CHAPTER 3
COPY MOVE FORGERY DETECTION USING STATISTICAL AND DATA-DRIVEN TECHNIQUES

Copy-Move forgery is a type of digital image forgery, in which the region of an image is being copied and pasted to the different parts of the same image to conceal the information of the image. State-of-the-art techniques are reported by researchers in various literature to detect this type of forgery. In this thesis, these techniques are classified into three different classes. The first is block-based, the second is key-point based and the third is data-driven based techniques. There are a lot of limitations to these techniques. Considering these limitations some methods are developed to detect this type of forgery in the digital image. This chapter of the thesis is dedicated to the proposed methods for the detection of copy-move forgery. The chapter is divided into four sections- background and research gaps of reported literature are discussed in the first and second section; proposed models are given in the third section with their experimental result and analysis; summary of the models are given in the fourth section. The third section (proposed models) is divided into two subsections- in the first subsection, a combination of the block-based and key-point-based techniques is developed to overcome the challenges of individuals. A data-driven deep learning technique is developed in another subsection to overcome the challenges of the conventional technique. This chapter also provides an evaluation of proposed approaches on different datasets and their performance is also compared with other state-of-the-art techniques.

3.1 Background

Copy-move forgery is one of the major image forgery techniques. In this, a duplicate patch of the digital image is placed over the targeted area with or without any

image processing operation. If duplication is done without any operation, patches will be identical but if duplication is done with any operation patches become un-identical. This is performed either to enhance the visual content of the image or to hide the essential information. The replaced region is from the same image, therefore, the texture and pattern of the replaced patch may same as that of the original image. Hence it is nearly impossible to recognize with the naked human eye (see Figure 3.1). So, to establish the veracity of an image and to identify the authenticity of an image forensic tool is needed.

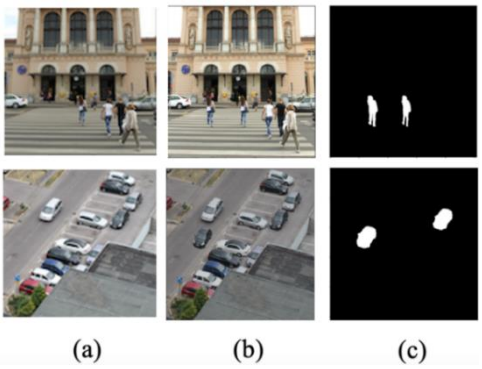

Figure 3.1: Examples of Copy Move Forgery (CoMoFoD dataset [68]*) (a) Original Image (b) Forged Image (c) Ground Truth mask of Forged Image*

Copy-Move Forgery Detection (CMFD) is one of the passive forgery detection techniques. The first CMFD technique was given by Jessica Fridrich et. al [27] in 2003. This is a statistical-based method in which the image is divided into patches and patches are matched together by two different methods. One is an exact match, and another is a robust match. However, this method works well when no operation is performed over the duplicated region. But this method fails with any type of geometrical transformation of divided block e.g. rotation (when rotation angle is more than 5^0) and scaling. After this, a lot of methods came based on block-matching during the last two decades [28], [29], [31], [69]–[71]. During copy-move forgery, several post-processing techniques act as hurdles towards its detection. Several techniques like contrast adjustment, brightness change, blurring, noise addition, jpeg and other compression and transformation

techniques thwart the efficient process of CMFD. Block matching techniques based on DCT can handle such post-processing techniques but is not invariant to scaling, rotation, and other major geometric transformation [26]. Most copy-move image forgery detection techniques fail to detect multiple image forgeries which have undergone such transformations on a single image. Geometric transformations mentioned above can be tackled using key-point-based matching techniques [36], [37], [72]. Techniques like SIFT which are invariant to rotation and scaling are based on their descriptors and key points. But the problem with these methods is these methods have poor performance when duplicated regions are very small. Deep learning CNN models are gaining huge momentum in almost every application of image processing and computer vision. The deep learning networks are highly inspired by human networks that are biological neurons, which constitute multiple nonlinear layers for processing simple objects parallelly. From 2018 to 2020, a lot of CMFD methods have been given using deep learning methods [39], [41], [73]. But they are either only classification methods (i.e. image analysis not content) or have poor performance with geometrical transformation.

3.2 Research Gaps

From the literature survey of copy-move forgery detection given in chapter 2, the following points summarize the identified challenges and limitations:

- Block-based CMFD approaches are not invariant to geometric transformation (especially scaling and rotation).
- Mild processing operations such as brightness enhancement, contrast stretching is not invariant to keypoint based features. In such cases, key-point extraction-based CMFD approaches don’t work well.
- Machine learning techniques don’t give better results than deep learning techniques and don’t localize the forged regions in an image.
- Based on these challenges an approach based on both types of features can be made, which can be invariant to both mild processing and geometric

transformation. Well, a block-based approach uses overlapped blocks to extract features which takes much time to detect the forged region. And the combination of the two approaches can take a higher computation cost for the forgery detection.

These research gaps motivated us to develop CMFD techniques that can determine the forged image as well as a forged location in the image. In this regard, two different techniques are developed to detect the copy-move forgery in an image. One is key-point cum block-based and another is a data-driven deep learning technique. The following section (Proposed Models) is dedicated to these techniques.

3.3 Proposed Models

Some challenges and gaps were found in the reported literature of CMFD. The proposed models tried to fill those gaps. In this section, two techniques are discussed. The first one is a combination of key-point extraction and the block-based the and the second one is based on a deep learning model. Here these techniques are discussed one by one-

3.3.1 Copy-Move Image Forgery Detection using DCT and ORB Feature Set

In this work, an approach is presented to detect and locate copy-move forged regions in an altered image. It utilizes a combination of both block-based along with a traditional key-point-based method. Invariance to contrast changes, brightness changes, Gaussian noise, and some degree of blurring is provided by block-based methods. On the contrary, key-point-based methods have an invariance towards geometric altercations like rotation changes and scale changes on image segments. Employing a combination of DCT (a block-based method) and ORB [74] (a key-point-based method) aforementioned attacks can be tackled. Compared to other key-point/block-based, the proposed method is robust towards geometrical transformation. Thus, it is more pragmatically applicable. The technique is also robust towards multiple forgeries in a single image.

3.3.1.1 Method and Model

The proposed work is segmented into the four following sub-sections and the overall process is depicted in the framework given in Figure 3.2.

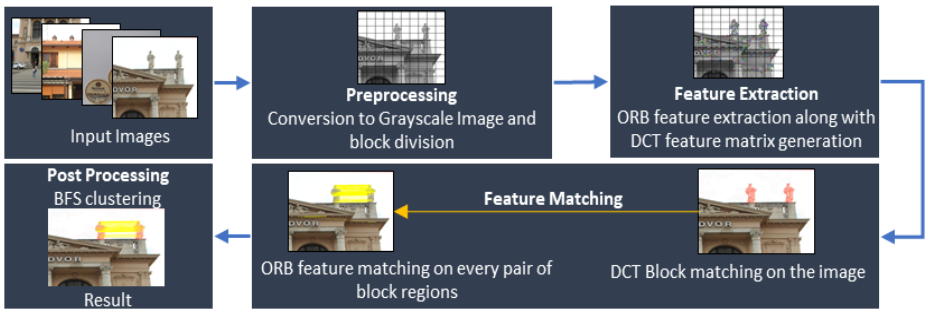

Figure 3.2: The framework of the proposed CMFD technique

3.3.1.1.1 Pre-processing

The image on which CMFD is required to be done is converted to a grayscale image if it already isn't. Further processes are done on this grayscale image since color doesn't play a significant role in the process. To apply DCT to the image, it is required to be divided into overlapping blocks. 8×8 pixel-sized blocks were utilized to meet this purpose. Key points are distributed in several regions within the image. 50×50 pixel-sized non-overlapping blocks were used to divide the image into regions that housed the key points in that particular region. Since discrete cosine transform is applied to all the overlapping blocks in the image, if the size of the image is extremely large, it would take a large amount of time to convert all the blocks to their transform vector representations.

3.3.1.1.2 Feature Extraction

Two features are extracted from the image, these are-

DCT Feature: The blocks are square in dimension. Let the region size be b=8, if the dimensions of the image are p × q pixels then the number of blocks for DCT will be (p-b+1) × (q-b+1). Here every block will contain a feature vector the size of the feature vector space will then be (p-b+1) × (q-b+1) ×b^2 for the image. These features are extracted

by sliding over a b × b sized block over the entire image. The feature space is now a 3-dimensional matrix. DCT is now applied to each of the two-dimensional blocks. The feature vector is extracted as shown in Figure 3.3.

