible to set up a machine learning environment from scratch. Still, it is more practical to use the Keras framework as a high-level API that is integrated with TensorFlow as the low-level API. Using Keras and TensorFlow enables the possibility of setting up a neural network environment that is up and running in a fast manner.

The Keras API offers a variety of pre-trained models that can be used for transfer learning. The models are pre-trained on the ImageNet dataset, an image database of more than 1000 categories [52]. ResNet-50 and ResNet-101 are two of the backbone models for image classification offered by Keras, and they are the two transfer learning models used in this study [23]. Keras applications come with pre-trained weights, so one does not need to train a model from scratch. The philosophy is that transfer learning models can reuse pre-trained weights from one classification problem in another classification problem. The method is also practical to use when the number of data samples is limited, because the amount of data samples is less vital when a transfer learning model is trained for a new classification problem, and this is very desirable because the dataset used in this project only contains about 12, 000 images in total.

Keras offers different possibilities on how to use ResNet models for training. The third approach with freezing all weights except from the top layer is selected for this research, because the pre-trained parameters ensure that the network is trained in a faster manner compared with training it from scratch. This approach is also preferable because the number of data samples is limited. The ResNet transfer learning model architectures selected for our research are a ResNet-101 model with a proposed custom architecture on top after the pretrained layers, and a ResNet-50 model with custom architecture from Hebbar et al. [53]. A ELA-CNN model with the architecture from Sari et al. [50] is also selected.

3.3 Evaluation

To answer the third research question from section Section 1.3, on *how do the models and pre-processing pipelines developed from RQ1 and RQ2 perform on tampered images that have been processed with other transformations than those present in the training images?*, the model is evaluated with the metrics described in section Section 2.2 which are F1, recall and precision score.

During training, the models that achieve the best validation accuracy, which in this context is F1, precision, and recall score, will be selected as the models used for further testing. A pre-trained model is loaded into a separate script that executes the test and displays accuracy metrics and a confusion matrix for each test. Each pre-trained model will be tested on different test datasets, and one of them will be a dataset that has not gone through any extra transformations before ELA is applied. This is to ensure that the results contain a baseline of the model's accuracy when it tests on images in the same format as the training and validation image set. The other test datasets will contain different transformations before applying ELA on each input image.

3.3.1 Confusion Matrix

FP, FN; TP, TN

3.4 Pre-processing

One of the steps in the startup phase of setting up a machine learning model is to pre-process the data that will be fed to the model. This process aims to prepare the data for the particular classification problem. It can eliminate irrelevant noise and ensure that the model can process each image faster because less data is processed. Some typical techniques for pre-processing images are changing file format, grayscaling, blurring, brightening, and resizing, to mention a few.

We converted images in *tif* format to *jpeg* format because ELA is used to evaluate the difference in JPEG images by comparing the error levels between the original image and a resaved image with a lower quality level, in this case, 90% quality. We performed data augmentation on the ResNet-101 model with height shift range, rotation range of 20%, vertical flip, and horizontal flip. The CASIA 2.0 dataset is limited in the number of data samples. One mitigation to the issue of data limitation is to implement data augmentation, which provides the model with several more data samples compared with the original dataset [54]. Augmentation can also be applied to prevent overfitting due to the philosophy that models may achieve better accuracy when training on big datasets, which is not always possible to create. The mentioned augmentation methods were selected because they differ from the test datasets' transformations, and the model can still benefit from data augmentation.

Chapter 4

Experimental Setup

4.1 Environmental setup Setup

We did the experiments on a Windows 10 desktop and a Ubuntu server. The Windows 10 desktop had an Intel Core i7-7700k processor running at 4.20GHz, with four cores and eight threads. In addition, the computer has 16 GB of RAM and an NVIDIA GTX 970 graphics card with 4GB of video memory (VRAM). For storage, a quick M.2 SSD was used. The Ubuntu server with 90 GB RAM, 8vCPU, and a 1/4 of a Tesla v100 with 8 GB GPU-RAM [55]. In addition, the server has an NVIDIA driver and CUDA packages. We used Python version 3.9.7 as the programming language to implement the models with the listed packages in table Section 4.1.

4.2 TensorFlow and Keras

TensorFlow is an open-source machine learning platform created by the Google Brain team, and it works as a backend together with high-level APIs. The library makes it possible for people to set up state-of-the-art machine learning environments and machine learning models. TensorFlow also has a great community and documentation that gives users a lot of resources when using the library. In addition, it is compatible with a selection of programming languages, such as Python and C++. This is why the framework was selected for this study. Keras is another open-source library used together with TensorFlow as a high-level API. The library provides functionality for fine-tuning ml models with optimizers and layers for activation functions, dropout, and batch normalization.

4.3 Preprocessing

Preprocessing is a method used to prepare data before training, validation, and testing in a machine learning model. The outcome of the preprocessing stage, as described in section Section 3.4, can eliminate data that is unnecessary, reduce the computational complexity by modifying the data, and provide the model with

Table 4.1: List of Python packages

Package	**Version**
Tensorflow	2.6.0
Tensorflow-addons	0.15.0
Keras	2.6.0
Keras-Applications	1.0.8
Keras-Preprocessing	1.1.2
Keras-resnet	0.2.0
Matploitlib	3.5.1
Scikit-learn	1.0.1
Numpy	1.19.5
opencv-python	4.5.3.56
Pillow	8.3.2
Imutils	0.5.4

consistent data. We converted images from tif format to JPEG format with the Python library Pillow. The library returns an image object that is resaved with the *Image.save()* [56] function in JPEG format with a quality of 95%, which is the highest compression quality Pillow offers [57].

A small amount compresses Tif images after JPEG converting. Section 4.3 shows the difference between a tif image before and after converting. The image is zoomed in to demonstrate the difference more clearly. Pillow does not resize images unless specified when it saves the image, so all the converted images are not resized in this stage. The disadvantage of converting images to JPEG is that they lose some quality in the converting stage and can become more difficult for the model to detect than images not converted to JPEG format. We used the Imutils library to rotate images 45°, and OpenCV to mirror images and rotate them 180°. ELA is applied on training, validation, and ordinary test images after image formating to JPEG. We created the test datasets by applying the respective transformation for that particular test before applying ELA to the images.

Color channels of images are converted from RGB to BGR with *preprocess_input* because it is a prerequisite for the ResNet Keras Application. The color channels are then individually zero-centered with respect to the imageNet dataset, and This process is preceded before the images are fed to the model [58].

(a) Tif image **(b)** Converted JPEG image

Figure 4.1: Image a is the original image before it was converted to JPEG format in image b.

4.4 Proposed Model Architectures

Early stopping was implemented to monitor the validation loss metric for all three models tested in this experiment. For the ResNet-50 and ELA-CNN model, the patience was set to 10, meaning that the training will stop if the validation loss metric has not improved over the last 10 epochs. For ResNet101, the early stopping patience was set to 50 epochs. A class weight metric was implemented in the ResNet101 model to mitigate imbalanced data. The CASIA 2.0 dataset contains 7484 data samples for the authentic class, and 5130 data samples from the tampered class. The authentic class makes up approximately 68% of the entire dataset, indicating that the dataset may be a little bit imbalanced. The authors of the ResNet-50 model [53] implemented a ReduceLROnPlateau function that can reduce the learning rate in a defined metric that has not improved [59]. The same function was also implemented in the ResNet-101 model with $factor = 3$, and the factor variable was defined as $factor = 2$ in ResNet-50. Min_lr was set to 0.001 and a patience of 10 epochs was defined in both the models. In both ResNet models, the validation loss function is monitored, and the patience is defined as for each 10th epoch, the validation loss must have improved.

A dropout layer of 0.15 was implemented after the first dense layer, and a dropout of 0.25 was implemented after the second dense layer in ResNet-101. Finally, a dropout of 0.25 was implemented after the dense layer in ResNet-50. The purpose of using ResNet-101 and ResNet-50 as base models, and the ELA-CNN model is to answer research question 1, "*Which combinations of backbone models, hyperparameters, and optimizers provide good accuracy for the detection of CM and SF images in standard datasets?*". The purpose of comparing ResNet-50 and ResNet-101 is to test if using a deeper model will have any effect when it comes to testing on images the model has not trained on compared with less deep architectures. The custom architecture on to of the pre-trained ResNet-50 model in Figure 4.3 is from the research of Hebbar et al. [53], and the same architecture inspires the custom architecture of ResNet-101, see Figure 4.2. The ELA-CNN model architecture is from [50]. The pre-trained weights are frozen in the ResNet models by defining the *include_top* parameter in the backbone model declaration is set to *false*. Input images are resized to $224x224x3$ in the proposed ResNet-101 model, and images are resized to $256x256x3$ in the ResNet-50 and ELA-CNN model. The optimizer function Adam with a learning rate 0.0001 was used in ResNet-50, SGD with learning rate 0.0001 was used in the ResNet-101 model, and a learning rate of 0.00001 for the ELA-CNN model. The relu activation function was used in all dense and convolutional layers in all of the three models.