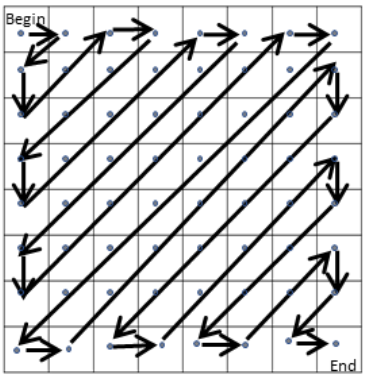

Figure 3.3: The order in which a block's features are extracted. Coefficients on the diagonal have the same frequency

Features are extracted in this fashion since the lower frequency terms in a DCT are rather more important in determining the true form of an image than the higher frequency terms which can even be avoided. Thus, we take only the coefficients until the diagonal of the square block. (For example, in this case of an 8×8 block first 36 features are only extracted). This is done to decrease further computational costs. The 36-length vector is appended with the Cartesian coordinates of the first block. The vector's length is now 38. Now, we have a two-dimensional feature space. This dimensional feature space is now lexicographically sorted.

ORB Feature: The key points and their respective descriptors are obtained using Oriented FAST and Rotated BRIEF respectively [74]. The key points are drawn out using the FAST(Features from Accelerated Segment Test) algorithm [75].

Oriented FAST: The FAST [75] method focuses on every pixel and a corresponding 16 length feature vector obtained by a radius of 3 from the point of the pixel. This is the Bresenham circle as depicted in Figure 3.4. The features are then divided into three different classes based on a fixed threshold (for example 20% of the intensity

of the pixel. The 16 values are assigned the three different states using the following equation:

$$S_{pc} \coloneqq \begin{cases} 1, & I_{pc} \leq I_{px} - T_h \\ 2, & I_{px} - T_h < I_{pc} < I_{px} + T_h \\ 3, & I_{px} + T_h \leq I_{pc} \end{cases} \tag{3.1}$$

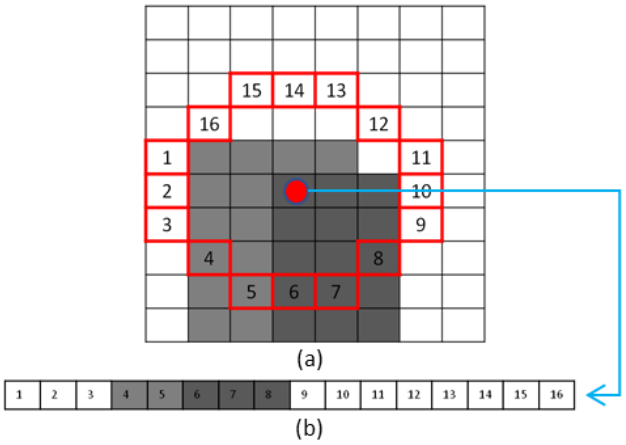

Figure 3.4: (a)The red dot depicts the pixel under consideration. The surrounding pixels values that correspond to its feature are depicted with a red border. (b)The extracted feature vector of length 16 [75]

Where S_{pc} is the state of the pixel and I_{px} is the intensity of the pixel which is described by the feature vector. I_{pc} is the intensity value of the pixel under consideration. The pixel to be classified as a key-point is determined by a variable I_{px} which is true if the point is to be considered as a key-point and false otherwise. Each subset is fed to an ID3 algorithm which classifies the key-point. ID3 algorithm works on entropy minimization. This is applied recursively to all three subsets. The entropy for a set *P* is given as:

$$H(P) \coloneqq (c + \bar{c})log_2(c + \bar{c}) - clog_2 c - \bar{c}log_2 \bar{c} \tag{3.2}$$

Here *H(P)* is the entropy, *c* denotes the number of components in the positive class and $\bar{c}$ quantifies the others. (The number of corners accounts in the positive class and the number of non-corners accounts in the other class). To produce multi-scale features, a scale pyramid of the image is used, and FAST features are filtered by the Harris corner filter at each level of the pyramid. A pyramid of scales is utilized along with FAST

features refined by Harris corner detection at every level of the pyramid to extract multiple scaled features.

To find the inclination of the FAST key points [76] the intensity centroid as a measure is used. It is expected that a point's intensity value is not aligned at that point, and this vector can be utilized to find the inclination. The moment of order (r + s) of a key-point P is given below. The neighborhood NB(x, y) is located at the first quadrant with P as origin in a Cartesian coordinate system. The resulting point of interest is called an oFAST point (oriented FAST point).

$$IM_{r,s} = \sum_{x,y} x^r y^s I(x,y) \tag{3.3}$$

The Centroid of the neighborhood is given by:

$$C = \left(\frac{IM_{10}}{IM_{00}}, \frac{IM_{01}}{IM_{00}}\right) \tag{3.4}$$

The orientation θ of the keypoint 'P' is the angle formed by $\overrightarrow{OC}$ which is determined by the equation given below where atan2 signifies a variant of arctan that also denotes the quadrant.

$$\theta = atan2(IM_{01}, IM_{10}) \tag{3.5}$$

rBRIEF Descriptors: These descriptors are a string of binary values that describe a patch of smooth image (*Sim*) from a set of binary intensity tests [77]. These binary tests are described as:

$$T(Sim; x_1, x_2) \coloneqq \begin{cases} 1, & Sim(x_1) < Sim(x_2) \\ 0, & Sim(x_1) \geq Sim(x_2) \end{cases} \tag{3.6}$$

Here, *Sim(x)* represents the intensity of *Sim* at a point *x.*

A feature consists of many binary tests. The feature vector is defined as follows:

$$FV_n(Sim) \coloneqq \sum_{i=1}^{n} 2^{i-1} T(Sim; x_{1i}, x_{2i}) \tag{3.7}$$

A vector of length 256 is chosen and a Gaussian distribution of tests around the center of the patch was used. To make this invariant to the same plane rotation steered BRIEF[77] features are used. An efficient way to perform steering of BRIEF is through the orientation of oFAST point x_i. Let S be a two-dimensional matrix given by S and rotation matrix for orientation θ_i is given by R_θ and S_θ by $R_\theta.P$:

$$\sum = \begin{pmatrix} x_{11}, \dots x_{1n} \\ x_{21}, \dots x_{2n} \end{pmatrix} R_{\theta_i} = \begin{bmatrix} cos\,\theta_i & -sin\,\theta_i \\ sin\,\theta_i & cos\,\theta_i \end{bmatrix} \tag{3.8}$$

$$S_\theta \coloneqq R_{\theta_i}.S$$

The ORB descriptors of oFAST are given by:

$$ORB(i) = FV_n(Sim) | (x_{1i}, x_{2i}) \in S_\theta \tag{3.9}$$

ORB (i) extraction is implemented on every scale. Each of these features and key points is assigned in their particular regions based on their positions in the image. Extremely blurred regions and perfectly geometrical shapes like a perfect circle without any detail form a roadblock for feature extraction as they generally do not contain distinctively contrasting areas for the identification and extraction of interesting key points.

3.3.1.1.3 Feature Matching

Every adjacent pair of rows in the lexicographically sorted feature space is compared. Rows are viewed as indistinguishable if the Euclidean separation between the two is not beyond a defined threshold(D_T). A shift vector is calculated for similar rows and the count of the shift vector is incremented. Finally, all the shift vector counts are checked and the ones that exceed a particular count threshold(C_T), the blocks that constitute it are considered as matched blocks.

For matching, features of ORB key points a brute force k-Nearest Neighbors (k-NN) method is used. For every pair of regions in the image as described in the above section (Feature Extraction) that houses the extracted key points and their descriptors, the above method is applied. The k-NN matcher returns the best k matches based on Hamming distance between the descriptors. We use hamming distance since we have binary descriptors for the key points. Here we take k=2 and apply Lowe's ratio test [78] to gain accurate matches between key points. The resulting matches are appended to a matched list.

3.3.1.1.4 Post Processing

To reduce false matches and refine the resulting image a breadth-first search-based clustering technique is implemented. The original and the key-point that is matched are clustered separately for every key point that is included in matches. The clusters are created based on a distance parameter, D_P and a grouping size parameter G_P. When the breadth-first search is applied to cluster key-points into a single cluster, key-points that are at a distance less than D_P are only considered to be in that cluster. This process is continued until all the key points have been clustered. Now, only those clusters whose membership count exceeds G_P are considered in the final clusters that make up the matches. If both the matched key points are a part of the final clusters then these matches are included in the resulting final matches.

Finally, the block-based matches along with key-point matches are mapped onto the image for depicting the forged regions.

3.3.1.2 Result Analysis and Discussion

The experiment is performed on Microsoft Windows 10 Home with system type x64-based PC and Intel(R) Core™ i5-8300H CPU of clock speed 2.30GHz with 4 Core(s) Processor. The system has installed physical memory of 8 GB (RAM).

The process is tested on the CoMoFoD dataset [68] in the small image category (discussed in the theoretical background of chapter 2). Various transformations like translation, rotation, scaling, and distortion have been applied to the images to obtain the forged images. Every image includes a bunch of an image, a mask with color, a black and white mask for the forged image. Every image set also consists of a blurred image set, a set of images with added noise, a set of images with changed brightness, a color-reduced image set, and a contrast-adjusted image set.

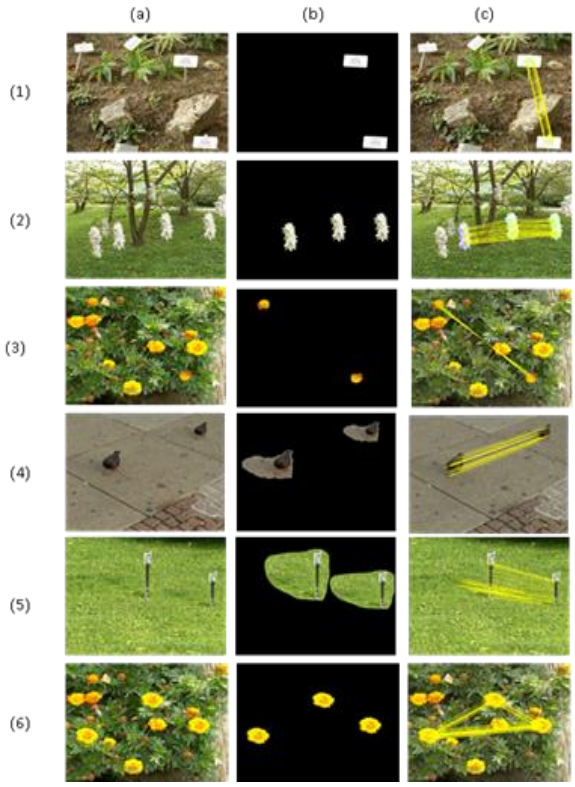

Figure 3.5: Image [a1-a6]: Forged Images where a1: Copy-move, a2: multiple copy-move, a3: copy-rotate-move, a4: copy-scale-move, a5: copy-scale-move, a6: combination of all; [b1-b6]: Ground truth images related to [a1-a6]; and [c1-c6]: Results of the proposed methods

The presented technique is assessed on sets of 200 PNG images from the CoMoFoD small dataset [68]. Images 001_F to 40_F consist of plain copy-move forgery attack, images 041_F to 080_F consist of copy-rotate-move forgery attack, images 081_F to 120_F consists of scaled copy-move forgery attack, images 121_F to 160_F consist of distorted copy-move forgery attack while images from 161_F to 200_F consist of images

with a combination of aforementioned attacks. Hence, each group consists of 40 images. For DCT feature matching D_T = 0.001 and C_T = 200. For ORB the, a maximum number of key points is set at 30000, the distance parameter D_P is set to 18 while the grouping parameter G_P is set to 6.

Table 3.1: Performance evaluation on simple copy-move forgery attack on CoMoFoD dataset

Type of attack	*p* (Precision)	*r* (Recall)	*f1* (F1-Score)
Translation	88.63	97.50	92.85
Rotation	88.09	92.50	90.24
Scaling	80.95	85.00	82.92
Distortion	85.00	85.00	85.00
Combination	78.90	75.00	76.90

The performance metrics on these groups are depicted in Table 3.1. The method is evaluated along with BusterNet [79] and Zernike [80]. The result of comparison on the base set with no post-processing operations performed on 200 images (001_F to 200_F) is given in Table 3.2.

Table 3.2: Performance comparison. #Passed represents the number of forged images that were successfully detected from a set of 200 forged images

Technique	#Passed	*p* (Precision)	*r* (Recall)	*f1* (F1-Score)
Zernike [80]	90	96.27	69.84	79.93
BusterNet [79]	117	83.52	78.75	63.13
Proposed	174	84.31	87	85.58

The robustness towards scaling rotation distortion and multiple copy-move forgeries are depicted in Figure 3.6. From performance curves depicted in Figure 3.6(a) to Figure 3.6(e), it can be concluded that the method outperforms other techniques by a large margin in detecting copy-move forged images even when post-processing operations like Brightness change, Color reduction, Contrast Adjustment, and added noise. The proposed method is susceptible to blurring as in a blurred image a large number of key points cannot be found and hence the dipping curve. Zernike [80] fails to detect forged images in images with high noise while the proposed technique is effective in

detecting forged regions in different types of noisy images. Even though Zernike [80] is algebraically invariant to geometric transformations, quantization and interpolation errors plague the technique hence, it is still underperforming under images forged by copied regions with different scaling and radical rotational angles. BusterNet [79] is limited by training over synthetic data and hence fails to correctly determine copied regions that undergo professional editing that is usually imperceptible to the human eye. From the curves, it can be seen that the proposed technique is quite robust to different post-processing operations applied along with different geometric transformations and is thus effectively better at determining forged images.

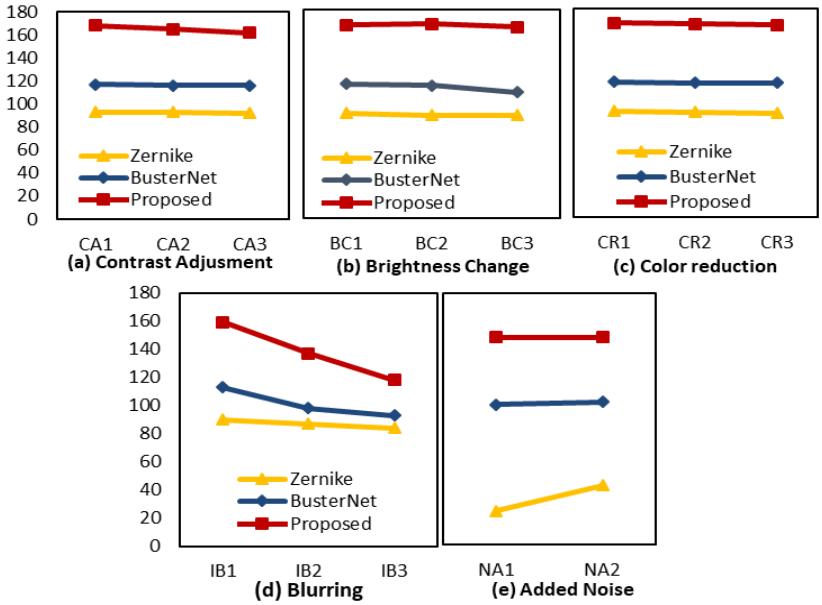

Figure 3.6: (a) to (e) depicts the comparison charts for various levels of post-processing operations and the respective number of images passed by the techniques

3.3.2 Detection of copy-move forgery in digital image using a multi-scale, multi-stage deep learning model

The concept of the multi-stage and multi-scale image has been used here to develop a deep learning-based CMFD technique. In this work, a deep learning CNN network has been developed that has two different- parts one is an encoder and another is the decoder. The major contributions that have been made in this work are-

- Proposed deep learning-based CMFD technique using multi-scale input image with a multi-stage convolutional network to overcome the challenge of scale-invariant.
- Quantitative and visual result analysis of the proposed model on two different publicly available datasets using different performance measures. The result produced by the proposed model is compared with other state-of-the-art techniques.

3.3.2.1 Method and Model

This section gives an architecture of the proposed model for CMFD based on the deep learning CNN model. In case copied objects are scaled or rotated and pasted in the image, most of the existing techniques are unable to detect the manipulated region. Therefore, a method is required that should be scale and rotation invariant. Deep learning CNN models are gaining huge momentum in almost every application of image processing and computer vision. This motivated us to develop the CMFD technique using the deep convolutional network (Conv-Net). The proposed architecture is purely designed for the segmentation of copied and pasted objects to the same image. To tackle the challenge of scaling, the concept of multiple scales input of image is taken from [81] and features are extracted at multiple levels which gives an advantage of scaling robustness. These multiple scales are used to extract features at multiple levels. Then from different levels of different layers, these features are concatenated with the first level of different pooled layers (Figure 3.7).