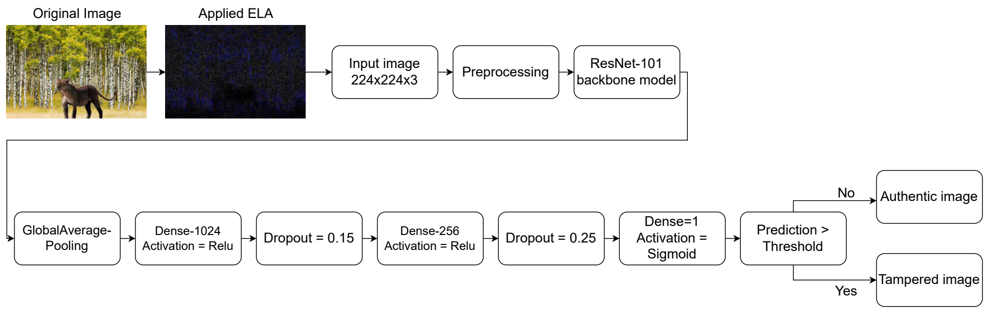


Figure 4.2: Architecture for pipeline utilising ResNet-101

Data augmentation is applied to avoid overfitting and for accuracy improvement in the proposed ResNet-101 model. The rotation range is set to 30% and indicates that the images are rotated randomly with a rotation range up to 30%. Vertical flip indicates that the images are tilted randomly at a vertical angle, and the same random flip is applied for horizontal flipping. The last applied augmentation is the height shift range which is set to 20%, and it indicates that the height of the images is shifted randomly by up to 20%. None of the pre-trained weights in the ResNet-101 model are updated during training, details around the advantages of not updating pre-trained weights are described in section Section 3.2. Outputs from the pre-trained backbone model are fed to the first layer in the ELA-CNN model which is a global average pooling (GAP) layer. The GAP layer is followed by a flatten layer that flattens the input tensor and converts it to a one-dimensional tensor. The third layer is a dense layer with 1024 dimensions and a relu activation function, followed by a dropout layer of 0.15. The fifth layer is a dense layer with 256 dimensions, relu activation function, followed by a dropout layer of 0.25. Dense layer three and five have also included the Keras l2 kernel regularizer function [60] that is calculated with equation Equation (4.1). The function applies a penalty to the parameters in a layer when the parameters are optimized during training.

$$loss = l2 * reducesum(square(x)) \tag{4.1}$$

The last layer is a dense layer with one dimension and a sigmoid activation function used for the final binary classification.

Parameters	**Value**
Epochs	250
Image resize	224x224
Batch size	32
Learning rate	0.0001

Table 4.2: List of parameters for Resnet-101

4.5 ResNet-50

The first layer in the custom architecture on top of the ResNet-50 transfer learning model is a GAP layer, followed by a dense layer of 256 dimensions. The dense layer is followed by a dropout layer of 0.25. The last layer is a dense layer with one dimension and a sigmoid activation function used for the final binary classification. The model was trained for 29 epochs in total. Figure 4.3 demonstrates the ResNet-50 model described in this section. Figure 4.4 shows F1, precision and recall graphs from training and validation.

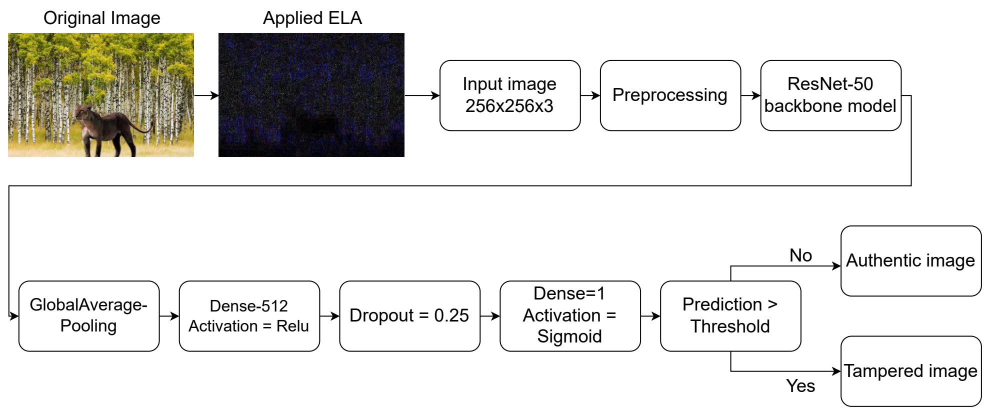
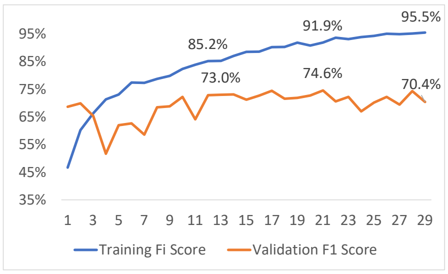
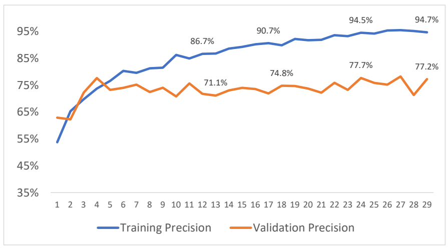


Figure 4.3: Architecture for pipeline utilising ResNet-101 [53]

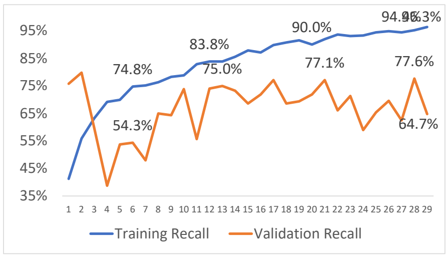
The first layer in the ELA-CNN architecture consists of a convolutional layer with a kernel size of $3x3$ and 32 dimensions. The third layer is a max-pooling layer with a pool size of $2x2$. The third layer is a convolutional layer with 64 dimensions and a kernel size of $3x3$, followed by a second max-pooling layer with a pool size of $2x2$. The last layer is a dense layer with one dimension and the activation function is sigmoid.



(a) Training and validation F1 scores



(b) Training and validation precision scores



(c) Training and validation recall scores°

Figure 4.4: Training and validation graphs from the training of ResNet-50. (a) is F1 score, (b) is precision score and (c) is recall score

4.5.1 Testing

The models that obtained the highest F1, recall, precision score during training, and k-fold cross-validation are saved for later testing. The saved models are loaded into a testing script developed with Keras [61] and TensorFlow [62]. The test script passes the pre-trained model to a predicting generator that extracts a NumPy array of predicted labels. We defined a threshold of 0.5 to determine if the predicted labels from a test were either *“authentic*" or *“tampered*". The outcome of each prediction is a NumPy array with the model’s confidence level of what each test image should be classified as [63]. Because the classification problem is binary, any confidence level below 0.5 means that the model predicted the class as *authentic*, and anything above 0.5 means that the model predicted the class as *tampered*. The predicted labels from the data generator are then compared with true labels from the testing dataset.

After the test predictions have been extracted, it is possible to display information that can be used to determine how well the model performed in terms of predicting number of true positive (TP), false positive (FP), true negative (TN), and false negative (FN). The results are displayed in a confusion matrix for better visualization of the test, and the results are used for computing F1, precision, and recall evaluation scores. For each test, the scores are also displayed as a classification report which provides scores for how well the model predicted each class,

Parameters	**Value**
Epochs	12
Image resize	256x256
Batch size	32
Optimizer	SGD
Learning rate	0.00001

Table 4.3: List of parameters for CNN model

and it shows the weighted average sum for each prediction. We used the Python library scikit-learn [64] to generate the confusion matrix, and classification report because it is a handy library for data visualization and analysis in computer vision.

Chapter 5

Results

The experimental results described in this chapter are used to answer the third research question, *How do the models and pre-processing pipelines developed from RQ1 and RQ2 perform on tampered images that have been processed with other transformations than those present in the training images?*, described in Section 2.2. The term “*accuracy*" is used as a collective description of the three metrics F1, precision and recall when describing them in general for each test. A further discussion with comparisons between the models and different variables that could have affected the results is presented in the discussion in Chapter 6. All experiments are performed with the three pre-trained models that have shown the best results from the training stage, also described earlier in **??**. There have been done nine different tests to evaluate the accuracy of the models.