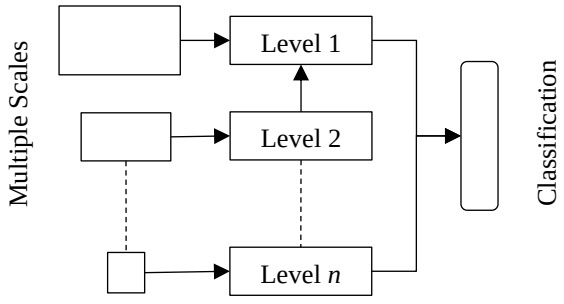

Figure 3.7: Visual Representation of Multi-Scale Network

3.3.2.1.1 Proposed Multi-Scale Multi-Stage Network

This architecture has an encoder network and corresponding to this encoder network, a decoder network exists. Followed by decoder network a sigmoid activation function is there for pixel-wise classification. In this way, the whole architecture is divided into three phases first is the encoder phase second is the decoder phase and the third is the classification phase as shown in Figure 3.8. These three phases are explained below subsections:

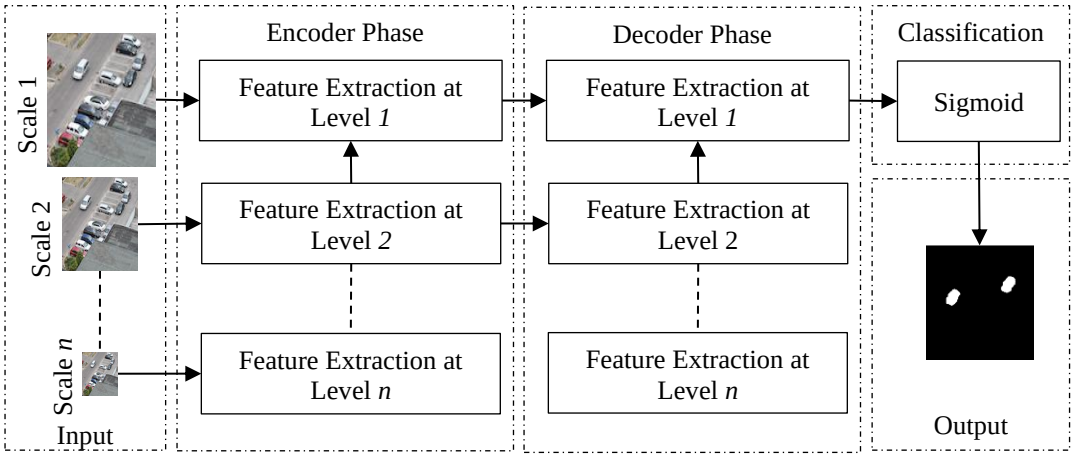

Figure 3.8: Block-Diagram of the Proposed Model

Encoder Phase: First input images are scaled(half-sampled) multiple times for multiple levels of the architecture. At every level of the model scaled image is taken as input. The proposed model takes an input image of dimension 256×256 and this input image is half-sampled to the dimension of 16×16. The scaled image is then passed through the convolution layer, and batch normalization layer. Then elementwise rectified linear non-linearity (ReLU) activation function ($\max[0, x]$) is applied over normalized feature space. At the first level of the model the activated feature space is again down-sampled i.e. max-pooling is applied over 2×2 patches. However, max pooling is not beneficial for object segmentation when input images are much scaled, in the proposed method multiple scaled are used and higher dimension feature space is used automatically. Then this max-pooled layer is concatenated with activated feature space

of similar dimension of next level as shown in Figure 3.9. These layers are continued till their lowest scaled input dimension. Also, at each convolution layer from beginning to end of the encoder phase number of filters are increasing. For every convolution layer, stride value is taken as one, and padding is taken as 'same' i.e. feature space should be covered fully and output dimension should be the same as the input dimension.

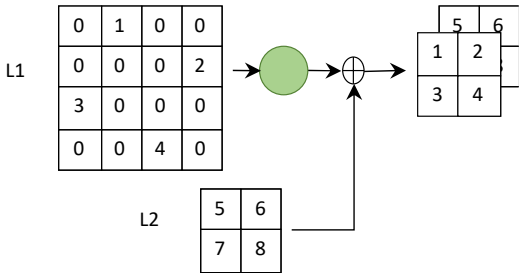

Figure 3.9: An Illustration of max-pooling of activated feature space and then the concatenation of another level feature space with first level feature space

Decoder Phase: After the encoder phase of the model, the dimension of the output feature gets smaller which is not appropriate for the pixel-wise localization of the manipulated region. To localize the forged region in tampered image training of the model is done with a ground truth segmented image of the input image. The dimension of the output feature space should be sufficient for visualization as well as to segment the manipulated object. Therefore, the output feature of the encoder phase needs to be upsampled. So, corresponding to each max-pooling layer of the encoder phase there is one upsampling layer is added in the decoder phase. This upsampling is also done using a 2×2 window size. The output feature of the upsampling layer is then convoluted with the number of filters. The convoluted features are batch normalized and then activated through the ReLU activation function. Every level of activated feature space is concatenated with the first level of corresponding output of upsampled layer as shown in Figure 3.10. This process is continued until the dimension of the output feature is not match the dimension of the input image.

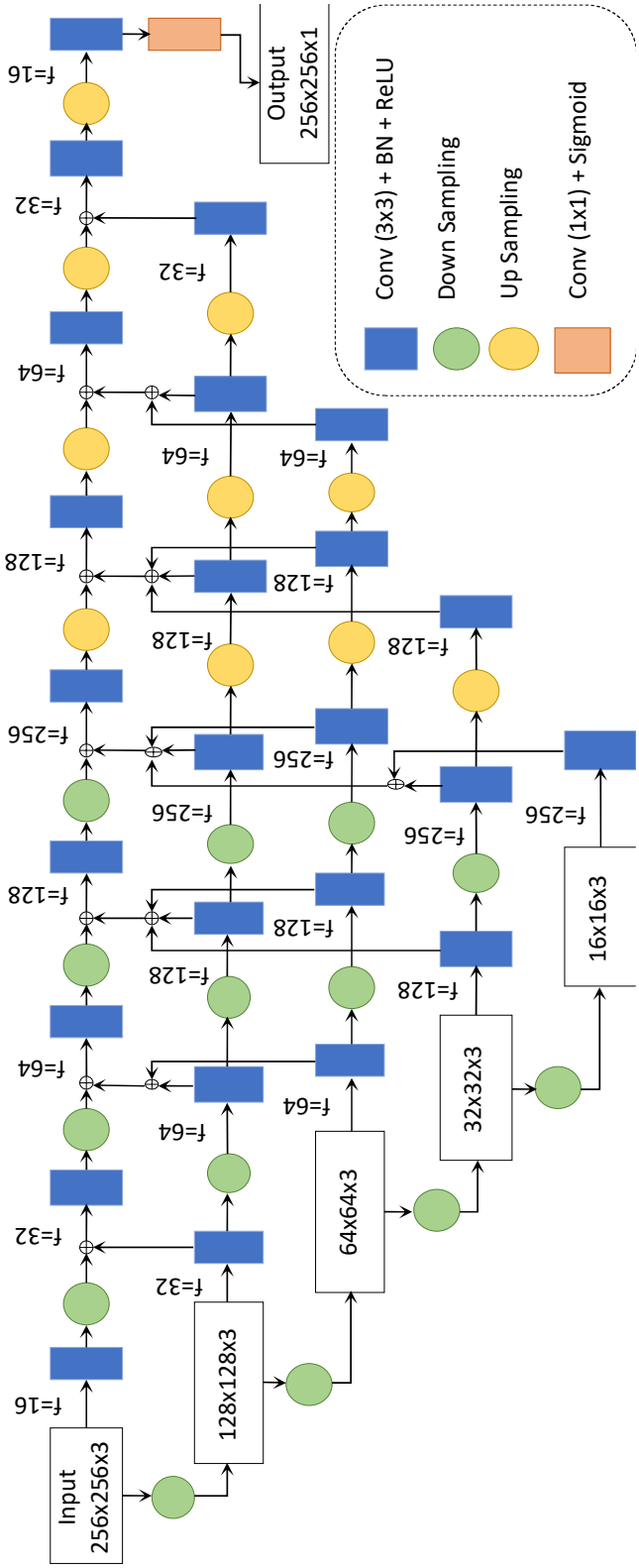

Figure 3.10: Architecture of proposed model for copy-move forgery detection using deep learning CNN model

Classification Phase: The decoder phase of the model gives higher depth output feature space but the same height and width as the input image. To train the classifier we need a single depth feature space whose height and width should be the same as the input ground truth image. Thus, higher depth feature space is convoluted with a 1×1 kernel

size filter with 'same' padding and stride value 'one'. This convoluted output feature is then passed through the sigmoid activation function which is generally used for binary class classification problems. To detect and localize manipulated regions in given input image pixels of the image should be classified into white and black pixels i.e. forged and authentic pixels. This is a binary class classification problem and the sigmoid activation function is better for such problems. Suppose y is input to the sigmoid function [82] and $f(x)$ is the feature space of the output layer of the model and 'w' is weight then-

$$y = w \times f(x) + b \tag{3.10}$$

Here, 'b' is a bias value-added to the function. If we assume that the predicted class of pixel $I = 0$ denotes that pixel is forged and $I = 1$ denotes that pixel is authentic, then probabilities using function will be-

$$p(I = 0) = \frac{1}{1 + e^{-y}} \tag{3.11}$$

$$p(I = 1) = \frac{e^{-y}}{1 + e^{-y}} \tag{3.12}$$

Now, to evaluate the classifier's prediction capability there will be a need for loss function. Generally, for binary class classification binary cross-entropy loss function is used which is also known as a log loss function. Mathematically, loss function can be defined by [83]:

$$L = -I \log\big(p(I)\big) + (1 - I)\log\,(1 - p(I)) \tag{3.13}$$

Here, I is the pixel level value of ground truth mask at location i.e. 1 for forged pixel and 0 for authentic pixel and $p(I)$ is the probability of a pixel being forged for all pixels.

3.3.2.1.2 Training of the Model

For the training of the proposed model, two different datasets have been used in this work. One is CoMoFoD [84] and another is CMFD [85]. The input size of the image

is taken as 256x256. The kernel size for the convolution has been taken as 3x3Stride value is taken as one which means feature extraction is done on the sliding window of the image. Optimization function Adam is used for the training of the model. This optimizer updates the weight of the network after each epoch based on the given condition of validation accuracy i.e. weight will update with maximum patience of ten validation accuracy. After this condition weights will be saved in the middle of the iteration. This optimization is a stochastic gradient descent method that is based on the adaptive estimation of the first order and second-order moments whose default learning rate is 0.001 is taken in this training.

Pseudo Code: Training of the proposed model

input: *D*: Input Training Dataset
output: *W*: Resultant Weight corresponding to the trained model
functions: resize: Resize original image into (h, w, d) form
split: Split Training Dataset into Training and Validation Data
scale: down sample input data
encoder_phase: consists of multiple convolution, down sample and concatenation blocks
decoder_phase: consists of multiple convolution, up sample and concatenation blocks
classification: classification of feature map
compile_train: compile and train the model using optimizer and loss function

1: **initialize** *D*;
2: *images* = resize (*D*, [256, 256, 3]);
3: *masks = resize* (*D*, [256, 256, 1]);
4: [*T, V*] = split(D,0.2) //80% for training and 20% for validation
5: **Function** Model(*T, V*)*:*
6: *S[n]* = scale(*T,V*, [2,2]); //half sampling of input data
7: *En* = encoder_phase(*S*[n]) //Feature Map from encoder phase
8: *De* = decoder_phase(*En*); //Feature Map from decoder phase
9: *R* = classification(*De*, sigmoid); //Classification of feature map using sigmoid activation
10: **Return** model*;*
11: Model = model(*T, V*);
12: *W* = compile_train(Model, optimizer = adam, loss = binary_cross_entropy)
13: **return** *W*;

By modifying the layers of the architecture variants of this model can be designed. To analyze the performance, we have modified the layers of the architecture and made their variants. The kernel size of the convolution layers has been changed from 3x3 to 5x5 and 7x7. It is observed that if we extract features from large size kernels training as

well as validation accuracy get reduced. Table 3.3 summarizes the training and validation accuracy with the size of trained weights.

Table 3.3: Training result of the proposed model on the various kernel size of convolutional layers (i.e. 3x3, 5x5 and 7x7*)*

	3x3		5x5		7x7	
	CoMoFoD	CMFD	CoMoFoD	CMFD	CoMoFoD	CMFD
Training Accuracy	0.9963	0.9931	0.9958	0.9886	0.9949	0.9671
Training Loss	0.0044	0.0201	0.0053	0.313	0.0080	0.1318
Validation Accuracy	0.9956	0.9915	0.9955	0.9793	0.9948	0.9732
Validation Loss	0.0072	0.0256	0.0065	0.0904	0.0083	0.4307
Size of Weight	41 MB	42 MB	114 MB	114 MB	223 MB	222 MB
No. of Params	10.71M	10.71M	29.73M	29.73M	58.26M	58.26M

As mentioned, the maximum number of epochs has been taken as 200 for training and validation with a condition of validation accuracy should not degrade till 10 epochs. With this condition, the training of the CMFD dataset was run till 91 epochs after which validation accuracy was decreasing and after ten epochs of patience, training weight was saved. Similarly, on the CoMoFoD dataset model run till 97 epochs and after 97th iteration validation accuracy was decreasing and hence training weight of 107th epochs was stored.

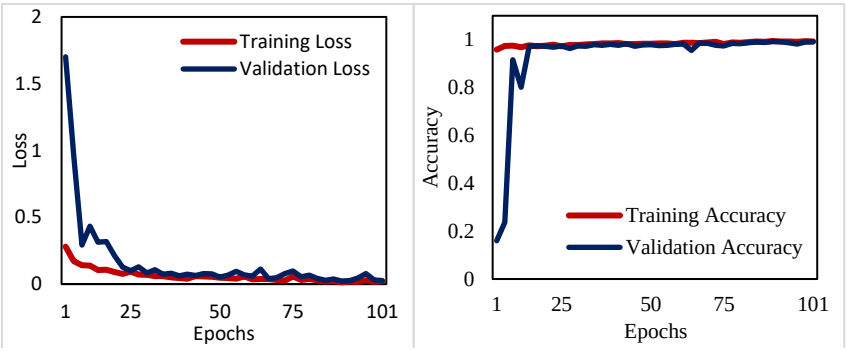

Figure 3.11: Accuracy and Loss of model (3x3) during training on CMFD dataset

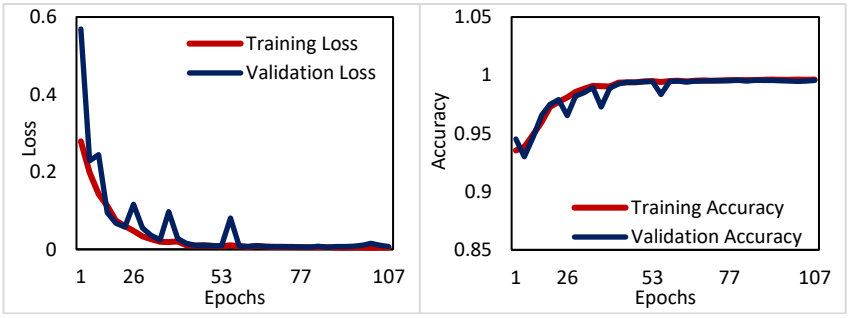

Figure 3.12: Accuracy and Loss of model (3x3) during training on CoMoFoD dataset

Training and validation accuracy with training and validation loss of each epoch on the CMFD dataset is shown using a line graph in Figure 3.11 and the same on the CoMoFoD dataset is shown in Figure 3.12.

3.3.2.2 Result Analysis and Discussion

This section explains the experiment performed for the validation of the proposed model on two different publicly available datasets. Simultion of the proposed model is performed on Ubuntu Server with the allocation of 16GB memory and a single graphics processing unit. Simulation code of architecture is written in python language with the help of different python libraries. TensorFlow GPU is used for backend and Keras libraries for front end coding of the architecture. Additional helping libraries are used like os, matplotlib, NumPy and OpenCV. The written simulation code is submitted to the server using a job scheduler and training weights are stored on the hard disk. These weights are transferred to the local system where testing is done on test cases.

Two publicly available datasets have been used for training and validation purpose. One is CMFD [85] and another is CoMoFoD [84]. Details of these datasets are already discussed in the theoretical background section of chapter 2. To evaluate the performance of the proposed model and comparison of the proposed model with other state-of-the-art methods image level as well as pixel-level analysis has been used. The performance measures defined in chapter 2 are used here for the evaluation of the proposed model.

From datasets, 10% of the images have been taken out to test the proposed model and the rest of the images are used to train and validate the model. Except for these images from both datasets 100 non-forged images (50 from each) have been also taken to test the model. The average precision, recall, accuracy, specificity (aka TNR), FNR (aka miss-rate), F1-score and MCC values are calculated and shown in Table 3.4.

Table 3.4: Average test result using performance measures precision, recall, accuracy, TNR, FNR, F1-score and MCC value on different datasets

Dataset	*p*	*r*	*a*	*tnr*	*fnr*	*f1*	*mcc*
CMFD [85]	0.9892	0.9982	0.9878	0.7764	0.0018	0.9936	0.8329
CoMOoFoD [84]	0.9863	0.9962	0.9839	0.8247	0.0038	0.9909	0.8578

Meaning of post-processing operations given in Table 3.5-

F: Only translation without post-processing, BC1-BC3: Brightness change, CA1-CA3: Contrast Adjustment, CR1-CR3: Color Reduction, IB1-IB3: Image blurring, JC1-JC9: JPEG Compression and NA1-NA3: Noise addition.