5.1 Tests

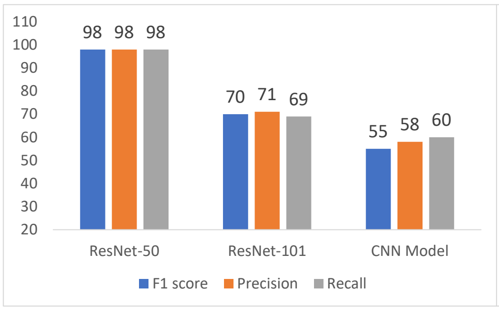
This section describes results from the nine tests done during testing.

5.1.1 Testing on normal test images

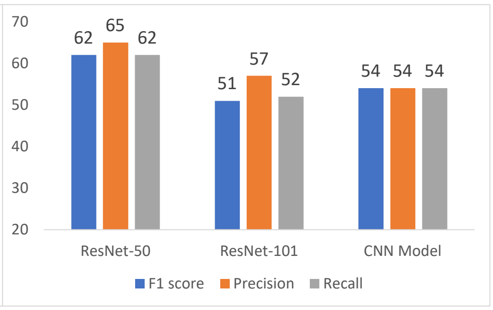
The ResNet-50 model achieved the best results from test 1 with a weighted average score of 98% in F1, recall, and precision.
The proposed ResNet-101 model achieved an F1 score of 70%, a precision score of 71%, and a recall score of 69%. The ELA-CNN model achieved an F1 score of 47%, a precision score of 58%, and a recall score of 60%.

5.1.2 Testing on brightened images

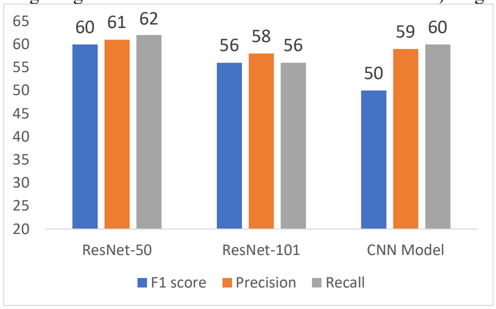
Test number two involved testing on images pre-processed with brightening. The test was done to imitate cases where the image’s creator has brightened the image as a mechanism to avoid the image being detected as fake [65]. The application of brightening is one of the simple methods that can be done by an editor using an image editing program. In this test, ResNet-50 got an F1 and recall score of 62%,



a) Normal training images



b) Brightened images



c) Gaussian Blur

Figure 5.1: F1, precision and recall score

and a precision score of 65%. ResNet-101 got an F1 score of 51, a precision score of 57%, and a recall score of 52%. The ELA-CNN model achieved an F1, precision, and recall score of 54%.

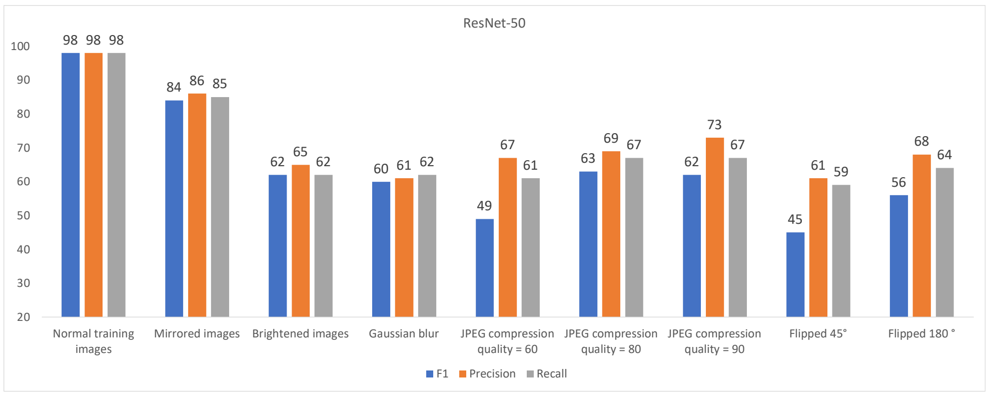


Figure 5.2: F1, precision, and recall score from tests done on ResNet-50

5.1.3 Testing on Gaussian Blur

Test number three used images pre-processed with the Gaussian blur blurring algorithm to imitate blurring attacks involving edges on tampered areas. Gaussian blur, in particular, is a technique typically used for blurring areas in the image and making other subjects draw the viewer's attention. Some purpose for using Gaussian blur by image editors is to smooth out regions that will not be the focus of the image. In other words, other subjects will pop out and drag the viewer's attention [66]. The Gaussian blur function from OpenCV [67] was used in the post-processing script with a kernel size of $5x5$. The ResNet-50 model achieved an F1 and a recall score of 62%, and a precision score of 65%. The ResNet-101 model achieved an F1 score of 51%, a precision score of 57%, and a recall score of 52%. The ELA-CNN model achieved an F1, precision, and recall score of 54%.

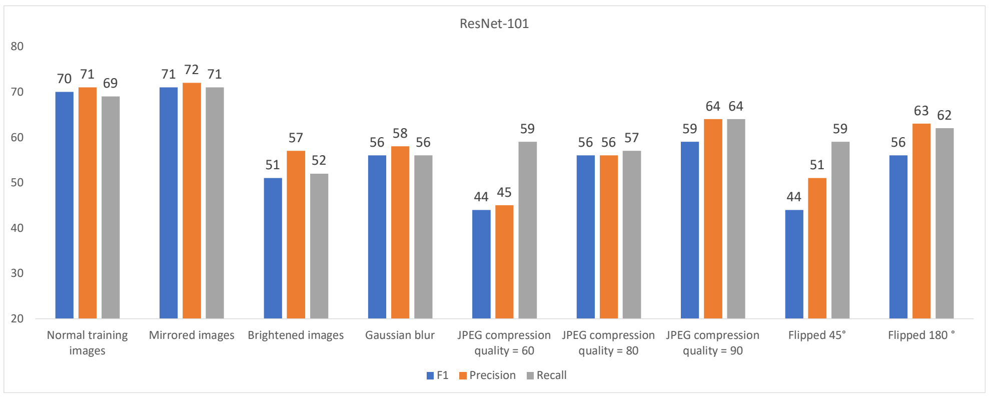


Figure 5.3: F1, precision, and recall score from tests done on the ResNet-101 model

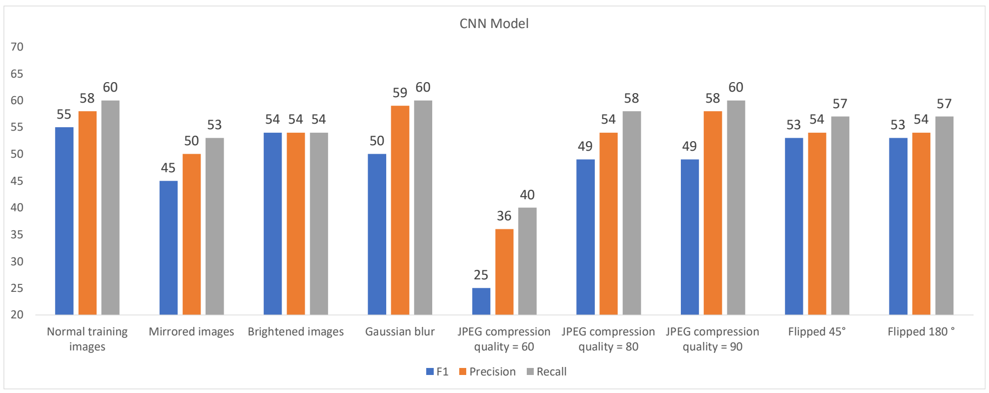
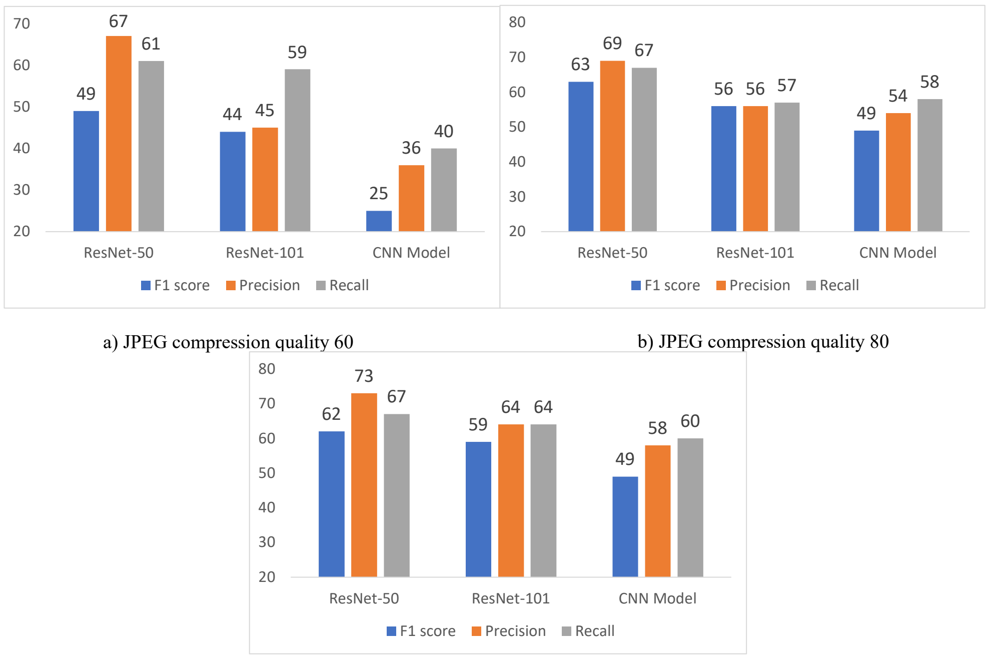


Figure 5.4: F1, precision and recall score from tests done on the CNN model.

5.1.4 JPEG Compression

JPEG compression happens, for example, when images are resized or uploaded on Facebook [11]. This test aimed to find out if there are any differences in accuracy between tests with different compression.



a) JPEG compression quality 60

b) JPEG compression quality 80

c) JPEG compression quality 90

Figure 5.5: F1, precision and recall score

JPEG Compression of Quality 60

The fourth test was done with a JPEG compression of 60%, the test with the lowest compression rate. ResNet-50 obtained the highest F1 score of 49%, a precision score of 67%, and a recall score of 61%. The ResNet-101 model achieved the second-best F1 score of 44%, a precision score of 45%, and a recall score of 59%. The ELA-CNN model achieved an F1 score of 25%, a precision score of 36%, and a recall score of 40%.

JPEG Compression of Quality 80

The fifth test was done with a compression quality of 80%. ResNet-50 got an F1 score of 63%, a precision score of 69%, and a recall score of 67%. ResNet-101 had an F1 and precision score of 56% and a recall score of 57%. The confusion matrix in Table 5.1 shows that the model predicted 522 out of 742 "*authentic*" images, and 197 out of 519 "tampered" images. The ELA-CNN model achieved an F1 score of 49%, a precision score of 53%, and a recall score of 48%.

Table 5.1: Confusion matrix of JPEG compression quality 80% test for ResNet-101

		Predicted class	
		Authentic	Tampered
True class	Authentic′	TP 522	FN 220
	Tampered′	FP 322	TP 197

Table 5.2: Confusion matrix of JPEG compression quality 90% test for ResNet-101

		Predicted class	
		Authentic	Tampered
True class	Authentic′	TP 667	FN 77
	Tampered′	FP 383	TP 136

JPEG Compression of Quality 90

The sixth test was done with a compression quality of 90%, which is also the same percentage as the ELA images are compared with. The ResNet-50 model achieved an F1 score of 62%, a precision score of 73%, and a recall score of 67%. The ResNet-101 model achieved an F1 score of 59% and a precision and recall score of 64%. The confusion matrix in Table 5.2 shows that in this test, the ResNet-101 model predicted 667 out of 742 images correctly as the "*authentic*" class, and it predicted 136 out of 519 images correctly as the "*tampered*" class. The ELA-CNN model achieved an F1 score of 41%, a precision score of 49%, and a recall score of 53%.

5.1.5 Flipped Images

Testing on Images Flipped 45°

The seventh test in this research tested on images rotated with 45°as seen in Figure 5.6. The image contains edges filling the gaps from where the picture has rotated. ResNet-50 achieved an F1 score of 45%, a precision score of 61%, and a recall score of 59%. The ResNet-101 model achieved an F1 score of 44%, a precision score of 51%, and a recall score of 59%. The ELA-CNN model achieved the highest F1 score with 53%, a precision score of 54%, and a recall score of 57%.

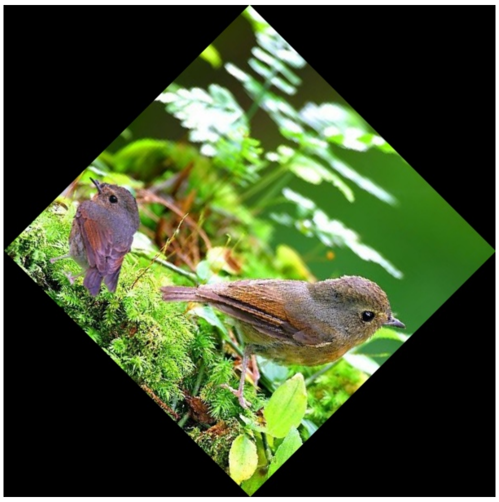


Figure 5.6: Image sample from test set with the 45°rotation

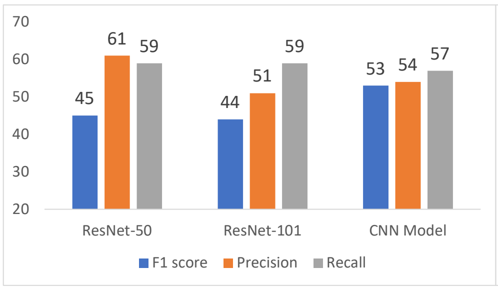
Flipped 180°

ResNet-50 achieved an F1 score of 56%, precision of 68%, and a recall score of 64%. ResNet-101 also achieved an F1 score of 56% and a precision score of 63%. The recall score was 62%. The ELA-CNN model obtained an F1 score of 44%, a recall score of 52%, and a recall score of 57%.

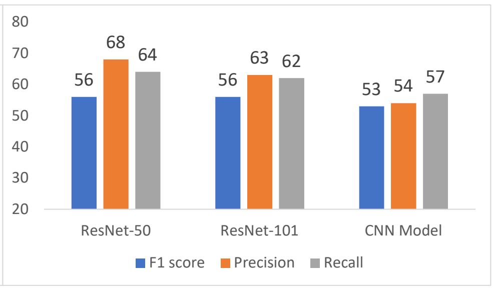
Mirrored Images

A mirrored image is an image that has been rotated 225°.

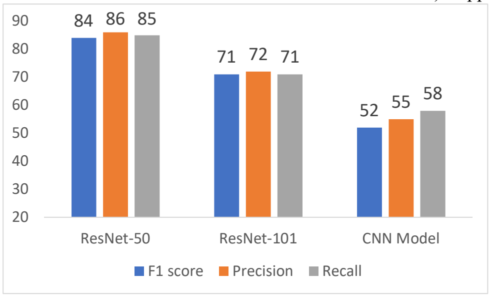
ResNet-50 obtained a significantly better performance compared to the other models, as seen in figure Figure 5.7, with an F1 score of 84%, precision of 86%, and recall score of 65%. ResNet-101 achieved an F1 and recall score of 71% and a precision score of 72%. The ELA-CNN model achieved an F1 score of 52%, a precision score of 55%, and a recall score of 58%.



a) Flipped 45°

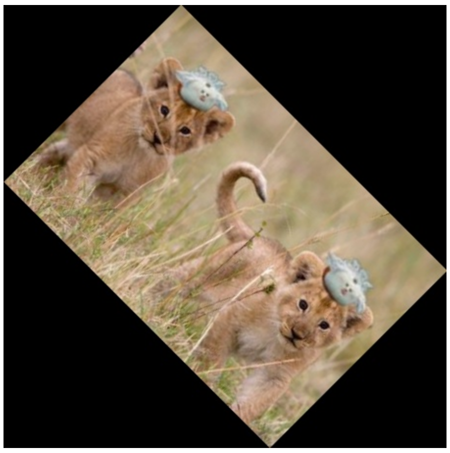


b) Flipped 180°

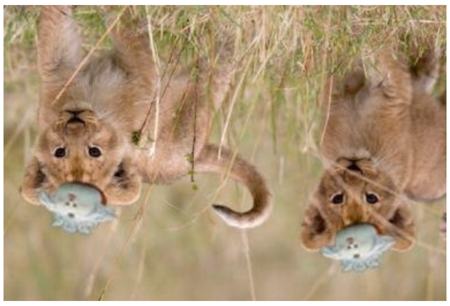


c) Mirrored images

Figure 5.7: F1, precision and recall score



(a) Vertical image



(b) Image rotated 180°

Figure 5.8: Demonstration of test image samples rotated with 45°and 180°.