Table 3.5: Average test result using performance measures precision, recall, accuracy, TNR, FNR, F1-score and MCC value on CoMoFoD dataset on different post-processing operations

Post-processing	*p*	*r*	*a*	*tnr*	*fnr*	*f1*	*mcc*
F	0.9863	0.9961	0.9838	0.8215	0.0039	0.9909	0.8558
BC1	0.9862	0.9961	0.9837	0.8186	0.0039	0.9908	0.8537
BC2	0.9860	0.9961	0.9835	0.8152	0.0039	0.9907	0.8510
BC3	0.9852	0.9961	0.9827	0.8029	0.0039	0.9903	0.8419
CA1	0.9864	0.9961	0.9840	0.8261	0.0039	0.9910	0.8589
CA2	0.9865	0.9962	0.9841	0.8293	0.0038	0.9910	0.8613
CA3	0.9858	0.9963	0.9835	0.8279	0.0037	0.9907	0.8607
CR1	0.9863	0.9961	0.9838	0.8213	0.0039	0.9909	0.8556
CR2	0.9863	0.9961	0.9838	0.8212	0.0039	0.9909	0.8556
CR3	0.9863	0.9961	0.9838	0.8214	0.0039	0.9909	0.8556
IB1	0.9861	0.9964	0.9838	0.8221	0.0036	0.9909	0.8575
IB2	0.9862	0.9964	0.9840	0.8217	0.0036	0.9910	0.8579
IB3	0.9862	0.9964	0.9841	0.8195	0.0036	0.9910	0.8560
JC1	0.9862	0.9962	0.9838	0.8277	0.0038	0.9909	0.8595
JC2	0.9860	0.9962	0.9835	0.8268	0.0038	0.9907	0.8573
JC3	0.9864	0.9963	0.9841	0.8287	0.0037	0.9910	0.8610
JC4	0.9867	0.9962	0.9843	0.8300	0.0038	0.9911	0.8611
JC5	0.9865	0.9961	0.9840	0.8298	0.0039	0.9910	0.8604
JC6	0.9865	0.9962	0.9841	0.8281	0.0038	0.9910	0.8602
JC7	0.9863	0.9961	0.9837	0.8261	0.0039	0.9909	0.8573
JC8	0.9863	0.9961	0.9837	0.8245	0.0039	0.9909	0.8570
JC9	0.9871	0.9961	0.9846	0.8456	0.0039	0.9913	0.8710
NA1	0.9869	0.9962	0.9846	0.8269	0.0038	0.9913	0.8600
NA2	0.9868	0.9961	0.9843	0.8278	0.0039	0.9911	0.8598
NA3	0.9868	0.9961	0.9843	0.8273	0.0039	0.9912	0.8593

This result is the combined and average result of the geometrical transformation as well as post-processing operations. Individual results on different post-processing and

different geometrical transformation are shown in different tables. Visual results are also shown for the individual dataset. Performance analysis of the proposed model is also shown using a line graph on different post-processing operations for both datasets. Table 3.5 shows the average result on the CoMoFoD dataset for different post-processing operations using performance measures precision, recall, accuracy, true negative rate, false-negative rate, f1-score and MCC values.

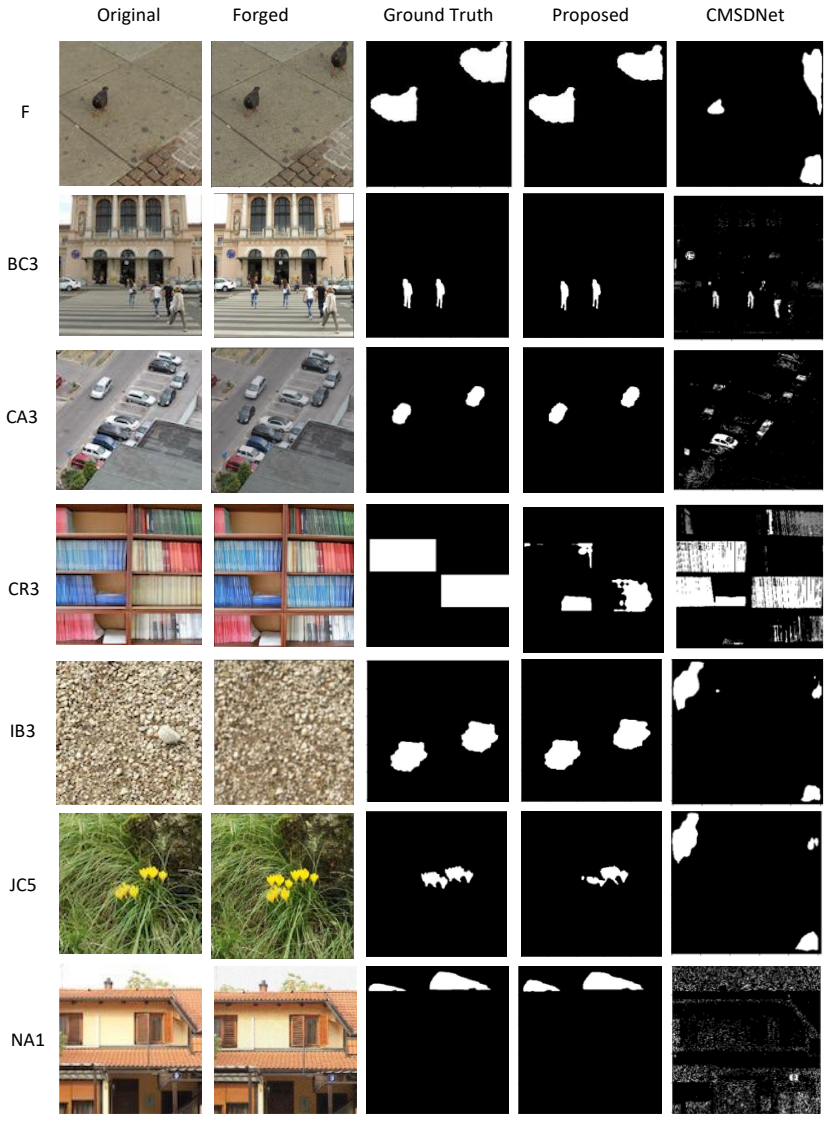

Figure 3.13: Visual result of the proposed model on test images of CoMoFoD dataset

Visual results obtained by the proposed model on the CoMoFoD dataset can be seen in Figure 3.13. From all post-processing, a random image is taken and validated

through the proposed model. Its predicted visual result can be seen in the fourth column of the shown figure. The first column of the figure shows the original image, the second column represents copied and pasted region from the original image and made it tampered whose ground truth mask is shown in the third column. The visual results shown in the fifth column is the predicted output of the state-of-the-art CMSDNet [86]. Seven rows of the visual result represent different post-processing operations performed on the tampered region. These operations are brightness enhancement, contrast adjustment, colour reduction, image blurring, JPEG compression and noise addition respectively.

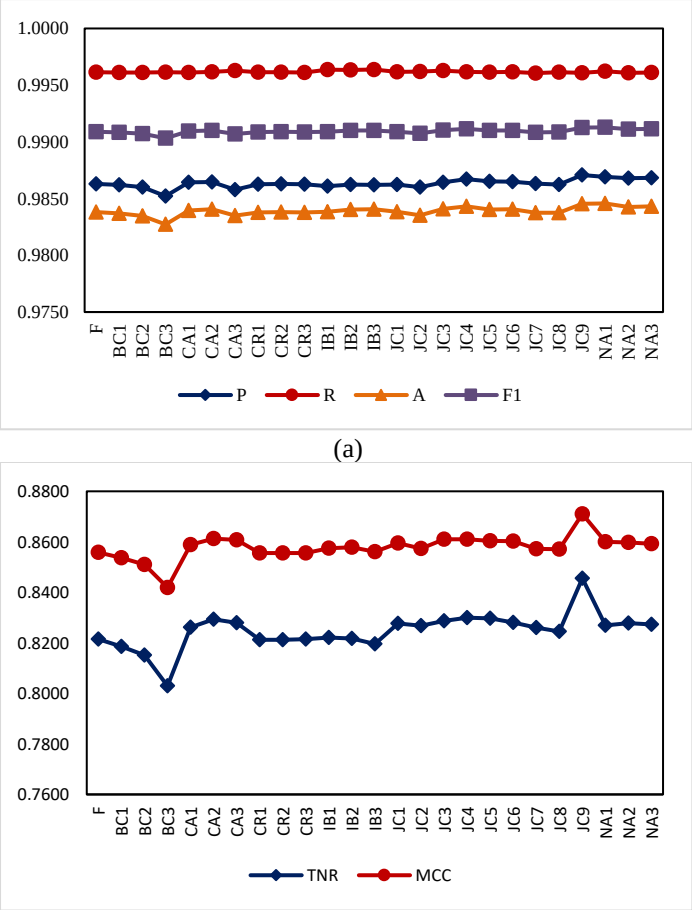

Figure 3.14: Performance analysis of the proposed model using line graph on CoMoFoD dataset (a) Precision, recall, accuracy and F1-score (b) TNR and MCC values

From these visual results, it can be concluded that the proposed model gives better results in all the cases of post-processing operations of the forged region. However, the only case of color reduction shown in the visual result tells that the proposed method

doesn't work well. Though the visual result of this case doesn't look very impressive as the method doesn't localize the whole forged region in the output result. The reason behind the false positives is that the image has multiple similar texture regions, not because of the color reduction of the forged region. Multiple similar textures may confuse the model whether the region is copied or not. Hence, the proposed model gets confused between regions and performance becomes lower. Except this in all cases result by the proposed model gives better performance.

The result analysis of the proposed model on different post-processing operations of the CoMoFoD dataset using a line graph is shown in Figure 3.14. In the first line graph precision, recall, accuracy and f1-score are shown which is not suitable in case of an imbalanced number of pixels in an image. So, another line graph is shown using MCC and true negative rate values. From the first line graph, it can be seen that in almost all post-processing operations accuracy is more than 98% and the F1-score is more than 99%. MCC values in all cases are lesser than 86%, however, MCC is a better performance measure, and 86% performance is much acceptable.