Chapter 6

Discussion

The purpose of this chapter is to discuss the results and analysis from chapter Chapter 5. In addition, the discussion will answer the objectives of the research questions.

6.1 Dataset

To answer research question number three; *how do the models and pre-processing pipelines developed from RQ1 and RQ2 performs on tampered images that have been processed with other transformations than those present in the training images?*, we used the CASIA 2.0 dataset for training and testing. Normal testing images and the post-processed images serve to find out if transformations done to the image will have any effects on the model's capability of detecting tampered images. We observed that the ResNet-101 model depended on training and validating enough data samples to learn how to classify the images correctly. In contrast, the ELA-CNN model had the opposite issue and acquired fewer data samples during training to classify images in tests. Only testing samples were post-processed before ELA was applied. The image samples that were only going to be post-processed with different rotations could have been transformed after ELA was applied. However, one of the weaknesses of ELA is that information in the image can be lost when JPEG are resaved after some retouching or editing. Because of that, it was more suitable for the project to test the model on images that potentially had lost some information after post-processing.

6.2 Baseline testing

The first test was the baseline test, and it was done with testing images without any transformations to create a baseline for how well the models performed when they predicted data samples that had not yet been transformed. The baseline testing is part of the research done to answer research question 3 because the baseline test is used to determine how the model is performing in comparison with tests on transformed images. The results from the nine tests presented in Chapter 5 contribute to answering how well the model performed when it was represented with transformed images compared with the baseline test. One hypothesis was that the models would achieve the best scores in the baseline test because the test samples would be closest to the training samples. Another hypothesis was that models would not drop in performance when represented with images transformed with different rotation ranges because edges and corners around the tampered areas would not have been exposed to any extra post-processing.

6.3 Analysis

This section presents an analysis of the results that have been represented in this chapter. The analysis will take a deep dive into different variables in the tests and compare the respective tests.

6.3.1 ResNet-50 Results Analysis

The first baseline test done on testing images without any extra post-processing gave the ResNet-50 model the best testing results, see Figure 5.2, and it was the overall best test results compared with the two other models we tested in this research, shown in Figure 5.1a. Another interesting testing result is the mirrored image test, where the accuracy of ResNet-50 dropped a little bit, but not drastically as we would expect. One explanation can be that the JPEG compression after the transformation did not result in a too great feature loss. Another variable is that the mirrored transformation does not introduce an entirely new hypothesis space for the model, so it can still extract relevant features from the test images. By a too large hypothesis space, we mean too complex and too many different images for the model to extract useful features. The three tests with images compressed with three different qualities show a minimal difference in accuracy when comparing the results from JPEG compression with 80% and 90%. However, we expected the results to be more different when considering that JPEG compression is lossy and ELA can only highlight tampered areas as long as the image has not lost too much information in post-processing and compression processes. However, when we compared the results from JPEG compression with 60% quality, we saw a different result with a more significant drop in F1 score, see Figure 5.2, which can indicate that the test images contained less useful features that ELA could highlight.

The more significant difference in F1 score compared to recall and precision can also indicate that the model predicts several more data samples as the same class, compared with the two previous tests on JPEG compression. The two tests with Gaussian blur and brightening are two tests that involve typical post-

Table 6.1: An evaluation report from the JPEG compression 60 test of ResNet-50 with macro average and weighted average scores

	Precision	Recall	F1
Authentic	0.60	0.99	0.75
Tampered	0.77	0.06	0.12
Accuracy			0.61
Macro average	0.68	0.53	0.43
Weighted average	0.67	0.61	0.49

processing mechanisms used for hiding hard edges and lines that can reveal that something in the image has been tampered with. The test results are similar to the test results from the JPEG compression with 80%, so there is an indicator that the test images have lost some relevant features because of JPEG compression after the images were resaved after post-processing.

6.3.2 ResNet-101 Results Analysis

This section contains an analysis of the testing results of the proposed ResNet-101 model, and the testing results are shown in Figure 5.3.

The baseline test with normal testing images gave an F1 score of 70%, a precision score of 71%, and a recall score of 69%. Therefore, one of the hypotheses we had was that the trained models would achieve the best testing results on the test images closest to the training samples, which was not completely correct in this test. However, the ResNet-101 transfer learning model accomplished the best testing results in the test with mirrored images instead of the baseline test, even though the performance only increased by 1-2%. One explanation can be that the ResNet-101 model has trained with data augmentation, and it may have learned useful features from the rotation range, vertical flip, and horizontal flip augmentations. The model may also extract features with ELA because the JPEG compression after the transformation did not result in a too significant loss, and the images did not introduce a new hypothesis space for the model. In contrast, the tests on images transformed with brightening and Gaussian blur show that the detection rate is more affected by typical post-processing operations. It can be because edges and corners that pop out with ELA may be more smooth, and the image may have lost information due to JPEG compression when we resaved the photo after the post-processing. Our ResNet-101 model did not train on blurred and brightened images, so the new testing transformations may have introduced a much more complex hypothesis space.

The ResNet-101 model had its most significant performance drop in the test with images compressed with a 60% quality, where the F1 score decreased to 44%. The totality of the test indicates that the model classifies most of the images as the "*authentic*" class, which means that the test had many FP and FN predictions. The model accuracy in the two other tests on images compressed with 80% and 90% quality did not drop as much, and the results were similar to the Gaussian blur and brightening test. Out of the three tests with different JPEG compression, the ResNet-101 model achieved the best scores on the test with a compression quality of 90%, which we expected because the test samples were similar to the data the model had trained on. However, the results indicate that the model struggles to classify images as the correct class when the ELA pre-processing lose features. Also, the main difference between the prediction results from JPEG compression with 90% and JPEG compression with 80%, see confusion matrix in Table 5.1, is that the model predicts fewer false positives on images compressed with 90% quality, demonstrated in Table 5.2. One explanation for the significant difference between the scores in the JPEG compression tests is that ELA has fewer artifacts to light up in the image, which indicates that the features that the ResNet-101 model uses to classify images correctly are less present in the data samples compressed with 60% compared to images compressed with 80 and 90%.

The model achieved the second-lowest prediction results were achieved on the test on images flipped 45°. The F1 score is much lower than the precision and recall score, indicating that the model predicts many test samples as the same class, which gives many false positives and false negatives in the test. One can also argue that the tampered objects should have been highlighted by ELA. Still, because the 45°rotation introduced black corners, we created a new element that may have confused the model because it may have introduced new artifacts that the model was unfamiliar with. In other words, the 45°transformation may not be a practical test for a real-world scenario. However, the result can indicate that the inclusion of abnormal elements in images results in a more drastic performance drop. Another interesting variable is that the ResNet-101 model trained on images augmented with a rotation range of 30%. One would expect that the model would have been more prepared to detect rotated images, but training on a rotation range of 30% may have been too small compared to 45°. Another third variable is that the rotation range implementation in Keras may exclude black edges from rotation, height shift range, and vertical and horizontal flip augmentation methods. It indicates that the model is not extracting features from black edges.

6.3.3 ELA-CNN Results Analysis

We trained the ELA-CNN model on what we thought was the complete training and validation dataset, but because of a human error, the ELA-CNN model did a validation split on the training images, instead of using the initially intended validation data. We trained the model on the complete training and validation set after the bug was discovered, but the model did not produce any useful test results. Adding several more dense layers and dropout layers were also tested to create a bigger and more complicated network, but the tests did not improve the results any further. One theory is that images in the validation split may have been too complex images for the model to extract useful features from. Another variable is introducing a too large hypothesis space with many different and several more difficult images for the model to train on. The increase in number of data samples when the validation set was introduced can also be the cause to why the model was not able to produce results. Not producing any results is defined as all images was predicted as the "*authentic*" class. The solution to the issue was that the ELA-CNN model trained on 72% of the initially intended training data, and validated on the other 8% of the training set. The model produced useful testing results with the modified training and validation split. We tested the ELA-CNN model on the same testing sets as the ResNet-50 and ResNet-101 models. The ELA-CNN model had not trained or validated on any of the images in the testing set, but the model had trained on a less amount of images compared to the two ResNet models.

In the first test with baseline testing samples, the model achieves its best testing results with an F1 score of 55%, a precision score of 58%, and a recall score of 60%. The result from the baseline test was as expected, and it provided the best testing scores for the model. However, what was not expected was that the rest of the eight tests provided very similar accuracy scores as the baseline test. One test that stood out was the test on JPEG compression with a quality of 60%, where the model achieved a significantly low F1 score of 25%. The result can reveal that the compression has erased a significantly more amount of information that the ELA pre-processing is depending on and that the convolutional layers, in general, are struggling with extracting useful features for image classification.

One less likely explanation is that the model can detect post-processed tampered images on the same level as the baseline test because it is not as affected by the transformations as the two ResNet models. Another theory is that because the model was struggling to produce results when trained on the same training and validation sets as the ResNet models, it is probably not deep or advance enough to learn which features are helpful when distinguishing between tampered and legitimate images. One observation regarding the dataset that the ELA-CNN architecture was initially trained and tested on in the research of Sari et al. [50], is that the dataset is a small selection of images from CASIA 2.0, The images are also similar in the style of tampering. In other words, it trained on more straightforward data samples that may not require an as deep or advanced ML model to distinguish between tampered and legitimate images. However, in this research, the model has been tested on more sophisticated tampering than the dataset used in the study of [50]. In addition, it has been tested on images with extra post-processing as vel. One explanation for why the ELA-CNN model has a much lower performance than the ResNet networks is that the model's architecture is not complex enough to learn useful features. However, the depth of the transfer learning models of 50 and 101 layers also indicates that the ResNet models can extract more complex features in the deeper layers, compared with the depth of the architecture of the ELA-CNN [28]. Also, suppose the ELA-CNN is better at learning basic features such as edges and corners. In that case, the results from the research of Sari et al. [50] can be explained by the fact that a more unified dataset contains more similar edges and corners that are easier to learn for a less complex model.

6.4 Comparison of the Models

This section provides a comparison of the models that were tested in this research and related work. The results of ResNet-50 are also compared with performance and accuracy results from Hebbar et al. [53].

The ResNet-50 model was trained for 29 epochs which is eight epochs more compared with the 10 epochs the model was trained on in [53]. However, we improved the F1, precision, recall, and accuracy score. The training time per epoch was also reduced from taking 20 minutes per epoch in [53], to taking 108 seconds using the desktop, and 67 seconds on the Ubuntu server, shown in Table 6.2. Our improved F1, precision, and recall score results can be because of our fine-tuning of the ReduceLROnPlateau function with *factor=2* and *min_lr=0.001* was better. We selected the respective parameters after testing several hyper-parameters because the authors of [53] did not define which hyper-parameters they used. Table 6.3 shows a score comparison of the ResNet-50 model used in our research and scores from related work that have used ResNet-50 to detect CMF and SF.

The ResNet-50 did also achieve better performance compared with the proposed ResNet-101 model. One explanation to the difference in accuracy can lay in the depth of the two networks. The authors of residual network architecture [46] explains that deeper networks can have higher training errors because of the depth and dimensions in the architecture. The increase in training error does not indicate that the model is overfitting. Instead it has started to converge, and adding more layers and complexity is not the solution to the problem. In our case, we have added more layers on top of the ResNet-101 architecture which means that we have added more complexity to a already complex network.

Our proposed ResNet-50 model also got better accuracy and recall score than Jaiswal et al. [31] on testing on standard CASIA 2.0 images. Our ResNet-101 and ResNet-50 models did not achieve better performance on JPEG compression tests on the CASIA 2.0 dataset than Zhou et al. [37] on JPEG compression tests with quality 50% and 70% on the NIST16 [38] dataset, where they god an F1 pixel score of 67.7%. However, their model was trained with the augmentation method of image flipping, and the augmentation improved their results. Similarly, in our proposed ResNet-101 model, we used data augmentation to improve the model. In addition, we used several more augmentation methods, such as height shift range, rotation range and. Our results are not entirely comparable to the study of Zhou. They used a pixel-based F1 score and AUCscore; because they also detected the location of tampered areas.

Table 6.2: Table of seconds per epoch during training on desktop and Ubuntu server

Model	**Desktop**	**Ubuntu Server**
ResNet-101	155 seconds	88 seconds
ResNet-50	108 Seconds	67 seconds
ELA-CNN	8 seconds	6 seconds

Table 6.3: Comparison between the presented results from the ResNet-50 transfer learning model and related work that have used ResNet-50 in CMF and SF.

Model	**Dataset**	**F1**	**Precision**	**Recall**	**Accuracy**
ResNet-50 model	CASIA 2.0	98	98	98	98%
Nath et al. [68]	CASIA 2.0	95.4%	96.6%	94.1%	97.58%
Jaiswal et al. [31]	CASIA 2.0	-	-	63.37	70.26%

Compared with state-of-the-art research, our proposed ResNet-101 model has not achieved better performance than other studies that have used ResNet-101 as a backbone model for detecting CMF and SF. Pandey et al. [27] achieved and weighted average F1 score of 81%, but the precision, and recall scores are not provided in their research, so those metrics are not compared with our study. However, Pandey's transfer learning model did not use any custom layers on top of the ResNet-101 architecture. The added complexity of introducing more layers on top of the 101 layered ResNet-101 architecture can be one of the factors why our proposed ResNet-101 architecture did not achieve a better weighted average F1 score. Table 6.4 shows the difference in performance between our ResNet-101 model and related work that have used ResNet-101 in their architectures.

Ahmed et al. got better performance when using ResNet-50 than ResNet-101, but they used the same architecture on top of the backbone models, which is not done in this research where the ResNet models have different architectures on top of the pre-trained models. However, they concluded that using ResNet-101 did not improve the detection rate in their research. They concluded that useful features for detecting tampered objects are more fundamental features such as edges and corners, which can be extracted in the top layers. Therefore, they do not need

Table 6.4: A comparison between results from our proposed ResNet-101 model and related work that have used the ResNet-101 architecture for transfer learning to detect CMF and SF.

Model	Dataset	F1	Precision	Recall	AUC	F1 pixel	Accuracy
ResNet-101 model	CASIA 2.0	70%	71%	69%	-	-	69%
Pandey et al. [27]	CASIA 2.0	88%	-	-	-	-	
Kadam et al. [69]	CASIA 1.0	66%	67%	66%	-	-	
Zhou et al. [37]	CASIA 2.0/1.0	-	-	-	79.5%	79.5%	-

Table 6.5: Comparison of F1, precision and recall score with similar research with ResNet-101 as the backbone model. The - indicates that the authors have used other evaluation metrics. The F1 pixels score indicates how well the model detected pixels within a tampered object for object detection

deeper layers that are more difficult to train to extract the respective features. Liu et al. also considered using ResNet-101 as their feature extractor. Still, they concluded that the architecture involved too many deep layers with almost twice the amount of parameters compared with the ResNet-50 architecture. One of the variables that could have affected the final testing results in the baseline test on the proposed ResNet-101 model is that we introduced more custom layers with more parameters in a network that already consists of 101 layers. One of the more interesting results is that the model achieved almost the same score in the test with mirrored images. The result can be a coincidence, or it can also be because we chose a network with deep convolutional layers that learns other patterns, as well as the most standard patterns for edges and corners, so it was not affected by the test, compared with the ResNet-50 model in figure Figure 5.2. Another test that speaks against this theory regarding the ResNet-101 model being able to learn useful features in the deeper layers, which are helpful for geometric attacks, is the test with 180°flipped images. One can argue that the transformation is not too different from the transformation of the mirrored images, but the detection rate decreased. One theory is that objects within the picture have switched locations. However, it would be logical to assume that ELA would highlight tampered areas. Compared with the brightened and blurred testing images, the testing images would not have lost too much information when they were resaved. However, the test result was almost the same as the brightened image test.

6.5 Implementation issues

One issue with how ResNet-101 was implemented was that the deprecated Keras Application package might conflict with the new core Keras repository. The issues with the includes can also have a root cause in that the author had little experience with the implementation of transfer learning architectures. With that being said, the model was up and running after some trial and error, which is part of learning new technologies and skills. The author has learned significantly more about how transfer learning models are used in practice, residual networks, and why the architecture stands out compared to other models. New knowledge about how models are fine-tuned, how data augmentation affects learning, and the effect of early stopping to avoid overfitting has also been obtained during the project.