Table 3.6: The compared result of the proposed model with state-of-the-art methods on images of CoMoFoD dataset without any post-processing and with JPEG compression quality factor (qf) = 90

Method	Transformation	Without any post-processing			JPEG compression qf=90		
		p	*r*	*f1*	*p*	*r*	*f1*
J. Li et al. [87]	Rotation	0.5594	0.8281	0.5979	0.5809	0.8817	0.6285
	Scaling	0.5542	0.8492	0.6059	0.5630	0.8448	0.6037
	Distortion	0.6425	0.9045	0.6961	0.6490	0.8860	0.6902
E. Silva et al. [88]	Rotation	0.5532	0.7451	0.5573	0.3990	0.3779	0.2708
	Scaling	0.4966	0.6971	0.5008	0.4858	0.3567	0.3000
	Distortion	0.5814	0.7878	0.6550	0.5625	0.3825	0.3571
Y. Liu et al. [39]	Rotation	0.6833	0.9006	0.7174	0.5977	0.8014	0.6369
	Scaling	0.5696	0.7516	0.5864	0.5169	0.7248	0.5449
	Distortion	0.6631	0.8516	0.6986	0.5963	0.7787	0.7175
B. Chen et al. [86]	Rotation	0.9845	0.9834	0.9839	0.9910	0.9965	0.9937
	Scaling	0.9907	0.9973	0.9940	0.9956	0.9367	0.9652
	Distortion	0.9045	0.9559	0.9295	0.9742	0.9409	0.9573
Proposed	Rotation	0.9943	0.9973	0.9958	0.9942	0.9974	0.9958
	Scaling	0.9964	0.9970	0.9967	0.9964	0.9971	0.9967
	Distortion	0.9808	0.9956	0.9877	0.9807	0.9956	0.9877

The comparisons of the proposed result with stat-of-the-art techniques using precision, recall and f1-score are given in tables from Table 3.6 to Table 3.8. In these compared methods two methods are based on deep learning networks. Table 3.6 shows the comparison of the performance of the proposed model with state-of-the-art techniques on the various geometrical transformed images of the CoMoFoD dataset without any post-processing operations and with the post-processing operation JPEG compression of quality factor 90. However, images are also compressed from a quality factor of 20 to a quality factor of 90 with a gap of 10. In this table quality factor, 90 is compared because these images have maximum quality factor compression.

Table 3.7: The compared result of the proposed model with state-of-the-art methods on images of CoMoFoD dataset with Noise addition (variance = 0.0005) and Image Blurring

Algorithm	Transformation	Noise addition (var = 0.0005)			Image Blurring		
		p	*r*	*f1*	*p*	*r*	*f1*
J. Li et al. [87]	Rotation	0.6202	0.8399	0.6528	0.4481	0.8753	0.5280
	Scaling	0.5673	0.7438	0.5849	0.4514	0.9096	0.5304
	Distortion	0.6806	0.7821	0.7013	0.5022	0.9449	0.5953
E. Silva et al. [88]	Rotation	0.5924	0.6481	0.5265	0.5183	0.7043	0.5335
	Scaling	0.6159	0.5115	0.4987	0.5281	0.6994	0.5212
	Distortion	0.6828	0.5627	0.5270	0.6243	0.8292	0.6048
Y. Liu et al. [39]	Rotation	0.6385	0.8076	0.6578	0.5114	0.8591	0.5945
	Scaling	0.5838	0.6840	0.5677	0.4890	0.7836	0.5540
	Distortion	0.7380	0.8411	0.7627	0.5715	0.8949	0.6611
B. Chen et al. [86]	Rotation	0.9907	0.9933	0.9920	0.9908	0.9968	0.9938
	Scaling	0.9948	0.9005	0.9453	0.9907	0.9966	0.9937
	Distortion	0.9661	0.9084	0.9363	0.9177	0.9426	0.9300
Proposed	Rotation	0.9944	0.9972	0.9958	0.9940	0.9977	0.9958
	Scaling	0.9965	0.9969	0.9967	0.9961	0.9974	0.9967
	Distortion	0.9815	0.9956	0.9881	0.9806	0.9957	0.9877

The CoMoFoD dataset constitutes images with three types of additive noise in which noises are added in the image with the variance of 0.005, 0.009 and 0.0005. From which result on images with additive noise of variance of 0.0005 is compared. Image blurring is a post-processing operation that is performed on images using different kernel sizes. In this dataset, image blurring is performed on three different kernel sizes i.e. 3x3, 5x5 and 7x7. Table 3.7 shows the compared result of the proposed model with state-of-the-art methods on different test images of the CoMoFoD dataset. These images are either

having post-processing of noise addition or having image blurring. After these post-processing operations, some transformations like rotation, scaling and distortion are also performed. Distortion comprises of different skews.

Table 3.8: The compared result of the proposed model with state-of-the-art methods on images of the CoMoFoD dataset with brightness change, color reduction and contrast adjustment

Algorithm	Transformation	Brightness Change			Colour Reduction			Contrast Adjustment		
		p	*r*	*f1*	*p*	*r*	*f1*	*p*	*r*	*f1*
J. Li et al. [87]	Rotation	0.5601	0.8445	0.5933	0.5692	0.8340	0.6174	0.5447	0.8428	0.5972
	Scaling	0.5537	0.7860	0.5926	0.5628	0.8969	0.6251	0.5714	0.8400	0.5973
	Distortion	0.6464	0.8964	0.6892	0.6352	0.9226	0.6955	0.6445	0.8994	0.6941
E. Silva et al. [88]	Rotation	0.5272	0.7314	0.5333	0.5432	0.6567	0.5066	0.5722	0.6987	0.5321
	Scaling	0.3977	0.6305	0.4378	0.5004	0.7092	0.5129	0.5334	0.7227	0.5059
	Distortion	0.4834	0.7168	0.5137	0.6147	0.7584	0.5813	0.6450	0.7854	0.6080
Y. Liu et al. [39]	Rotation	0.6075	0.9063	0.6531	0.6583	0.8415	0.6891	0.6157	0.8675	0.6590
	Scaling	0.5350	0.7290	0.5526	0.5984	0.7832	0.6267	0.5517	0.7987	0.5948
	Distortion	0.6342	0.7814	0.6609	0.6652	0.8193	0.6827	0.6610	0.8476	0.6924
B. Chen et al. [86]	Rotation	0.9908	0.9972	0.9940	0.9907	0.9972	0.9939	0.9906	0.9974	0.9940
	Scaling	0.9909	0.9971	0.9940	0.9908	0.9972	0.9940	0.9905	0.9975	0.9940
	Distortion	0.9895	0.9877	0.9886	0.9562	0.9303	0.9431	0.9781	0.9780	0.9781
Proposed	Rotation	0.9938	0.9972	0.9955	0.9943	0.9973	0.9957	0.9943	0.9973	0.9957
	Scaling	0.9957	0.9968	0.9963	0.9964	0.9969	0.9966	0.9936	0.9974	0.9955
	Distortion	0.9795	0.9957	0.9870	0.9807	0.9956	0.9877	0.9815	0.9956	0.9880

Another post-processing operation performed on images of the CoMoFoD dataset is brightness change. The result produced by the proposed model on these images and comparison of the result with other state-of-the-art methods is shown in Table 3.8. In this table, the proposed model is compared with other state-of-the-art methods on the images of the CoMoFoD dataset with color reduction and contrast stretching. A post-processing operation ‘color reduction’ is performed on images of the dataset three times with different parameters. Contrast stretching is also performed on these images. A comparison of the proposed method with other state-of-the-art methods is also done in this table. From these comparisons, it can be deduced that the proposed deep learning approach is far better than the conventional approaches. Although in the case of distortion (i.e. horizontal skewness and vertical skewness) result is lesser than rotation and scaling, the overall result of the proposed model is greater than other state-of-the-art methods.

Another dataset is CMFD on which the proposed model is evaluated using different performance measures. Here also, pixels are imbalanced concerning forged and non-forged classes. Thus, MCC value is much important, so except precision, recall, accuracy and F1-score other performance measures are calculated like MCC, miss rate and specificity. The geometric transformation-wise average quantitative result on the given dataset is shown in Table 3.9.

Table 3.9: The proposed model result on CMFD dataset on different transformation

Transformation	*p*	*r*	*a*	*tnr*	*fnr*	*f1*	*mcc*
Translation	0.9750	0.9862	0.9633	0.4475	0.0138	0.9805	0.4325
Rotation	0.9972	0.9988	0.9961	0.8825	0.0012	0.9980	0.9136
Scaling	0.9765	0.9989	0.9762	0.6290	0.0011	0.9874	0.7443

The visual results of the proposed model on the given dataset are shown in Figure 3.15.