Chapter 7

Conclusion

We have presented three CNN models, one with a CNN architecture, and two of them were different transfer learning architectures. We trained the selection of different models on the CASIA 2.0 dataset to answer the first research question, "*Which combinations of backbone models, hyperparameters, and optimizers provide good accuracy and performance for the detection of CM and SF images in standard datasets?*". The analysis of the baseline test revealed that the ResNet-50 backbone model provided the best accuracy scores out of the three models. We also improved the training time per epoch. We improved the F1, precision, recall, and accuracy score of the ResNet-50 transfer learning model compared to the previous study that used the same architecture by fine-tuning the reduced learning rate function ReduceLROnPlateau. The results can also indicate that the ResNet-50 architecture provides a proper depth compared to the ResNet-101, which may be too deep to detect CM and SF, because of the converging issue with deeper networks. We also discovered that the ELA-CNN model did not provide valid results when trained and validated on the complete train and validation set. We conclude that the ELA-CNN architecture is not advanced and deep enough for the classification task in this thesis.

We used the pre-processing technique ELA to highlight tampered areas in images to answer the second research question, "Which image pre-processing techniques can assist in CMF and SF detection?". ELA has shown to be an effective pre-processing technique when it compares error levels with JPEG images that are not compressed several times, as proven in the baseline test of the ResNet-50 and ResNet-101 models. The drawback of ELA is the single point of failure with JPEG compression, which is also the factor that makes ELA brilliant as a pre-processing technique to highlight tampered objects. Results from the nine experiments in this thesis indicate that ELA is vulnerable when it is represented with images that have been compressed multiple times on, e.g. SM sites that use compression or after image editing. ELA was not proven to be as helpful for the ELA-CNN model. However, as described earlier, the results may have a root cause in the model's architecture, not necessarily because of the pre-processing technique.

We tested all the three models on the same nine test datasets, where eight of them were post-processed with a rotation, compression, blurring, or brightening attack before they were post-processed with ELA. The tests were done to answer the third research question, "How do the models and pre-processing pipelines developed from RQ1 and RQ2 perform on tampered images that have been processed with other transformations than those present in the training images?" The objective of the experiments was to test if we could trigger a failure in the detection accuracy in any of the models by presenting them with test images that they had not trained on. The results achieved in the proposed experiments revealed that the Two ResNet models achieved significantly lower scores when they were tested on images with different transformations from the training set. The only exception is the test with mirrored images, where the ResNet-101 achieved better accuracy scores than the baseline test, and the ResNet-50 achieved decent performance. The ELA-CNN model achieved almost identical accuracy scores on eight out of nine tests, supporting our claim regarding the lack of complexity in the model's architecture. The results from the experiments do also indicate that good detection accuracy in the baseline test did not equal good performance in the other test with different post-processings.

The overall results in this thesis show that the post-processing of images can affect the detection accuracy of CNN and transfer learning models when using ELA for pre-processing. Therefore, our recommendation for future work is to use other pre-processing techniques such as DCT and DWT with ResNet or other transfer learning models to detect image manipulation in post-processed images. We also recommend testing unfreezing of more layers in the ResNet-50 architecture to test if it can affect the detection accuracy.

Bibliography

[1] G. Pennycook and D. G. Rand, 'The psychology of fake news,' *Trends in cognitive sciences*, vol. 25, no. 5, pp. 388–402, 2021.

[2] Forskning.no, *De fleste unge følger nyheter i sosiale medier*, `https://forskning.no/barn-og-ungdom-media-medievitenskap/de-fleste-unge-folger-nyheter-i-sosiale-medier/1749251` [Accessed: May 4 2022], 2022.

[3] B. Singh and D. K. Sharma, 'Siteforge: Detecting and localizing forged images on microblogging platforms using deep convolutional neural network,' *Computers & Industrial Engineering*, vol. 162, p. 107 733, 2021.

[4] Paint.net, *Paint.net*, `https://www.getpaint.net/` [Accessed: May 25 2022], 2022.

[5] GIMP, *Gimp gnu image manipulation program[internet]*, Available from: `https://www.gimp.org/` cited 2021 November 7, 2021.

[6] Adobe, *Skap utrolige ting med photoshop.* `https://www.adobe.com/products/photoshop.html` cited; 2021 November 7, 2021.

[7] A. Diwan, R. Sharma, A. K. Roy and S. K. Mitra, 'Keypoint based comprehensive copy-move forgery detection,' *IET Image Processing*, 2021.

[8] A. Images, *Iran july 2008 distributed by the iranian revolutionary guard[internet]*, Available from: `http://www.alteredimagesbdc.org/sepah-news` [cited 2021 December 10]., 2021.

[9] J. Tijdink, R. Verbeke and Y. Smulders, 'Publication pressure and scientific misconduct in medical scientists,' *Journal of empirical research on human research ethics : JERHRE*, vol. 9, pp. 64–71, Dec. 2014. DOI: `10.1177/1556264614552421`.

[10] D. Mangal and D. K. Sharma, 'A framework for detection and validation of fake news via authorize source matching,' in *Micro-Electronics and Telecommunication Engineering*, D. K. Sharma, L. H. Son, R. Sharma and K. Cengiz, Eds., Singapore: Springer Singapore, 2021, pp. 577–586, ISBN: 978-981-33-4687-1.

[11] M. L. Mele, D. Millar and C. E. Rijnders, 'The web-based subjective quality assessment of an adaptive image compression plug-in.,' in *VISIGRAPP (2: HUCAPP)*, 2017, pp. 133–137.

[12] J. Dong, W. Wang and T. Tan, 'Casia image tampering detection evaluation database,' in *2013 IEEE China Summit and International Conference on Signal and Information Processing*, IEEE, 2013, pp. 422–426.

[13] N. Jindal *et al.*, 'Copy move and splicing forgery detection using deep convolution neural network, and semantic segmentation,' *Multimedia Tools and Applications*, vol. 80, no. 3, pp. 3571–3599, 2021.

[14] Y. Liu, H.-X. Wang, H.-Z. Wu and Y. Chen, 'An efficient copy-move detection algorithm based on superpixel segmentation and harris key-points,' in *International conference on cloud computing and security*, Springer, 2017, pp. 61–73.

[15] D. G. Lowe, 'Distinctive image features from scale-invariant keypoints,' *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[16] N. B. Abd Warif, A. W. A. Wahab, M. Y. I. Idris, R. Ramli, R. Salleh, S. Shamshirband and K.-K. R. Choo, 'Copy-move forgery detection: Survey, challenges and future directions,' *Journal of Network and Computer Applications*, vol. 75, pp. 259–278, 2016.

[17] I. Goodfellow, Y. Bengio and A. Courville, *Deep Learning*. MIT Press, 2016, `http://www.deeplearningbook.org`.

[18] I. Zafar, G. Tzanidou, R. Burton, N. Patel and L. Araujo, *Hands-on convolutional neural networks with TensorFlow: Solve computer vision problems with modeling in TensorFlow and Python*. Packt Publishing Ltd, 2018.

[19] Y. Ho and S. Wookey, 'The real-world-weight cross-entropy loss function: Modeling the costs of mislabeling,' *IEEE Access*, vol. 8, pp. 4806–4813, 2019.

[20] D. P. Kingma and J. Ba, 'Adam: A method for stochastic optimization,' *arXiv preprint arXiv:1412.6980*, 2014.

[21] V.-H. Nhu, N.-D. Hoang, H. Nguyen, P. T. T. Ngo, T. T. Bui, P. V. Hoa, P. Samui and D. T. Bui, 'Effectiveness assessment of keras based deep learning with different robust optimization algorithms for shallow landslide susceptibility mapping at tropical area,' *Catena*, vol. 188, p. 104 458, 2020.

[22] N. Goel, S. Kaur and R. Bala, 'Dual branch convolutional neural network for copy move forgery detection,' *IET Image Processing*, 2021.

[23] Keras, *Keras applications*, `https://keras.io/api/applications/` [Accessed: May 9. 2022], 2022.

[24] Z. Xu, K. Sun and J. Mao, 'Research on resnet101 network chemical reagent label image classification based on transfer learning,' in *2020 IEEE 2nd International Conference on Civil Aviation Safety and Information Technology (ICCASIT*, 2020, pp. 354–358. DOI: `10.1109/ICCASIT50869.2020.9368658`.

[25] Y. Wu, W. Abd-Almageed and P. Natarajan, 'Deep matching and validation network: An end-to-end solution to constrained image splicing localization and detection,' in *Proceedings of the 25th ACM international conference on Multimedia*, 2017, pp. 1480–1502.

[26] K. D. Kadam, S. Ahirrao and K. Kotecha, 'Efficient approach towards detection and identification of copy move and image splicing forgeries using mask r-cnn with mobilenet v1,' *Computational Intelligence and Neuroscience*, vol. 2022, 2022.

[27] A. Pandey and A. Mitra, 'Detecting and localizing copy-move and image-splicing forgery,' *arXiv preprint arXiv:2202.04069*, 2022.

[28] B. Ahmed, T. A. Gulliver and S. alZahir, 'Image splicing detection using mask-rcnn,' *Signal, Image and Video Processing*, vol. 14, no. 5, pp. 1035–1042, 2020.

[29] Y. Liu and X. Zhao, 'Constrained image splicing detection and localization with attention-aware encoder-decoder and atrous convolution,' *IEEE Access*, vol. 8, pp. 6729–6741, 2020.

[30] T. Pomari, G. Ruppert, E. Rezende, A. Rocha and T. Carvalho, 'Image splicing detection through illumination inconsistencies and deep learning,' in *2018 25th IEEE International Conference on Image Processing (ICIP)*, IEEE, 2018, pp. 3788–3792.

[31] A. K. Jaiswal and R. Srivastava, 'Image splicing detection using deep residual network,' in *Proceedings of 2nd International Conference on Advanced Computing and Software Engineering (ICACSE)*, 2019.

[32] M. Castro, D. M. Ballesteros and D. Renza, 'A dataset of 1050-tampered color and grayscale images (cg-1050),' *Data in brief*, vol. 28, p. 104 864, 2020.

[33] M. Castro, D. Ballesteros and D. Renza, 'Cg-1050: Original and tampered images (color and grayscale),' *Mendeley Data*, 2019.

[34] B. Wen, Y. Zhu, R. Subramanian, T.-T. Ng, X. Shen and S. Winkler, 'Coverage—a novel database for copy-move forgery detection,' in *2016 IEEE international conference on image processing (ICIP)*, IEEE, 2016, pp. 161–165.

[35] I. Amerini, L. Ballan, R. Caldelli, A. Del Bimbo and G. Serra, 'A sift-based forensic method for copy–move attack detection and transformation recovery,' *IEEE transactions on information forensics and security*, vol. 6, no. 3, pp. 1099–1110, 2011.

[36] E. Ardizzone, A. Bruno and G. Mazzola, 'Copy–move forgery detection by matching triangles of keypoints,' *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 10, pp. 2084–2094, 2015.

[37] P. Zhou, X. Han, V. I. Morariu and L. S. Davis, 'Learning rich features for image manipulation detection,' in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 1053–1061.

[38] H. Guan, M. Kozak, E. Robertson, Y. Lee, A. N. Yates, A. Delgado, D. Zhou, T. Kheyrkhah, J. Smith and J. Fiscus, 'Mfc datasets: Large-scale benchmark datasets for media forensic challenge evaluation,' in *2019 IEEE Winter Applications of Computer Vision Workshops (WACVW)*, IEEE, 2019, pp. 63–72.

[39] Y. Rodriguez-Ortega, D. M. Ballesteros and D. Renza, 'Copy-move forgery detection (cmfd) using deep learning for image and video forensics,' *Journal of Imaging*, vol. 7, no. 3, p. 59, 2021.

[40] N. T. Pham, J.-W. Lee, G.-R. Kwon and C.-S. Park, 'Hybrid image-retrieval method for image-splicing validation,' *Symmetry*, vol. 11, no. 1, p. 83, 2019.

[41] S. Walia, K. Kumar, M. Kumar and X.-Z. Gao, 'Fusion of handcrafted and deep features for forgery detection in digital images,' *IEEE Access*, vol. 9, pp. 99 742–99 755, 2021.

[42] N. Goel, S. Kaur and R. Bala, 'Dual branch convolutional neural network for copy move forgery detection,' *IET Image Processing*, vol. 15, no. 3, pp. 656–665, 2021.

[43] X. Ying, 'An overview of overfitting and its solutions,' in *Journal of Physics: Conference Series*, IOP Publishing, vol. 1168, 2019, p. 022 022.

[44] TensorFlow, *Classification on imbalanced data*, `https://www.tensorflow.org/tutorials/structured_data/imbalanced_data` [Accessed: May 12 2022], 2022.

[45] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, 'Dropout: A simple way to prevent neural networks from overfitting,' *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[46] K. He, X. Zhang, S. Ren and J. Sun, 'Deep residual learning for image recognition,' in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778. DOI: `10.1109/CVPR.2016.90`.

[47] W. Commons, *File:resnet.png — wikimedia commons, the free media repository*, [Online; accessed 30-May-2022], 2020. [Online]. Available: `https://commons.wikimedia.org/w/index.php?title=File:Resnet.png&oldid=500024146`.

[48] T. S. Gunawan, S. A. M. Hanafiah, M. Kartiwi, N. Ismail, N. F. Za'bah and A. N. Nordin, 'Development of photo forensics algorithm by detecting photoshop manipulation using error level analysis,' *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 7, no. 1, pp. 131–137, 2017.

[49] S. Jabeen, U. G. Khan, R. Iqbal, M. Mukherjee and J. Lloret, 'A deep multimodal system for provenance filtering with universal forgery detection and localization,' *Multimedia Tools and Applications*, vol. 80, no. 11, pp. 17 025–17 044, 2021.

[50] W. P. Sari and H. Fahmi, 'The effect of error level analysis on the image forgery detection using deep learning,' *Kinetik: Game Technology, Information System, Computer Network, Computing, Electronics, and Control*, 2021.

[51] P. Sovathana, *Casia dataset*, `https://www.kaggle.com/sophatvathana/casia-dataset` [Accessed: May 20 2022], 2018.

[52] Image-net.org, *About imagenet*, `https://www.image-net.org/about.php` [Accessed: May 9 2022], 2020.

[53] N. K. Hebbar and A. S. Kunte, 'Transfer learning approach for splicing and copy-move image tampering detection,' *ICTACT Journal on Image and Video Processing*, vol. 11, no. 4, pp. 2447–2452, 2021.

[54] C. Shorten and T. M. Khoshgoftaar, 'A survey on image data augmentation for deep learning,' *Journal of big data*, vol. 6, no. 1, pp. 1–48, 2019.

[55] E. Obrestad, *Using gpu instances*, `https://www.ntnu.no/wiki/display/skyhigh/Using+GPU+instances` [Accessed: May 5 2022], 2022.

[56] A. Clark, *Image module*, `https://pillow.readthedocs.io/en/stable/reference/Image.html` [Accessed: May 25 2022], 2022.

[57] A. Clark, *Pillow*, `https://pillow.readthedocs.io/en/stable/handbook/image-file-formats.html` [Accessed: May 25 2022], 2022.

[58] Keras, *Resnet and resnetv2*, `https://keras.io/api/applications/resnet/` [Accessed: May 7. 2022], 2022.

[59] Keras, *Reducelronplateau*, `https://keras.io/api/callbacks/reduce_lr_on_plateau/` [Accessed: May 12 2022], 2022.

[60] Keras, *Layer weight regularizers*, `https://keras.io/api/layers/regularizers/` [Accessed: May 13 2022], 2022.

[61] Keras, *Simple. flexible. powerful.* `https://keras.io/` [Accessed: May 15 2022], 2022.

[62] TensorFlow, *An end-to-end open source machine learning platform*, `https://www.tensorflow.org/` [Accessed: May 15 2022], 2022.

[63] TensorFlow, *Tf.keras.model*, `https://www.tensorflow.org/api_docs/python/tf/keras/Model` [Accessed: May 15 2022], 2022.

[64] Scikit-learn, *Machine learning in python*, `https://scikit-learn.org/stable/` [Accessed: May 15 2022], 2022.

[65] A. Dixit and S. Bag, 'A fast technique to detect copy-move image forgery with reflection and non-affine transformation attacks,' *Expert Systems with Applications*, vol. 182, p. 115 282, 2021.

[66] K. Waltz and A. Gonzalez, *Demystifying gaussian blur.* `https://www.adobe.com/creativecloud/photography/discover/gaussian-blur.html` [Accessed: May 16 2022], 2022.

[67] O. team, *About*, `https://opencv.org/about/` [Accessed: May 16 2022], 2022.

[68] S. Nath and R. Naskar, 'Automated image splicing detection using deep cnn-learned features and ann-based classifier,' *Signal, Image and Video Processing*, vol. 15, no. 7, pp. 1601–1608, 2021.

[69] K. Kadam, S. Ahirrao, K. Kotecha and S. Sahu, 'Detection and localization of multiple image splicing using mobilenet v1,' *IEEE Access*, vol. 9, pp. 162 499–162 519, 2021.

NTNU
Norwegian University of
Science and Technology