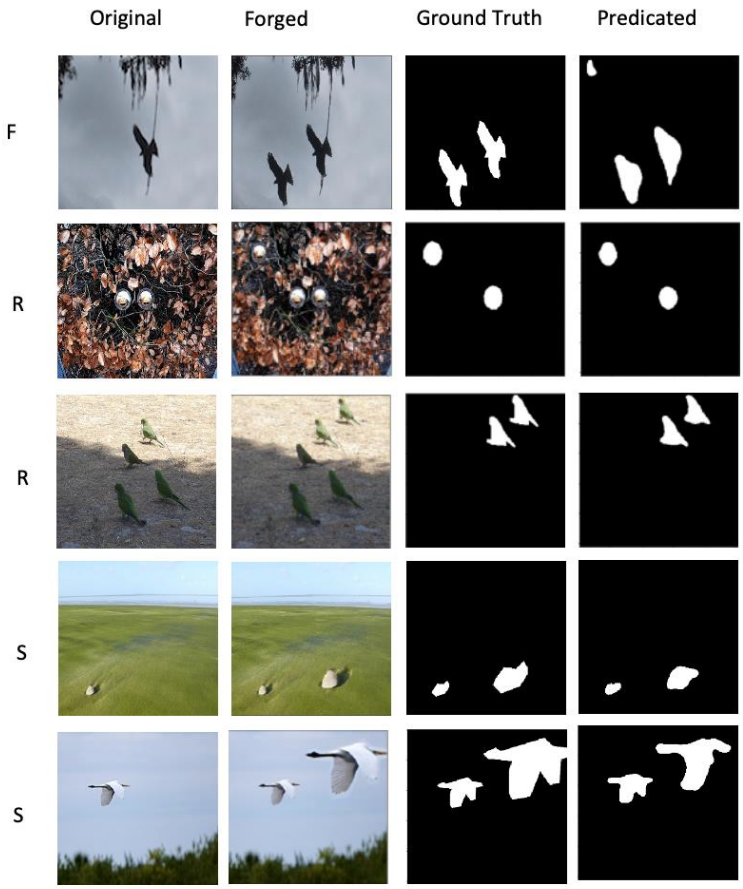

Figure 3.15: The visual results of the proposed model on images of the CMFD Dataset

In this visual result, the first row contains a direct translation of the copied region and made it tampered with using a copy-move forgery attack. The second and the third row represent the rotation of the copied region and the fourth, fifth rows show scaling geometric transformation. The first column of visual result shows the original image, the second column shows tampered image, the third column represents the ground truth mask of a tampered image and the last column represents the visual outcome from the proposed model. The visual result shows that the proposed model is scaling invariant. The fourth and fifth rows have tampered images in which seashell and bird are scaled up and the proposed model can identify the forged region. To analyze the performance of the proposed model on different transformed images of given dataset line graphs of all performance measures are shown in Figure 3.16. Precision, recall, accuracy and f1-score are shown in Figure 3.16(a) and true negative rate and MCC values are shown in Figure 3.16(b).

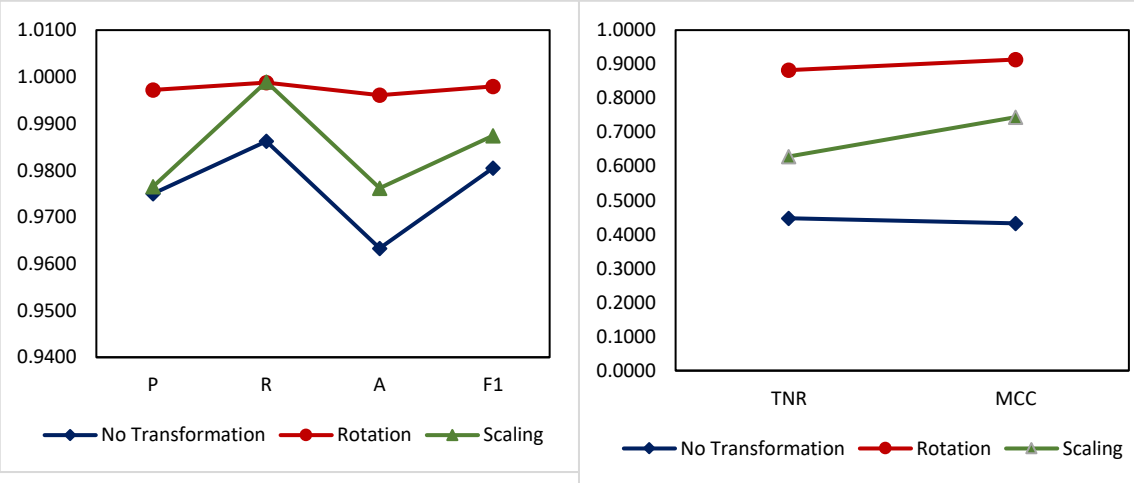

*Figure 3.16: Performance analysis of the proposed model using line graph on CMFD dataset (a) **p**recision, **r**ecall, **a**ccuracy and **F1**-score (b) TNR and MCC values*

The above-mentioned results and comparisons are pixel-level analyses. In pixel-level analysis pixels of an image can be classified into forged and non-forged pixels. Hence, forged region in an image can be localized in the pixel-level analysis. Another

analysis is image level, in this analysis, the only image is to be checked whether it has been tampered with or not.

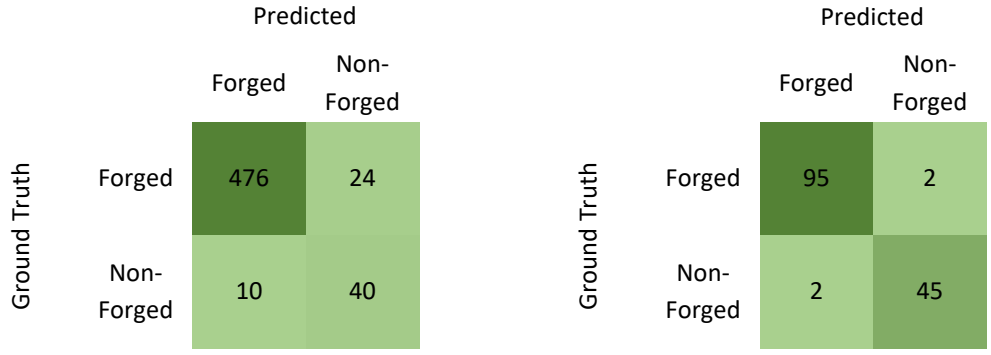

Figure 3.17: Image level analysis of the proposed model on datasets (a) CoMoFoD (b) CMFD

The forged region is not to be localized in the image level analysis. In this work image-level analysis is also done based on the number of true-negative, false-positive, and false-negative pixels. From both datasets, 50-50 original images have been taken and given to the trained model. In the case of the model predicted zero true negative pixels and very lesser false positive and false negative pixels. Then the image is counted non-tampered image else it will be counted as tampered image. Based on these tampered and non-tampered images, the confusion matrix for both datasets is shown in Figure 3.17. Based on these confusion matrix performance measures of the proposed model are also calculated. Evaluated results obtained on both datasets are given in the following Table 3.10.

Table 3.10: Image level analysis of the proposed model on different datasets

	p	*r*	*a*	*tnr*	*fnr*	*f1*	*mcc*
CoMoFoD [84]	0.9794	0.9520	0.9382	0.8000	0.0480	0.9655	0.6742
CMFD [85]	0.9794	0.9794	0.9722	0.9574	0.0206	0.9794	0.9368

3.4 Summary

Copy-move forgery is a commonly used forgery attack in the digital image to conceal any information in the image. A lot of literature has already been reported for the detection and localization of such types of forgery attacks. These are block-based approaches, keypoint-based approaches and data-driven approaches. Block-based approaches are suffering from the problems of transformation of forged region and computation cost. Keypoints based approaches are suffering from the problem of clustering in case of small region duplication and post-processing operations of the forged region (i.e. noise addition, brightness change etc). Data-driven approaches are either based on image-level analysis or have poor performance results. Considering the limitations of state-of-the-art techniques, we have proposed two different copy-move-forgery detection techniques, one is a combination of block-based and keypoint-based techniques and another is a data-driven-based deep learning technique.

The first approach is a copy-move image forgery detection based on ORB and DCT features along with a distance-based clustering algorithm and k-NN matching based on Hamming distance. This method can detect copy-move regions without any prior information about a forged image. Compared to the previously established key point and block-based methods this technique is more effective at detecting forged images. Experimental results show the proposed technique is robust to not only post-processing operations like contrast reduction, blurring, noise and brightness change but also geometric altercations like changes in the degree of rotation, changes in scale, distortion, an amalgamation of them and multiple copy-move forged regions in a single image. Another is a deep learning-based method using a multi-scale multistage network. A multi-scale input-based deep learning convolution neural network is developed to localize the forged region in a copy-move forged image. The multi-scale input image and its

convolution features are scale-invariant this is the reason why the scaled forged region is also identified by the proposed model. The proposed model is trained and validated on two different publicly available datasets. In this work, image-level analysis, as well as pixel-level analysis is done. The performance result of the proposed model shows that the method is robust against geometrical transformation as well as post-processing operations.